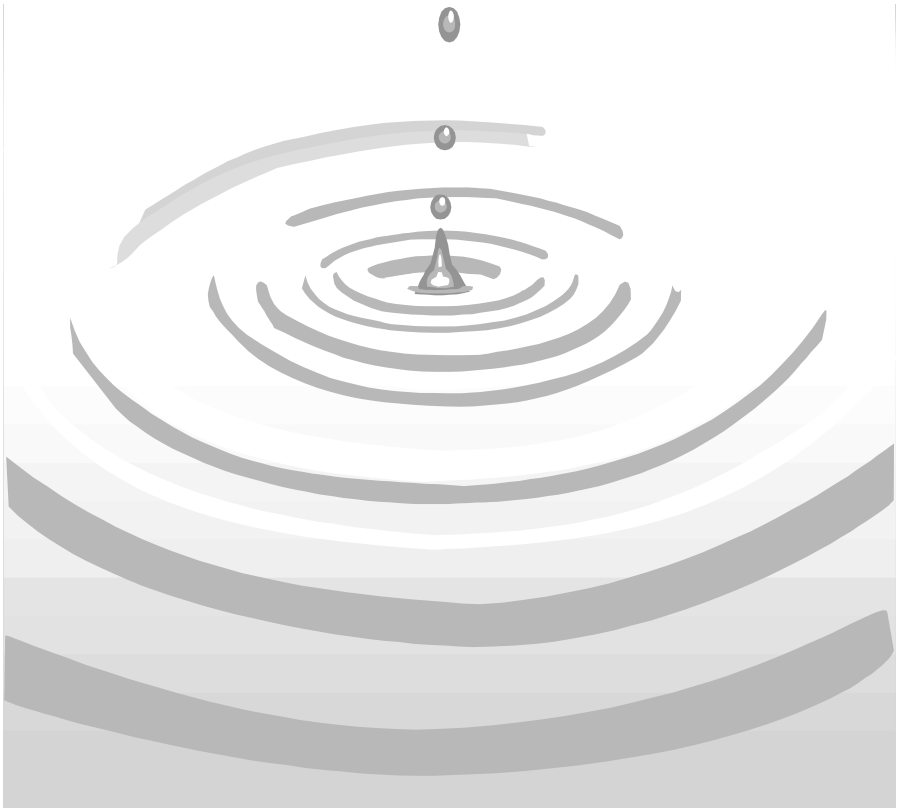
Philippe Grosjean
(phgrosjean@sciviews.org)

Frédéric Ibanez
(ibanez@obs-vlfr.fr)

PASTECS



Package for Analysis of Space-Time Ecological Series

Manuel de l'utilisateur de la librairie de fonctions pour R et pour S+

http://www.sciviews.org/pastecs

PASTECS version 1.2-0 pour R v. 2.0.0 & version 1.0-1 pour S+ 2000 rel 3, le 03/10/2004

*Nous tenons à remercier **Benoît Beliaeff (IFREMER Nantes**) pour son soutien sans faille, et surtout pour ses bonnes idées qui ont permis d'arriver au résultat actuel. L'idée de traduire les algorithmes du logiciel PASSTEC 2000 en une librairie S+ est sienne, de même que d'autres suggestions pertinentes à divers niveaux. C'est d'une collaboration étroite avec Benoît pendant deux années dans la gestion de ce projet que PASTECS a pu naître en France au sein du **PNEC** (Programme National d'Ecologie Côtière), et plus particulièrement au sein de "l'**ART4**". Enfin, Benoît a été le principal et le plus efficace "beta-testeur" tout au long de la phase de développement.*

*Nous remercions également **Michèle Etienne (Laboratoire Océanologique de Villefranche-sur-mer**) pour sa contribution à la programmation des algorithmes de PASSTEC 2000 duquel la librairie PASTECS s'est inspirée. Nos remerciements vont également aux nombreux étudiants d'IUT qui ont contribué à la réalisation de PASSTEC 2000 pour la même raison.*

Table des matières

PASTECS

Manuel de l'utilisateur de la librairie de fonctions pour R et pour S+

Introduction

Plusieurs progiciels d'analyse de données en écologie ont été développés dans des laboratoires scientifiques; on notera par exemple l'existence du progiciel **PRIMER** (Plymouth Routines In Multivariate Ecological Research) en Angleterre, plus particulièrement orienté vers le traitement des données benthiques marines, consécutivement à la demande des spécialistes du domaine. De manière analogue, en réponse à une demande des participants du PNOC SLT (Programme National en Océanographie Côtière, thème « séries à long-terme ») en France, la conception du Progiciel **PASSTEC** (Programme d'Analyse des Séries Spatio-Temporelles en Ecologie Côtière, disponible auprès de Frédéric Ibanez, ibanez@obs-vlfr.fr) a démarré dès 1993. L'ensemble de ses applications est axé sur l'analyse des séries chronologiques hydro-climatiques et biologiques. F. Ibanez et M. Etienne (Laboratoire Océanologique de Villefranche-sur-mer) ont développé plusieurs dizaines d'applications en Fortran ou en Basic, et en ont assuré progressivement l'implantation en Visual Basic dans l'environnement PC-Windows. Ce travail s'est essentiellement effectué avec la collaboration d'étudiants stagiaires en informatique de l'Institut Universitaire de Technologie de Nice - Sophia Antipolis.

Compte tenu de la nécessité actuelle de synthèse des enregistrements hydro-climatiques fournis par les réseaux d'observation, et de la valorisation de toutes les données biologiques récoltées par les laboratoires côtiers, PASSTEC a mis à la disposition des écologistes non informaticiens, un outil performant permettant de traiter les séries chronologiques. Après une large diffusion du progiciel auprès des participants du PNOC SLT, PASSTEC a été mis à la disposition de nombreux chercheurs de la communauté océanographique française et étrangère (notamment à l'occasion du cours européen NAME (Numerical Analysis in Marine Ecology, juillet 1996, Villefranche sur Mer).

Avec le temps, les limitations imposées par le choix de Visual Basic comme langage de programmation du logiciel, ainsi que l'évolution de langages de troisième génération spécialisés dans le calcul scientifique, tels que S+, R ou Matlab, ont fait apparaître la nécessité de réécrire l'ensemble des routines dans un langage plus performant pour le calcul matriciel. Cette librairie aurait pu s'appeler logiquement PASSTEC+, mais comme le nom "PassTec" est déjà enregistré pour un autre programme, nous avons décidé de le modifier légèrement tout en anglicisant son acronyme (puisque la bibliothèque est écrite en anglais, comme toutes les librairies Splus ou les packages R). Le nom de cette librairie devient ainsi PASTECS pour "Package for Analysis of Space-Time Ecological Series". L'ouverture de l'ancien PASSTEC prioritairement donnée aux non-spécialistes numériciens sera maintenue et développée ultérieurement, mais les besoins précis d'écologistes plus spécialisés, notamment dans le cadre du PNEC Art4 en France, sont également pris en compte dans PASTECS.

La librairie PASTECS reprend le concept développé dans l'ancien PASSTEC: rassembler dans un même progiciel des analyses –originales pour certaines– utiles au biologiste écologiste numéricien qui travaille sur des séries spatio-temporelles. PASTECS offre cependant la possibilité aux programmeurs d'inclure ces méthodes au sein de routines

personnelles en langage S. Il est donc, de ce point de vue, infiniment plus souple et puissant que l'ancien PASSTEC. Etant donné que des versions de S+ ou de R sont exécutables sur de nombreuses plate-formes différentes (PC, Mac, stations Unix), PASTECS peut être également utilisé en dehors du couple PC-Windows, absolument indispensable pour exécuter l'ancien PASSTEC.

En séparant totalement les routines de calcul (librairie 'pastecs') de l'interface utilisateur, cette nouvelle version donne libre court au développement d'interfaces personnelles, éventuellement plus adaptées à un traitement donné. Ainsi à l'avenir, l'utilisateur aura le choix entre au moins 2 interfaces différentes: la fenêtre en ligne de commande de S+ ou R, et une interface graphique à menus et boites de dialogues qui se rapproche de celle disponible sous l'ancien PASSTEC dans sa version 2000 (et il pourra même mixer les deux interfaces librement).

Enfin, le code source de la librairie PASTECS est distribué librement, alors que l'ancien PASSTEC n'était distribué que sous forme binaire. De plus, PASTECS étant programmé en langage matriciel, cela permet d'avoir un code plus compact et plus proche de la formulation mathématique du traitement, et donc, plus lisible. Les avantages sont triples. Tout d'abord, l'utilisateur peut vérifier lui-même le code pour s'assurer que le traitement effectué correspond bien à ce qu'il désire faire. Ensuite, il peut corriger les bugs lui-même lorsqu'il en découvre (**aucun** logiciel n'est totalement exempt de bugs!). Cette caractéristique est vitale pour un logiciel non commercial qui ne dispose par d'un service technique « après-vente » structuré. Enfin, les utilisateurs peuvent adapter le code et étendre les fonctions eux-même, au gré de leurs besoins personnels, voire de l'évolution des connaissances dans le domaine du traitement des données. Comme cela a été déjà démontré maintes fois pour les logiciels du domaine public ou ceux diffusés en « open source », une telle approche garantit l'évolution du logiciel et sa relative pérennisation.

En définitive, la librairie PASTECS offre beaucoup plus de liberté à l'utilisateur quant à la façon dont il peut utiliser les routines d'analyse disponibles. Sans compter que, étant intégré dans un environnement S+ ou R, des milliers de fonctions statistiques, mathématiques ou graphiques propres à cet environnement ou disponibles à partir d'autres librairies sont également présentes pour le traitement des données.

> *Le présent manuel a pour but de permettre l'utilisation des fonctions de la librairie PASTECS, soit à partir de la fenêtre de commande, soit encore dans un script en langage S. Il ne présente pas l'interface utilisateur graphique de PASTECS. Il s'adresse donc aux utilisateurs* ***qui ont au moins des notions élémentaires du langage S*** *(nous rappelerons cependant les notions clefs au fur et à mesure de leur apparition dans ce manuel), et qui veulent tirer le maximum des analyses en ayant accès à toutes les fonctions, directement en entrant du code.*

Organisation de ce document

Le présent manuel détaille toutes les fonctions de la librairie PASTECS, version 1.2-0 sous R et 1.0-1 sous S+ 2000. Ces fonctions se présentent de manière identique entre les deux versions S+ et R. Seules quelques subtiles différences existent entre ces deux implémentations de la librairie. Elles sont signalées dans le texte. La première partie est une brève introduction théorique des différentes méthodes d'analyse reprises dans les fonctions de PASTECS, suivie d'application de ces méthodes sur des exemples pratiques. Ces exemples correspondent, pour l'essentiel, à ceux présentés dans l'aide en ligne de la librairie de fonction. La seconde partie, plus technique, présente les différentes fonctions de la librarie par ordre alphabétique, et détaille la signification des différents arguments des fonctions. Cette partie est quasiment la traduction en français de l'aide en ligne en anglais de la librairie.

Les conventions typographiques adoptées dans ce manuel sont les suivantes. Le code à entrer à la ligne de commande est présenté en police non proportionnelle (courrier) et en rouge, précédé du prompt ('>' affiché par le programme en début de ligne de commande, et qu'il ne faut pas taper soi-même). Les résultats textuels retournés par le moteur de calcul dans la fenêtre de commande sont présentés dans la même police, mais en bleu. L'exemple suivant illustre une ligne de code entrée à la ligne de commande, avec son prompt, suivie d'une ligne de résultat textuel retourné par le moteur de calcul:

```
> (1:3)^2

[1] 1 4 9
```

A noter qu'il est permit de rentrer une ligne de commande très longue sur plusieurs lignes **à condition que la commande soit syntaxiquement incomplète** avant de taper <enter>. Le logiciel comprend alors qu'il doit attendre la suite avant de traiter la commande et le signale par un prompt modifié, en forme de signe '+' à la place du pompt '>' classique. Naturellement, ce signe '+' ne doit pas non plus être tapé: il est également renvoyé automatiquement par le programme en début de ligne de commande, tout comme le '>':

```
> (1:3
+ )^2

[1] 1 4 9
```

Dans le texte, le nom des fonctions est présenté en gras et italique, suivi de parenthèses comme: ***une.fonction()***. Les noms de variables sont également en gras italique, mais non suivis de parenthèses, comme ***ma.variable***. Les noms de classes d'objets S apparaissent en italique dans le texte, éventuellement entourés de guillements simples s'il s'agit de leur première définition, ou pour insister sur leur nature, comme dans *'classe'*. Les arguments de fonction sont présentés en gras: **un.argument = 1**. Enfin, les entrées de menus sont présentées en gras, italique et en couleur brune, comme ***File -> Open…*** qui signifie: sélectionner le menu "File", et cliquer sur l'élément de sous-menu "Open…".

Licence de la librairie PASTECS

PASTECS se démarque de l'ancien PASSTEC par le fait que le code source est distribué librement. Ainsi, tout utilisateur peut télécharger le logiciel et/ou son code source (voir le site web http://www.sciviews.org/pastecs) et l'adapter à ses propres besoins librement, à condition qu'il respecte la « **GNU General Public License** » version 2 ou ultérieure sous laquelle la librairie PASTECS est distribuée. Cette licence impose notamment à l'utilisateur de redistribuer le fruit de son travail sous la même licence s'il effectue des modifications à PASTECS. Les termes de la licence originale, rédigée en anglais, sont disponibles à: http://www.gnu.org/copyleft/gpl.html. Le lecteur intéressé trouvera une traduction en français de cette license en annexe. Il va sans dire que seule la version officielle en anglais établi les termes légaux de la distribution de logiciels sous license GPL.

> *Outre les dispositions relatives à la GNU General Public License, tout utilisateur qui emploie une ou plusieurs fonctions de la librairie PASTECS pour réaliser une analyse qu'il publie ensuite (que ce soit sous forme d'un article scientifique, d'un chapitre ou d'un livre en entier, ou d'un rapport public ou confidentiel)* ***est expressément tenu de citer le programme PASTECS dans le document en question****, et d'indiquer où il est possible de se le procurer (adresse du site Web).*

En cas de problèmes…

Si vous avez des difficultés à installer ou à utiliser la librairie PASTECS, référez-vous en tout premier lieu à son site web (http://www.sciviews.org/pastecs), et en particulier, vérifiez s'il n'y a pas un fichier 'ReadMe' ou 'FAQ' (Foire Aux Questions, ou Frequently Asked Questions, en anglais) récent qui répondrait à votre question. Si votre première recherche est infructueuse, vous pouvez contacter Frédéric Ibanez (ibanez@obs-vlfr.fr) pour les aspects méthodologiques, Benoit Beliaeff (benoit.beliaeff@ifremer.fr) pour la gestion du projet ou Philippe Grosjean (phgrosjean@sciviews.org) pour des questions techniques ou concernant le code source de l'application. Notez que vous pouvez aussi remplir des formulaires de **suggestion** ou de **rapport de bug** directement sur le site web (dans la section EXCHANGE de http://www.sciviews.org).

Partie I: utilisation de la librairie PASTECS

Les différentes fonctions qui constituent la librairie PASTECS sont décrites ici. Elles sont présentées par thème de manière à regrouper dans un même paragraphe toutes les fonctions apparentées. De plus, une courte synthèse théorique des méthodes statistiques traitées introduit chaque chapitre. Pour un descriptif complet de chaque fonction, veuillez vous référer à la partie II, 'fonctions de PASTECS'.

Organisation de la librairie PASTECS

L'ensemble du code de traitement statistique de PASTECS est regroupé dans la librairie S+ ou le package R 'pastecs', tous deux communément dénommés ici « librairie PASTECS ». L'interface utilisateur graphique, donc, les menus et boites de dialogue ne sont pas contenus dans cette librairie. La motivation de cette séparation est essentiellement de regrouper les fonctions de traitement qui peuvent s'exécuter sur tout système (versions PC sous Windows ou Linux, versions Unix sur stations de travail, versions Macintosh) dans la librairie PASTECS. L'interface utilisateur graphique étant beaucoup plus « système-dépendante », elle est plus limitée au niveau de la compatibilité des systèmes. Cette organisation permet également de concevoir facilement des interfaces utilisateur alternatives (spécialisées pour un traitement donné ou pour une équipe de recherche spécifique), sans toucher à la librairie de fonctions sous-jacente. Il est également possible qu'au cours du développement, d'autres subdivisions s'avèrent nécessaires.

Ce manuel présente uniquement les fonctions qui se retrouvent dans la librairie PASTECS. Les différentes fonctions implémentées dans la version actuelle de la librairie (version 1.2-0 sous R et 1.0-1 sous S+ 2000) sont résumées dans le tableau I.

Tableau I: *fonctions implémentées dans la librairie PASTECS, regroupées par thème.*

Fonction PASTECS	page	Description
daystoyears()	p. 189	Converti les unités de temps de ‘days’ à ‘years’
yearstodays()	p. 278	Converti les unités de temps de ‘years’ à ‘days’
disjoin()	p. 199	Recodage disjonctif complet des données
buysbal()	p. 185	Crée un tableau de Buys-Ballot
stat.desc()	p. 261	Statistiques descriptives classiques
stat.pen()	p. 262	Statistiques descriptives de Pennington
pennington()	p. 226	Estimateur de Pennington
stat.slide()	p. 263	Statistiques glissantes
print.stat.slide()	p. 243	Résultat des statistiques glissantes
plot.stat.slide()	p. 234	Graphe des statistiques
lines.stat.slide()	p. 222	Ajoute un tracé au graphe
escouf()	p. 201	Crée un objet ‘escouf’ (méthode d’Escoufier)
print.escouf()	p. 241	Résultats de l’analyse d’Escoufier
summary.escouf()	p. 266	Résultats détaillés de l’analyse
plot.escouf()	p. 230	Graphe de l’analyse
lines.escouf()	p. 220	Trace un seuil sur le graphique
identify.escouf()	p. 213	Identification interactive d’un seuil sur le graphe
extract.escouf()	p. 204	Extrait les variables à un seuil donné
abund()	p. 183	Crée un objet ‘abund’ (tri par abondance)
print.abund()	p. 240	Résultats de l’analyse
summary.abund()	p. 265	Résultats détaillés de l’analyse
plot.abund()	p. 228	Graphe de l’analyse
lines.abund()	p. 219	Trace un repère sur le graphique
identify.abund()	p. 212	Identification interactive d’un repère sur le graphe
extract.abund()	p. 203	Extrait les variables les plus abondantes
regul()	p. 252	Régularise une série spatio-temporelle
print.regul()	p. 242	Résultats de la régularisation
summary.regul()	p. 267	Résultats détaillés de la régularisation
specs.regul()	p. 259	Récupère les paramètres d’une régularisation
plot.regul()	p. 232	Graphe de la régularisation
lines.regul()	p. 221	Ajoute une régularisation sur un graphe
identify.regul()	p. 215	Identifie des points sur le graphe
hist.regul()	p. 211	Histogramme des écarts dans le temps
extract.regul()	p. 205	Extrait une ou plusieurs séries régulières
match.tol()	p. 225	Détermine les points qui coïncident entre séries
regul.screen()	p. 257	Détermine le meilleur pas de temps
regul.adj()	p. 255	Histo. des écarts dans le temps avant traitement
tseries()	p. 273	Convertit un objet ‘regul’ ou 'tsd' en ‘time series’
is.tseries()	p. 217	L’objet est-il une ‘time series’?
regconst()	p. 249	Régularisation par la méthode constante

reglin()	p. 250	Régularisation linéaire
regspline()	p. 251	Régularisation par courbes splines
regarea()	p. 247	Régularisation par les aires
AutoD2()	p. 184	Calcul et graphe de l'AutoD2
CrossD2()	p. 188	Calcul et graphe du CrossD2
CenterD2()	p. 187	Calcul et graph du D2 au centre
GetUnitText()	p. 210	Formatte les unités de temps pour les graphes
turnpoints()	p. 276	Calcule les points de retournement (PR)
print.turnpoints()	p. 246	Résultat du calcul des PR
summary.turnpoints()	p. 270	Résultat détaillé du calcul des PR
plot.turnpoints()	p. 239	Graphe de l'information associée aux PR
lines.turnpoints()	p. 223	Enveloppes maxi, mini et points médians des PR
extract.turnpoints()	p. 208	Position des PR dans la série originelle
turnogram()	p. 274	Calcule un turnogramme
print.turnogram()	p. 245	Résultats du turnogramme
summary.turnogram()	p. 269	Résultats détaillés du turnogramme
plot.turnogram()	p. 237	Graphe du turnogramme
identify.turnogram()	p. 216	Identifie un intervalle d'extraction sur le graphe
extract.turnogram()	p. 207	Extrait (recalcule) la série avec un intervalle donné
first()	p. 209	Extrait le premier élément d'un vecteur
last()	p. 218	Extrait le dernier élément d'un vecteur
pgleissberg()	p. 227	Probabilité associée à une distribution de Gleissberg
vario()	p. 277	Calcule et trace le semi-variogramme
disto()	p. 200	Calcule et trace un distogramme
trend.test()	p. 271	Détermine s'il y a une tendance dans la série
local.trend()	p. 224	Mets en évidence des tendances locales
identify.local.trend()	p. 214	Identifie interactivement des tendances locales
tsd()	p. 272	Décompose une ou plusieurs séries régulières
print.tsd()	p. 244	Résultats de la décomposition
summary.tsd()	p. 268	Résultats détaillés de la décomposition
specs.tsd()	p. 260	Récupère les paramètres d'une décomposition
plot.tsd()	p. 235	Graphe de la décomposition
extract.tsd()	p. 206	Extrait une ou plusieurs séries régulières
decdiff()	p. 192	Décomposition par la méthode des différences
decaverage()	p. 190	Décomposition par les moyennes mobiles
decmedian()	p. 197	Décomposition par les médianes mobiles
decevf()	p. 194	Décomposition par les vecteurs propres
decreg()	p. 198	Décomposition par régression
decloess()	p. 195	Décomposition par la méthode loess
deccensus()*	p. 192	Décomposition par la méthode census II

* *Dans la version 1.2-0 sous R uniquement.*

Jeux de données exemples de la librairie PASTECS

En plus des fonctions décrites dans le paragraphe précédent, la librairie PASTECS contient aussi quatre jeux de données d'exemple qui nous serviront à illustrer les différentes techniques d'analyse présentées dans ce manuel: ***bnr***, ***marbio***, ***marphy*** et ***releve***.

Comme conséquence de son "origine océanologique" (le nom de l'ancien PASSTEC signifiait "Programme d'Analyse des Séries Spatio-Temporelles en Ecologie Côtière"), ces données d'exemple correspondent à des communautés animales ou végétales marines. Elles sont toutefois représentatives également du type de données que l'on rencontre en écologie de manière plus générale, y compris terrestre, puisqu'il s'agit de matrices d'abondance espèces/taxa x stations le long d'un transect (séries spatiales unidimensionnelles ou séries spatio-temporelles lorsque le temps nécessaire à la mesure des différentes stations ne peut pas être considéré comme négligeable) avec (***marbio*** / ***marphy***) ou sans (***bnr***) mesures correspondantes du milieu, ou alors de mesures effectuées à différents moments à une seule station (séries temporelles, ***releve***).

bnr, 163 espèces dénombrées dans une communauté animale benthique le long d'un transect

Le *data frame* ***bnr*** contient 163 colonnes et 103 lignes. Chaque colonne correspond à une espèce identifée dans une communauté benthique et numérotée "S1" à "S163". Les 103 lignes sont autant de stations de mesure régulièrement espacées le long d'un transect. La valeur dans chaque case est le nombre d'individus de l'espèce donnée observé dans un quadrat de la station correspondante (matrice type espèces x stations). Ces données ne sont pas publiées. Elles nous serviront principalement à illustrer la méthode de tri par abondance afin d'éliminer les espèces rares (présentant beaucoup de zéros), mais également à extraire les espèces rares mais localement abondantes (présentes en très peu de stations, mais lorsqu'elles sont présentes, elles le sont en grand nombre d'individus).

marbio, zooplancton observé le long d'un transect à travers un front

Ce jeu de données de 24 colonnes (groupes taxonomiques de zooplancton) et 68 lignes (stations) échantillonné le long d'un transect à travers le front Liguro-Provençal en mer Méditerranée entre Nice (France) et Calvi (Corse). Le transect s'étend du Cap Ferrat (près de Nice) jusqu'à 65 km au large dans la direction de Calvi (cap de 123°). Les 68 stations sont régulièrement espacées le long de ce transect. Les abondances sont exprimées en nombre d'individus par mètre cube d'eau de mer. Les groupes taxonomiques identifiés sur des échantillons fixés sont les suivants:

- ***Acartia***: *Acartia clausi* – herbivore,
- ***AdultsOfCalanus***: *Calanus helgolandicus* (adults) – herbivore,
- ***Copepodits1***: idem (copépodites 1) – omnivore,
- ***Copepodits2***: idem (copépodites 2) – omnivore,
- ***Copepodits3***: idem (copépodites 3) – omnivore,
- ***Copepodits4***: idem (copépodites 4) – omnivore,
- ***Copepodits5***: idem (copépodites 5) – omnivore,
- ***ClausocalanusA***: *Clausocalanus* (classe de taille A) – herbivore,
- ***ClausocalanusB***: *Clausocalanus* (classe de taille B) – herbivore,

- ***ClausocalanusC***: *Clausocalanus* (classe de taille C) – herbivore,
- ***AdultsOfCentropages***: *Centropages tipicus* (adultes) – omnivore,
- ***JuvenilesOfCentropages***: *Centropages typicus* (juvéniles) – omnivore,
- ***Nauplii***: nauplies de copépodes – filtreur,
- ***Oithona***: *Oithona sp.* – carnivore,
- ***Acanthaires***: différentes espèces d'acanthaires – misc,
- ***Cladocerans***: différentes espèces de cladocères – carnivore,
- ***EchinodermsLarvae***: larves d'échinodermes – filtreur,
- ***DecapodsLarvae***: larves de décapodes – omnivore,
- ***GasteropodsLarvae***: larves de gastéropodes – filtreur,
- ***EggsOfCrustaceans***: oeufs de crustacés – misc,
- ***Ostracods***: différentes espèces d'ostracodes – omnivore,
- ***Pteropods***: différentes espèces de ptéropodes – herbivore,
- ***Siphonophores***: différentes espèces de siphonophores – carnivore,
- ***BellsOfCalycophores***: cloches de calycophores – misc.

Pour les caractéristiques physico-chimiques des masses d'eaux aux stations correspondantes, voir ***marphy*** (paragraphe suivant). Les données ***marbio***, autant que ***marphy*** seront utilisées pour illustrer plusieurs techniques, et notamment les méthodes d'identification de discontinuités dans les séries (correspondant aux différentes masses d'eaux rencontrées le long du transect, voir paragraphe suivant).

Références:

Boucher, J., F. Ibanez & L. Prieur (1987) Daily and seasonal variations in the spatial distribution of zooplankton populations in relation to the physical structure in the Ligurian Sea Front. *Journal of Marine Research*, 45:133-173.

Fromentin, J.-M., F. Ibanez & P. Legendre (1993) A phytosociological method for interpreting plankton data. *Marine Ecology Progress Series*, 93:285-306.

marphy*, caractéristiques des masses d'eaux à travers un front aux mêmes stations que *marbio

Le *data frame* ***marphy*** a 68 lignes (stations) et 4 colonnes. Il s'agit de mesures de l'eau de mer à une profondeur variant entre 3 et 7 m aux mêmes stations que le jeu de données ***marbio***. Les 4 mesures sont:

- ***Temperature***: la température de l'eau de mer, exprimée en °C,
- ***Salinity***: la salinité en g/kg,

- ***Fluorescence***: la fluorescence de la chlorophylle α (indication de la densité du phytoplancton),

- ***Density***: l'excès de masse volumique de l'eau de mer, en g/l (notez que cette mesure est "redondante" avec la salinité; elle a été mesurée de manière indépendante de cette dernière).

Les masses d'eaux sont identifiées comme suit (voir figures dans Boucher et al, 1987):

- stations 1 à 17: zone périphérique,

- stations 17 à 25: D1 (zone de divergence 1),

- stations 25 à 30: C (zone de convergence),

- stations 30 à 41: zone frontale,

- stations 41 à 46: D2 (zone de divergence 2),

- stations 46 à 68: zone centrale.

Référence:

Boucher, J., F. Ibanez & L. Prieur (1987) Daily and seasonal variations in the spatial distribution of zooplankton populations in relation to the physical structure in the Ligurian Sea Front. *Journal of Marine Research*, 45:133-173.

releve, six taxa phytoplanctoniques suivis au cours du temps à une seule station

Les séries ***releve*** sont de réelles séries temporelles, mais observées de manière assez irrégulière. Elles nous serviront donc à illustrer les méthodes de régularisation, mais aussi de décomposition de séries. Le *data frame* a 61 lignes (observations) pour 8 colonnes (le jour de l'observation, la date et six taxa phytoplanctoniques dénombrés).

- ***Day***: le numéro du jour, la première observation étant effectuée au jour 1,

- ***Date***: une chaine de caractère de type "dd/mm/yyyy" indiquant la date de l'observation (donc, au format "d/m/Y", voir la fonction daystoyears),

- ***Astegla***: l'abondance de *Asterionella glacialis*,

- ***Chae***: l'abondance de *Chaetoceros sp*.,

- ***Dity***: l'abondance de *Ditylum sp*.,

- ***Gymn***: l'abondance de *Gymnodinium sp*.,

- ***Melosul***: l'abondance de *Melosira sulcata* + *Paralia sulcata*,

- ***Navi***: l'abondance de *Navicula sp*.

Ces comptages ont été effectués dans le cadre du suivi du phytoplancton toxique par l'IFREMER (programme REPHY; petit échantillon seulement de l'ensemble des données, voir Belin & Raffin, 1998).

Référence:

Belin, C. & B. Raffin, 1998. Les espèces phytoplanctoniques toxiques et nuisibles sur le littoral français de 1984 à 1995, résultats du REPHY (réseau de surveillance du phytoplancton et des phycotoxines). Rapport IFREMER, RST.DEL/MP-AO-98-16. IFREMER, France.

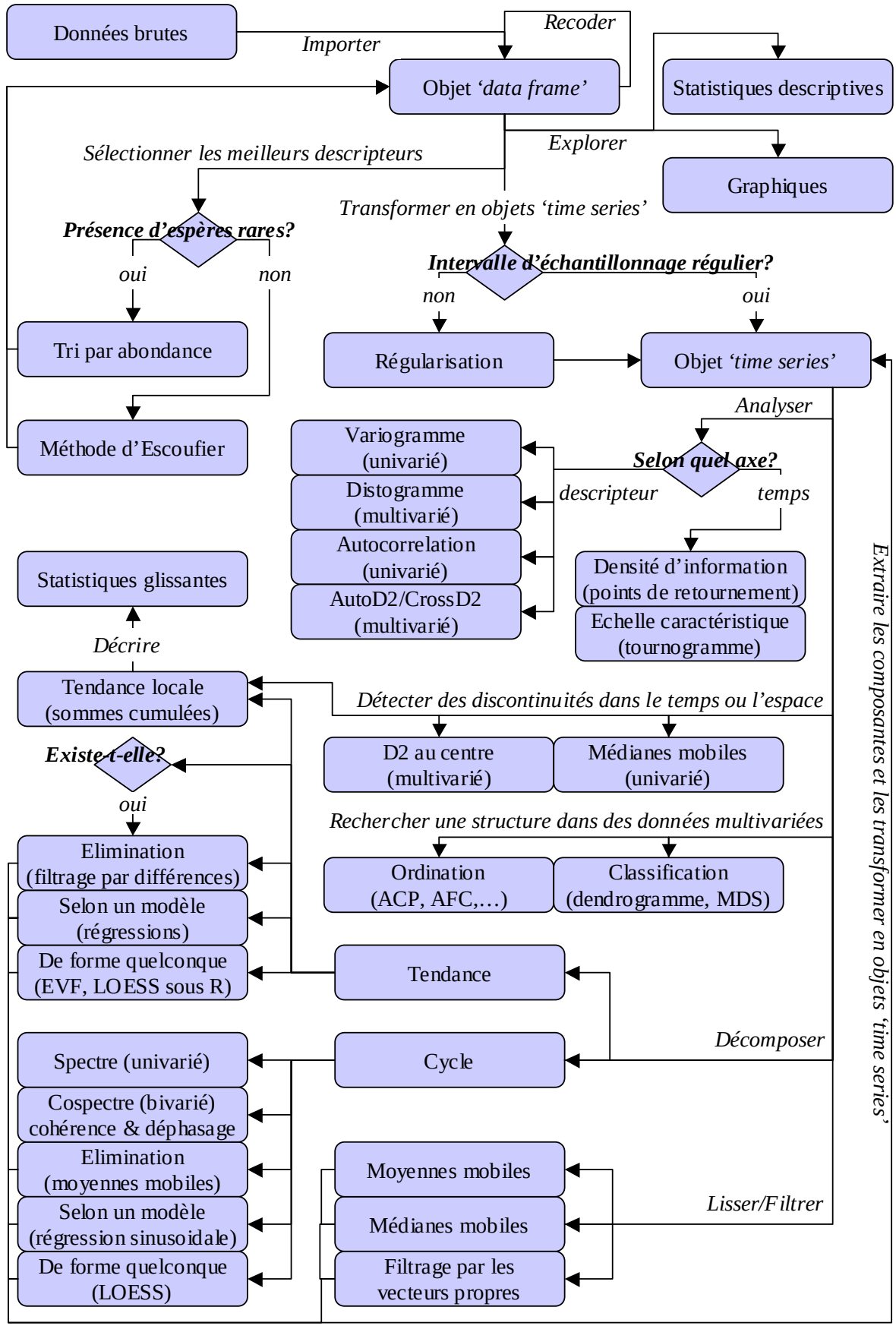
Diagramme décisionnel des méthodes d'analyse

Le diagramme à la page suivante devrait faciliter le choix des méthodes d'analyse de PASTECS en fonction des objectifs de votre étude.

En partant de **données brutes** encodées (nous considèrerons qu'il s'agit d'un format de fichier supporté par S+ ou R; si les données sont dans un format non supporté, il faudra préalablement les convertir en un tableau ASCII à partir du logiciel qui a servi à les générer), nous aborderons d'abord très brièvement l'**importation** des données pour nous focaliser sur le résultat de cette importation: un *'**data frame**'*. Nous verrons ainsi comment manipuler ce *data frame*, et comment éventuellement **recoder** les données dans S+ ou R. Nous évoquerons aussi les différentes possibilités du logiciel en matière de présentation **graphique** de ces mêmes données brutes, ainsi que les fonctions offertes tant part S+ ou R que par la librairie PASTECS pour obtenir des **statistiques descriptives**. Ces techniques (**importation**, **recodage** et **exploration des données**) constituent des outils de base à connaître avant d'aborder l'analyse des séries spatio-temporelles proprement dites.

La première question que l'on peut être amené à se poser face à un jeu de données brutes est de savoir si ***tous*** les descripteurs sont pertinents, ou bien si l'on peut en éliminer certains parce qu'ils sont redondants (par exemple, deux mesures qui se rapportent à peu de choses près au même phénomène physique ou chimique) ou parce qu'ils sont peu porteurs d'information (citons le cas des espèces rares dans des études d'abondance sur des matrices espèces x stations). La librairie PASTECS propose deux méthodes pour **sélectionner les descripteurs**: la méthode des **vecteurs équivalents d'Escoufier** qui va classer les descripteurs par ordre de leur contribution à la variance totale du jeu de données (au sens de l'ACP), et le **tri par abondance** qui va permettre de repérer et d'éliminer les espèces rares. Ces deux méthodes permettent d'extraire un *data frame* simplifié contenant moins de descripteurs, tout en conservant l'essentiel de l'information. Elles vont donc faciliter toutes les analyses ultérieures.

Toutes les méthodes présentées jusqu'ici travaillent sur des *data frames*, c'est-à-dire, des tableaux généraux, alors que la librairie PASTECS est spécialisée dans l'analyse des séries spatio-temporelles. L'étape suivante, avant de pouvoir utiliser la grosse partie des fonctions de PASTECS qui ont trait aux séries sera donc de **transformer** un ou plusieurs descripteurs du *data frame* en un objet *'time series'* (en fait, un objet *'rts'* sous S+ ou *'ts'* sous R, voir chapitre **manipulation de base des séries régulières**). Les séries spatio-temporelles manipulées par les fonction PASTECS doivent être ***régulières***. Cela signifie que le pas d'échantillonnage doit être constant et que les séries ne peuvent pas présenter de ***trous*** (i.e, de valeurs manquantes). Si l'intervalle d'échantillonnage dans le temps ou dans l'espace sont réguliers, la transformation d'un *data frame* en un objet *time series* est très facile. Sinon, PASTECS propose toute une panoplie de techniques pour **régulariser** des séries irrégulières ou à trous. Il est en effet assez rare d'obtenir des séries spatio-temporelles parfaitement régulières en écologie et les méthodes de régularisation permettent de palier aux aléas de l'échantillonnage sur le terrain en "reconstruisant" les valeurs manquantes.



Choix des méthodes d'analyse dans PASTECS en fonction des objectifs.

Une fois que l'on a une ou plusieurs séries spatio-temporelles chargées en mémoire dans des objets *time series*, différentes études sont envisageables: l'**analyse de la série**; la **détection de discontinuité dans le temps ou dans l'espace**, la **recherche d'une structure dans des données multivariées**, ou bien encore la **décomposition** ou le **lissage/filtrage** des séries.

Selon l'axe temporel, on peut déterminer quelles sont les parties de la série qui sont riches en information et quelles sont, au contraire, les endroits où la série se rapproche d'une variation aléatoire peu porteuse d'information. Dans ce but, nous analyserons les **points de retournement** (c'est-à-dire la succession des extremas, pics ou creux, qui apparaissent dans la série). Issu également de la théorie de l'information et de l'étude des points de retournement, le **tournogramme** fournira des informations relatives à l'**échelle caractéristique** du phénomène étudié, c'est-à-dire qu'il répondra à la question: "quel est l'interval optimal entre deux mesures dans le temps pour optimiser la quantité d'information portée par la série?"

Au niveau de l'**analyse des descripteurs**, S+ ou R proposent des fonctions classiques pour calculer et tracer le graphe de l'**autocorrelation**. En effet, une série est principalement caractérisée par le fait que les différentes observations ne sont pas indépendantes entre elles. Une observation réalisée au temps t+1 sera dépendante de celle faite au temps t, voir d'autres réalisées encore avant. Le graphe de l'autocorrelation est donc un outil de base qui sert à visualiser la dépendance plus ou moins forte entre les observations en fonction de l'écart temporel entre elles. De manière similaire, le **variogramme** étudie la variation du descripteur pour différents décalages temporels. PASTECS offre aussi des formes multivariées (plusieurs descripteurs étudiés simultanément) assimilables au graphe de l'autocorrelation et au variogramme qui sont tous deux utilisables uniquement pour des séries univariées: il s'agit respectivement du **distogramme** et de l'**AutoD2/CrossD2**. Ces méthodes sont nettement moins connues, mais néanmoins fort utiles à l'écologiste.

Une variante de l'analyse des séries basée sur le D2 est le **D2 au centre**. Dans le cas de cette analyse, on va pouvoir **détecter des discontinuités dans une série multivarié** (par exemple indiquant qu'un événement majeur mais ponctuel s'est produit dans le temps, ou marquant la frontière entre des zones de propriétés différentes dans l'espace). Avec des séries univariés, on pourra utiliser soit les **médianes mobiles**, soit la méthode des **sommes cumulées** pour détecter des transitions (dans le second cas, il s'agit d'un changement dans la valeur moyenne des observations dans la série, c'est-à-dire, un changement de pente dans la tendance générale considérée comme linéaire). Une fois que les discontinuités sont identifiées, on pourra décrire chaque période séparément à l'aide de **statistiques glissantes**.

Si l'on a affaire à des données multivariées, on peut aussi **rechercher une structure dans ces données** à l'aide de **méthodes d'ordination ou de classification** (analyse en composantes principales, analyse factorielle des correspondances, dendrogramme, cadrage multidimensionnel…). Toutes ces techniques, qui ne sont pas spécifiques aux séries, ont pour but de réduire les dimensions du jeu de données, d'en effectuer la synthèse ou d'en ressortir des regroupements d'entités semblables, et ce tout en limitant le plus possible la perte d'information que génèrent ces simplifications.

Une série peut aussi être le résultat d'une superposition de plusieurs effets; typiquement, une **tendance** générale à long terme se superpose à un ou plusieurs **cycles** (saisonnier, lunaire, circadien, ou autre) et à un bruit de fond constitué par une fluctuation aléatoire. On peut **décomposer** une série pour séparer ces différentes composantes qui pourront à leur tour être transformées en séries et être analysées, voire décomposées une nouvelle fois.

Nous avons déjà évoqué la méthode des **sommes cumulées** pour l'étude de la **tendance locale**. Un test est proposé dans la librairie PASTECS pour **déterminer s'il existe une tendance générale significative** croissante ou décroissante dans la série. Si cette

tendance existe, on peut l'**éliminer** par un **filtrage avec la méthode des différences**. Si on a une idée assez précise de l'allure que doit avoir la tendance générale recherchée (linéaire, exponentielle, sigmoidale,…), il est possible de l'estimer par **régression selon un modèle** donné. Si on a pas d'idée *a priori* de sa forme, on peut utiliser des méthodes qui vont extraire une tendance générale de forme quelconque, telle que le **filtrage par les vecteurs propres** (**EVF**) ou la **décomposition par LOESS** qui, sous R, permet aussi d'extraire une tendance générale en plus d'un cycle saisonnier. La version 1.2-0 de PASTECS permet aussi la **décomposition par CENSUS II.**

Les **cycles** sont étudiés par **analyse harmonique et spectrale**, une technique classique et déjà implémentée dans S+ ou R. La variation cyclique de deux séries l'une par rapport à l'autre (pour répondre aux questions: "est-ce que les deux séries présentent les mêmes variations cycliques –**cohérence**–, et est-ce que ces variations cycliques sont décalées l'une par rapport à l'autre dans le temps –**déphasage**–?") se fait grâce à l'étude du **cospectre** ou encore appelée **analyse spectrale croisée**. Une fois qu'un ou plusieurs cycles ont été identifiés, on peut les **éliminer** (par exemple par un lissage par **moyennes mobiles**), ou alors on peut en estimer la forme (soit selon un **modèle sinusoidal par régression**, soit sans *a priori* sur sa forme à l'aide de la **décomposition par LOESS**).

Finalement, on peut être intéressé par un **lissage** ou un **filtrage** d'une manière générale pour une ou plusieurs séries présentant de très fortes fluctuations erratiques parasites. Nous avons déjà évoqué plus haut les 3 techniques principales proposées par PASTECS pour atteindre ce but: le lissage par les **moyennes mobiles**, par les **médianes mobiles**, et le **filtrage par les vecteurs propres**.

Equivalence entre PASSTEC 2000 et PASTECS

Afin de faciliter la transition pour les habitués de PASSTEC 2000, les équivalences entre les entrées de menu de PASSTEC 2000 et les fonctions de PASTECS sont présentées dans le tableau II. Etant donné la façon dont les différentes routines sont implémentées dans chaque logiciel respectif, cette équivalence n'est pas toujours exacte. Certaines fonctions utilisent des algorithmes différents entre les programmes, et peuvent donc également renvoyer dans certains cas des résultats différents. Le lecteur est invité à consulter la description la plus à jour de chaque fonction directement dans l'aide en ligne respective de chaque logiciel.

Dans sa version 1.2-0, la librairie PASTECS implémente déjà quasiment toutes les fonctions de l'ancien PASSTEC 2000 qui n'existaient pas encore dans S+ ou dans R. Toutefois, diverses fonctions, seront probablement ajoutées dans des versions ultérieures. Vérifiez de temps en temps sur le site web (http://www.sciviews.org/pastecs) s'il n'apparaît pas une nouvelle version qui offrirait des possibilités supplémentaires à celle que vous possédez actuellement.

Tableau II: *équivalences PASSTEC 2000 / PASTECS. (U) signifie une entrée de menu en mode univarié seulement dans PASSTEC 2000, (M) signifie une entrée de menu en mode multivarié seulement. NA signifie que la fonction est sans objet dans PASTECS et (+) signifie que la fonction est décrite dans ce document.*

Entrées de menu PASSTEC 2000	**Fonctions equivalentes dans PASTECS (ou langage S)**
Menu Fichier	
Nouveau…	Data <- c(….) ou Data <- matrix(…)(+)
Ouvrir Fichier Standard…	read.table()(+) ou menu ***File -> Import Data -> From File*** (S+)
Ouvrir Fichier Excel…	menu ***File -> Import Data -> From File*** (S+)
Importer	read.table()(+) ou menu ***File -> Import Data -> From File*** (S+)
Enregistrer / Enregistrer sous	write.table() ou menu ***File -> Workspace -> Save…*** (S+)
Cacher	NA
Quitter	q()
Menu Afficher	
Barres d'outils	NA
Afficher Graphiques	NA
Afficher Tableur	NA
Afficher PressePapier	NA
Menu Edition	
Centrer sur la cellule courante	NA
Vue sur…	NA
Sélection > Effacer la zone	Data[…] <- NA
Sélection > Colonne courante	Data[colnbr,]
Sélection > Observation courante	Data[colnbr, rownbr]
Sélection > Toute une zone	Data[i:j,k:l](+)
Sélection > Toute la matrice	Data
Options > Renommer la feuille	Data2 <- Data1; remove("Data1")
Options > Format > Centré	NA
Options > Format > à Gauche	NA
Options > Format > à Droite	NA
Menu Description	
Dates en entiers (U)	daystoyears()(+), yearstodays()(+)
Découpage en classes (U)	cut()(+), disjoin()(+)
Estimateurs de Pennington (U)	stat.pen()(+), pennington()(+)
Méthode d'Escoufier	escouf()(+), print()(+), summary()(+), plot()(+)
Points de retournement (U)	turnpoints()(+), print()(+), summary()(+), plot()(+), lines()(+)
Statistiques (U)	stat.desc()(+)
Statistiques glissantes (U)	stat.slide()(+), print()(+), plot()(+)
Tournogramme (U)	turnogram()(+), print()(+), summary()(+), plot()(+)
Variogramme (U)	vario()(+)
Distogramme (M)	disto()(+)
Graphe (U)	plot()(+)
Menu Tranformation	
Centrage	scale()
Renommer	names(Data) <- newnames
Opérations > Addition	Data3[colnbr3,] <- Data1[colnbr1,] + Data2[colnbr2,]
Opérations > Soustraction	Data3[colnbr3,] <- Data1[colnbr1,] - Data2[colnbr2,]

Opérations > Multiplication	Data3[colnbr3,] <- Data1[colnbr1,] * Data2[colnbr2,]
Opérations > Division	Data3[colnbr3,] <- Data1[colnbr1,] / Data2[colnbr2,]
Opérations > Pourcentage	Data2[colnbr2,] <- Data1[colnbr1,] / 100
Opérations > Cumul (Sommation)	Data2[colnbr2,] <- cumsum(Data1[colnbr1,])
Opérations > Exponentielle	Data2[colnbr2,] <- exp(Data1[colnbr1,])
Opérations > Logarithme népérien	Data2[colnbr2,] <- log(Data1[colnbr1,] + 1)(+)
Valeurs aléatoires	rnorm()(+)
Valeurs manquantes	Data[Data == codeValManquante] <- NA(+)
Réduction	scale()
Menu Régulation	
Méthode des aires (U)	regul()(+), regarea()(+), summary()(+), plot()(+), tseries()(+)
Splines (U)	regul()(+), regspline()(+), summary()(+), plot()(+), tseries()(+)
Méthode ZET (M)	Pas encore implémenté
Menu Filtrage (U)	
Buys Ballot + Census (U)	buysbal()(+), tsd()(+), plot.tsd()(+), deccensus()(+)
Census (U)	tsd()(+),deccensus()(+) (R)
Elimin. de tendance ou de cycle (U)	tsd()(+),plot.tsd()(+), decdiff()(+), decaverage()(+), decloess()(+)
Filtrage par les vecteurs propres (U)	tsd()(+),plot.tsd()(+), decevf()(+)
Menu Tendance (U)	
Test d'existence (U)	trend.test()(+)
Estimation statistique (U)	tsd()(+),plot.tsd()(+), decaverage()(+), decmedian()(+), decreg()(+)
Tendance locale (U)	local.trend()(+), identify()(+)
Menu Cycles	
Analyse harmonique (U)	spectrum()(+)
Analyse spectrale (U)	spectrum()(+)
Period. Conting. (U)	Pas encore implémenté
Méthode D2 (M)	AutoD2()(+), CrossD2()(+)
D2 au Centre (M)	CenterD2()(+)
Analyse spectrale croisée (M)	spectrum()(+), plot()(+)
Menu Analyses (M)	
Analyse des Compos. Princ. (M)	princomp()(+), summary()(+)
Analyse Factor. des Corresp. (M)	corresp(), ca()
ACP biplot (M)	biplot()(+)
Cadrage multidimensionnel (M)	isoMDS(), sammon(), cmdscale()
Dendrogramme (M)	dist()(+), hclust()(+)
Fidélités Cumulées (M)	Pas disponible (?)
Méthodes d'Ordin. Canoniques (M)	Voir librairie CoCoAn sous R
Méthode du quatrième coin (M)	Pas disponible (?), mais quasi-immédiat en langage matriciel
Menu Mode	
Univarié	NA
Multivarié	NA
Menu Aide	
Aide	help(…)
A propos de PASSTEC	Menu ***Help -> About S-plus***

Description des fonctions de PASTECS

> ***Attention:*** *les sorties produites dans les exemples sont issues de l'environnement R. Elles ont aussi été testées sous S+ 2000 release 3 sous Windows et doivent fonctionner dans cet environnement, sauf lorsque cela est stipulé. Toutefois, le lecteur doit s'attendre à quelques différences de présentation, entre autres: le format d'affichage des nombres (notation scientifique ou non), les couleurs par défaut pour les traits, ainsi que le style de traits (plein, pointillé,…) et les polices choisies par défaut dans les graphiques. Plutôt que d'homogénéiser le tout dans PASTECS, nous avons préféré garder les valeurs par défaut dans chaque environnements respectif afin de conserver une certaine homogénéité au sein d'un même programme. Toutefois, toutes ces options sont modifiables par l'utilisateur (voir notamment* `?par` *pour les graphiques). Ces exemples doivent aussi fonctionner sur d'autres systèmes d'exploitation que Windows (MacOS, Linux, Unix), bien que n'ayant pas été testés de manière approfondie.*

Fonctions de base du langage S et routines diverses PASTECS

Une connaissance de base du langage S est indispensable pour comprendre et utiliser les différentes fonctions présentes dans la librairie PASTECS. Comme précisé dans l'introduction, nous conseillons vivement aux utilisateurs qui débutent sous cet environnement de se familiariser avec son langage par l'intermédiaire des documents fournis avec les logiciels respectifs. S+ propose un « Getting started guide » qui présente essentiellement l'interface utilisateur du produit. Dans le « S+ user's guide », au chapitre 9 « Command window », l'utilisateur trouvera une introduction au langage S lui-même (à noter que seule la version professionnelle de S+, pas la version standard, donne accès à la fenêtre de commande indispensable à la manipulation des routines PASTECS). De même, R propose « An introduction to R » accessible à partir du menu d'aide.

Pour ceux qui préfèrent une introduction en français, le document de 16 pages « R pour débutants » disponible en ligne sur le site http://cran.r-project.org (section documentation – contributed) est un excellent point de départ. Ceux qui désirent apprendre le langage sur base d'exemples pratiques pourront se tourner vers l'excellent livre de Venables & Ripley (2002).

> ***Attention:*** *après avoir installé la librairie PASTECS, il faut la charger à l'aide de la fonction* ***library()*** *avant de pouvoir utiliser ses fonctions. Donc, avant chaque session utilisant les fonctions de PASTECS, il faudra prendre soin de rentrer la commande suivante, sinon le programme renvoie un message d'erreur indiquant que les fonctions sont inconnues:*

```
> library(pastecs)
```

Obtenir de l'aide sur une fonction

L'aide en ligne est un excellent outil pour apprendre à utiliser une fonction ou pour se rafraîchir la mémoire concernant sa synthaxe. Connaissant le nom d'une fonction, il suffit de rentrer `help(ma.function)` ou encore `?ma.function`, pour afficher l'aide en ligne relative à la fonction ***ma.function()***. Par exemple:

```
> ?AutoD2
```

N'hésitez pas à user et à abuser de ***help()***, ou ***?*** en abbrégé. Si par contre, vous ne savez pas quelles fonctions sont disponibles dans la librairie PASTECS, il suffit alors de consulter le fichier 'INDEX.txt' sous S+ ou 'INDEX' sous R, dans le répertoire

d'installation de la librairie (généralement: <program.dir>\library\pastecs, où <program.dir> est le programme d'installation du logiciel respectif S+ ou R. Dans R, on peut également activer l'aide au format html (par exemple, menu ***Help -> R language (html)*** dans 'Rgui' sous Windows), ensuite cliquer sur le lien '**Packages**' et sélectionner l'entrée '**pastecs**' pour obtenir la liste exhaustive de toutes les fonctions disponibles dans la librairie. La fonction ***args()*** est également très utile pour se rappeler de la synthaxe d'une fonction sans nécessairement activer toute l'aide en ligne:

```
> args(AutoD2)
function (series, lags = c(1, nrow(series)/3), step = 1,
    plotit = TRUE, add = FALSE, ...)
NULL
```

Importation de données dans un 'data frame'

L'une des premières questions que se pose l'utilisateur novice la plupart du temps est: « comment puis-je importer mes données dans S+ ou R? ». S+ 2000 sous Windows offre de nombreux formats d'importation des données via le menu ***File -> Import Data -> From File…*** Il est très facile d'éditer des fichiers de données au format Excel ou PASSTEC 2000 par exemple pour permettre ensuite leur importation dans S+. R n'offre pas les mêmes facilités, mais une commande en ligne utilisant la fonction ***read.table()*** permet de récupérer des données à partir d'un fichier texte. Si le fichier est de type texte ASCII, avec les données séparées par des tabulations (imaginons qu'un tel fichier se nomme ***c:\temp\mydata.txt***), la commande suivante importera le dit fichier dans la variable ***dat***:

```
> dat <- read.table("c:/temp/mydata.txt", header=TRUE, sep="\t",
+ na.strings="NA")
```

Notez que la chaîne de caractères indiquant le chemin d'accès utilise le slash (/), et non le back-slash (\) comme séparateur (on peut aussi utiliser le double back-slash \\ à la place). Il s'agit en fait de la présentation Unix qui est également utilisée sous S+ ou R dans l'environnement Windows! Si un entête est présent, utilisez **header = TRUE**, dans le cas contraire, précisez **FALSE**. Le séparateur de données étant une tabulation, utilisez **sep = "\t"**. Selon la nature du séparateur décimal dans le fichier, utilisez soit **dec = "."** ou **dec = ","**. Enfin, si le fichier contient des données manquantes, précisez **na.strings = "<code>"** où code est la chaîne de caractères utilisée pour représenter les données manquantes (NA, -99, ., etc.).

Pour certains fichiers PASSTEC 2000 importés, on aura deux colonnes supplémentaires indiquant respectivement le temps selon une échelle décimale dont l'unité est le jour (voir chapitre régularisation pour plus de détails) et les dates correspondantes au format texte (par exemple "23/04/1998") ou au format date selon la façon dont on a importé les données. Il est parfois utile d'éliminer ces deux colonnes pour ne conserver que les données proprement dites. Cela se fait facilement à l'aide de: `var2 <- var[,3:n]`, où **n** est le nombre de colonnes contenues dans ***var***. D'autre part, la fonction ***daystoyears()*** modifie l'échelle de temps de la colonne temps pour avoir l'année comme unité (voir chapitre régularisation). Enfin, la fonction ***yearstodays()*** a l'effet inverse.

Dans le cas particulier des jeux de données inclus dans des libraries (comme ***marbio***, ***marphy***, ***releve*** et ***bnr*** de la librairie PASTECS), ils sont disponibles immédiatement dans S+. Dans R, leur chargement dans l'espace de travail doit se faire grâce à la fonction ***data()*** avant de pouvoir les utiliser:

```
> data(marbio)   # Dans R uniquement
```

> *A noter que tout ce qui suit le caractère # sur une ligne de commande est un commentaire et n'est pas interprété par le programme.*

On peut vérifier que les entêtes, puis les données (affichage des 5 premières lignes et 5 premières colonnes) ont été lues correctement (c'est naturellement valable aussi et surtout pour vos données personnelles chargées à partir d'un fichier externe) grâce aux instructions suivantes:

```
> names(marbio)
 [1] "Acartia"               "AdultsOfCalanus"
 [3] "Copepodits1"           "Copepodits2"
 [5] "Copepodits3"           "Copepodits4"
 [7] "Copepodits5"           "ClausocalanusA"
 [9] "ClausocalanusB"        "ClausocalanusC"
[11] "AdultsOfCentropages"   "JuvenilesOfCentropages"
[13] "Nauplii"               "Oithona"
[15] "Acanthaires"           "Cladocerans"
[17] "EchinodermsLarvae"     "DecapodsLarvae"
[19] "GasteropodsLarvae"     "EggsOfCrustaceans"
[21] "Ostracods"             "Pteropods"
[23] "Siphonophores"         "BellsOfCalycophores"

> marbio[1:5,1:5]
  Acartia AdultsOfCalanus Copepodits1 Copepodits2 Copepodits3
1      32               0           0           0           1
2      99               0           0           0           4
3     124               0           0           0           9
4      82               0           0          12          12
5      48               0           0           0          12
```

Un fichier contenant un tableau est lu dans une variable de type *'data frame'*, tel que ***marbio***. Un *data frame* est un tableau bidimensionnel représentant les variables (en colonne) *versus* les observations (en ligne). Un *data frame* diffère d'une simple matrice bidimensionnelle en ce que chaque colonne peut être d'un type différent. Ainsi, une matrice peut contenir des nombres, du texte, etc. Mais dans une matrice numérique, **toutes** les colonnes doivent être numériques. De même dans une matrice textuelle, toutes les colonnes doivent contenir des chaines de caractères. Un *data frame* est plus souple et permet de mixer des colonnes de type différent, par exemple: du texte (le nom de la station mesurée, l'auteur de la mesure, un commentaire,…), des nombres (les mesures quantitatives), des « facteurs » (type *'factor'* pour les mesures semi-quantitatives ou qualitatives tels que grand/moyen/petit ou bleu/vert/rouge,…), des valeurs logiques (type *'boolean'*: vrai/faux ou oui/non). Le data frame est donc particulièrement adapté au stockage de données hétérogènes telles qu'on les rencontre le plus souvent en statistiques. Naturellement, il ne convient pas forcément pour le stockage de tous les types de données. Le language S offre également d'autres classes d'objets pour stocker d'autres types de données. Nous verrons plus loin en particulier les classes *'ts'* et *'rts'* (regular time series) très utiles pour stocker des séries spatio-temporelles régulières telles que celles qui sont couramment manipulées à l'aide de la librairie PASTECS.

Gestion des variables, des environnements de travail et des scripts

Autres fonctions de base pour la manipulation des variables en langage S, ***objects()*** et ***remove()***. La première fonction liste les objets chargés en mémoire, et la seconde en élimine. Veuillez à utiliser régulièrement ces fonctions pour éviter d'accumuler des données qui ne sont plus utilisées en mémoire:

```
> new.data <- 1
> objects()
 [1] "marbio"     "new.data"
> remove("new.data")
> objects()
 [1] "marbio"
```

Notez que d'autres objets peuvent être listés en plus selon l'état du système. L'explorateur d'objets de S+ 2000 offre également un accès graphique aux variables stockées en mémoire, mais il est possible de s'en passer complètement et de gérer les variables intégralement à l'aide du langage S.

A propos des objets que vous manipulez dans une *session*, le programme vous donne l'occasion de les sauvegarder à la fermeture du programme, pour pouvoir les retrouver à la prochaine session lorsque vous allez ouvrir à nouveau le logiciel. S+ et R diffèrent sensiblement à ce niveau. Sous S+, lorsque vous quittez le programme (soit à l'aide du menu ***File -> Exit***, soit via la commande `q()`), celui-ci vous propose une liste des variables nouvellement créées ou modifiées lors de la dernière *session*. Vous pouvez sélectionner dans cette liste les variables que vous désirez conserver. Notez que si vous entrez `q("no")`, le programme ne vous demandera rien et ne sauvera aucune de vos modfications. Ceci est utile si vous avez juste exploré certains aspects de S+, mais sans y effectuer un travail que vous voulez poursuivre par après.

Dans R, vous utiliserez aussi ***File -> Exit*** (versions Windows ou Macintosh seulement), `q()` ou `q("no")`. Dans les deux premiers cas, le programme vous demandera si vous voulez sauver une « *image* » de l'environnement de travail (save workspace image?). A ce niveau, vous n'avez pas la possibilité de sélectionner les variables à sauvegarder et celles à ignorer, ce qui signifie que vous devez faire le nettoyage à l'aide de la fonction ***remove()*** *avant* de quitter le programme! Si vous répondez oui à la sauvegarde de l'image, deux fichiers seront créés **dans le répertoire courant**: ***.Rdata*** et ***.Rhistory***. Le premier fichier contient les données proprement dites, le second contient l'historique des instructions que vous avez entrées à la ligne de commande. Le fait que ces fichiers soient sauvegardés dans le répertoire courant permet de conserver différentes sessions indépendantes en attribuant un répertoire différent à chacune d'elles. La fonction ***getwd()*** permet de savoir quel est le répertoire de travail courant, et la fonction ***setwd(dir)*** permet de le changer (n'oubliez pas de présenter le répertoire à la mode Unix, par exemple pour **c:\temp**, vous écrirez soit **"c:/temp"**, soit **"c:\\temp"**!). Pour ouvrir R dans une session particulière, vous devez changer le répertoire courant *avant* de lancer R. Sur les systèmes Unix/Linux, vous le changez dans le shell (***cd …***). Dans Windows, vous pouvez créer un raccourci de R (dans l'explorateur de fichiers, cliquez avec le bouton droit sur **Rgui.exe** qui se trouve dans **<R dir>\bin** et sélectionnez **créer un raccourci** dans le menu contextuel). Ensuite vous éditez ce raccourci (clic bouton droit sur le raccourci, puis sélection de **propriétés**) et vous modifiez le champ **démarrer dans** pour qu'il contienne le répertoire dans lequel vous avez souvegardé votre session. Vous pouvez maintenant renommer le raccourci (par exemple en l'appelant *"R pastecs"*) et le déplacer (posez-le sur la fenêtre du bureau, par exemple, pour pouvoir y accéder plus rapidement). Pour en savoir plus sur les fonctions de sauvegarde et de chargement d'environnements de travail sous R, tapez `?save.image` et `?load`.

> *Au fur et à mesure que vous apprendrez le langage S, vous vous rendrez compte qu'il devient indispensable de sauvegarder vos différentes commandes pour pouvoir les réutiliser. Vous pouvez les copier et les éditer dans une **fenêtre d'édition** dans Splus et dans R pour le Macintosh. R sous Unix/Linux et sous Windows n'offre pas d'éditeur intégré au programme, mais vous pouvez utiliser tout éditeur de texte classique. Certains logiciels sont même spécialisés dans l'édition de code (par exemple **Textpad** sous Windows, disponible à partir de http://www.textpad.com, et qui propose un fichier de définition du langage S pour obtenir la coloration synthaxique du code S, voir aussi http://www.r-project.org/GUI). Vous ferez un copier-coller entre votre fenêtre d'édition et la ligne de commande, ou encore, vous pouvez utiliser la fonction **source()** pour lire tout un fichier de commande en une seule fois (voir l'aide en ligne de cette fonction). Notez que vous trouverez un fichier script qui contient tous les exemples de ce manuel sur le site de PASTECS (http://www.sciviews.org/pastecs). Ainsi, en l'ouvrant dans un éditeur de texte et en copiant les portions de code qui vous intéressent dans la fenêtre de commande, vous pourrez recalculer tous les exemples sans devoir en retaper le code.*

Créer ses propres fonctions en langage S

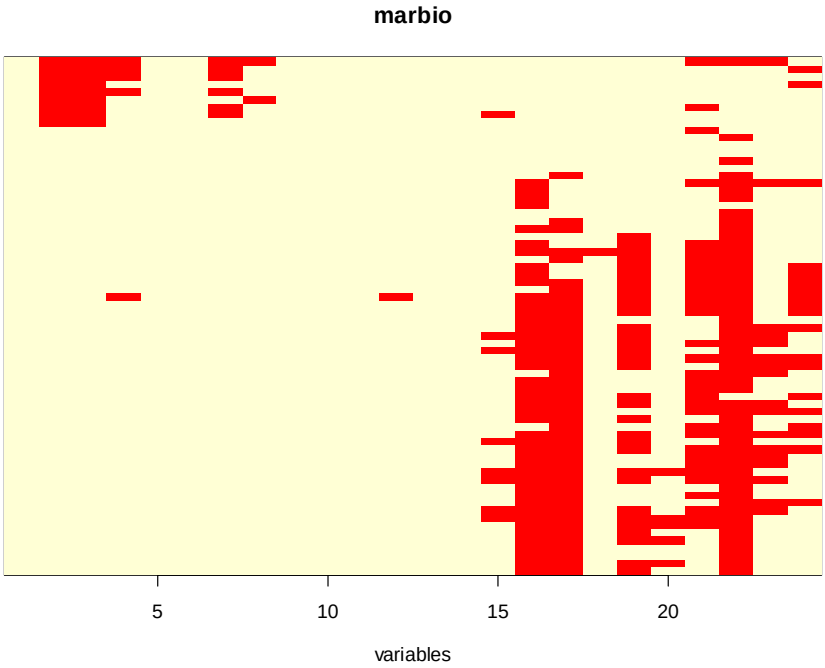
Certaines fonctions du langage S ou de PASTECS sont prévues pour explorer un data frame. Mais lorsque la fonction n'existe pas, il est très facile de la créer dans le langage S. Par exemple, si nous voulons une fonction pour mettre en évidence graphiquement quelles sont les données qui sont supérieures ou inférieures à un seuil donné dans une matrice ou un data frame, on peut créer une fonction, que l'on appelera par exemple ***levelmap()***:

```
> levelmap <- function(x, level=0){
+ data <- deparse(substitute(x))
+ x <- as.matrix(x)
+ image(1:ncol(x), 1:nrow(x), t(x > level)[,nrow(x):1], yaxt="n",
+ xlab="variables", ylab="", main=data) }
```

Comme la définition de notre fonction est longue, nous voyons ici tout l'intérêt de pouvoir la splitter sur plusieurs lignes! Le prompt '+', nous rappelle que le programme attend la suite de la commade. Il n'est pas indispensable que l'utilisateur novice comprenne le code que nous venons d'entrer. Il lui suffit simplement de retenir qu'il peut créer lui-même ses propres fonctions. Nous venons donc de créer notre propre fonction ***levelmap()*** dont la synthaxe complète est `levelmap(x, level=0)`. Ce qui se trouve entre parenthèses représente les *arguments* de la fonction, c'est-à-dire, les variables et les paramètres nécessaires au calcul de la fonction. Ces arguments sont nommés (**x** et **level**) et séparés par une virgule. Les arguments peuvent recevoir une valeur par défaut, comme ici, **level=0** qui indique que la valeur par défaut de **level** est zéro. Ces arguments qui reçoivent une valeur par défaut sont dits *facultatifs*, dans le sens qu'il n'est pas nécessaire de les repréciser à chaque appel de la fonction. S'ils ne sont pas précisés, la valeur par défaut est utilisée, soit zéro dans le cas présent. Ainsi, seul l'argument **x** doit être précisé à lappel de cette fonction, par exemple: `levelmap(x=marbio)`. Toutefois, il n'est pas indispensable de taper **arg=** si la valeur de l'argument est fournie à la même position que sa définition dans la fonction. Ainsi, comme l'argument **x** est le premier, la commande abbrégée `levelmap(marbio)`est strictement équivalente à la précédente, et le programme comprend **x=marbio** en lieu et place de **marbio** dans la parenthèse de la fonction. Nous renvoyons le lecteur aux différentes introductions du langage S pour la compréhension et l'utilisation des arguments dans les fonctions.

Nous ne nous étendrons pas plus sur le sujet ici. Revenons à notre fonction ***levelmap()***, et appliquons-là donc au jeu de données ***marbio***:

```
> levelmap(marbio)
```

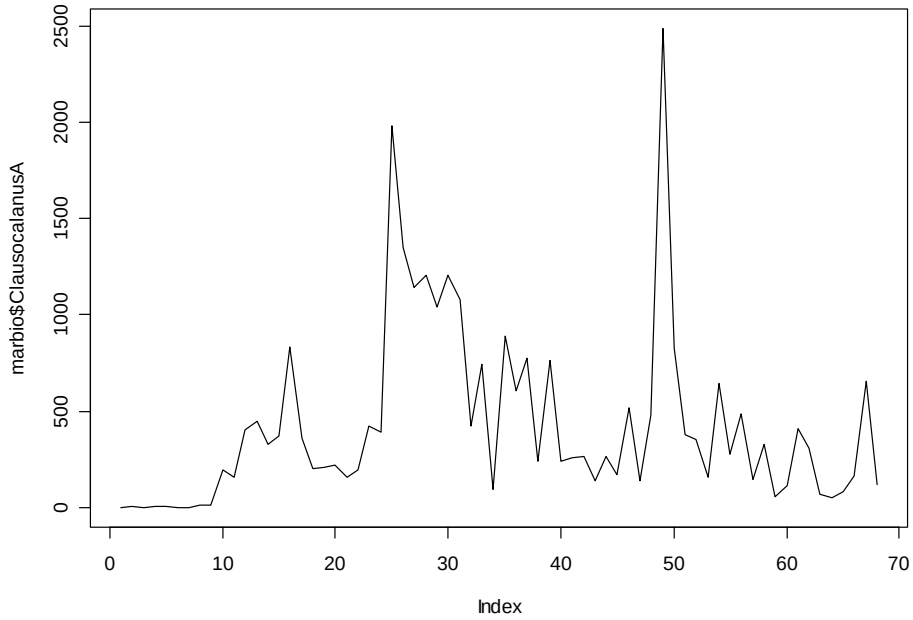


Les valeurs nulles ou négatives (puisque le seuil **level=0**) apparaissent en rouge; les valeurs positives apparaissent en jaune pâle. On constate tout de suite que 7 descripteurs sur 24 présentent beaucoup de zéros.

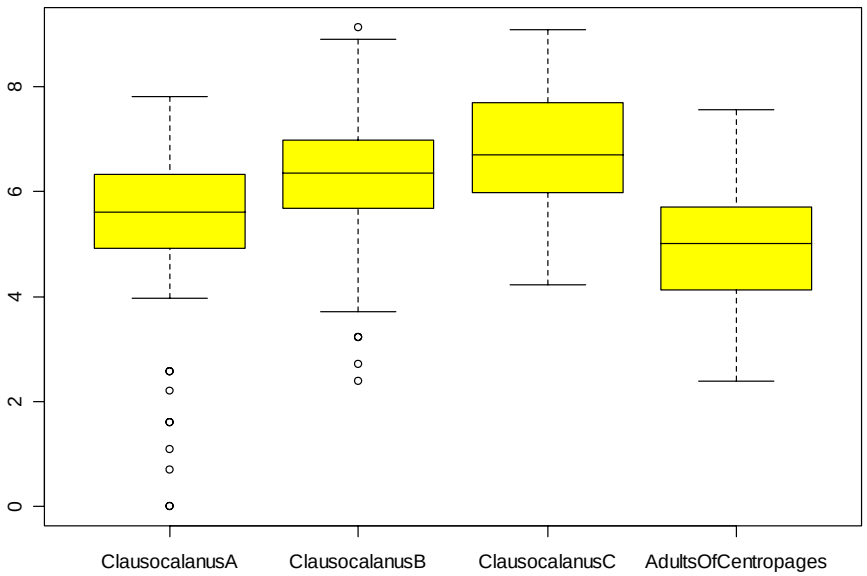
Exploration et manipulation des données d'un 'data frame'

Voyons ce que l'on peut faire d'autre avec le langage S en matière d'exploration de données. Pour un ou plusieurs descripteurs, on peut obtenir diverses représentations graphiques classiques. Tracer le graphique d'une des variables présente dans le data frame ***marbio***, par exemple ***ClausocalanusA*** (ce qui se note ***marbio$ClausocalanusA***), se fait grâce à la fonction ***plot()*** (l'argument **type="l"** indique que l'on désire relier les points par un ligne brisée; voir `?plot`):

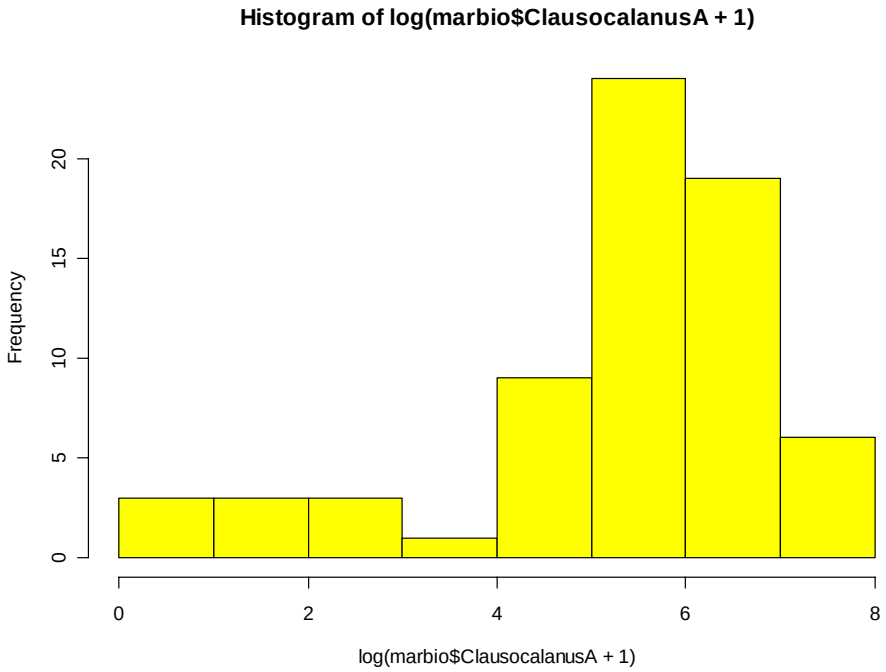
```
> plot(marbio$ClausocalanusA, type="l")
```



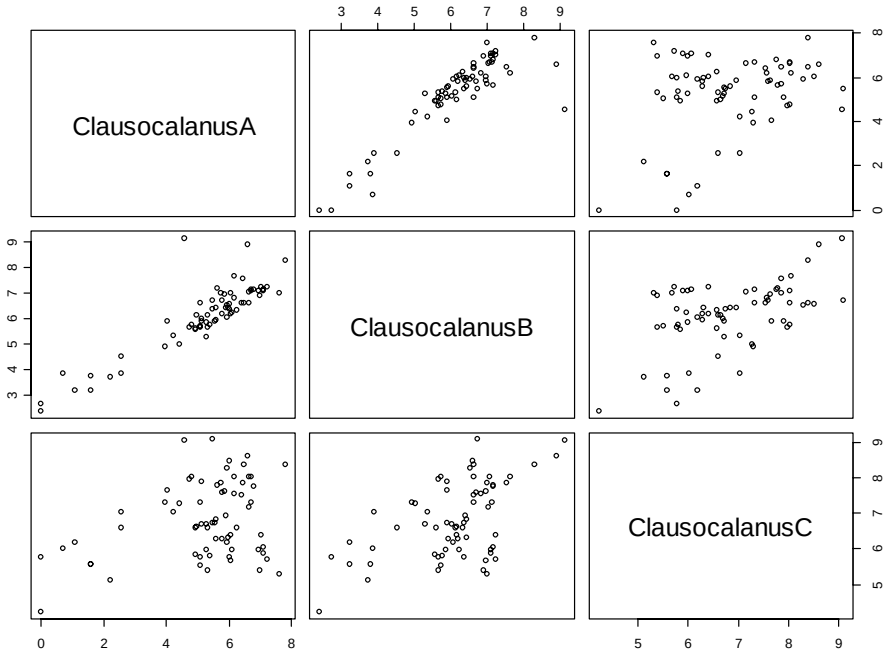

```
> boxplot(as.data.frame(log(marbio[8:11]+1)), col=7)
```



```
> hist(log(marbio$ClausocalanusA+1), col=7)
```




```
> pairs(log(marbio[8:10]+1))
```



Comme on le voit, de nombreuses fonctions permettent de représenter graphiquement des séries de données uni- ou multivariées. Outre le passage aux logarithmes népériens à l'aide de ***log()*** que nous avons plusieurs fois utilisé (à noter que dans R, la fonction ***log1p()*** effectue directement $\ln(x + 1)$, avec une plus grande précision également pour $|x| << 1$), d'autres fonctions mathématiques de base sont utilisables en langage S (***exp()***, ***cos()***, ***sin()***,…). Elles sont considérées comme connues (nous renvoyons à l'aide en ligne de S+ ou de R pour plus d'informations). Nous pouvons très facilement remplacer des données qui corresponde à un critère précis par une commande du type `data[data ==`

`/ > / < / != valeur] <- nouvelle.valeur`. En particulier, si le fichier importé utilise un code spécial pour représenter les valeurs manquantes (-99 par exemple), nous pouvons très facilement remplacer ce code par **NA**, valeur particulière qui représente les valeurs manquantes de manière plus explicite dans S+/R:

```
> marbio2 <- as.matrix(marbio)  # Requis dans R!
> marbio2[marbio2 == -99] <- NA  # Remplace tous les –99 par NA
```

Remarquons que, dans le cas précédent, nous avons été obligé de transformer le data *frame* ***marbio*** en une matrice à l'aide de la fonction ***as.matrix()***. Ceci ne pose pas de problèmes particuliers dans le cas présent parce que ***marbio*** ne contient que des données numériques. Ce n'est toutefois pas possible si le data frame contient également du texte. Il faut alors retirer les colonnes textuelles avant le traitement. Eventuellement, on désire recoder les données en leur appliquant une transformation: +, -, *, /, ^ (exposant), ***sqrt()*** (racine carrée), ***log()*** (logarithme népérien), ***log10()***, logarithme en base 10, ***exp()*** (exponentielle), ***cumsum()*** (sommes cumulées), ***rank()*** (calcul de rang), etc… Ces fonctions travaillent en général élément par élément. Ainsi:

```
> X <- matrix(c(1,2,3,4), nrow=2, ncol=2)
> X
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> Y <- matrix(c(5,6,7,8), nrow=2, ncol=2)
> Y
     [,1] [,2]
[1,]    5    7
[2,]    6    8

> X*Y                          # Multiplication élément par élément
     [,1] [,2]
[1,]    5   21
[2,]   12   32
```

est la multiplication de chaque élément X_{ij} de la matrice X par chaque élément Y_{ij} correspondant de la matrice Y. Cela ne correspond donc **pas** à une multiplication matricielle de X par Y, comme c'est le cas notamment sous Matlab (l'équivalent Matlab de ce qui est réalisé ici est `X .* Y`, `X * Y` exprimant bien la multiplication matricielle sous Matlab alors que sous S+/R, il faudra utiliser `X %*% Y`):

```
> X %*% Y                      # Multiplication matricielle
     [,1] [,2]
[1,]   23   31
[2,]   34   46
```

On peut très facilement centrer des données en soustrayant la moyenne de ces données à chaque élément. De même, on peut aisément obtenir des données standardisées (i.e., centrées-réduites, donc de moyenne nulle et d'écart type unitaire) en divisant le résultat par l'écart type de ces données. La fonction ***scale()*** facilite le travail:

```
marbio.cen <- scale(marbio, center=TRUE, scale=TRUE)
```

L'argument **center** indique s'il faut centrer les données (moyenne nulle) et l'argument **scale** indique s'il faut les réduire (écart type égal à un). Pour centrer seulement les observations de la colonne 1 de **marbio2** (on travaille ici sur une copie pour ne pas altérer directement les données originales) et les replacer dans cette même matrice (en écrasant les données de départ) on utilisera:

```
> marbio2 <- marbio                          # Copie de marbio
> marbio2[1,] <- scale(marbio2[1,], center=TRUE, scale=FALSE)
```

On peut éventuellement désirer recoder les données en plusieurs classes. La fonction ***cut()*** effectue cette opération. On peut préciser soit **breaks = n** où n est le nombre de classes équidistantes que l'on souhaite, soit fournir un vecteur indiquant les limites des classes (dans l'exemple, **breaks = 0:5** indique de faire des classes entre les valeurs 0 et 1, 1 et 2, …, 4 et 5). L'argument **labels** indique ce que l'on veut utiliser comme chaînes de caractères pour représenter les différentes classes (il faut $n - 1$ chaînes). Dans le cas présent, on nomme les classes simplement par 1, 2, 3,…, 5. Enfin, la fonction ***table()*** permet de représenter les résultats sous forme de tableau de contingence, mais comme ici, on n'y a qu'un seul facteur, il s'agit d'un tableau à une seule entrée (à noter que ***rnorm()*** crée des données pseudo-aléatoires à distribution normale et ***abs()*** en prend la valeur absolue):

```
> Z <- abs(rnorm(10000))
> table(cut(Z, breaks=0:5, labels=1:5))
   1    2    3    4    5
6833 2739  402   25    1
```

Cela nous amène à discuter des différentes façons de coder les données en classes.

Le codage en classes et le tableau de Buys-Ballot

Fréquemment, on considère les valeurs des descripteurs comme des fréquences observées dans des classes. Le graphique correspondant est un histogramme. De multiples applications nécessitent cette transformation, citons les tests de normalité, la construction de tableaux de contingence, la plupart des techniques relevant de la théorie de l'information, l'analyse des correspondances, les tests non-paramétriques...

Cependant soit les fréquences (ou les probabilités) sont utilisées dans les calculs statistiques, soit seul le découpage est utilisé afin de définir une succession d'états pour le descripteur. Dans ce cas là on cherche à coder une variable comme on le ferait pour des variables nominales.

Plusieurs possibilités sont offertes pour un découpage en classes:

1. Traiter soit les valeurs arithmétiques estimées par l'échantillonnage, soit une transformation afin de réduire la variance (logarithmes).

2. Fixer ou non relativement arbitrairement le nombre de classes. Dans le premier cas deux possibiltés sont à envisager:

- soit *p* classes d'amplitudes égales,

- soit *p* classes de fréquences égales (distribution uniforme). En fonction du pourcentage de zéros, on a les alternatives suivantes:

* considérer une classe particulière, identifiée pour les 0, puis continuer la partition en partant du minimum. Dans ce cas on devra fixer le nombre de classes restantes. Cette option est utile lorsque les données correspondent à des nombres d'individus par espèce. En effet souvent les récoltes peuvent correspondre à des valeurs nulles (les vrais zéros mesurés, sans considérer les valeurs manquantes éventuelles) sans que l'on puisse savoir s'il s'agit de l'absence de l'espèce ou d'un biais dû à l'échantillonnage.

* prendre le zéro s'il correspond au minimum dans le cas où un pourcentage de zéros trés faible ne suggère pas un biais de l'échantillonnage.

3. Utiliser le codage disjonctif, ce qui donne pour chaque descripteur, plusieurs variables binaires (0 et 1). Par cette méthode un descripteur va être représenté par plusieurs vecteurs de 0 et de 1. Les traitements de données montrent que ce sytème est très riche. On va s'apercevoir que même si un descripteur biologique, globalement, ne peut être relié

significativement à un paramètre de l'environnement, au contraire, ses valeurs faibles ou trés fortes, elles, sont vraiment indicatrices des changements du milieu.

Voici d'autres exemples de codage en classe, en utilisant le language S, et illustrant ces différentes méthodes:

```
> Z <- c(abs(rnorm(8000)), rep(0, 2000))   # Don. artif, 20% zeros
> table(cut(Z, breaks=5)) # Laisse le programme choisir la coupure
(-0.00429,0.855] (0.855,1.72] (1.72,2.57] (2.57,3.43] (3.43,4.29]
            6848         2461         611          76           4
```

Dans cet exemple, on peut préférer une classe particulière pour les zéros et un découpage en 4 autres classes d'effectifs égaux. Les coupures au niveau de ces classes sont alors déterminées grâce aux quantiles:

```
> Z2 <- Z[Z != 0]            # Données à l'exclusion des 0 dans Z2
> cuts <- c(-1e-10, 1e-10, quantile(Z2, 1:5/5, na.rm=TRUE))
> cuts
                                      20%           40%
-0.0000000001  0.0000000001  0.2493478137  0.5255144907
          60%           80%          100%
 0.8453325252  1.2862470498  4.2897873304

> table(cut(Z, breaks=cuts))

(-1e-010,1e-010]   (1e-010,0.249]    (0.249,0.526]
            2000             1600             1600
   (0.526,0.845]     (0.845,1.29]      (1.29,4.29]
            1600             1600             1600
```

La première ligne extrait un vecteur contenant toutes les valeurs non nulles. La seconde ligne crée un vecteur contenant les limites des classes. La première classe contient les zéros uniquement. Comme la function ***cut()*** n'accepte pas d'intervalles nuls, nous devons spécifier un intervalle très petit entourant zéro, soit par exemple -1^{-10} à 1^{-10}. Les intervalles irréguliers des autres classes d'effectifs égaux sont déterminés grâce à la fonction ***quantile()***, en spécifiant le quantile désiré. Par exemple, pour séparer en 2 classes d'effectifs égaux, on spécifiera: `quantile(Z2, c(0, 0.5, 1))`. Dans le cas présent, on veut 4 classes supplémentaires et la borne inférieur de la première classe est déjà définie, puisqu'il s'agit de la borne supérieure de la classe zéro (soit 1^{-10}). On utilise donc les quantiles 0.2, 0.4, 0.6, 0.8 et 1, ou en abbrégé: `1:5/5`. La dernière ligne de code effectue la séparation en classes et crée le tableau de contingence correspondant à cette séparation. Sur les 10.000 valeurs générées artificiellement, on obtient bien 2000 valeurs dans la classe zéro, et un même effectif de 1600 valeurs dans chacune des quatre autres classes.

Le codage disjonctif complet (création d'une variable par classe et attribution de la valeur 0 ou 1 selon que l'élément considéré appartient ou non à cette classe) est réalisé grâce à la fonction PASTECS ***disjoin()***. Ainsi, en reprenant le même exemple et les mêmes classes que ci-dessus, voici les dix premières lignes obtenues:

```
> disjoin(cut(Z, breaks=cuts))[1:10,]
      (-1e-010,1e-010] (1e-010,0.249] (0.249,0.526] (0.526,0.845]
 [1,]                0              0             0             0
 [2,]                0              1             0             0
 [3,]                0              0             0             0
 [4,]                0              0             0             0
 [5,]                0              0             0             0
 [6,]                0              0             1             0
 [7,]                0              0             0             1
 [8,]                0              0             1             0
 [9,]                0              0             1             0
[10,]                0              0             0             1
```

```
      (0.845,1.29] (1.29,4.29]
 [1,]            1           0
 [2,]            0           0
 [3,]            0           1
 [4,]            0           1
 [5,]            1           0
 [6,]            0           0
 [7,]            0           0
 [8,]            0           0
 [9,]            0           0
[10,]            0           0
```

Au niveau du recodage des données, PASTECS fournit aussi la fonction ***buysbal()*** qui crée un tableau de Buys-Ballot à partir d'un vecteur temps et d'un vecteur d'observations correspondantes, ou bien à partir d'une série temporelle régulière unique (objet *'rts'* ou *'ts'*, voir chapitre régularisation) dont le temps est exprimé en unité année ("years") ou journée ("days"). Le jeu de données ***releve*** contient une colonne ***Day*** qui représente les indications temporelles sur les observations (le jour de chaque observation, compté à partir de 1 pour la première observation). Etant donné que l'échelle de temps dans ***releve$Day*** est la journée, et que d'autre part, le temps y est exprimé à partir du jour 1 (donc, une référence relative par rapport au début de l'expérience, et non un temps en valeur absolue), nous devons préciser **units="days"** d'une part, et aussi recaler ce "jour 1" grâce à **datemin="21/03/1989"** d'autre part. L'argument **frequency** de ***buysbal()*** indique le nombre de colonnes désirées dans la table (préciser 4 pour des données trimestrielles, 12 pour des données mensuelles, 24 pour des données bi-mensuelles, etc…). Si les séries de départ contiennent plusieurs valeurs dans les intervalles considérés, elles seront moyennées. Le tableau de Buys-Ballot pour des valeurs trimestrielles de la série ***Melosul*** sera donc obtenu par:

```
> data(releve)   # Dans R uniquement
> buysbal(releve$Day, releve$Melosul, frequency=4, units="days",
+ datemin="21/03/1989", dateformat="d/m/Y")
             1         2         3     4
1989 10500.000 14000.000  4666.667 23530
1990  5406.667  2550.000  5950.000 41920
1991  8800.000 21166.667  5668.333  9900
1992  7566.667  6816.667  4000.000   364
```

On peut déterminer le nombre d'observations qui ont servi à remplir chaque case du tableau si on précise **count = TRUE**. Ceci est utile pour déterminer si les observations sont bien balancées entre les cases du tableau et si la valeur de **frequency** est acceptable pour la série calculée (réduire sa valeur si le nombre d'observations dans certaines cases est trop faible):

```
> buysbal(releve$Day, releve$Melosul, frequency=4, units="days",
+ datemin="21/03/1989", dateformat="d/m/Y", count=TRUE)
      1  2  3  4
1989  1  1  3  2
1990  3  2  6  5
1991  3  3  6  6
1992  6  6  6  2
```

Références:

Fromentin J.-M., F. Ibanez & P. Legendre, 1993. A phytosociological method for interpreting plankton data. *Mar. Ecol. Prog. Ser.*, 93:285-306.

Gebski, V.J., 1985. Some properties of splicing when applied to non-linear smoothers. *Comput. Stat. Data Anal.*, 3:151-157.

Grandjouan, G., 1982. Une méthode de comparaison statistique entre les répartitions des plantes et des climats. *Thèse d'Etat, Université Louis Pasteur, Strasbourg*.

Ibanez, F., 1976. Contribution à l'analyse mathématique des événements en Ecologie planctonique. Optimisations méthodologiques. *Bull. Inst. Océanogr. Monaco*, 72:1-96.

Venables, W.N. & B.D. Ripley, 2002. Modern applied statistics with S-plus. 4th ed. Springer, New York. 495 pp.

Organisation objet des routines PASTECS

Afin de comprendre l'organisation des routines au sein de la librairie PASTECS, il est utile de comprendre comment S+ et R gèrent les données en tant qu'entités indépendantes appelées *objets*. En général, une analyse donnée nécessite plusieurs étapes de calcul. Il faut souvent préparer les données (transformation), effectuer un ou plusieurs tests qui permettent de confirmer les hypothèses de départ nécessaires à l'analyse (distribution normale, variances égales entre les groupes, par exemple). De même, une fois l'analyse effectuée, les résultats peuvent habituellement être présentés de manières différentes (tableaux, graphiques,…). Enfin, certaines analyses conduisent à extraire une partie de l'information ou à construire un modèle qui servira ensuite à une autre analyse.

L'ensemble de ces actions peut difficilement être rassemblé au sein d'une fonction unique, ou alors, celle-ci devrait admettre une quantité énorme d'arguments en entrée afin de prendre en compte toutes les possibilités. S+ et R utilisent une autre stratégie. Pour un traitement donné, il existe une fonction qui effectue le travail de base. Il crée un *objet* (une structure de données particulière qui contient toutes les informations nécessaires pour permettre la suite du traitement). Cet objet est identifié par sa *classe* (le type de l'objet qui porte souvent le même nom que la fonction qui a servi à le créer, mais pas obligatoirement). Toute une panoplie de fonctions *liées à cet objet* peut ensuite lui être appliquée pour poursuivre l'analyse. Les fonctions « liées » à des objets sont appelées *méthodes* de l'objet. En général, elles portent un nom générique tel que ***print()***, ***summary()***, ***plot()***, ***hist()***, ***lines()***, ***rect()***, ***identify()***, ***predict()***, ***extract()***, ***specs()***, …

Pour chaque objet, les méthodes se comportent différemment, afin de s'adapter au contexte. Ainsi, la méthode ***plot()*** de l'objet *data.frame*, aussi appelée ***plot.data.frame()*** trace un graphique différent de la méthode ***plot.ts()*** qui trace le graphique de séries temporelles régulières (objets *'ts'*, sous R). Comme le programme reconnaît le type d'objet donné en argument à la méthode ***plot()***, il n'est pas utile de préciser ***plot.data.frame()*** ou ***plot.ts()***. Dans tous les cas, ***plot(obj)*** appellera la méthode de l'objet ***obj*** automatiquement selon qu'il est reconnu comme *data.frame*, *ts* (ou tout autre object offrant une méthode ***plot()*** spécifique). Si l'objet n'expose pas de méthode spécifique, alors, la fonction ***plot.default()*** est utilisée.

La librairie PASTECS est *orientée objet*, c'est-à-dire, qu'elle suit ce concept pour pratiquement tous les traitements proposés. Ceci explique le nombre important de fonctions disponibles pour chaque traitement (habituellement, entre 5 et 7 fonctions). Une fois que l'on comprend l'organisation objet de ces traitements, on apprécie beaucoup mieux leur souplesse et leur facilité d'utilisation **mais il est vrai qu'il s'agit probablement là de l'aspect le plus difficile à assimiler pour un débutant, voire pour un utilisateur habitué à un langage fonctionnel non orienté objet, tel que Matlab**. Afin de vous familiariser avec cette organisation, la description des fonctions sera suivie d'exemples détaillés qui utilisent la plupart des méthodes de l'objet, afin de montrer l'étendue des possibilités qu'elles offrent.

Les principaux objets de S+ ou de R manipulés par les fonctions PASTECS sont les *data frames*, mais aussi et surtout les *time series* (c'est-à-dire des objets *'rts'* sous S+ et *'ts'* sous R) qui doivent être régulières. Cela implique de régulariser des séries irrégulières, ou à trous, en passant par un objet *'regul'*. Ensuite, les séries sont décomposées (objet *'tsd'* ou *time series decomposition*), et les composantes sont analysées, voire décomposées encore à leur tour (voir schéma ci-dessous).

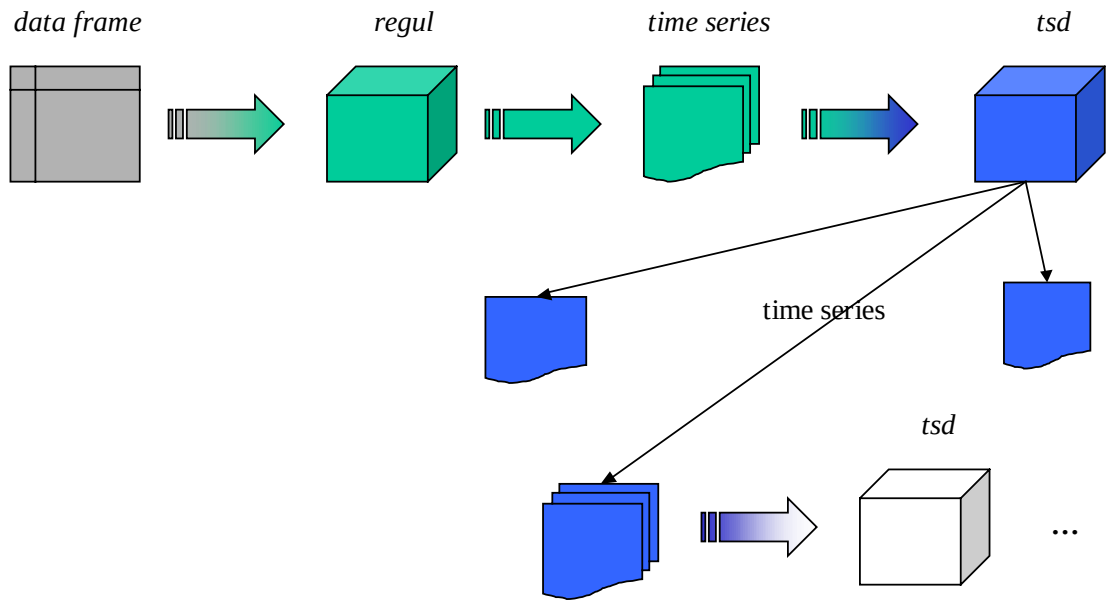


Schéma montrant le flux des données au cours d'une analyse avec PASTECS, ainsi que les différents objets qui entrent en jeu à chaque étape du traitement. Seules les étapes principales sont indiquées, à savoir: régularisation, transformation en time series, décomposition (objets 'tsd') et extraction des composantes en nouvelles time series, éventuellement décomposées à nouveau. D'autres analyses génèrent également des objets (non représentés dans ce schéma), mais leur logique est similaire.

Lorsque l'on a compris la logique de cette organisation objet, on a assimilé l'essentiel du processus de traitement que l'on fera à l'aide de PASTECS. Le tableau III présente la liste de tous les objets créés par une ou plusieurs des fonctions de la librairie.

Tableau III: *objets générés par une ou plusieurs fonctions de la librairie PASTECS.*

Objet PASTECS	page	Description
stat.slide	p. 43	Statistiques glissantes d'une série
escouf	p. 47	Méthode d'Escoufier appliquée à un *data frame*
abund	p. 53	Tri par abondance d'un *data frame*
regul	p. 57	Régularisation de données irrégulièrement espacées
D2	p. 91	Analyse AutoD2, CrossD2 ou D2 au centre
turnpoints	p. 106	Calcul des points de retournement
turnogram	p. 111	Calcul d'un tournogramme
local.trend	p. 125	Tendance locale par les sommes cumulées
tsd	p. 127	Décomposition d'une ou de plusieurs séries

Statistiques descriptives

Dans PASTECS, plusieurs fonctions permettent de résumer les données et d'afficher des statistiques descriptives soit classiques (***stat.desc()***), soit selon les estimateurs de Pennington (***stat.pen()***, ***pennington()***), soit enfin pour différentes fenêtres temporelles au sein d'une même série (statistiques glissantes, ***stat.slide()***).

Des informations de base sur les séries de données contenues dans des *data frames* sont obtenues par la méthode ***summary()***. Par exemple, pour les variables 13, 14, 15 et 16 de ***marbio***, on obtient:

```
> data(marbio)          # Dans R uniquement
> summary(marbio[,13:16])
    Nauplii          Oithona       Acanthaires      Cladocerans
 Min.   :  3.0   Min.   :  39   Min.   : 0.00   Min.   :  0.000
 1st Qu.: 67.5   1st Qu.: 759   1st Qu.: 4.00   1st Qu.:  0.000
 Median :104.0   Median :1228   Median :12.00   Median :  0.000
 Mean   :132.6   Mean   :1546   Mean   :14.51   Mean   :  5.794
 3rd Qu.:168.0   3rd Qu.:1939   3rd Qu.:24.00   3rd Qu.:  4.000
 Max.   :636.0   Max.   :7792   Max.   :52.00   Max.   :200.000
```

Toutefois, de telles informations sont très sommaires. Une fonction plus élaborée a été ajoutée à la librairie PASTECS pour afficher un tableau de statistiques descriptives courantes sur les variables contenues dans un *data frame*: la fonction ***stat.desc()***. Sur les mêmes données de ***marbio***, cela donne:

```
> stat.desc(marbio[,13:16], basic=TRUE, desc=TRUE,
+ norm=TRUE, p=0.95)
                  Nauplii      Oithona  Acanthaires  Cladocerans
nbr.val      6.800000e+01 6.800000e+01 6.800000e+01 6.800000e+01
nbr.null     0.000000e+00 0.000000e+00 8.000000e+00 4.500000e+01
nbr.na       0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
min          3.000000e+00 3.900000e+01 0.000000e+00 0.000000e+00
max          6.360000e+02 7.792000e+03 5.200000e+01 2.000000e+02
range        6.330000e+02 7.753000e+03 5.200000e+01 2.000000e+02
sum          9.018000e+03 1.051590e+05 9.870000e+02 3.940000e+02
median       1.040000e+02 1.228000e+03 1.200000e+01 0.000000e+00
mean         1.326176e+02 1.546456e+03 1.451471e+01 5.794118e+00
SE.mean      1.254198e+01 1.554392e+02 1.458903e+00 3.035968e+00
CI.mean.0.95 2.503389e+01 3.102580e+02 2.911983e+00 6.059818e+00
var          1.069648e+04 1.642972e+06 1.447311e+02 6.267629e+02
std.dev      1.034238e+02 1.281785e+03 1.203042e+01 2.503523e+01
coef.var     7.798644e-01 8.288530e-01 8.288439e-01 4.320802e+00
skewness     2.037948e+00 2.579061e+00 9.098177e-01 6.945251e+00
skew.2SE     3.504462e+00 4.434961e+00 1.564525e+00 1.194307e+01
kurtosis     6.433595e+00 8.866041e+00 5.389480e-01 5.039083e+01
kurt.2SE     5.604129e+00 7.722966e+00 4.694629e-01 4.389407e+01
normtest.W   8.239310e-01 7.538447e-01 9.182656e-01 2.253479e-01
normtest.p   1.507167e-07 2.457390e-09 2.736212e-04 3.745681e-17
```

Le test de normalité Shapiro-Wilk (**normtest.W** et **normtest.p**) n'est, pour l'instant, pas disponible sous S+. Dans cet environnement, les colonnes correspondantes sont donc remplies de NAs!

Estimateur de Pennington

Un problème particulièrement complexe en écologie marine est la notion de zéro. En effet l'échantillonnage « en aveugle » d'une population de poissons par exemple aboutit souvent à des tableaux d'abondance avec un pourcentage très élevé de valeurs nulles. Est-ce que ces valeurs indiquent l'absence de poissons ou bien doit-on les considérer comme des valeurs manquantes, c'est-à-dire que dans ce cas, le poisson était rare mais présent, et c'est l'évitement de l'engin de pêche qui est en cause?

Autrement dit, quelle est la représentativité des valeurs nulles? Si nous avons réalisé 100 pêches et trouvé 80 valeurs nulles, comment peut-on estimer la moyenne par unité de pêche? Doit-on diviser le nombre total de poissons par 20 ou par 100?

On doit à Pennington (1983) le mérite d'avoir appliqué, en halieutique, la théorie probabiliste proposée par Aitchison (1955). Ce dernier avait établi une technique d'estimation de la moyenne et de la variance pour une variable aléatoire qui aurait une probabilité non nulle de présenter des zéros, et dont la distribution conditionnelle correspondant à ses autres valeurs (non nulles), suivrait une distribution normale. Pratiquement, la distribution lognormale est observée le plus couramment pour les valeurs non nulles. Ce résultat est très répandu en ichtyologie et se rencontre presque toujours pour le plancton comme nous avons pu le vérifier.

Considérons n observations x_1, x_2, …, x_n et m valeurs non nulles parmi elles. Si la distribution de ces m valeurs est lognormale (distribution de Aitchison), pour tous les x_i non nuls, on a: $y_i = \log(x_i)$ admettant une distribution normale de moyenne $\bar{y}$ et de variance σ^2. Les estimateurs non biaisés de la moyenne ($\hat{P}$) et de la variance ($\hat{V}$) des observations $x_1 \ldots x_n$ initiales peuvent se calculer comme suit:

$$\hat{P} = \begin{cases} \dfrac{m}{n} e^{\bar{y}} G_m\left(\dfrac{1}{2}\sigma^2\right) & \text{pour } m>1 \\ \dfrac{x_m}{n} & \text{pour } m=1 \\ 0 & \text{pour } m=0 \end{cases}$$

$$\hat{V} = \begin{cases} \dfrac{m}{n} e^{2\bar{y}} \left[G_m\left(2\sigma^2\right) - \dfrac{m-1}{n-1} G_m\left(\dfrac{m-2}{m-1}\sigma^2\right) \right] & \text{pour } m>1 \\ \dfrac{x_m^{\,2}}{n} & \text{pour } m=1 \\ 0 & \text{pour } m=0 \end{cases}$$

avec:

$$G_m(t) = 1 + \frac{m-1}{m} t + \sum_{j=2}^{\infty} \frac{(m-1)^{2j-1}}{m^j (m+1)(m+3)\ldots(m+2j-3)} \frac{t^j}{j!}$$

Une estimation non-biaisée de la variance de $e^{\bar{y}} G_m\left(\dfrac{1}{2}\sigma^2\right)$ devient dans la même notation que ci-dessus:

$$e^{2\bar{y}} \left[G_m^2\left(\frac{1}{2}\sigma^2\right) - G_m\left(\frac{m-2}{m-1}\sigma^2\right) \right]$$

On démontre alors que la variance estimée de la moyenne $\hat{V}(\hat{P})$, se calcule par:

$$\hat{V}(\hat{P}) = \begin{cases} \frac{m}{n} e^{2\bar{y}} \left[\frac{m}{n} G_m^2 \left(\frac{1}{2}\sigma^2 \right) - \frac{m-1}{n-1} G_m \left(\frac{m-2}{m-1}\sigma^2 \right) \right] & \text{pour } m>1 \\ \left(\frac{x_m}{n} \right)^2 & \text{pour } m=1 \\ 0 & \text{pour } m=0 \end{cases}$$

Quand le nombre de valeurs nulles est grand, ce qui est souvent le cas pour des données ichtyologiques ou planctoniques, $\hat{P}$ est considérablement plus précis que la moyenne arithmétique ordinaire.

Références:

Aitchison, J., 1955. On the distribution of a positive random variable having a discrete probability mass at the origin. *J. Amer. Stat. Ass.*, 50:901-908.

Pennington, M., 1983. Efficient estimations of abundance for fish and plankton surveys. *Biometrics*, 39:281-286.

Dans PASTECS, les estimateurs de Pennington sont calculés à l'aide de la fonction ***pennington()***. Néanmoins, pour obtenir une sortie plus complète, et pour pouvoir calculer plusieurs variables simultanément, la fonction ***stat.pen()*** est disponible.

Exemple:

On peut calculer les estimateurs de Pennington pour le jeu de données ***marbio***, ainsi que d'autres statistiques utiles pour la comparaison pour les variables présentes dans les colonnes 4, 14, 15 et 16 par:

```
> data(marbio)         # Dans R uniquement
> stat.pen(marbio[,c(4,14:16)], basic=TRUE, desc=TRUE)
             Copepodits2     Oithona Acanthaires Cladocerans
nbr.val        68.000000      68.000   68.000000   68.000000
nbr.null        5.000000       0.000    8.000000   45.000000
percnull        7.352941       0.000   11.764706   66.176471
nbr.na          0.000000       0.000    0.000000    0.000000
median         24.500000    1228.000   12.000000    0.000000
mean           39.632353    1546.456   14.514706    5.794118
var          1619.877744 1642971.655  144.731124  626.762950
std.dev        40.247705    1281.785   12.030425   25.035234
pos.median     28.000000    1228.000   13.000000    4.000000
pos.mean       42.777778    1546.456   16.450000   17.130435
pos.var      1613.788530 1642971.655  131.980508 1705.754941
pos.std.dev    40.171987    1281.785   11.488277   41.300786
geo.mean       31.504949    1161.198   12.030807    6.482851
pen.mean       38.604561    1624.345   16.011929    4.183929
pen.var      1302.882154 2463485.626  389.856084  144.116540
pen.std.dev    36.095459    1569.549   19.744774   12.004855
pen.mean.var   18.476177   34230.002    5.325181    1.900941
```

Une telle présentation est tout à fait similaire à ***stat.desc()***. On aussi calculer plus simplement les estimateurs de Pennington pour une seule série, par exemple, la série ***Copepodits2*** par:

```
> pennington(marbio[,"Copepodits2"])
       mean         var    mean.var
  38.604561 1302.882154    18.47618
```

Pour ne calculer que la moyenne de Pennington, tout en éliminant d'éventuelles valeurs manquantes et en introduisant le résultat dans une variable ***Pm***, on utilisera:

```
> Pm <- pennington(marbio[,"Copepodits2"], calc="mean",
+ na.rm=TRUE)
> Pm
[1] 38.604561
```

Statistiques glissantes

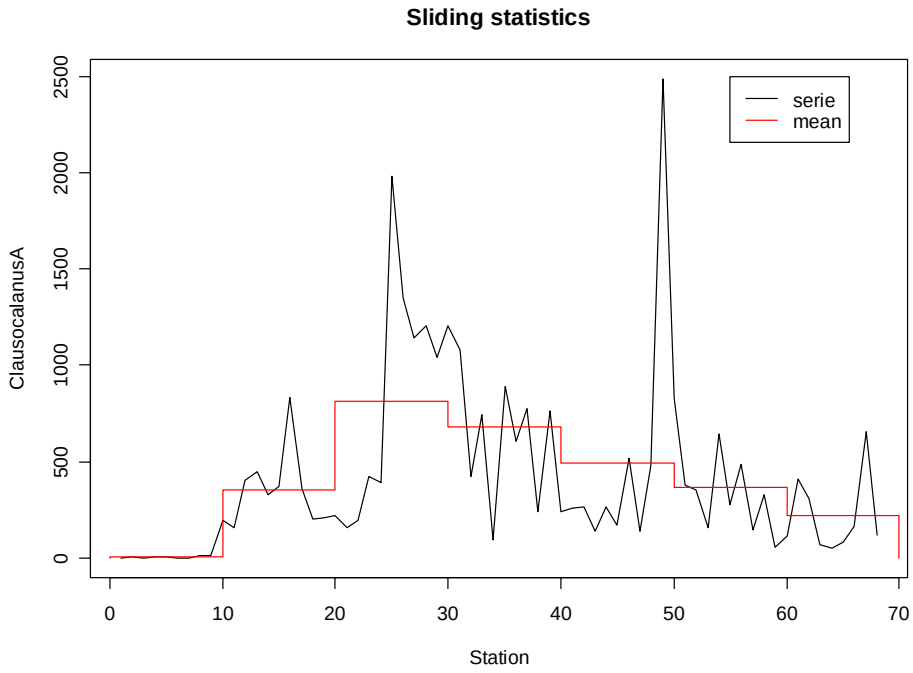
Les paramètres statistiques ne sont pas constants tout au long d'une série: la moyenne et la variance chaque année ou pendant des intervalles particuliers définis par des évènements très reconnaissables, apparition d'une pollution, hivers froids, etc. Il est très utile de disposer d'un programme qui fournit des estimations de ces paramètres statistiques de base pour des intervalles de temps précis. Les statistiques glissantes (« sliding statistics » en anglais) correspondent à l'analyse de blocs successifs de données suivant un axe spatio-temporel.

Exemple:

On peut calculer des statistiques glissantes pour la série ***ClausocalanusA*** de ***marbio*** par groupe de 10 stations, les imprimer et faire un graphique de la « moyenne glissante » à l'aide des instructions suivantes:

```
> data(marbio)          # Dans R uniquement
> statsl <- stat.slide(1:68, marbio[,"ClausocalanusA"], xmin=0,
+ n=7, deltat=10)
> statsl
          [0,10[ [10,20[ [20,30[ [30,40[ [40,50[ [50,60[ [60,70[
xmin        0.00   10.00   20.00   30.00   40.00   50.00   60.00
xmax       10.00   20.00   30.00   40.00   50.00   60.00   70.00
nbr.val     9.00   10.00   10.00   10.00   10.00   10.00    9.00
nbr.null    2.00    0.00    0.00    0.00    0.00    0.00    0.00
nbr.na      0.00    0.00    0.00    0.00    0.00    0.00    0.00
min         0.00  160.00  158.00   96.00  136.00   56.00   52.00
max        12.00  832.00 1980.00 1204.00 2484.00  824.00  656.00
median      4.00  344.00  732.00  752.00  260.00  340.00  120.00
mean        4.78  350.80  810.10  682.00  494.40  364.80  219.11
std.dev     4.79  196.97  619.86  350.06  711.05  234.91  202.42
```

```
> plot(statsl, stat="mean", leg=TRUE, lpos=c(55,2500),
+ xlab="Station", ylab="ClausocalanusA")
```



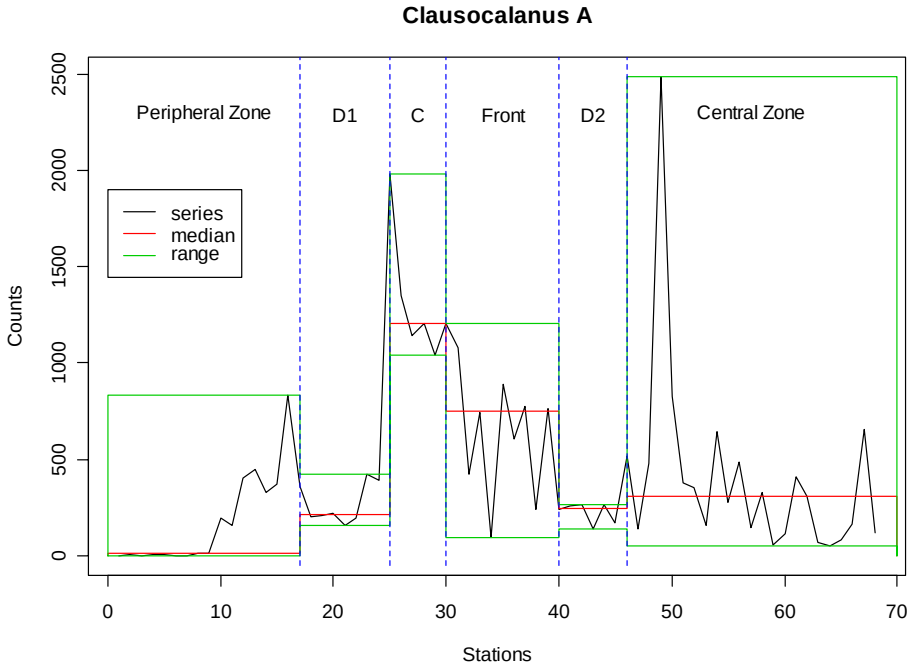
Toujours pour la même série, on peut calculer toutes les statistiques sur des intervalles irréguliers (seules les 4 premiers sont imprimés), puis représenter l'étendue (minimum, maximum) et la médiane pour chaque intervalle comme suit:

```
> statsl2 <- stat.slide(1:68, marbio[,"ClausocalanusA"],
+ xcut=c(0,17,25,30,41,46,70), basic=TRUE, desc=TRUE, norm=TRUE,
+ pen=TRUE, p=0.95)
> statsl2
                [0,17[       [17,25[       [25,30[       [30,40[
xmin        0.000000e+00  1.700000e+01  2.500000e+01  3.000000e+01
xmax        1.700000e+01  2.500000e+01  3.000000e+01  4.000000e+01
nbr.val     1.600000e+01  8.000000e+00  5.000000e+00  1.000000e+01
nbr.null    2.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
nbr.na      0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
min         0.000000e+00  1.580000e+02  1.040000e+03  9.600000e+01
max         8.320000e+02  4.240000e+02  1.980000e+03  1.204000e+03
range       8.320000e+02  2.660000e+02  9.400000e+02  1.108000e+03
sum         2.785000e+03  2.151000e+03  6.716000e+03  6.820000e+03
median      1.200000e+01  2.120000e+02  1.204000e+03  7.520000e+02
mean        1.740625e+02  2.688750e+02  1.343200e+03  6.820000e+02
SE.mean     6.066894e+01  3.704796e+01  1.668120e+02  1.106992e+02
CI.mean.95  1.293128e+02  8.760450e+01  4.631443e+02  2.504190e+02
var         5.889153e+04  1.098041e+04  1.391312e+05  1.225431e+05
std.dev     2.426758e+02  1.047875e+02  3.730029e+02  3.500616e+02
coef.var    1.394188e+00  3.897255e-01  2.776973e-01  5.132868e-01
skewness    1.248551e+00  4.181831e-01  8.469063e-01 -2.088529e-01
skew.2SE    1.106268e+00  2.780098e-01  4.638697e-01 -1.519941e-01
kurtosis    7.020336e-01 -1.852389e+00 -1.193537e+00 -1.256496e+00
kurt.2SE    3.218053e-01 -6.254351e-01 -2.983842e-01 -4.708629e-01
normtest.W  7.557989e-01  8.300441e-01  8.162687e-01  9.670516e-01
normtest.p  7.467702e-04  5.943017e-02  1.092400e-01  8.622442e-01
pos.median  8.600000e+01  2.120000e+02  1.204000e+03  7.520000e+02
pos.mean    1.989286e+02  2.688750e+02  1.343200e+03  6.820000e+02
geo.mean    3.935062e+01  2.522408e+02  1.307640e+03  5.595596e+02
```

```
pen.mean    3.351203e+02  2.683489e+02  1.340628e+03  7.291947e+02
pen.var     1.907110e+06  1.039770e+04  1.119418e+05  3.505851e+05
pen.std.dev 1.380981e+03  1.019691e+02  3.345771e+02  5.921022e+02
pen.mean.var5.558396e+04  1.298434e+03  2.238628e+04  3.437841e+04
```

Le test de normalité Shapiro-Wilk n'étant pas encore disponible sous S+, les colonnes **normtest.W** et **normtest.p** renvoient NA dans cet environnement.

```
> plot(statsl2, stat="median", xlab="Stations", ylab="Counts",
+ main="Clausocalanus A")                          # Médiane
> lines(statsl2, stat="min")                       # Minimum
> lines(statsl2, stat="max")                       # Maximum
> lines(c(17,17), c(-50,2600), col=4, lty=2)       # Séparations
> lines(c(25,25), c(-50,2600), col=4, lty=2)
> lines(c(30,30), c(-50,2600), col=4, lty=2)
> lines(c(41,41), c(-50,2600), col=4, lty=2)
> lines(c(46,46), c(-50,2600), col=4, lty=2)
> text(c(8.5,21,27.5,35,43.5,57), 2300, labels=c("Peripheral
+ Zone", "D1", "C", "Front", "D2", "Central Zone")) # Labels
> legend(0, 1900, c("series", "median", "range"), col=1:3, lty=1)
```



Enfin, on peut extraire différentes statistiques, les valeurs de la série initiale (y), les valeurs de temps de la série (x), et le vecteur temps de coupure des périodes (xcut) à partir de l'objet *stat.slide* renvoyé:

```
> statsl2$stat[c("mean", "pos.mean", "geo.mean", "pen.mean"),]
            [0,17[  [17,25[  [25,30[  [30,40[  [40,46[  [46,70[
mean     174.06250 268.8750 1343.200 682.0000 221.3333 401.5652
pos.mean 198.92857 268.8750 1343.200 682.0000 221.3333 401.5652
geo.mean  39.35062 252.2408 1307.640 559.5596 214.6600 252.7195
pen.mean 335.12029 268.3489 1340.628 729.1947 221.8544 390.5421

> statsl2$y
 [1]    0    4    2    8    4    0    1   12   12  196  160  406
[13]  448  328  372  832  360  200  206  218  158  193  424  392
[25] 1980 1348 1144 1204 1040 1204 1080  424  744   96  888  608
[37]  776  240  760  240  256  264  136  264  168  516  136  480
```

```
[49] 2484  824  376  352  160  640  280  488  144  328   56  112
[61]  408  308   68   52   84  164  656  120

> statsl2$x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[21] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
[41] 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
[61] 61 62 63 64 65 66 67 68

> statsl2$xcut
[1]  0 17 25 30 40 46 68
```

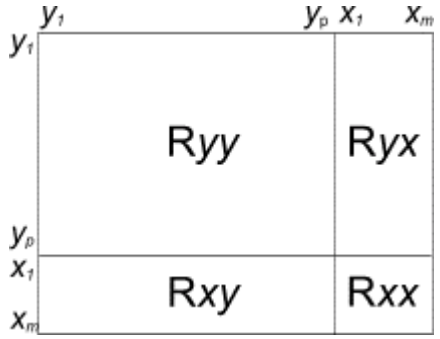
Sélection des descripteurs

Lorsque les matrices de données brutes contiennent beaucoup de descripteurs, il peut être utile d'en supprimer certains pour alléger les calculs. Encore faut-il pouvoir déterminer ceux qui ont une importance moindre dans la représentation du phénomène étudié. PASTECS offre deux techniques complémentaires: la méthode des **vecteurs équivalents d'Escoufier** (***escouf()***, utile pour des variables environnementales), et le **tri par ordre d'abondance** (***abund()***, adaptées aux matrices de type espèces/stations).

Méthode d'Escoufier

La méthode des vecteurs équivalents d'Escoufier (1970) a pour but d'extraire de p variables mesurées, un échantillon de q variables ($q < p$) de telle sorte que les composantes principales de l'échantillon q soient aussi proches que possible de celles du groupe initial à p variables. On obtient un classement des variables dans l'ordre décroissant de la variance totale expliquée.

Prenons le cas général de deux séries multivariables x et y. On peut calculer une matrice de corrélation entre les variables x et y telle que:



Matrice de corrélation entre les séries de l'ensemble y et de l'ensemble x.

La sous-matrice R_{yy} contient les coefficients de corrélation pour chaque combinaison de descripteurs de la série y pris deux à deux. La sous-matrice R_{xx} contient similairement les coefficients de corrélation pour les descripteurs de la série x, pris deux à deux. Les sous-matrices R_{xy} et R_{yx} contiennent les coefficients de corrélation pour les descripteurs des deux séries croisés entre eux.

La corrélation RV entre les séries y et x, telle qu'elle puisse être maximum (égale à 1) lorsque les composantes principales de y et de x sont proportionnelles (c'est-à-dire de même direction), se calculera par:

$$RV_{xy} = \frac{\mathrm{Tr}(R_{yx}R_{xy})}{\sqrt{\mathrm{Tr}(R_{yy}^2)\,\mathrm{Tr}(R_{xx}^2)}}$$

RV varie entre 0 et 1.

Le problème de sélection des variables les plus représentatives d'un groupe de descripteurs peut se résoudre en considérant des descripteurs x qui correspondent en fait à un sous-ensemble de y (Cambon, 1974) et en choisissant ce sous-ensemble de manière à obtenir le plus grand RV possible (meilleure corrélation entre l'ensemble des descripteurs d'origine et le sous-ensemble des descripteurs retenus). Pour définir la variable de y qui est la plus corrélée à la première composante extraite de Ryy, on va calculer le coefficient RV avec toutes les séries de y envisagées successivement comme une série x unique accolée au tableau y. Ensuite, tout en conservant cette première variable la plus corrélée comme première série de x, on va tester de la même manière toutes les séries de y

restantes pour obtenir le groupe *x* de 2 variables qui donne le plus grand *RV*. On continuera le calcul de façon similaire pour des sous-ensembles *x* de 3, 4,..., *p* séries de *y*. Evidemment *RV* vaut 1 avec les *p* séries formant *x* (puisqu'alors les deux ensembles *x* et *y* sont rigoureusement égaux).

On obtient finalement un classement des *p* séries de *y* tel que la valeur de *RV* soit la plus grande pour chaque sous-ensemble *q* de séries extraites dans l'ordre ainsi défini, et ce, pour *q* allant de 1 à *p* (c'est-à-dire que pour chaque valeur de *q*, le sous-ensemble des *q* premières séries classées est celui qui donne le plus grand *RV* parmi toutes les combinaisons possibles de *q* séries). En examinant la variation de *RV* en fonction de *q*, ou mieux, la variation de *RV*', soit $RV_q - RV_{q-1}$, on peut décider quel sous-ensemble conserver, par exemple par identification d'un point d'inflexion dans cette courbe.

Cette méthode est très importante pour condenser l'information d'un tableau et reconnaître les variables les plus représentatives de sa structure au sens donné par l'analyse en composantes principales (ACP). De plus elle a le mérite, par rapport à la sélection des premières composantes de *y* par une ACP, de pouvoir représenter les objets dans l'espace des variables les plus corrélées aux axes principaux, plutôt que par rapport aux composantes principales qui sont, elles, une combinaison linéaire de toutes les variables.

Références:

Cambon, J., 1974. Vecteur équivalent à un autre au sens des composantes principales. Application hydrologique. *DEA de Mathématiques Appliquées, Université de Montpellier*.

Escoufier, Y., 1970. Echantillonnage dans une population de variables aléatoires réelles. *Pub. Inst. Stat. Univ. Paris*, 19:1-47.

Jabaud, A., 1996. Cadre climatique et hydrobiologique du lac Léman. *DEA d'Océanologie Biologique Paris*.

Dans PASTECS, la méthode d'Escoufier est calculée à partir de la fonction ***escouf()***. Fidèle à la philosophie orientée objet de S+, la fonction ne calcule néanmoins que les données de base du traitement et renvoie un objet spécifique de classe '*escouf*' qui contient toutes les données nécessaires pour l'affichage d'un résumé de l'analyse (***summary()***), le traçage de graphique (***plot()***, ***lines()***, ***identify()***), ou pour extraire le résultat final du traitement dans un nouveau data frame (***extract()***). Ces différentes fonctions sont utilisées tour à tour dans l'exemple ci-dessous.

Exemple:

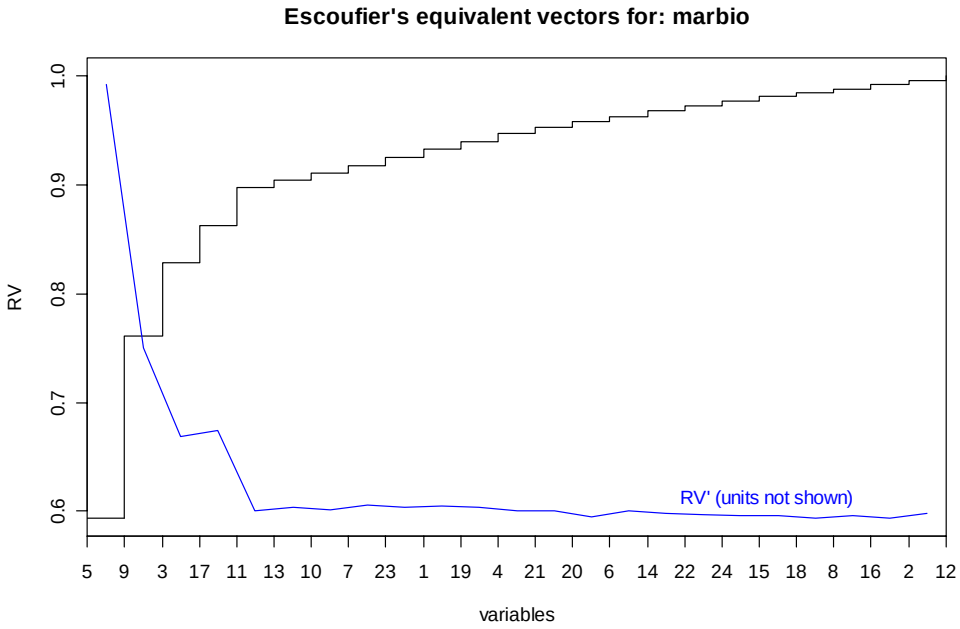
On peut analyser le jeu de données ***marbio*** selon la méthode d'Escoufier de la manière suivante (l'affichage des étapes de calcul est indiqué par **verbose = TRUE**, mais cela n'est pas toujours utile):

```
> data(marbio)         # Dans R uniquement
> marbio.esc <- escouf(marbio, verbose=TRUE)
Variable   5 incorporated, RV = 0.5938735
Variable   9 incorporated, RV = 0.7607149
Variable   3 incorporated, RV = 0.8280316
Variable  17 incorporated, RV = 0.8620636
Variable  11 incorporated, RV = 0.8981089
Variable  13 incorporated, RV = 0.9039953
```

```
Variable  10 incorporated, RV = 0.9109498
Variable   7 incorporated, RV = 0.917168
Variable  23 incorporated, RV = 0.9254146
Variable   1 incorporated, RV = 0.932456
Variable  19 incorporated, RV = 0.9399177
Variable   4 incorporated, RV = 0.9469293
Variable  21 incorporated, RV = 0.952844
Variable  20 incorporated, RV = 0.9584662
Variable   6 incorporated, RV = 0.9621404
Variable  14 incorporated, RV = 0.9679197
Variable  22 incorporated, RV = 0.9727763
Variable  24 incorporated, RV = 0.9773901
Variable  15 incorporated, RV = 0.981146
Variable  18 incorporated, RV = 0.9850263
Variable   8 incorporated, RV = 0.9882042
Variable  16 incorporated, RV = 0.9921904
Variable   2 incorporated, RV = 0.995268
Variable  12 incorporated, RV = 1

> summary(marbio.esc)
Escoufier's method of equivalent vectors for: marbio

Calculation stopped at level: 1
24 variable(s) calculated on a total of 24
RV:
          Copepodits3        ClausocalanusB           Copepodits1
            0.5938735             0.7607149             0.8280316
    EchinodermsLarvae   AdultsOfCentropages               Nauplii
            0.8620636             0.8981089             0.9039953
       ClausocalanusC          Copepodits5         Siphonophores
            0.9109498             0.9171680             0.9254146
              Acartia     GasteropodsLarvae           Copepodits2
            0.9324560             0.9399177             0.9469293
             Ostracods     EggsOfCrustaceans           Copepodits4
            0.9528440             0.9584662             0.9621404
               Oithona             Pteropods   BellsOfCalycophores
            0.9679197             0.9727763             0.9773901
           Acanthaires        DecapodsLarvae         ClausocalanusA
            0.9811460             0.9850263             0.9882042
           Cladocerans       AdultsOfCalanus JuvenilsOfCentropages
            0.9921904             0.9952681             1.0000000
```

Le graphique correspondant est simplement obtenu par ***plot()***. A ce stade, aucun seuil d'extraction n'est défini. On peut en définir un pour l'option de tracé (option **level**), mais nous verrons plus loin d'autres méthodes pour le définir dans un second temps. Il est aussi possible de tracer RV' (la dérivée) en plus de RV (**diff = TRUE**). Il s'agit d'ailleurs d'une option sélectionnée par défaut. Le traçage du graphique à ce stade avec toutes les options par défaut donne:

```
> plot(marbio.esc)
```



Afin d'éviter de surcharger le graphique, les différentes variables reprises en abscisse sont indiquées par un nombre correspondant à leur position dans le tableau de départ (comme dans PASSTEC 2000). Il est très facile d'afficher la correspondance entre ces index et les noms des variables par la commande suivante:

```
> marbio.esc$vr
        Copepodits3        ClausocalanusB          Copepodits1
                  5                     9                    3
  EchinodermsLarvae  AdultsOfCentropages              Nauplii
                 17                    11                   13
     ClausocalanusC          Copepodits5         Siphonophores
                 10                     7                   23
            Acartia    GasteropodsLarvae          Copepodits2
                  1                    19                    4
           Ostracods    EggsOfCrustaceans          Copepodits4
                 21                    20                    6
             Oithona             Pteropods   BellsOfCalycophores
                 14                    22                   24
         Acanthaires        DecapodsLarvae        ClausocalanusA
                 15                    18                    8
         Cladocerans       AdultsOfCalanus JuvenilesOfCentropages
                 16                     2                   12
```

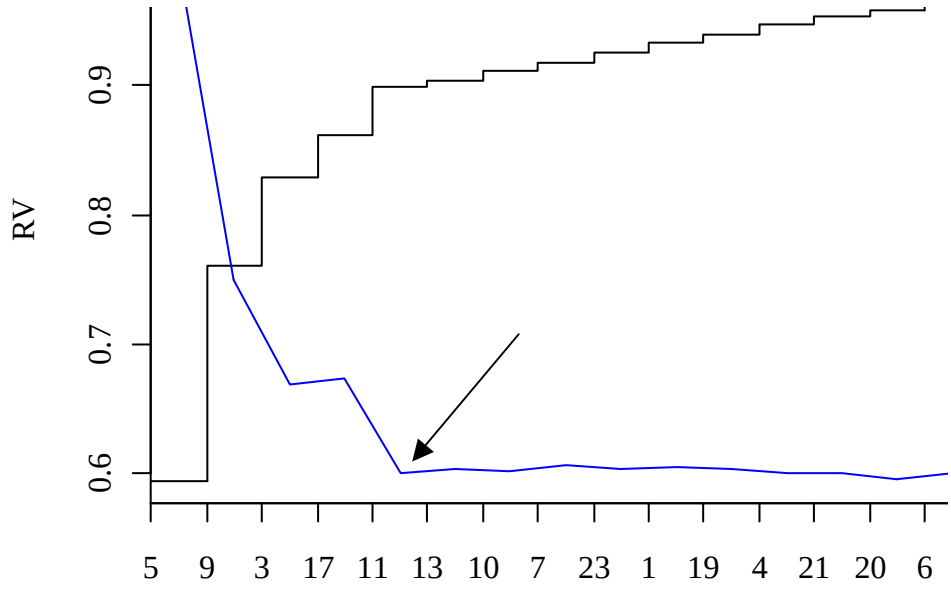
De même, on a accès aux RV par ***marbio.esc$RV*** et aux noms de variables classées par `names(marbio.esc$vr)`. Une fois le graphique construit, on peut y ajouter des repères qui indiquent l'endroit où l'on décide d'effectuer la coupure (extraction de toutes les variables à gauche, qui totalisent un RV inférieur ou égal au seuil). La méthode ***lines()*** permet de réaliser cela, soit en précisant **level** dans ses options, soit après avoir défini ***marbio.esc$level***, comme suit:

```
> marbio.esc$level <- 0.95
```

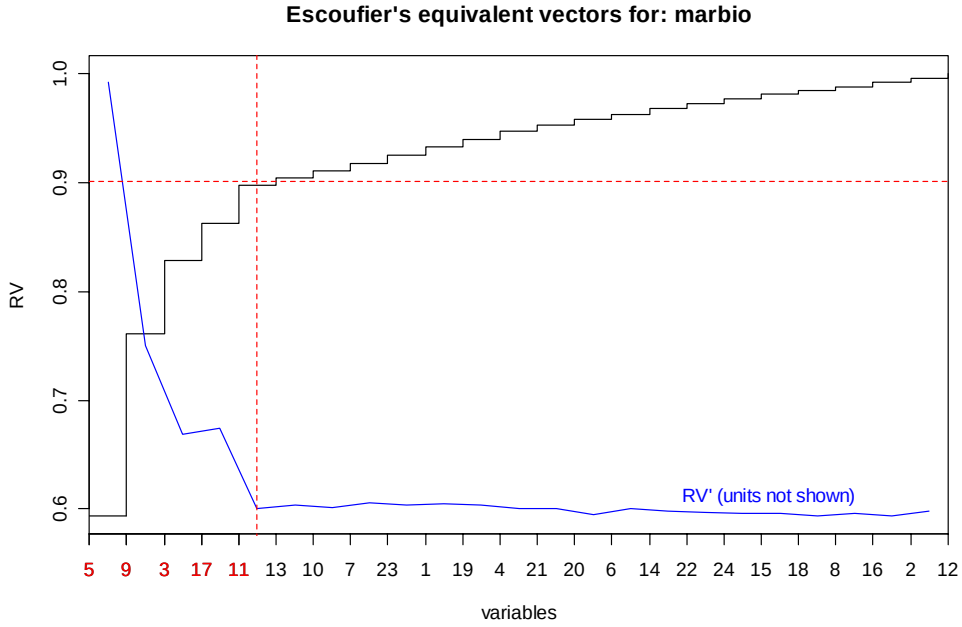
Il est cependant plus facile à ce stade d'utiliser la méthode ***identify()*** pour pointer directement sur le graphique la position voulue de la coupure, soit sur la courbe RV, soit si elle existe sur la courbe RV'. Le summum étant d'attribuer le résultat de cette opération directement à ***marbio.esc$level*** afin de l'enregistrer comme valeur par défaut pour l'extraction:

```
> marbio.esc$level <- identify(marbio.esc)
```

Le graphique est affiché et un pointeur de souris spécial apparaît pour indiquer qu'il faut cliquer un point sur le graphique. L'endroit préféré pour le seuil est, dans ce cas-ci, clairement visible sur la courbe RV'. Cliquez avec le bouton gauche de la souris à l'endroit indiqué par la flèche sur le graphique (zoom sur la partie concernée):



Le seuil correspondant est calculé et le graphique est complété comme suit (comme si la méthode ***lines()*** avait été appelée avec le même seuil comme argument):



Afin d'accroître la lisibilité, les variables qui seraient extraites à ce seuil sont également indiquées dans la couleur du trait (il est possible d'éviter cela en précisant l'option **lvars = FALSE**).

> *Si les méthodes **lines()** ou **identify()** sont appelées plusieurs fois de suite, la ou les lignes précédentes ne sont pas effacées. Cela permet de tester et de représenter éventuellement plusieurs lignes de seuil différentes sur un même graphe. Pour obtenir un graphique avec seulement la dernière ligne de seuil après tâtonnement, il suffit d'appeler la méthode **plot()** une fois de plus.*

Enfin, on peut extraire le tableau correspondant à la sélection des variables rangées selon la méthode d'Escoufier, à ce seuil, et l'introduire dans une nouvelle variable ***marbio2*** à l'aide de l'instruction suivante:

```
> marbio2 <- extract(marbio.esc)
> names(marbio2)
[1] "Copepodits3"         "ClausocalanusB"      "Copepodits1"
[4] "EchinodermsLarvae"   "AdultsOfCentropages"
```

Comme on peut le constater (à l'aide de la dernière instruction ***names()***), le tableau ***marbio2*** ne contient plus que les 5 variables qui sont le plus fortement corrélées avec les axes principaux de l'ACP, et ce, jusqu'au seuil choisi de corrélation (environ 0.90). En outre, les variables extraites sont également classées par ordre d'importance d'après l'analyse d'Escoufier. La méthode ***extract()*** permet également de redéfinir le seuil au moment de l'extraction (option **level**), ou d'indiquer une nombre fixe de variables à extraire (option **n**). Dans l'exemple présenté, nous concluons que seules 5 des 24 espèces initiales sont suffisantes pour représenter 90% de la variation totale observée.

Tri par ordre d'abondance

Lorsque la matrice représente des dénombrements d'espèces en différentes stations, certains descripteurs présentent beaucoup de valeurs nulles, et/ou très peu d'individus. Il peut être utile d'éliminer ces espèces mineures afin de réduire le nombre de descripteurs. Ibanez *et al.* (1993) utilisent une méthode basée sur le classement des descripteurs par ordre croissant selon le nombre de zéros présents. L'inconvénient de cette méthode est de ne pas faire ressortir les espèces rares (beaucoup de zéros), mais localement abondantes (dénombrements importants là où elles sont présentes). Le tri par abondance proposé dans PASTECS effectue un tri en fonction de deux critères:

1. Le nombre de valeurs nulles $N0_i$ par rapport au nombre d'observations totales N.

2. Le nombre total d'individus dénombrés NI_i dans l'ensemble des stations, exprimé en log et rapporté à une échelle variant entre 0 et 1 en divisant par le dénombrement le plus élevé NI_{max} (également en log).

Le tri, en valeurs croissantes, se fait sur le vecteur c suivant:

$$c = (1 - f).\frac{N0_i}{N} + f.\frac{\log(NI_i)}{\log(NI_{\max})}$$

Le coefficient f (non fixé *a priori*), strictement compris entre 0 et 1 permet de donner plus de poids à la fréquence des valeurs nulles (f proche de 0, méthode équivalente à Ibanez *et al*, 1993) ou à l'abondance des individus dénombrés (f proche de 1) dans le tri. Une valeur aux alentours de 0.2 pour f permet de récupérer à gauche les descripteurs avec peu de zéros (espèces abondantes), et à droite les descripteurs avec beaucoup de zéros, mais relativement abondantes (espèces rares, mais localement abondantes) que l'on peut extraire séparément. Entre les deux, on a d'une part les espèces moyennement abondantes (plus de 50-60% de valeurs nulles, accompagnées d'un dénombrement relativement faible), ainsi que les espèces peu significatives car présentant à la fois énormément de zéros et une abondance très faible et que l'on cherche à éliminer.

Afin d'aider à identifier la limite entre ces différents groupes, il est possible de faire un graphe des descripteurs classés en fonction de c croissants, et représentant à la fois le pourcentage de valeurs non nulles et le dénombrement des individus en log. Une courbe des différences cumulées entre ces deux courbes permet d'identifier facilement les limites entre les 4 groupes (voir l'exemple à la suite).

Référence:

Ibanez, F., J.-C. Dauvin & M. Etienne, 1993. Comparaison des évolutions à long terme (1977-1990) de deux peuplements macrobenthiques de la baie de Morlaix (Manche occidentale): relations avec les facteurs hydroclimatiques. *J. Exp. Mar. Biol. Ecol.*, 169:181-214.

Dans PASTECS, le tri des descripteurs par ordre d'abondance est calculé à partir de la fonction ***abund()***. Elle renvoie un objet spécifique de classe *'abund'* qui contient toutes les données nécessaires pour l'affichage d'un résumé de l'analyse (***summary()***), le traçage de graphique (***plot()***, ***lines()***, ***identify()***), ou pour extraire le résultat final du traitement dans un nouveau data frame (***extract()***). Son utilisation est donc très semblable à ***escouf()***.

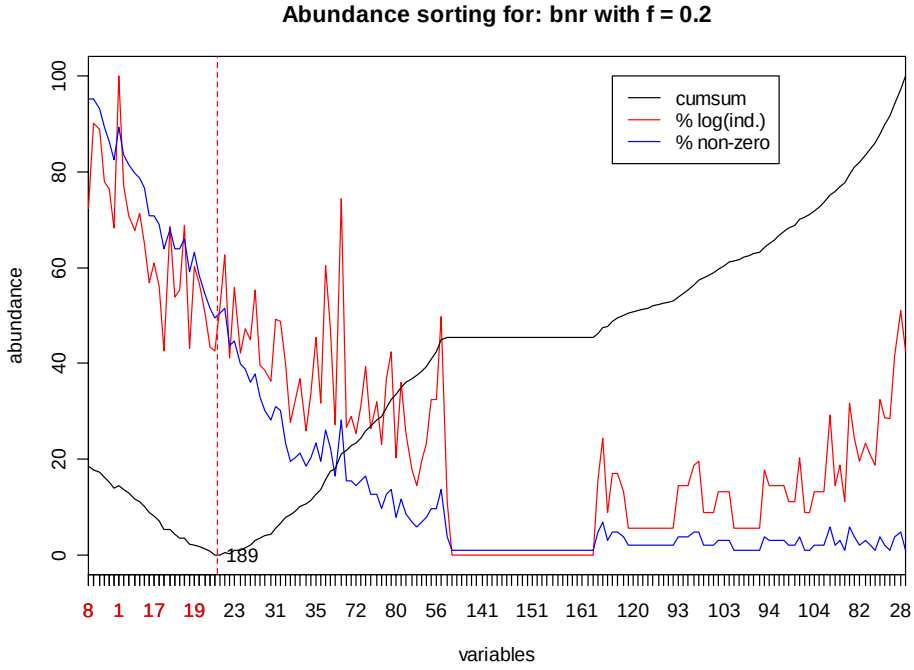
Exemple:

Nous traiterons ici un exemple contenant 163 espèces démonbrées en 103 stations différentes et nommé ***bnr***. Après l'avoir chargé, on peut effectuer le tri des descripteurs par ordre d'abondance à l'aide de la fonction ***abund()***:

```
> data(bnr)            # Dans R uniquement
> bnr.abd <- abund(bnr)
> summary(bnr.abd)

Sorting of descriptors according to abundance for: bnr

Coefficient f: 0.2
163 variables sorted

Number of individuals (% of most abundant in log):
        S8         S2         S3         S4         S6         S13
 72.273641  90.069317  88.739428  78.019800  76.302796  68.235798
        S1         S5        S10        S14         S9         S15
100.000000  77.095883  70.713663  67.746617  71.405852  65.019607
       S21        S17        S22        S39        S12         S26
 56.765809  61.013010  56.217342  42.731127  68.629878  53.892117
       S25        S11        S38        S19        S20         S29
 55.223940  68.848694  43.220902  60.065505  56.854955  50.145792
       S37        S41        S27        S16        S45         S23
 43.416565  42.561076  51.790497  62.798247  41.106427  55.899523
       S43        S33        S36        S24        S47         S49
 42.254638  47.328915  44.929107  55.322688  39.646573  38.667117
…

Percent of non-zero values:
        S8         S2         S3         S4         S6         S13
100.000000 100.000000  97.938144  93.814433  90.721649  86.597938
        S1         S5        S10        S14         S9         S15
 93.814433  87.628866  85.567010  83.505155  82.474227  80.412371
       S21        S17        S22        S39        S12         S26
 74.226804  74.226804  72.164948  67.010309  71.134021  67.010309
       S25        S11        S38        S19        S20         S29
 67.010309  69.072165  61.855670  65.979381  60.824742  56.701031
       S37        S41        S27        S16        S45         S23
 53.608247  51.546392  52.577320  53.608247  45.360825  46.391753
       S43        S33        S36        S24        S47         S49
 41.237113  40.206186  37.113402  39.175258  34.020619  30.927835
…
```

Le graphique correspondant est simplement obtenu par ***plot()***. Dans le cas présent, on peut décider de tracer les trois courbes (%logind, %nonnul et la somme cumulée, par défaut), ou bien uniquement la somme cumulée en précisant **all = FALSE**. La représentation du seuil d'extraction (le nombre *n* de variables les plus abondantes retenues) se fait de manière similaire à la méthode d'Escoufier, à l'aide des fonctions ***lines()*** ou ***identify()*** (remplacer dans le cas présent **level** par **n**). Notez qu'avec ***identify()***, il faut cliquer sur le point marqué "189" à sa droite sur le graphe (ce label n'apparaît pas sur le graphique). Par exemple:

```
> plot(bnr.abd, dpos=c(105,100))
> bnr.abd$n <- identify(bnr.abd)
Number of variables extracted: 26 on a total of 163
```



Afin d'éviter de surcharger le graphique, les différentes variables reprises en abscisse sont indiquées par un nombre correspondant à leur position dans le tableau de départ (comme pour la méthode d'Escoufier). Le nom des variables est obtenu par:

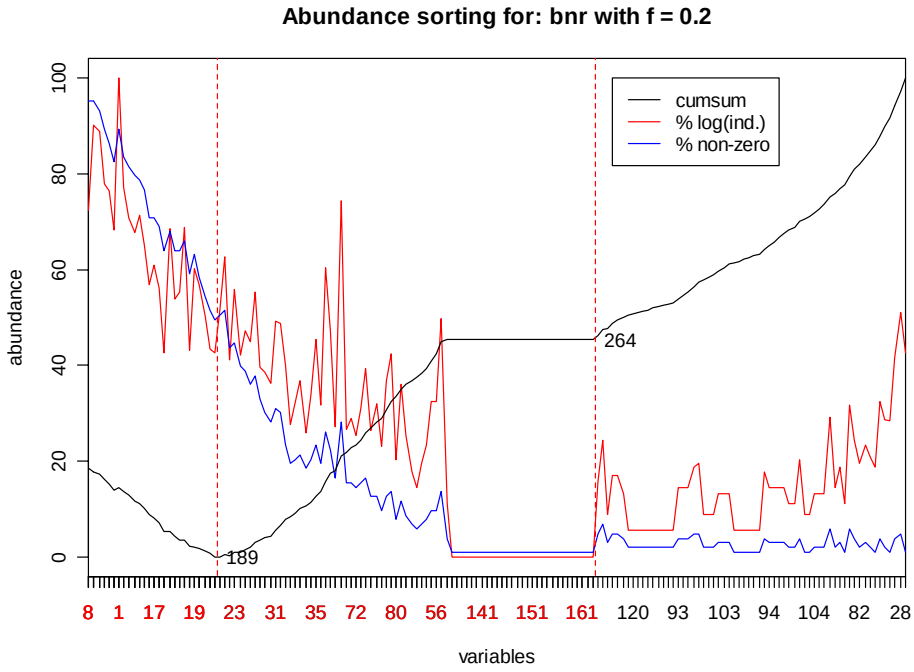
```
> bnr.abd$vr
  S8   S2   S3   S4   S6  S13   S1   S5  S10  S14   S9  S15  S21
   8    2    3    4    6   13    1    5   10   14    9   15   21
 S17  S22  S39  S12  S26  S25  S11  S38  S19  S20  S29  S37  S41
  17   22   39   12   26   25   11   38   19   20   29   37   41
 S27  S16  S45  S23  S43  S33  S36  S24  S47  S49  S52  S31  S32
  27   16   45   23   43   33   36   24   47   49   52   31   32
...
```

L'extraction de la fraction identifiée à gauche (les 26 descripteurs les plus abondants) se fait par:

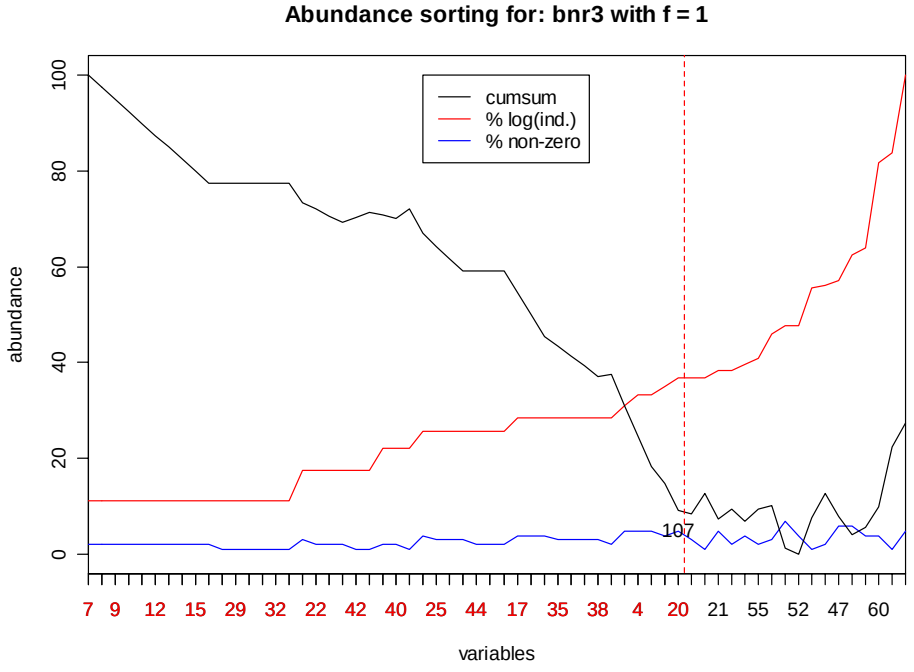
```
> bnr2 <- extract(bnr.abd)
> names(bnr2)          # Montre les variables extraites
[1] "S8"  "S2"  "S3"  "S4"  "S6"  "S13" "S1"  "S5"  "S10" "S14"
[11] "S9"  "S15" "S21" "S17" "S22" "S39" "S12" "S26" "S25" "S11"
[21] "S38" "S19" "S20" "S29" "S37" "S41"
```

On n'est toutefois pas tenu de conserver uniquement les plus abondants. On peut décider de conserver aussi tous les descripteurs moyennement abondants, donc jusqu'au plateau caractérisant les espèces rares. De même, on peut extraire séparément quelques-unes des espèces rares mais localement abondantes en indiquant une séparation après le plateau et en extrayant à droite (après avoir entré la commande ***identify()***, cliquez sur le point ayant le label "264" à sa droite, ce label n'apparaissant pas sur votre graphe). Généralement dans ce cas, on a intérêt à retrier par abondance (**f = 1**) afin de ne retenir au final que les descripteurs ayant une abondance non négligeable (de même, le label "107" à la droite du second graphe indique le point que vous aurez à cliquer):

```
> bnr.abd$n <- identify(bnr.abd)       # Identifie un second seuil
Number of variables extracted: 101 on a total of 163
```




```
> bnr3 <- extract(bnr.abd, left=FALSE)   # Extrait à droite
> plot(bnr3.abd <- abund(bnr3, f=1), dpos=c(26,100))
> bnr3.abd$n <- identify(bnr3.abd)       # Choisi le seuil
Number of variables extracted: 45 on a total of 62
> bnr4 <- extract(bnr3.abd, left=FALSE)  # Extrait à nouveau
> names(bnr4)
 [1] "S87" "S86" "S83" "S82" "S81" "S79" "S76" "S74" "S75" "S66"
[11] "S65" "S63" "S60" "S57" "S44" "S40" "S28"
```



Comme on peut le constater, les possibilités sont innombrables, en jouant sur la valeur du paramètre **f**, en sélectionnant une fraction, et en effectuant éventuellement un autre tri sur la fraction extraite.

Régularisation

Les méthodes présentées dans cette partie permettent de transformer une série spatio-temporelle échantillonnée avec un pas de temps irrégulier ou présentant des trous, en une série régulière. Cette régularisation est une opération préliminaire indispensable à beaucoup d'analyses de séries spatio-temporelles, que ce soit avec les outils PASTECS ou avec d'autres méthodes. Les objets *'ts'* ou *'rts'* doivent être *d'office* des séries spatio-temporelles régulières dans S+ ou R. Cette étape est donc requise pour adapter des *data frames* avant de pouvoir les transformer en time series! Plusieurs méthodes de régularisation sont disponibles dans PASTECS: régularisation par valeur constante (***regconst()***), linéaire (***reglin()***), par courbes splines (***regspline()***) et par la méthode des aires (***regarea()***). Afin de simplifier le travail de régularisation, et de faciliter ensuite la création des time series (voir conversion d'objets *'regul'* en séries régulières), toutes ces méthodes ont été regroupées sous une fonction commune ***regul()***.

> ***Attention!*** *Toutes ces méthodes sont à utiliser* ***avec précaution****. Il est très facile d'obtenir n'importe quoi si l'on choisit des mauvais paramètres! Il n'y a pas de règle absolue dans leurs choix, étant donné qu'ils sont liés à la nature-même des données à régulariser. Par exemple, il faut être extrêmement attentif à ne pas régulariser à tort des données physico-chimiques (par exemple à l'aide de la méthode des aires avec une très grande fenêtre).*

Quelle que soit la méthode utilisée, on doit veiller à obtenir un pas de temps de la série régulière aussi proche que possible du pas de temps (moyen) de la série échantillonnée. De même, on doit aussi essayer d'obtenir autant de valeurs non interpolées que possible dans la série régulière finale. Quant à l'extrapolation (obtenue à l'aide de l'argument **rule = 2**), elle est à proscrire. La fonction ***regul()*** admet beaucoup de paramètres. L'utilisateur est invité à essayer différents réglages sur ses propres données, et à visualiser le résultat à l'aide de ***plot.regul()*** et ***hist.regul()*** pour se familiariser avec ses possibilités.

Dans le cas où l'on voudrait condenser des données aléatoires (par exemple, transformer un échantillonnage mensuel en valeurs saisonnières ou des données quotidiennes en série par semaine ou par mois), il vaut mieux condenser les données en calculant des moyennes ou des médianes successives. On peut aussi régulariser la série avec le pas temps initial si ce dernier est un multiple du pas de temps visé, et utiliser ensuite la fonction ***aggregate()*** du langage S, présentée dans le chapitre suivant.

Choix du pas de temps pour la régularisation

Dans le cas de séries initiales très irrégulières, les fonctions ***regul.screen()*** et ***regul.adj()*** permettrent de déterminer quels sont les meilleurs paramètres temporels pour la série régularisée afin qu'un maximum d'observations de la série initiale coïncident avec les valeurs de temps échantillonnées pour la série régularisée. Une fenêtre de tolérance pour déterminer si les valeurs coïncident est utilisée. Celle-ci est calculée en interne à l'aide de la fonction ***match.tol()*** qui ne sera donc en principe pas utilisée directement. Un exemple d'utilisation de ces fonctions est donné dans le paragraphe suivant.

Fonction générale de régularisation regul()

Dans PASTECS, la régularisation d'une ou plusieurs séries se fait grâce à la fonction ***regul()*** qui renvoie un objet *'regul'* (en fait, une liste contenant un *data frame* qui représente la ou les séries régularisées, accompagnées d'un certains nombres d'attributs supplémentaires qui permettent le diagnostic du traitement, ainsi que l'extraction aisée des objets time series *'ts'* ou *'rts'* à partir de cette matrice à l'aide de ***extract()*** ou ***tseries()***. La méthode ***summary()*** résume le résultat de la régularisation. La méthode ***specs()*** récupère les spécifications d'un objet *'regul'* pour éventuellement les appliquer à une nouvelle régularisation. La méthode ***plot()*** permet de réaliser facilement le graphe superposé de la série initiale et de la série finale à des fins de comparaison. La méthode

lines() permet de superposer à un graphe déjà tracé une autre série régularisée (par exemple pour comparer deux méthodes de régularisation entre elles). La méthode ***identify()*** permet de sélectionner des points sensibles lors de la régularisation directement sur le graphe. La méthode ***hist()*** (sous S+, il s'agit de la fonction ***hist.regul()***) permet de représenter les points qui correspondent entre la série originale et la série régularisée, en fonction de la tolérance choisie dans le temps pour déclarer que les données sont équivalentes (voir description de cette fonction). Pour l'utilisateur avancé, les fonctions de régularisations respectives ***regconst()***, ***reglin()***, ***regspline()*** et ***regarea()*** sont aussi disponibles mais ne prennent pas en charge tout le traitement annexe de diagnostic et d'extraction de *time series*. Par contre, ces dernières fonctions peuvent s'avérer plus pratiques dans des scripts et programmes personnels.

Dans la plupart des logiciels, le temps est exprimé par rapport à une date de départ arbitraire (par exemple, le 1er janvier 1960 par défaut dans S+, ou le 1er janvier 1970 par défaut dans R). Les dates antérieures à cette valeur de départ sont exprimées par des valeurs négatives. Les dates sont manipulées en interne comme des nombres décimaux classiques. La partie entière de la date représente le jour, la partie décimale représente l'heure. Par exemple, le nombre 1.5 représente le 2 janvier 1960, à 12:00:00 (le « second jour et demi » par rapport à la date de départ qui vaut 0) dans S+, si l'origine temporelle n'a pas été redéfinie. Il est possible de connaître l'origine temporelle utilisée sur votre système grâce à la function ***options()*** (ou ***GetOption()*** de préférence sous R), et la modifier en appelant la même fonction:

```
> options("chron.origin")           # Regarde sa valeur
$chron.origin
NULL

> options(chron.origin = c(month=1, day=1, year=1990))
> options("chron.origin")
$chron.origin
month   day  year
    1     1  1990

> options(chron.origin = NULL)      # Reset default value
> options("chron.origin")
$chron.origin
NULL
```

Lorsque **chron.origin = NULL**, cela signifie que c'est la valeur par défaut du 01/01/1960 dans S+ ou du 01/01/1970 dans R qui est prise en compte.

> ***Attention!*** *Si vous définissez des dates, et puis que vous modifiez* **chron.origin** *entre deux calculs, les mêmes variables risquent de se référer à des dates différentes avant et après le changement! Pour éviter cela, et aussi pour prendre en compte les fuseaux horaires et les horaires été/hiver, R utilise un format plus complexe de dates appelé 'POSIXt'. Nous invitons le lecteur intéressé à consulter l'aide en ligne sous R à la rubrique* `?DateTimeClasses`.

Les séries temporelles dans S+/R admettent toute autre représentation arbitraire du temps en mode décimal. On peut par exemple décider de décompter les secondes écoulées à partir d'un temps de référence à l'aide de la partie entière du nombre (dizièmes, centièmes, etc. de secondes pour la partie décimale). Cela pourra être renseigné par l'argument **units** de ***regul()*** dans lequel on entrera **"sec"** au lieu de **"days"** par défaut. L'utilisateur peut donc se référer à n'importe quelle échelle de temps, avec une unité au choix qu'il peut préciser dans l'argument **units**. Cependant, il est conseillé de s'en tenir aux unités reconnues par les fonctions de la libraire PASTECS, parce que cela permettra notamment un formattage idéal des unités dans les sorties graphiques à l'aide de la fonction ***GetUnitText()*** (utilisée en interne, et donc, normalement pas appelée directement par l'utilisateur). Ces unités reconnues sont: **"years"**, **"days"**, **"weeks"**, **"hours"**, **"min"** et **"sec"**.

Une échelle particulièrement utile pour manipuler des séries pluriannuelles est **"years"**. Dans ce mode, la partie entière représente l'année, et la partie décimale représente le mois, le jour, l'heure, etc… Ainsi par exemple, 1998.167 représente le 1er mars 1998 à 00:00:00 heures, soit 1998 + 2/12 (0/12 pour début janvier, 1/12 pour début février, 2/12 pour début mars, …, 11/12 pour début décembre). En général, on travaille sur des données mensuelles dans ce mode, et on arrondit au mois près. Il s'agit en réalité d'une approximation puisque tous les mois n'ont pas le même nombre de jours (ni toutes les années), mais c'est une représentation pratique qui permet de se débarrasser des tracas liés aux nombres de jours par mois, aux années bissextiles, etc… Dans la fonction ***regul()*** il est possible de préciser **units = "daystoyears"** qui convertit automatiquement des dates classiques où l'unité est le jour dans le mode d'unité **"years"**. De telles séries, si le pas est mensuel sont automatiquement représentées par une table de Buys-Ballot à l'impression dans S+/R, ce qui est bien pratique.

> *A noter que, d'une manière générale, l'analyse de cycles bien précis tels que l'effet saisonnier ou les cycles circadiens nécessitent d'exprimer le temps dans une unité qui correspond exactement au cycle étudié. Ainsi, les fonctions de décomposition des séries en composantes saisonnières (**deccensus()**, **decloess()** par exemple, voir chapitre décomposition des séries spatio-temporelles) imposent d'exprimer le temps en unité* **"years"**. *De même, il est fortement conseillé d'adopté l'unité* **"days"** *pour l'étude de cycles circadiens.*

La fonction ***regul()*** offre encore une autre particularité. Elle permet de ne pas interpoler des données qui seraient présentes dans la série de départ, ***avec une fenêtre de tolérance réglable***. Ainsi par exemple, si une mesure est présente au jour 123, mais aucune au jour 124 dans la série initiale, et qu'il faut calculer une valeur dans la série régularisée au jour 124 (pas d'une semaine ou plus), on peut décider que la valeur échantillonnée au jour 123 est une très bonne estimation de la valeur au jour 124 et ne pas effectuer d'interpolation. La fenêtre dans laquelle il faut aller voir si une valeur mesurée existe est réglable par le paramètre **tol** de ***regul()***. De plus, on peut décider d'aller voir de chaque côté, uniquement à gauche, ou seulement à droite (voir paramètre **tol.type**). Ainsi, on peut accroître sensiblement le nombre de valeurs non interpolées dans la série régularisée, en considérant une certaine « souplesse » dans les dates entre les séries échantillonnées et régularisées. Si cela n'est pas simple, un exemple concret va clarifier les choses…

Exemple:

Chargeons le jeu de données ***releve*** à partir de la librairie PASTECS. Etant donné que cette série est très irrégulière comme on peut le voir ci-dessous, il est très difficile de décider quel est le meilleur pas de temps (**deltat**), le meilleur temps initial (**xmin**), ainsi que le nombre de mesures optimal (**n**) pour interpoler le moins de valeurs possibles:

```
> data(releve)          # Dans R uniquement
> releve$Day
 [1]    1   51  108  163  176  206  248  315  339  356  389  449
[13]  480  493  501  508  522  554  568  597  613  624  639  676
[25]  697  723  751  786  814  842  863  877  891  906  922  940
[37]  954  971  983  999 1010 1027 1038 1054 1069 1081 1094 1110
[49] 1129 1143 1156 1173 1186 1207 1226 1235 1249 1271 1290 1314
[61] 1325
> length(releve$Day)
[1] 61
> ecarts <- releve$Day[2:61]-releve$Day[1:60]
```

```
> ecarts
 [1] 50 57 55 13 30 42 67 24 17 33 60 31 13  8  7 14 32 14 29 16
[21] 11 15 37 21 26 28 35 28 28 21 14 14 15 16 18 14 17 12 16 11
[41] 17 11 16 15 12 13 16 19 14 13 17 13 21 19  9 14 22 19 24 11
> range(ecarts)
[1]  7 67
> mean(ecarts)
[1] 22.06667
```

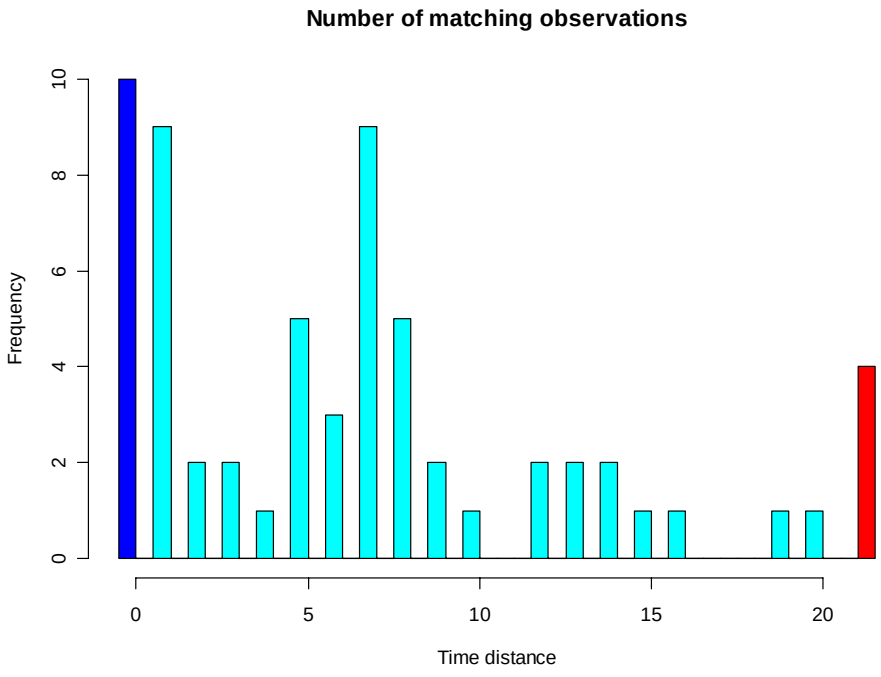
Nous choisirions *a priori* **xmin = 1** (la première valeur), **deltat = 22** (l'écart moyen arroundi à l'entier le plus proche) et **n = 61** (le nombre de valeurs observées dans la série initiale), mais est-ce la meilleure combinaison pour obtenir un maximum de valeurs qui coïncident entre la série de départ mesurée et la série régulière finale calculée? Sachant que l'on est prêt à accepter une fenêtre de tolérance de 1 jour dans le temps de part et d'autre de la valeur régularisée (prendre un peu plus, car les fenêtres de tolérance n'incluent pas les bornes), il est possible de « screener » différentes combinaisons de **xmin** et de **deltat** autour de ces premières approximations à l'aide de la fonction ***regul.screen()***:

```
> regul.screen(releve$Day, xmin=0:11, deltat=16:27, tol=1.05)


$tol
d=16 d=17 d=18 d=19 d=20 d=21 d=22 d=23 d=24 d=25 d=26 d=27
1.07 1.06 1.06 1.05 1.05 1.05 1.05 1.04 1.04 1.04 1.04 1.04

$n
     d=16 d=17 d=18 d=19 d=20 d=21 d=22 d=23 d=24 d=25 d=26 d=27
x=0    83   78   74   70   67   64   61   58   56   54   51   50
x=1    83   78   74   70   67   64   61   58   56   53   51   50
x=2    83   78   74   70   67   64   61   58   56   53   51   50
x=3    83   78   74   70   67   63   61   58   56   53   51   49
x=4    83   78   74   70   67   63   61   58   56   53   51   49
x=5    83   78   74   70   67   63   61   58   56   53   51   49
x=6    83   78   74   70   66   63   60   58   55   53   51   49
x=7    83   78   74   70   66   63   60   58   55   53   51   49
x=8    83   78   74   70   66   63   60   58   55   53   51   49
x=9    83   78   74   70   66   63   60   58   55   53   51   49
x=10   83   78   74   70   66   63   60   58   55   53   51   49
x=11   83   78   74   70   66   63   60   58   55   53   51   49

$nbr.match
     d=16 d=17 d=18 d=19 d=20 d=21 d=22 d=23 d=24 d=25 d=26 d=27
x=0     9   12   13    6    8    5    5    5   10   12   10   12
x=1    10   15   12    6    8    9    5    9   11    9    7   12
x=2    13    9   11   10   11   11    8   11   14    9    8   13
x=3    13   10   10    7    9   12   10   11   13    6    8    7
x=4    14    9    8   10    7    9   11    7   11    5    7    9
x=5    10   11    6    6    7    6    8    6    8   10    5    5
x=6    12   13    4    7    7    1    7    8    5    7    6    5
x=7     9   11    5    8   11    4    9    6    6    9    7    2
x=8    10   11    9   11   11   14   12    6    4    4    7    5
x=9    13   11    9   13   12   19    9    6    4    6    4    3
x=10   14   10   13   13   13   17    7   10    4    5    5    7
x=11   14    9   14   12    8    9    3    9    5    5    7    6

$nbr.exact.match
     d=16 d=17 d=18 d=19 d=20 d=21 d=22 d=23 d=24 d=25 d=26 d=27
x=0     3    7    4    1    2    1    2    2    2    1    2    3
x=1     3    3    6    4    3    3    2    1    5    7    3    7
x=2     4    5    2    1    3    5    1    6    4    2    2    2
x=3     6    1    3    5    5    3    5    4    5    0    3    4
x=4     3    4    5    1    1    5    4    1    4    4    3    2
```

```
x=5     5    4    0    4    1    1    2    2    2    1    1    3
x=6     2    3    1    1    5    0    2    3    2    5    1    0
x=7     5    6    3    2    2    0    4    3    2    1    4    2
x=8     2    2    1    5    4    4    3    0    2    3    2    0
x=9     3    3    5    4    5   10    5    3    0    0    1    3
x=10    8    6    3    4    3    5    1    3    2    3    1    0
x=11    3    1    5    5    5    2    1    4    2    2    3    4
```

Le premier tableau ***$tol*** renvoie la tolérance pour chaque deltat *d*. Le second tableau indique pour chaque combinaison de deltat *d* et de xmin *x*, quelle est la valeur maximale acceptable pour *n* sans devoir faire d'extrapolation. Les deux tableaux suivants renvoient, pour les mêmes combinaisons *d versus x*, respectivement le nombre d'observations qui coïncident dans la fenêtre de tolérance, et le nombre d'observation qui coïncident exactement. Nous cherchons à optimiser ces deux critères. Clairement, la combinaison **xmin = 9**, **deltat = 21**, **n = 63**, avec **tol = 1.05** renvoie le meilleur résultat puisque 10 observations coïncident parfaitement et 19 sont incluses dans la fenêtre de tolérance. Cela signifie que nous n'aurons pas à interpoler 19/63, soit un peu plus de 30% des valeurs dans la série régulière. Nous pouvons ensuite visualiser l'effet d'une modification de la fenêtre de tolérance dans le nombre d'observations qu'il ne faudra pas interpoler à l'aide de ***regul.adj()***:

```
> regul.adj(releve$Day, xmin=9, deltat=21)
$params
  xmin      n deltat    tol
     9     63     21     21

$match
[1] 59

$exact.match
[1] 10

$match.counts
 0  1  2  3  4  5  6  7  8  9 10 12 13 14 15 16 19 20 Inf
10 19 21 23 24 29 32 41 46 48 49 51 53 55 56 57 58 59 63
```



Cette fonction, par défaut, retourne des résultats chiffrés et dessine un histogramme de la distribution des valeurs non interpolées en fonction d'une fenêtre de tolérance croissante. La première barre en bleu foncé représente le nombre de valeurs qui coïncident exactement, et la dernière barre rouge représente le nombre de valeurs qui doivent être interpolées quelle que soit la taille de la fenêtre de tolérance au maximum égale à **deltat** (c'est-à-dire, les trous dans la série) On voit que dans le cas présent, il n'est effectivement pas utile d'augmenter la fenêtre de tolérance d'un jour ou deux, car seul un petit nombre de valeurs seraient rajoutées. Il faudrait ouvrir cette fenêtre à 5-7 jours de part et d'autre de la date de mesure pour augmenter significativement le nombre de valeurs qui coïncident, mais nous pouvons considérer qu'une telle fenêtre est trop large par rapport à un pas de temps de 21 jours (cela dépend en fait de la nature des données traitées!).

> *Il faut noter que comme les fonctions **regul.screen()** et **regul.adj()** ne manipulent que le vecteur temps, elles ne tiennent pas compte d'éventuelles valeurs manquantes dans la/les série(s) à régulariser. Si ces valeurs manquantes correspondent aux valeurs qui coïncident, on perd l'intérêt de ce « screening ». Ces fonctions sont donc plutôt adaptées aux séries très irrégulières, mais avec peu de trous. De toute manière, elles sont totalement superflues pour des séries régulières à trous (le pas de temps est déjà fixé et connu!), et…on est bien mal embarqué si l'on possède des séries irrégulières à trous au départ!!!*

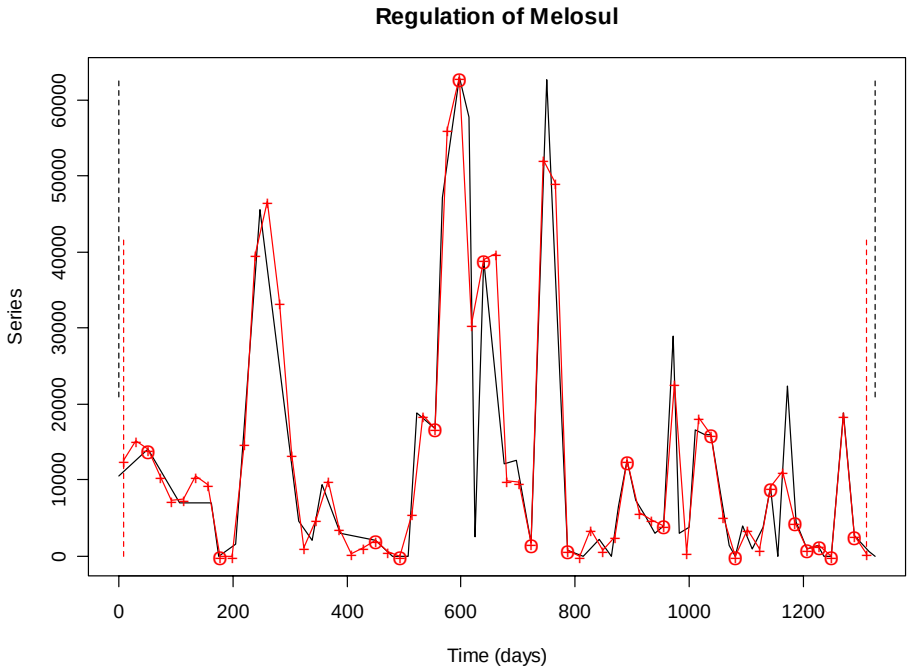
Maintenant que l'on a déterminé les paramètres temporels optimaux, il ne nous reste plus qu'à effectuer la régularisation de ***releve*** (dans notre exemple, les 6 premières séries qui suivent les premières colonnes ***releve$Day*** et ***releve$Date***, donc, les colonnes 3 à 8). Pour cela, nous devons choisir une éventuelle transformation des données (***log()***, ou autre…), une méthode de régularisation (éventuellement différente pour chaque série), ainsi que les paramètres de régularisation (tel que la taille de la fenêtre pour la méthode des aires, ou la valeur de **f** pour la régularisation constante). Une fois notre choix effectué, nous lançons la régularisation par ***regul()*** avec tous les paramètres choisis:

```
> rel.reg <- regul(releve$Day, releve[3:8], xmin=9, n=63,
+ deltat=21, tol=1.05, methods=c("s","c","l","a","s","a"),
+ window=21)
> rel.reg

Regulation of, by "method" :
   Astegla       Chae       Dity       Gymn    Melosul       Navi
  "spline" "constant"   "linear"     "area"   "spline"     "area"

Arguments for "methods" :
tol.type      tol     rule        f periodic   window    split
  "both"   "1.05"      "1"      "0"  "FALSE"     "21"    "100"

 44 interpolated values on 63 ( 0 NAs padded at ends )

Time scale :
      start      deltat   frequency
 9.00000000 21.00000000  0.04761905
Time units : days

call : regul(x = releve$Time, y = releve[3:8], xmin = 9, n = 63,
deltat = 21,      tol = 1.05, methods = c("s", "c", "l", "a", "s",
"a"), window = 21)
```

Ainsi, 44 valeurs sont interpolées pour chaque série à l'aide de sa méthode respective, mais cela nous le savions déjà puisque les paramètres de temps pour les séries régulières ont été déterminés précédemment afin que 19 valeurs coïncident dans la fenêtre de tolérance choisie. De même, aucune valeur n'a été extrapolée (aucun **NA** n'a dû être ajouté si **rule = 1** comme ici) puisque **n** a été choisi judicieusement pour ne pas déborder

de la série initiale. Il est possible d'avoir également la liste des points interpolés et leur valeur pour chaque série à l'aide de `summary(rel.reg)`. La représentation graphique de n'importe laquelle de ces séries, superposée à la série de départ correspondante est simplement obtenue par ***plot()*** en précisant le numéro de la série à représenter:

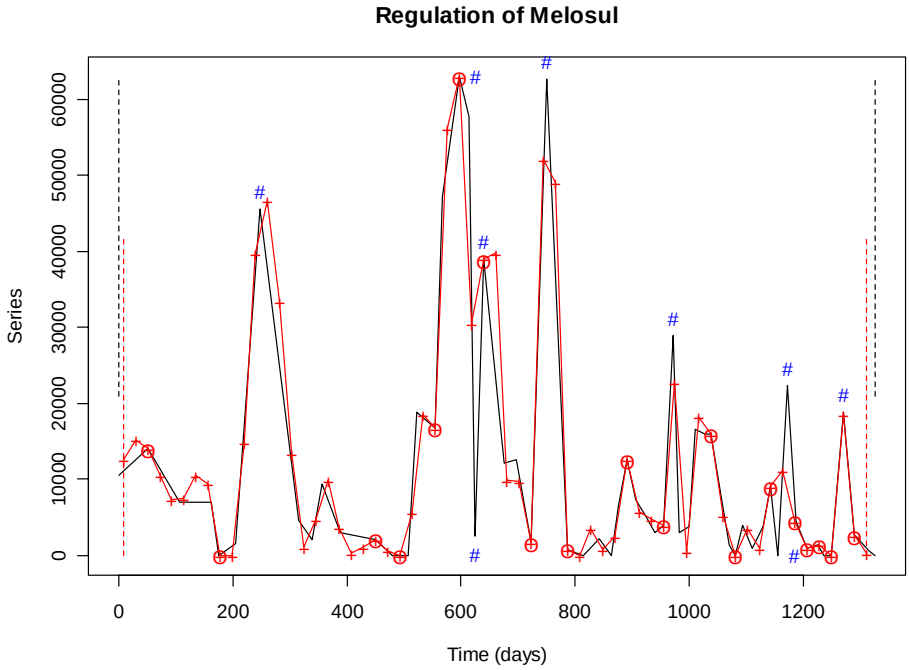
```
> plot(rel.reg, 5)
```



Les points interpolés sont représentés par une croix et les points qui coïncident à la série initiale (dans la fenêtre de tolérance) sont indiqués par une croix entourée d'un cercle. Les limites inférieures et supérieures des deux séries sont indiquées par des traits pointillés verticaux de couleur correspondante.

Les paramètres temporels et/ou la méthode de régularisation par les courbes splines ne permettent pas de conserver certains pics ou creux dans la série régularisée (à 630 jours et vers 1180-1190 jours). En général, on cherchera à les conserver au mieux, sauf si l'on peut considérer qu'il s'agit d'une fluctuation aléatoire qui n'apportera rien à l'analyse. Dans ce dernier cas, on pourra alors préférer une méthode de régularisation qui lisse également les données, comme la méthode des aires. On a intérêt à essayer plusieurs combinaisons de paramètres (toujours en s'aidant des tableaux renvoyés par ***regul.screen()*** pour choisir celles qui paraissent les plus favorables). On peut faire mieux aussi. Après un premier essai de régularisation, on peut appliquer la méthode ***identify()*** de l'objet *'regul'* obtenu afin de pointer sur le graphique les zones sensibles qui sont (potentiellement) mal représentées après régularisation. Le vecteur obtenu peut être utilisé directement ou après transformation (voir la suite de l'exemple ci-dessous) pour pondérer les résultats renvoyés par ***regul.screen()*** (argument **weight**). Ainsi, nous privilégierons les combinaisons de **xmin** et **deltat** qui font coïncider un maximum de points sensibles entre la série initiale et la série régularisée. Nous pouvons créer un vecteur qui donne un poids 5 aux valeurs sensibles et un poids 1 aux autres comme suit:

```
> weight <- identify(rel.reg, 5, col=4)*4+1
```




```
> weight
[1]  1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 5 1 5 5 1 1 1 5 1 1 1 1
[32] 1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 5 5 1 1 1 1 1 5 1 1 1
```

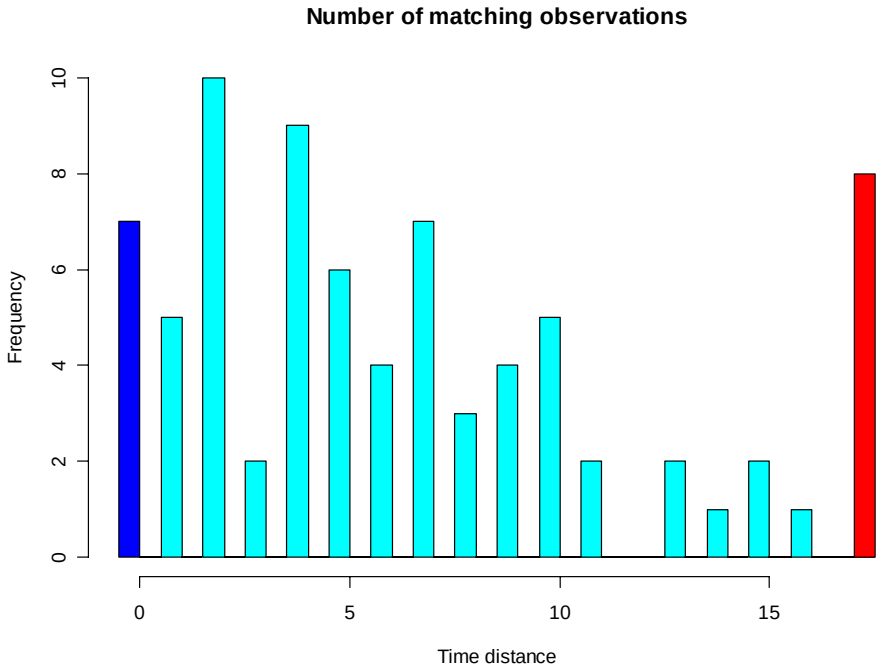
Lorsque la première ligne de commande est entrée, le programme attend que l'utilisateur clique un ou plusieurs points sur le graphe. Chaque point reconnu est identifié par un symbole '#' (systématiquement à droite du points dans S+, dans la direction du curseur dans R). On sélectionnera les pics ou les creux importants qui sont entourés par des valeurs très différentes dans la série initiale. Ce sont ces pics et creux qui sont susceptibles de disparaître lors de la régularisation si le pas de temps ne tombe pas dessus (ou en tout cas, très près). Une fois que l'on a sélectionné tous les points désirés, on clique avec le bouton droit de la souris sur le graphe (dans R, on sélectionne ensuite ***Stop*** dans le menu). A noter qu'un tel vecteur de pondération, contenant **tous** les pics et creux de la série peut être obtenu automatiquement à l'aide de la méthode ***extract()*** de ***turnpoints()*** (voir chapitre Analyse des séries spatio-temporelles, points de retounement). Ensuite, on relance ***regul.screen()***, mais cette fois-ci, en utilisant le vecteur de pondération que l'on vient de créer:

```
> regul.screen(releve$Day, weight, xmin=-1:10, deltat=c(16:23),
+ tol=2.2)

$tol
d=16  d=17  d=18  d=19  d=20  d=21  d=22  d=23
2.286 2.125 2.250 2.111 2.222 2.100 2.200 2.300

$n
     d=16 d=17 d=18 d=19 d=20 d=21 d=22 d=23
x=-1   83   79   74   70   67   64   61   58
x=0    83   78   74   70   67   64   61   58
x=1    83   78   74   70   67   64   61   58
x=2    83   78   74   70   67   64   61   58
x=3    83   78   74   70   67   63   61   58
x=4    83   78   74   70   67   63   61   58
x=5    83   78   74   70   67   63   61   58
x=6    83   78   74   70   66   63   60   58
```

```
x=7    83   78   74   70   66   63   60   58
x=8    83   78   74   70   66   63   60   58
x=9    83   78   74   70   66   63   60   58
x=10   83   78   74   70   66   63   60   58


$nbr.match
     d=16 d=17 d=18 d=19 d=20 d=21 d=22 d=23
x=-1   28   27   25   32   20   21   22   17
x=0    29   38   24   24   16   17   19   21
x=1    31   38   34   20   20   21   27   27
x=2    27   40   36   20   18   25   30   22
x=3    33   29   28   23   17   28   30   22
x=4    32   33   23   16   19   21   30   32
x=5    37   26   24   13   18   17   37   29
x=6    37   23   14   17   24   14   26   21
x=7    33   22   14   20   24   27   27   23
x=8    28   32   17   24   27   27   27   20
x=9    33   26   25   28   39   33   22   13
x=10   31   26   32   33   37   35   19   12

$nbr.exact.match
     d=16 d=17 d=18 d=19 d=20 d=21 d=22 d=23
x=-1   11    3    7    1    7    1    1    6
x=0     7   15    8    1    2    1    2    6
x=1     3    3    6    8    3    7    6    1
x=2     4   13    2    5    3    5    1    6
x=3     6    5   11    5    5    7   17    8
x=4     7    4    9    1    5    5    4    1
x=5    13    4    0    4    1    5    2    6
x=6     2    7    1    1    5    0    6   11
x=7     9    6    3    2    2    0    8    3
x=8     6    2    1    9   12    4    7    0
x=9     3    3    9    4    5   18    5    3
x=10    8   14    3    8    3    5    1    3
```

Cette fois-ci, d'autres combinaisons de **xmin** et **deltat** ressortent. Nous retiendrons finalement **xmin = 0**, **deltat = 17**, **n = 78**, avec **tol = 2.125**. Ici, nous avons avantage à élargir la fenêtre de tolérance à deux jours, comme on peut le voir sur le graphe suivant:

```
> regul.adj(releve$Day, xmin=0, deltat=17)
$params
  xmin      n deltat    tol
     0     78     17     17

$match
[1] 70

$exact.match
[1] 7

$match.counts
 0   1   2   3   4   5   6   7   8   9  10  11  13  14  15  16 Inf
 7  12  22  24  33  39  43  50  53  57  62  64  66  67  69  70  78
```

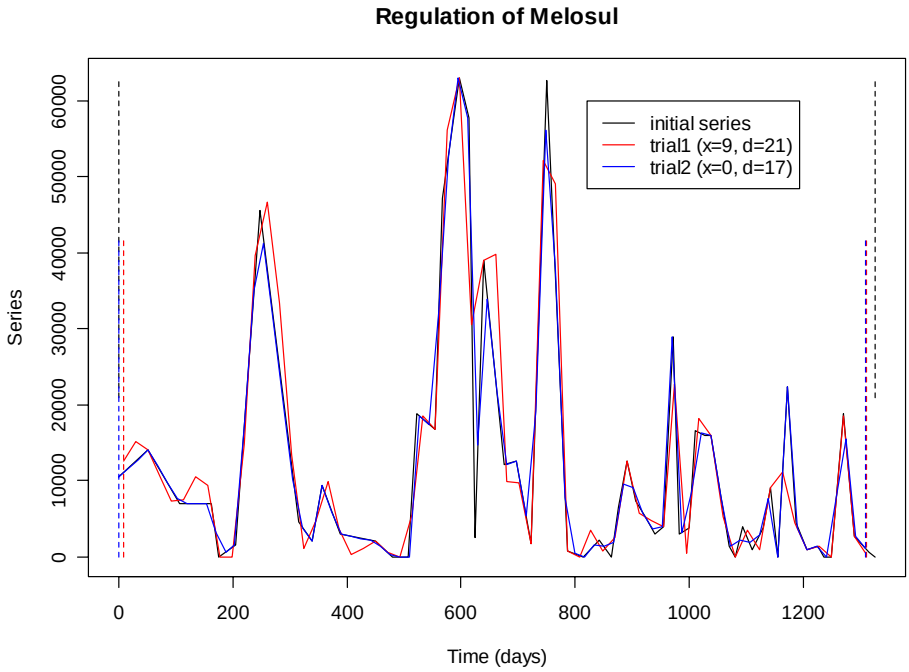


Pour la série ***Melosul*** représentée, nous avons finalement choisi une régularisation linéaire avec ces derniers paramètres:

```
> melo.reg <- regul(releve$Day, releve$Melosul, xmin=0, n=78,
+ deltat=17, tol=2.125, methods="linear")
```

Afin de comparer les deux options, on peut superposer cette dernière courbe au graphique précédent à l'aide de la méthode ***lines()*** (on n'affiche pas les points pour éviter de surcharger le graphe):

```
> plot(rel.reg, 5, plot.pts=FALSE)
> lines(melo.reg, col=4, plot.pts=FALSE)
> legend(820, 60000, c("initial series", "trial1 (x=9, d=21)",
+ "trial2 (x=0, d=17)"), col=c(1,2,4), lty=1)
```



Comme on peut le constater, cette dernière courbe régularisée (en bleu) suit beaucoup mieux les fluctuations de la série initiale irrégulière (en noir) que celle du premier essai (en rouge), en particulier, vers 630 et 1180-1190 jours. Naturellement, il y a encore deux critères que nous n'avons pas considérés ici: (1) les autres séries (il faut qu'une base de temps commune soit apte à représenter correctement TOUTES les séries, pas seulement une seule), et (2) des contraintes sur le pas de temps d'un point de vue pratique (par exemple, on veut conserver un pas de temps bi-mensuel ou mensuel au lieu de 17 ou 21 jours parce qu'on cherche à mettre en évidence des fluctuations saisonnières et que l'on ne veut pas de décalages du pas de temps d'une année à l'autre).

En particulier, nous n'avons pas respecté la dernière contrainte. Avec une série pluriannuelle comme celle-ci, nous voudrions certainement étudier d'éventuelles variations saisonnières sur la série régularisée. Or, beaucoup de fonctions de S+ ou R utilisables dans ce but (voir chapitre décomposition de séries) nécessitent que l'unité de temps corresponde à la durée du cycle, c'est-à-dire, **"years"** ici. Bien entendu, il faut au moins 3 ou 4 mesures par unité de temps pour pouvoir étudier le cycle. S+ et R imposent aussi que l'intervalle de temps soit un multiple exact de l'unité. Nous avons vu plus haut que, à plus ou moins un jour près, nous pouvons considérer qu'une année dure 365.25 jours. Toujours à un ou deux jours près dans la coupure à gauche et/ou à droite des mois, on peut considérer qu'un mois est équivalent à 1/12 d'une année, soit 365.25/12 = 30.44 jours. Si l'on veut s'en tenir à des valeurs bimensuelles ou « quinzaines », on doit donc considérer des intervalles **deltat** de 15.22 jours. L'utilisation de **tol** supérieur ou égale à 1 jour prend ici tout son sens! Ainsi, au lieu de 17 jours, nous serions contraints de prendre 15.22 jours comme intervalle pour pouvoir transposer le temps en **"years"**, avec une fréquence entière **frequency = 24** (24 « quinzaines » par an). Nous n'avons donc plus qu'un paramètre temporel libre pour ajuster au mieux la série régularisée afin d'interpoler le moins de points possibles. Les fonctions ***regul.screen()*** et ***regul.adj()*** nous renseignent toujours sur la valeur optimale de **xmin** dans ce cas précis (les résultats renvoyés ne sont pas imprimés):

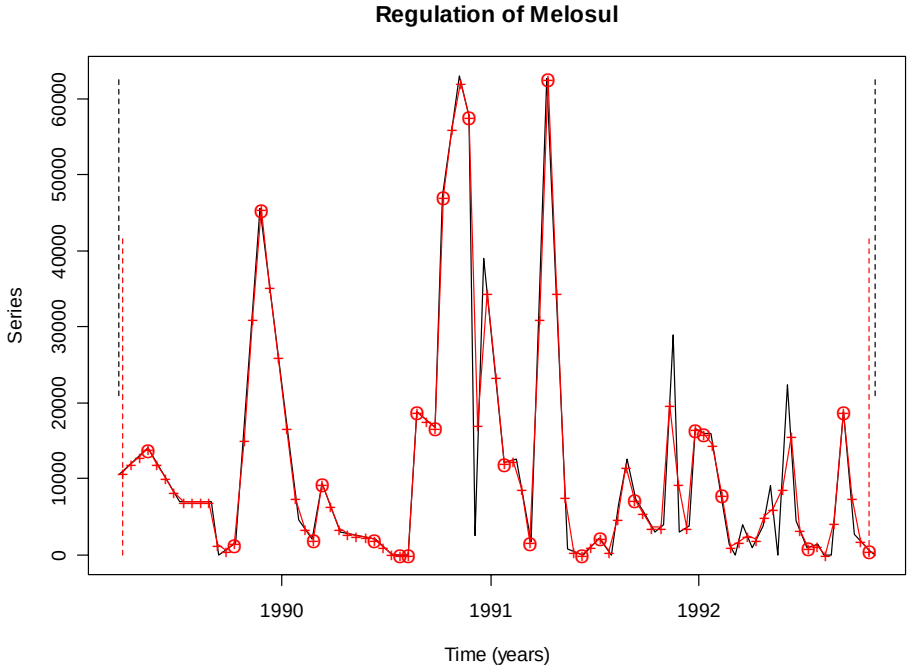
```
> regul.screen(releve$Day, weight, xmin=-1:14, deltat=365.25/24,
+ tol=2.2)
…
> regul.adj(releve$Day, xmin=6, deltat=365.25/24)
…
```

Nous retiendrons ainsi: **xmin = 6**, **n = 87** et **tol ≈ 2.2**. Avec ces valeurs, 24 observations sur les 87 dans la série régulière ne devront pas être interpolées, soit environ 27%. Nous préférons préciser **frequency = 24** à **deltat** (qui serait une fraction puisqu'il vaut 1/24 ≈ 0.0416667). La régularisation de la série ***Melosul***, avec une fréquence de 24 mesures par an exactement est donc obtenue par:

```
> melo.regy <- regul(releve$Day, releve$Melosul, xmin=6, n=87,
+ units="daystoyears", frequency=24, tol=2.2, methods="linear",
+ datemin="21/03/1989", dateformat="d/m/Y")

A 'tol' of 2.2 in 'days' is 0.00602327173169062 in 'years'
'tol' was adjusted to 0.00595238095238083
```

Le programme nous avertit que, puisque nous avons transformé l'échelle temporelle de "**days**" en "**years**" (argument **units="daystoyears"**), la valeur de **tol=2.2** qui était exprimée en jours devient 0.0602… lorsqu'elle est exprimée dans l'unité "**years**". D'autre part, **tol** doit aussi être une fraction entière de 1/**frequency** (c'est-à-dire de **deltat**, voir `?regul`). Or ce n'est pas le cas actuellement. La seconde ligne nous prévient donc que la valeur de **tol** a dû être ajustée à 0.00595 ans, soit 1/168 d'année, soit encore 1/168*365.25 jours, c'est-à-dire 2.174 jours. Nous savions que **tol** allait être ajustée, puisque nous n'avons fourni qu'une valeur indicative. On précisera **datemin = "21/03/1989"** ou **datemin = releve$Date[1]** afin de caler l'axe temporel correctement. Nous obtenons le résultat suivant:

```
> melo.regy
Regulation of, by "method" :
   Series
"linear"

Arguments for "methods" :
tol.type        tol  rule    f  periodic            window  split
  "both"  "0.00595”   "1"  "0"   "FALSE" "15.3953488372093"  "100"

 63 interpolated values on 87 ( 0 NAs padded at ends )

Time scale :
       start       deltat    frequency
2.109000e+03 4.166667e-02           24
Time units : years

call : regul(x = releve$Time, y = releve$Melosul, xmin = 6, n =
87,      units = "daystoyears", frequency = 24, datemin =
releve$Date[1],      tol = 2.2, methods = "linear")

> plot(melo.regy, main="Regulation of Melosul")
```



On fait un tout petit peu moins bien qu'avec notre pas de temps de 17 jours dans la « capture » des pics et des creux, mais cette fois-ci, on obtient une échelle temporelle utilisable pour étudier les variations saisonnières. En fin de compte, la régularisation retenue sera un compromis entre les meilleures représentations pour les différentes séries et les contraintes imposées. Une fois que l'on est satisfait du résultat obtenu, il est extrêmement facile de convertir l'objet *'regul'* issu de la régularisation en 'time series' régulière(s), c'est à dire, soit en objet *'rts'* sous S+, soit en objet *'ts'* sous R à l'aide de la fonction ***tseries()***:

```
> melo.ts <- tseries(melo.regy)
> is.tseries(melo.ts)
[1] TRUE
```

On peut aussi décider de n'extraire qu'une seule ou quelques séries régularisées contenues dans un objet *'regul'* qui en contient plusieurs à l'aide de la méthode ***extract()***:

```
> melo.ts2 <- extract(rel.reg, series="Melosul")
```

A partir d'ici, nous disposons de tous les outils de traitement de séries spatio-temporelles de S+/R, ainsi que des routines spécialisées de PASTECS pour continuer l'analyse: analyse spectrale, lissage, filtrage, décomposition, tendance générale ou saisonnière, etc. (voir chapitres suivants).

> *Les quatre fonctions suivantes (**regconst()**, **reglin()**, **regspline()** et **regarea()**) sont utilisées par **regul()**, mais sont aussi fournies séparément pour permettre à l'utilisateur avancé de réaliser ses propres routines de régularisation indépendamment de la fonction **regul()** et de l'objet 'regul' correspondant.*

Régularisation par valeur constante regconst()

Il s'agit de la méthode la plus simple de régularisation (mais aussi la moins puissante). Les valeurs interpolées X_j aux temps T_j sont calculées comme suit, si les valeurs mesurées qui encadrent X_j dans la série irrégulière initiale sont x_{i-1} et x_i aux temps t_{i-1} et t_i (i est choisi tel que $t_{i-1} < T_j \leq t_i$).

$$X_j = x_i . f + x_{i-1} . (1 - f)$$

avec f une constante comprise entre 0 et 1 et qui donne plus de poids à la valeur mesurée à gauche ($f < 0.5$), ou à droite ($f > 0.5$) de la valeur à interpoler. Avec $f = 1$ on a une fonction d'interpolation continue à gauche, avec $f = 0$ on a une fonction continue à droite et avec $f = 0.5$ on obtient la moyenne des deux valeurs encadrantes comme interpolation sur tout l'intervalle. Cette méthode est surtout utile avec $f = 0$, si la série de départ a été compressée avec un algorithme de type *RLE* (« Run-Length Encoding ») qui reprend chaque valeur et l'intervalle sur lequel elle reste constante, au lieu de considérer un pas de temps constant.

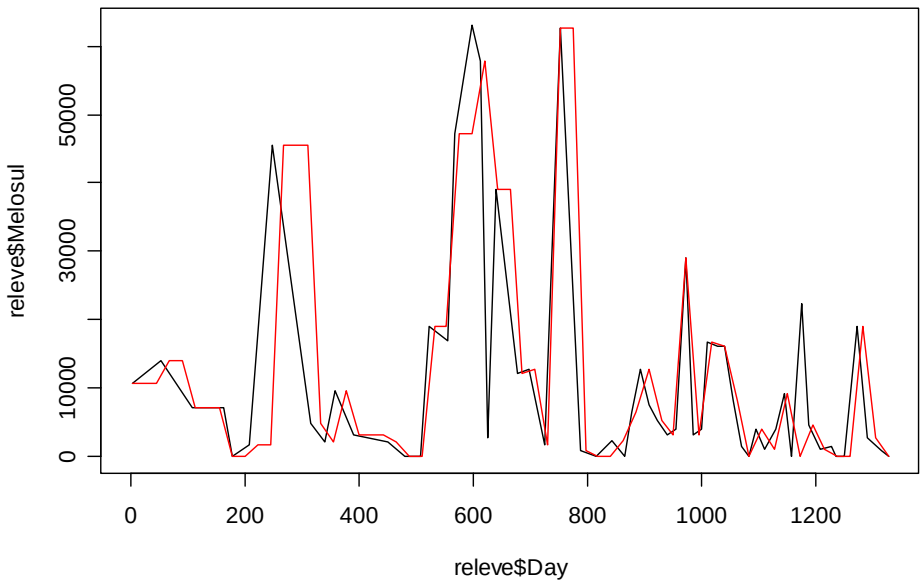
Exemple:

La série ***Melosul*** du jeu de données ***releve*** est régularisée avec la méthode de valeur constante comme suit:

```
> data(releve)          # Dans R uniquement
> reg <- regconst(releve$Day, releve$Melosul)
```

On peut tracer un graphique de la série de départ et de la série régularisée pour comparaison visuelle:

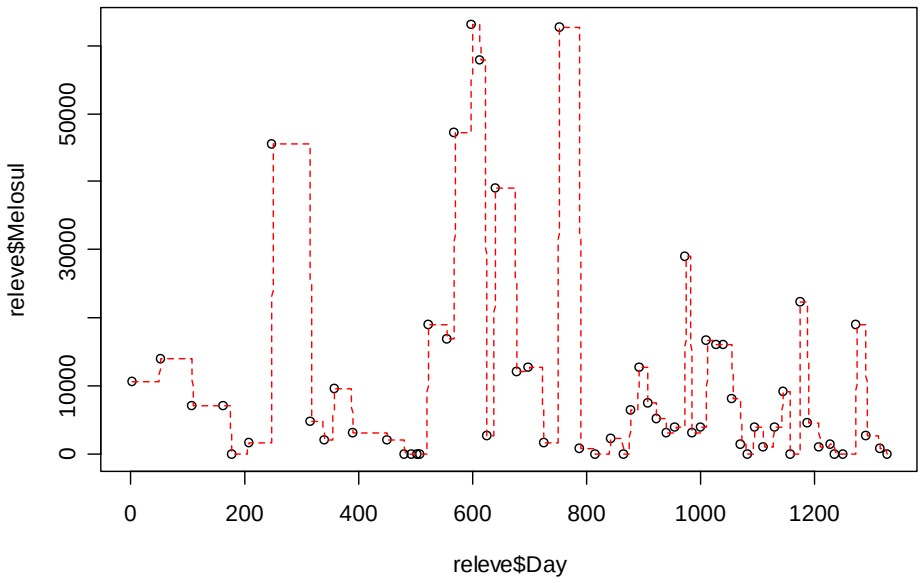
```
> plot(releve$Day, releve$Melosul, type="l")
> lines(reg$x, reg$y, col=2)
```



La série de départ apparaît en noir, et la série régularisée en rouge. Lorsqu'une interpolation est nécessaire, la valeur immédiatement à gauche dans la série de départ est utilisée. A noter toutefois que cette méthode est incluse dans ***regul()*** qui offre plus de

facilités pour le diagnostic et l'extraction des séries temporelles après régularisation. On peut visualiser les points de la série de départ superposés à la fonction continue calculée qui sert à la régularisation comme suit:

```
> plot(releve$Day, releve$Melosul, type="p")
> lines(regconst(releve$Day, releve$Melosul,
+ n=length(releve$Day)*10), col=2, lty=2)
```



Régularisation linéaire reglin()

Les points mesurés sont reliés par des segments de droite, et les valeurs interpolées sont obtenues à partir de la ligne brisée ainsi construite. L'équation qui donne les valeurs interpolées X_j au temps T_j à partir des valeurs encadrantes x_{i-1}, x_i aux temps respectifs t_{i-1}, t_i de la série irrégulière initiale est:

$$X_j = \frac{(t_i - T_j).x_{i-1} + (T_j - t_{i-1}).x_i}{t_i - t_{i-1}}$$

avec, pour tout $j = 1 \ldots p$, i est choisi tel que $t_{i-1} < T_j \leq t_i$.

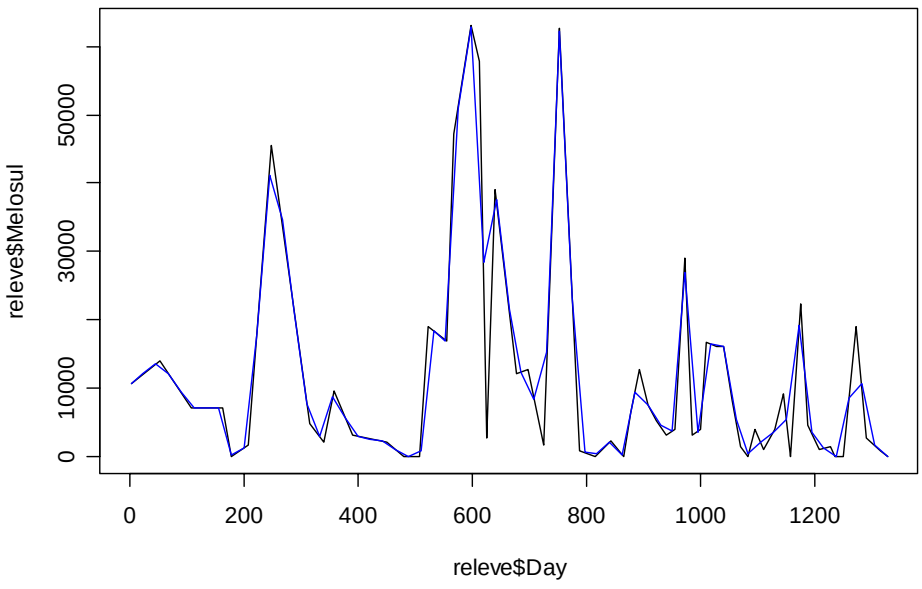
Exemple:

La série ***Melosul*** du jeu de données ***releve*** est régularisée avec la méthode linéaire comme suit:

```
> data(releve)          # Dans R uniquement
> reg <- reglin(releve$Day, releve$Melosul)
```

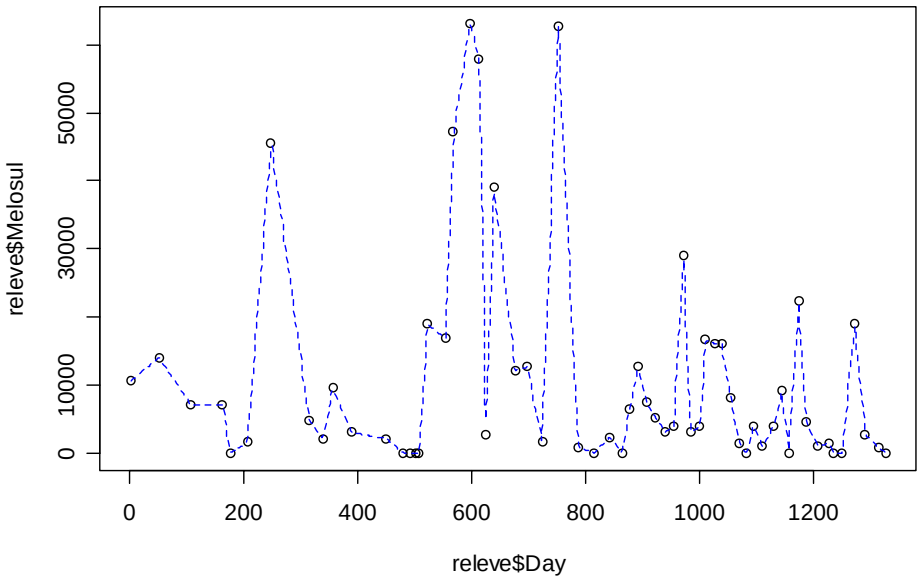
On peut tracer un graphique de la série de départ et de la série régularisée pour comparaison visuelle:

```
> plot(releve$Day, releve$Melosul, type="l")
> lines(reg$x, reg$y, col=4)
```



La série de départ apparaît en noir, et la série régularisée en bleu. L'interpolation est réalisée le long de la droite qui relie les deux valeurs encadrantes. Cette méthode est incluse dans ***regul()*** qui offre plus de facilités pour le diagnostic et l'extraction des séries temporelles après régularisation. On peut visualiser les points de la série de départ superposés à la fonction continue calculée qui sert à la régularisation comme suit:

```
> plot(releve$Day, releve$Melosul, type="p")
> lines(reglin(releve$Day, releve$Melosul,
+ n=length(releve$Day)*10), col=4, lty=2)
```


releve$Melosul
0
10000
30000
50000
0
200
400
600
800
1000
1200
releve$Day

Régularisation par courbes splines regspline()

L'ajustement par des fonctions splines consiste à définir un polynôme dont les valeurs aux points d'observation coïncident avec les valeurs de l'échantillon. Le problème d'interpolation entre p points, donnés par une fonction $g(x)$ dont les dérivées d'ordre 1 à k (avec $1 \leq k$) sont continues, peut être résolu en minimisant des fonctions du type:

$$S = \int_a^b [g^k(x)]^2$$

avec: $a < x_1 < \ldots < x_p < b$

Plus on est exigeant sur l'ordre k des dérivées que l'on souhaite continues ainsi que sur le degré du polynôme $g(x)$ choisi et plus l'expression de la fonction spline devient complexe. Nous considérerons des fonctions splines cubiques, c'est-à-dire constituées de polynômes du troisième degré dont les deux premières dérivées sont continues.

Soit n valeurs temporelles successives t_1, t_2, …, t_n ainsi que les observations correspondantes x_1, x_2, …, x_n de la série irrégulière initiale. L'algorithme de calcul est le suivant:

1. On calcule les écarts entre les abscisses temporelles: $H_i = t_i - t_{i-1}$.

2. On construit un vecteur colonne des différences V tel que:

$$V = \begin{Bmatrix} \frac{x_3 - x_2}{H_3} - \frac{x_2 - x_1}{H_2} \\ \frac{x_4 - x_3}{H_4} - \frac{x_3 - x_2}{H_3} \\ \vdots \\ \frac{x_n - x_{n-1}}{H_n} - \frac{x_{n-1} - x_{n-2}}{H_{n-1}} \end{Bmatrix}$$

   et une matrice carrée M telle que:

$$M = \begin{Bmatrix} \frac{H_2 + H_3}{3} & \frac{H_3}{6} & 0 & \cdots & 0 \\ \frac{H_3}{6} & \frac{H_3 + H_4}{3} & \frac{H_4}{6} & & 0 \\ 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{H_{n-1}}{6} & \frac{H_{n-1} + H_n}{3} \end{Bmatrix}$$

3. On calcule les dérivées secondes aux points expérimentaux. Pour cela il faut résoudre le système $M \,.\, DS = V$ ou encore $DS = M^{-1} \,.\, V$. On obtient ainsi dans le vecteur DS les dérivées secondes aux points d'origine. On attribue des dérivées secondes nulles aux points extrêmes de la courbe (argument **method = "fmm"** de ***regspline()***), ou on considère que la courbe forme une boucle (dernière valeur = première valeur, argument **method = "periodic"**).

4. On calcule la fonction spline pour chaque valeur de temps T_j de la série régularisée selon un pas préalablement choisi. En appelant X_j, les ordonnées calculées de la série régularisée correspondant à ces temps T_j, on a:

$$X_j = (t_i - T_j)^3 . \frac{DS_{i-1}}{6.H_i} + (t_i - T_j) . \frac{x_{i-1} - \frac{H_i^2}{6} . DS_{i-1}}{H_i} + (T_j - t_{i-1})^3 . \frac{DS_i}{6.H_i} - (t_{i-1} - T_j) . \frac{x_i - \frac{H_i^2}{6} . DS_i}{H_i}$$

avec *i* choisi tel que $t_{i-1} < T_j \leq t_i$ pour chaque valeur de $j = 1 \ldots p$.

> *Etant donné que les courbes splines peuvent dans certains cas produire des valeurs plus grandes ou plus petites que les valeurs extrêmes de la série initiale, la méthode utilisée dans PASTECS recadre ces extrémas pour éviter qu'ils ne soient plus grands ou plus petits que les valeurs extrêmes de la série de départ. Cela évite, par exemple, d'obtenir des valeurs négatives pour des comptages dans certains cas.*

Référence:

Lancaster, P. & K. Salkauskas, 1986. Curve and surface fitting. *Academic Press*, England, 280 pp.

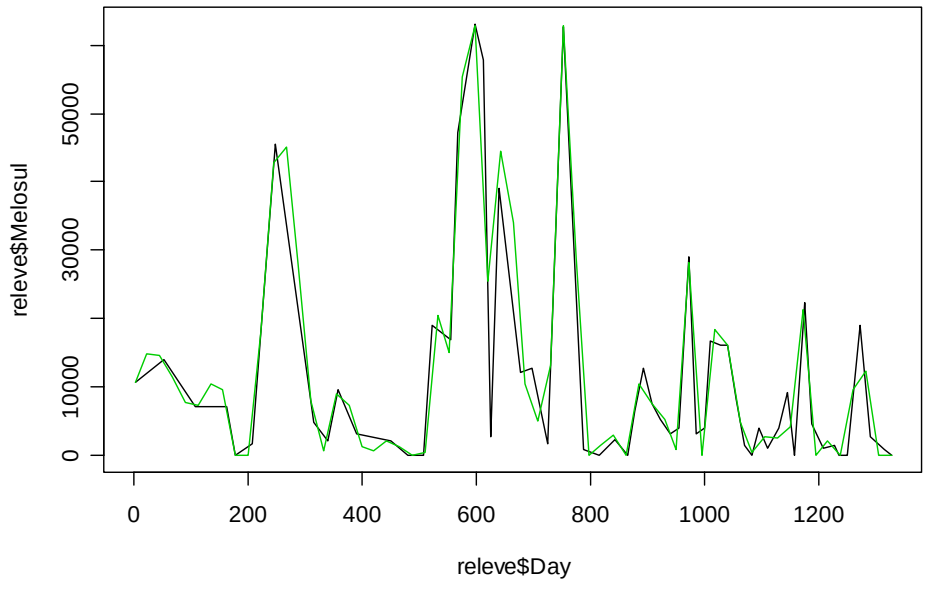
Exemple:

La série ***Melosul*** du jeu de données ***releve*** est régularisée avec des splines comme suit:

```
> data(releve)          # Dans R uniquement
> reg <- regspline(releve$Day, releve$Melosul)
```

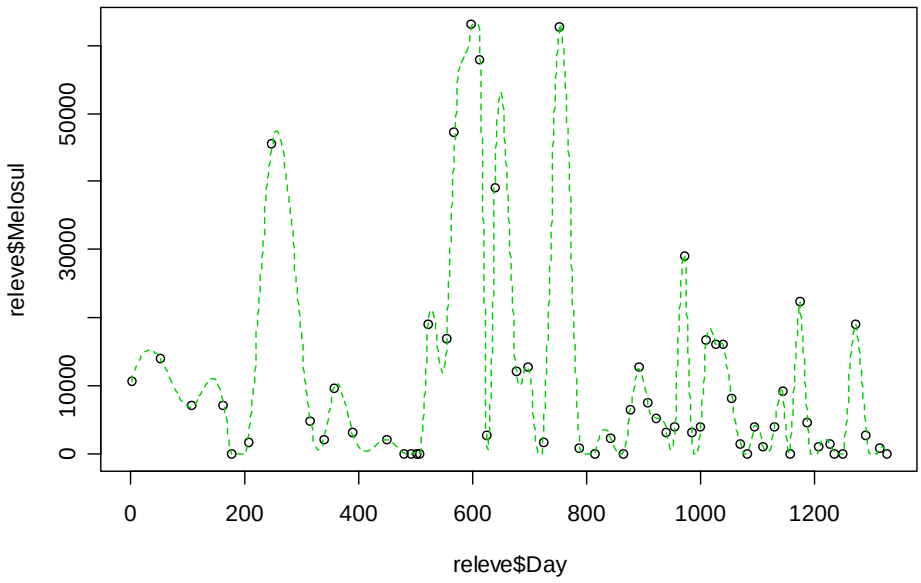
On peut tracer un graphique de la série de départ et de la série régularisée pour comparaison visuelle:

```
> plot(releve$Day, releve$Melosul, type="l")
> lines(reg$x, reg$y, col=3)
```



La série de départ apparaît en noir, et la série régularisée en vert. Les valeurs interpolées le sont le long de segments de courbes appelés splines. A noter toutefois que cette méthode est incluse dans ***regul()*** qui offre plus de facilités pour le diagnostic et l'extraction des séries temporelles après régularisation. On peut visualiser les points de la série de départ superposés à la fonction continue calculée qui sert à la régularisation comme suit:

```
> plot(releve$Day, releve$Melosul, type="p")
> lines(regspline(releve$Day, releve$Melosul,
+ n=length(releve$Day)*10), col=3, lty=2)
```



Régularisation par la méthode des aires regarea()

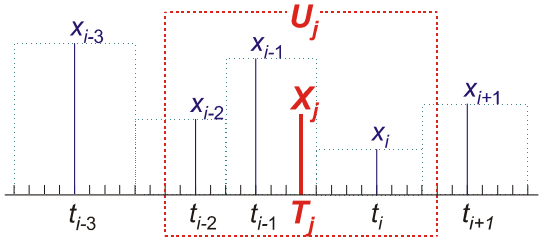
Une autre méthode d'interpolation linéaire correspond à l'interpolation par les aires décrites par Fox (1972). Cette méthode permet d'estimer les valeurs manquantes, mais plus généralement de substituer à une série irrégulière, des valeurs régulièrement espacées dans le temps en prenant en compte toutes les valeurs mesurées dans un intervalle constant à gauche et à droite de la valeur interpolée. Contrairement aux autres méthodes de régularisation, la courbe obtenue ne passe pas nécessairement par les points observés intiaux, c'est-à-dire qu'un lissage est effectué simultanément à l'interpolation. Ce lissage est d'autant plus important que la fenêtre d'interpolation est grande.

Soit une série d'observations x_1, x_2, ..., x_n aux temps t_1, t_2,..., t_n. On considère que chaque valeur x_i représente une estimation des valeurs comprises dans l'intervalle $f_i = \left[\frac{t_{i-1}+t_i}{2}; \frac{t_i+t_{i+1}}{2}\right]$ de longueur $F_i = \frac{t_i+t_{i+1}}{2} - \frac{t_{i-1}+t_i}{2}$, c'est-à-dire que l'on considère en fait des **aires** successives rectangulaires, plutôt que des observations ponctuelles discontinues dans le temps.

Un intervalle de temps U (une fenêtre), est choisi de préférence plus grand que la moyenne des durées minimums séparant les observations successives sur l'ensemble des séries initiales traitées. Considérons des fenêtres U_j centrées autour des valeurs T_j choisies pour la série régulière calculée. Ces fenêtres U_j sont limitées par les dates $U_{-j} = T_j - U/2$ et $U_{+j} = T_j + U/2$. Le cas le plus simple correspond à la coïncidence entre le point au centre de la fenêtre et la date d'observation. Dans ce cas il n'est pas effectué de régularisation et la valeur observée reste inchangée (on peut aussi décider d'effectuer le calcul sur ce point à l'aide de l'option **interp = T**, voir description de la fonction ***regarea()***). Dans les autres cas, l'interpolation prend en compte tous les intervalles f_i inclus dans la fenêtre. Soit $F_i \mid U_j$ la fraction de F_i appartenant à la fenêtre U_j. Pour chaque observation, trois cas peuvent apparaître:

- les observations telles que $F_i \mid U_j = F_i$ (aire complètement inclue dans la fenêtre),
- les observations telles que $F_i \mid U_j = 0$ (aire totalement en dehors de la fenêtre),
- les observations telles que $F_i > F_i \mid U_j > 0$ (intervalle coupant un bord de la fenêtre).

On a: $\sum_i (F_i \mid U_j) = U$. La valeur interpolée est donnée par $X_j = \left[\sum_i x_i (F_i \mid U_j)\right] \Big/ U$.



La figure illustre ce procédé. Il faut estimer la valeur X_j au temps T_j. Les points x_{i-2}, x_{i-1} et x_i sont inclus dans la fenêtre. Les points x_{i-3} et x_{i+1} sont les proches à l'extérieur. Les valeurs réelles observées sont: x_{i-3} = 10, x_{i-2} = 5, x_{i-1} = 9, x_i = 3, x_{i+1} = 6, et les $F_i \mid U_j$ correspondants: 0, 4, 6, 7, 1.

On a: $\sum (F_i \mid U_j) = 18$.

D'où: X_j = (10 . 0 + 5 . 4 + 9 . 6 + 3 . 7 + 6 . 1) / 18 = 5,6

Notons qu'une valeur nulle interpolée correspond soit:

- au cas où toutes les observations x_i satisfaisant $F_i \mid U_j > 0$ sont égales à 0,

- à la coïncidence de dates entre un 0 observé et le milieu de la fenêtre: $T_j = t_i$ avec $x_i = 0$ lorsque les données qui coïncident ne sont pas interpolées (argument **inter = F** de ***regarea()***).

L'algorithme utilisé dans l'ancien PASSTEC 2000 correspond à peu de choses près à cette description. Le code se caractérise par des boucles (sur toutes les valeurs extrapolées) et de nombreux tests de conditions. L'algorithme a dû donc être totalement réécrit en calcul matriciel pour la librairie PASTECS. Quatre vecteurs sont créés: t_-, t_+ et U_-, U_+. Les vecteurs t_- et t_+ contiennent respectivement les bornes inférieures et supérieures de chaque aire correspondant aux points x_i mesurés. Leur dimension est donc n, le nombre de points dans la série initiale. De même, les vecteurs U_- et U_+ contiennent les bornes inférieures et supérieures des fenêtres entourant chaque point X_j à extrapoler au temps T_j. Leur dimension est p, le nombre de points à obtenir dans la série régularisée. Les vecteurs t_- et t_+ sont recopiés p fois en colonnes, de manière à former deux matrices $n \times p$ (lignes × colonnes). De même, les vecteurs U_- et U_+ sont recopiés n fois en lignes pour former également deux matrices de dimension $n \times p$. On peut montrer que les $F_i \mid U_j$ pour toutes les valeurs à extrapoler sont obtenus dans les colonnes respectives de F calculé comme suit:

$$F_{ij} = \min(t_{+ij}, U_{+ij}) - \max(t_{-ij}, U_{-ij})$$

Et ce, à condition que:

1. toutes les valeurs contenues dans les vecteurs t_-, t_+, U_-, U_+ soient positives (il suffit de soustraire la plus petite valeur de temps rencontrée à chaque vecteur, si cette valeur est inférieures à 0, pour obtenir cette condition),

2. toutes les valeurs négatives obtenues pour les F_{ij} soient remplacées par des zéros (elles correspondent à des plages totalement en dehors de la fenêtre U_j).

Par conséquent, les valeurs interpolées sont obtenues dans le vecteur X_j par:

$$X_j = \left[\sum_i x_i . F_{ij} \right] \Big/ U$$

où i est l'indice des lignes et j est l'indice des colonnes pour la matrice F (somme par colonne du produit $x_i.F_{ij}$ divisé par la taille de fenêtre U).

L'inconvénient de cette méthode est de produire 4 matrices de très grandes tailles $n \times p$ pour des séries longues. De plus, comme la plupart des F_{ij} sont nuls (en dehors de U) pour de très longues séries de taille nettement supérieure à la taille de la fenêtre U, cette méthode utilise une grande quantité de mémoire vive et de calcul dans pareil cas. La solution est de diviser la série en sous-unités plus petites et d'effectuer l'extrapolation sur chaque sous-unité indépendamment le long de la « diagonale » $j \approx i$. Nous avons déterminé empiriquement que, sous S+ comme sous R, la taille de sous-unité qui optimise la vitesse de calcul est d'environ 100 valeurs interpolées calculées à la fois. Cette taille de sous-unité est toutefois ajustable par l'argument **split** de la fonction ***regarea()*** (voir description de la fonction). Si le nombre de valeurs dans la série initiale est sensiblement égal au nombre de valeurs extrapolées, les matrices ont une taille d'environ 100 × 100, soit 10000 éléments. Notre machine de référence (Pentium II 500 Mhz avec 128 Mo de mémoire vive) est capable de régulariser la série ***Melosul*** de la matrice exemple ***releve*** (200 valeurs extrapolées, et fenêtre de 50 jours) en 0.175 sec avec **split = 100**, contre 2.68

sec avec **split = 1** (correspondant à peu près à la méthode de l'ancien PASSTEC 2000 de calcul valeur par valeur).

Références:

Fox, W.T. & J.A. Brown, 1965. The use of time-trend analysis for environmental interpretation of limestones. *J. Geol.*, 73:510-518.

Ibanez, F., 1991. Treatment of the data deriving from the COST 647 project on coastal benthic ecology: The within-site analysis. *In:* B. Keegan (ed). *Space and Time Series Data Analysis in Coastal Benthic Ecology.* Pp 5-43.

Ibanez, F. & J.C. Dauvin, 1988. Long-term changes (1977-1987) on a muddy fine sand *Abra alba - Melinna palmata* population community from the Western English Channel. *J. Mar. Ecol. Prog. Ser.*, 49:65-81.

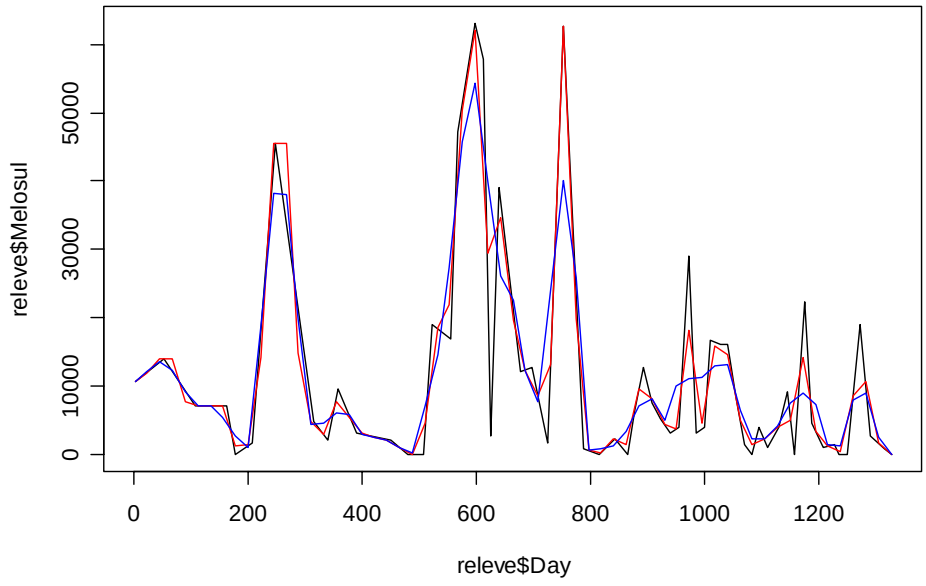
Exemple:

La série ***Melosul*** du jeu de données ***releve*** est régularisée avec la méthode des aires, en utilisant une fenêtre de 25, puis de 50 jours) comme suit:

```
> data(releve)          # Dans R uniquement
> reg <- regarea(releve$Day, releve$Melosul, window=25)
> reg2 <- regarea(releve$Day, releve$Melosul, window=50)
```

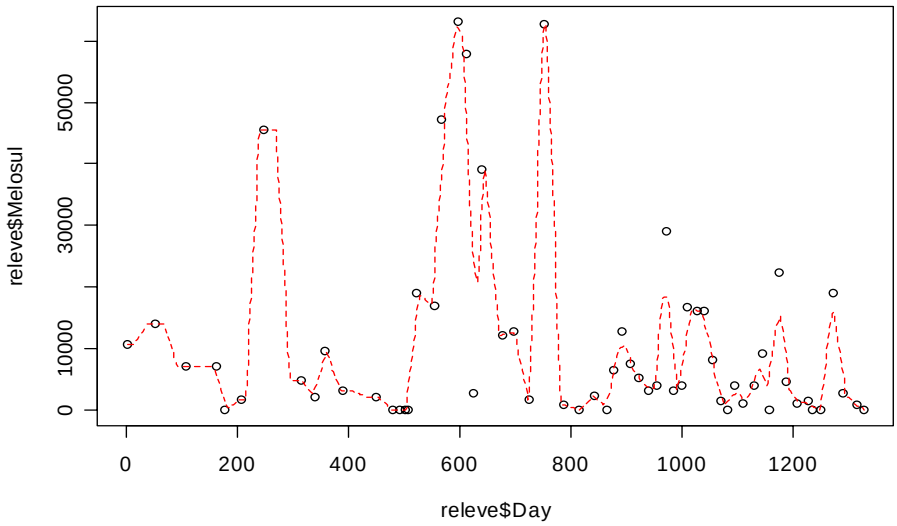
On peut tracer un graphique de la série de départ et des deux séries régularisées pour comparaison visuelle:

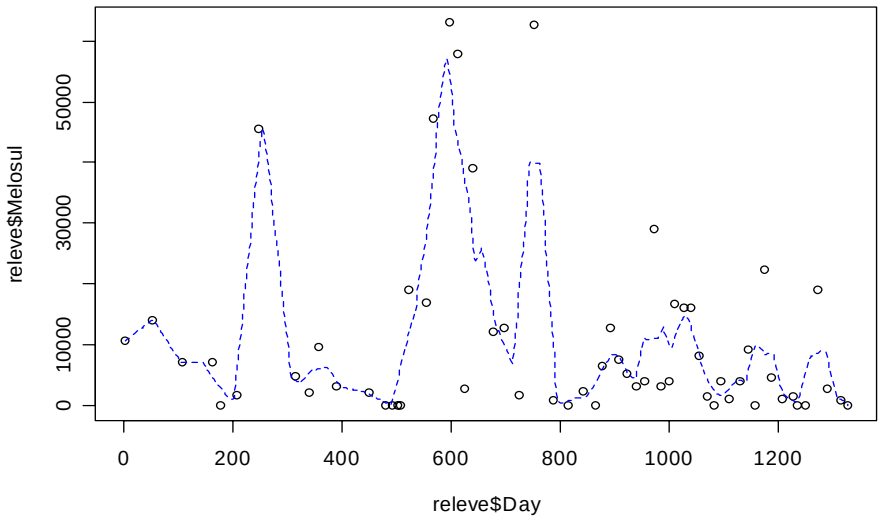
```
> plot(releve$Day, releve$Melosul, type="l")
> lines(reg$x, reg$y, col=2)
> lines(reg2$x, reg2$y, col=4)
```



La série de départ apparaît en noir, la série régularisée avec une fenêtre de 25 jours est tracée en rouge, et celle régularisée avec une fenêtre de 50 jours, en bleu. Lorsqu'une interpolation est nécessaire, une moyenne des aires incluses dans la fenêtre est réalisée. Plus cette fenêtre est large, plus les valeurs interpolées seront « lissée » par rapport à la série de départ. A noter toutefois que cette méthode est incluse dans ***regul()*** qui offre plus de facilités pour le diagnostic et l'extraction des séries temporelles après régularisation. Les deux graphiques suivants montrent les fonctions continues calculées qui servent à la régularisation de l'exemple précédent pour respectivement un fenêtre de 25 jours (en rouge) et de 50 jours (en bleu):

```
> plot(releve$Day, releve$Melosul, type="p")
> lines(regarea(releve$Day, releve$Melosul, window=25,
+ n=length(releve$Day)*10), col=2, lty=2)
> plot(releve$Day, releve$Melosul, type="p")
> lines(regarea(releve$Day, releve$Melosul, window=50,
+ n=length(releve$Day)*10), col=4, lty=2)
```





Conversion d'objets 'regul' en séries régulières

Comme nous l'avons vu au paragraphe présentant les fonctions générales de régularisation, la fonction ***tseries()*** permet de convertir un objet '*regul*' issu d'une régularisation (de même d'ailleurs qu'un objet '*tsd*' issu d'une décomposition, voir chapitre décomposition de séries spatio-temporelles) en objets '*rts*' ou '*ts*' contenant **toutes** les séries. Pour extraire **une partie** des séries présentes dans l'objet '*regul*', utiliser plutôt la méthode ***extract()*** de l'objet. Enfin, la fonction ***is.tseries()*** est utile pour tester si un objet est une série régulière, que ce soit sous S+ ou sous R, et indépendamment du fait que la série soit stockée en un objet '*rts*' ou '*ts*' en interne. Cette fonction est donc à préférer par rapport à ***is.rts()*** ou ***is.ts()*** proposées en standard dans S+ ou R, respectivement.

Manipulations de base des séries régulières *'rts'* et *'ts'*

Ce chapitre ne présente pas de nouvelles fonctions de la librairie PASTECS, mais montre comment les séries spatio-temporelles régulières (objets *'ts'* et *'rts'*) sont manipulées dans S+ ou R. Seules les bases indispensables à la compréhension des objets *'ts'* et *'rts'* pour pouvoir comprendre les fonctions de PASTECS sont présentées ici. Pour une introduction détaillée des séries temporelles et de leur traitement dans S+ ou R, nous renvoyons le lecteur intéressé au chapitre 14 de Venables & Ripley (2002), ainsi qu'aux documents annexes à cette référence disponibles au format PDF sur le Web (http://www.stats.ox.ac.uk/pub/MASS4/).

Dans le chapitre régularisation, nous avons vu comment extraire une série temporelle, c'est-à-dire, un objet de classe *'ts'* ou *'rts'* depuis un data frame régularisé (objet *'regul'*) à l'aide de ***tseries()***. S+ et R offrent toute une panoplie de fonctions pour manipuler, analyser et représenter graphiquement les objets *'ts'*. PASTECS y ajoute quelques fonctions spécifiques à l'analyse de données en écologie marine. Ces dernières seront décrites dans le chapitre suivant. Avant de les présenter, voyons d'abord ce que nous pouvons déjà faire avec les fonctions standard de S+ ou de R.

Première différence entre S+ et R: les fonctions 'time series' de base sont disponibles directement sous S+, alors que sous R (version < 1.9.0), il faut prendre soin de charger le package *'ts'* avant de pouvoir les utiliser. Notez qu'à partir de R 1.9.0, ces fonctions sont disponibles dans le package *'stats'* qui est chargé en mémoire au démarrage, par défaut.

```
> library(ts)    # Seulement sous R, version < 1.9.0
```

> *De plus, si R définit logiquement un objet 'ts' et offre des méthodes objet compatibles à la manipulation de séries spatio-temporelles, S+ (dans sa version 2000 release 3) est en pleine mutation à ce niveau. A terme, la volonté est de remplacer les objets 'ts' par des objets 'rts', par contraste avec des nouveaux objets 'cts' (calendar time series) et 'its' (irregular time series) récemment créés. Il en résulte que les deux types 'ts' et 'rts' coexistent pour l'instant. Alors que les objets 'rts' sont correctement définis à l'aide de leur classe 'rts', les objets 'ts' ne peuvent pas être définis avec leur classe, mais uniquement par la présence d'un attribut* ***'tsp'*** *(voir ci-dessous).*

Une série spatio-temporelle régulière est stockée dans S+ ou R en tant que vecteur contenant les données régulièrement espacées, ainsi qu'un attribut **tsp** sous S+ ou **tspar** sous R qui indique quelle est l'échelle de temps de la série (date initiale, date finale et fréquence des données dans l'unité de temps considérée, par exemple 12 pour des données mensuelles exprimée en année comme unité). Dans le cas de séries multiples, les données sont stockées en colonnes dans une matrice. Dans les deux cas, les valeurs manquantes ne sont pas acceptées par la plupart des traitements.

Création et représentation graphique d'une série

La fonction ***ts()*** permet de créer une série temporelle tout en précisant l'échelle de temps, par exemple: la date initiale et la fréquence des données, mais sous S+, on préfèrera ***rts()***. Un certain nombre de fonctions existent dans S+ ou R pour générer artificiellement des données, y compris des générateurs de nombres pseudo-aléatoires sous différentes distributions (par exemple ***rnorm()*** pour une distribution normale). Ces fonctions permettent de faire l'équivalent de ***Transformation -> Valeurs Aléatoires*** dans PASSTEC 2000, et même plus. Ceci est bien utile pour étudier le résultat de différentes analyses sur des données artificielles dont les paramètres sont connus. Par exemple, générons une série de données mensuelles (unité année et fréquence 12) commençant en avril 1998 (**start = c(1998, 4)**), contenant 100 valeurs et comportant une composante

sinusoïdale d'une période annuelle ainsi qu'une composante aléatoire ayant une distribution normale de moyenne nulle et d'écart type 0.5:

```
> tser <- ts(sin((1:100)/6*pi)+rnorm(100, sd=0.5), start=c(1998,
+ 4), frequency=12)
> tser
             Jan         Feb         Mar         Apr
1998                                         0.59897723
1999 -0.67748804 -0.88088289  0.78630325  0.13217563
2000 -0.62371255 -0.66990826  0.83449819  0.79783790
2001 -2.12436689 -0.52581156  0.81329477  1.06435920
2002 -1.52789141 -0.56730282 -0.39693053 -0.12925129
2003 -1.18286223 -0.05610947 -0.58263409  0.05027564
2004 -0.72529340 -1.24154295  0.23559678  0.46617355
2005 -0.89376790 -0.04580969  0.94160400  0.22996846
2006 -1.68152973 -0.38347542 -0.23450394  1.52118505
             May         Jun         Jul         Aug
1998  1.23475206  1.46628413 -0.09476006  0.94279765
1999  0.46350286  1.24294531 -0.07297615  0.71385640
2000  0.68925141  0.43853694  0.16452186  0.06254603
2001  0.97296652  0.57288689  2.00658200  0.26111531
2002 -0.28089107  1.75300222  2.19721514  0.45650542
2003  1.49109425  0.80772911  1.31615818  0.95046577
2004  0.74935101  0.01533196  0.89154582  0.47800842
2005  1.09686285  0.68573934  0.44475419  0.40692158
2006  0.39678139  1.01641059  0.37596144
             Sep         Oct         Nov         Dec
1998 -0.39671057 -0.83031027 -1.23241922 -0.61132178
1999  0.30010569 -0.95651594 -1.10585682 -0.88798084
2000  0.93703601 -0.48558293 -0.25026491 -0.49322370
2001 -0.89402673  0.28222829 -0.79804378  0.17403044
2002 -1.10050631 -0.79038756 -1.38900926 -0.54012760
2003 -1.10295575 -0.48710000 -1.25021704 -2.20668928
2004 -0.29999685 -0.42293269  0.12789873 -0.43545112
2005 -0.49430730 -0.73287274 -0.50872893 -1.23438588
2006
```

De même, pour créer un objet *'rts'* sous S+, il suffit d'utiliser ***rts()*** comme ***ts()*** sous R. Pour connaître la classe d'un objet, comme ***tser***, par exemple pour déterminer s'il s'agit d'un objet *'ts'* ou *'rts'*, il suffit de rentrer:

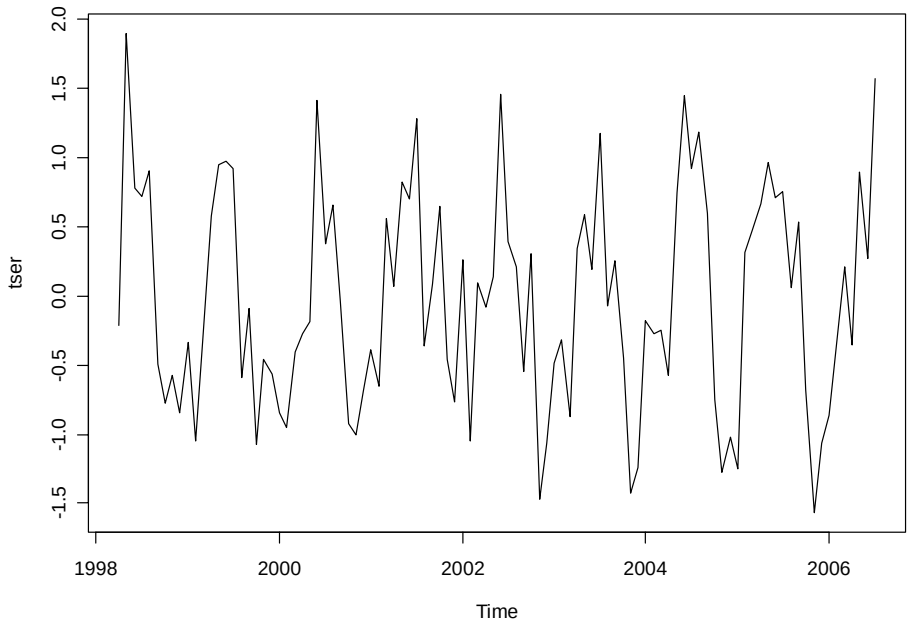
```
> class(tser)
```

Dans le cas présent, étant donné les mutations en cours dans S+, il vaut mieux utiliser la fonction ***is.tseries()*** de PASTECS présentée dans le chapitre précédent et qui renvoie **TRUE** dans tous les cas, que l'on ait affaire à un objet *'ts'* définit comme tel dans R, à un objet « non déclaré » *'ts'* de S+ ou encore à un objet *'rts'* toujours sous S+:

```
> is.tseries(tser) # retourne TRUE, si l'objet est une time series
[1] TRUE
```

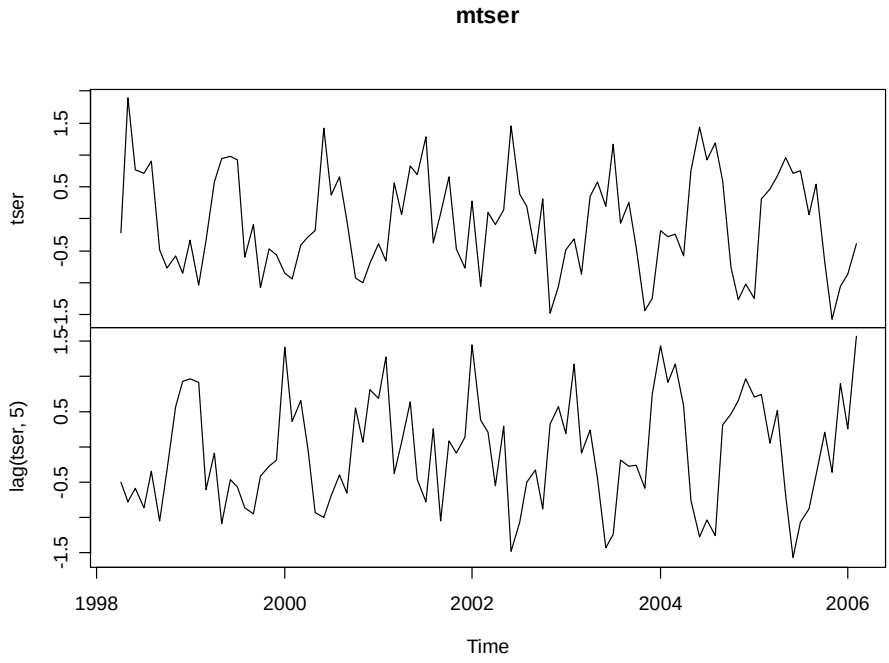
Il existe une fonction spécialisée dans le traçage de graphiques de séries spatio-temporelles dans S+ et R: ***ts.plot()***:

```
> ts.plot(tser)
```



Dans R seulement, la méthode ***plot()*** des objets *'ts'* offre des options avancées pour le traçage de séries multiples sur un même axe temporel:

```
> mtser <- ts.intersect(tser, lag(tser, 5))
> plot(mtser)          # Dand R uniquement
```



Manipulation des paramètres temporels d'une série

Les fonctions ***start()***, ***end()*** et ***frequency()*** permettent de récupérer les différents paramètres de temps de la série:

```
> start(tser)
[1] 1998    4
> end(tser)
[1] 2006    7
> frequency(tser)
[1] 12
```

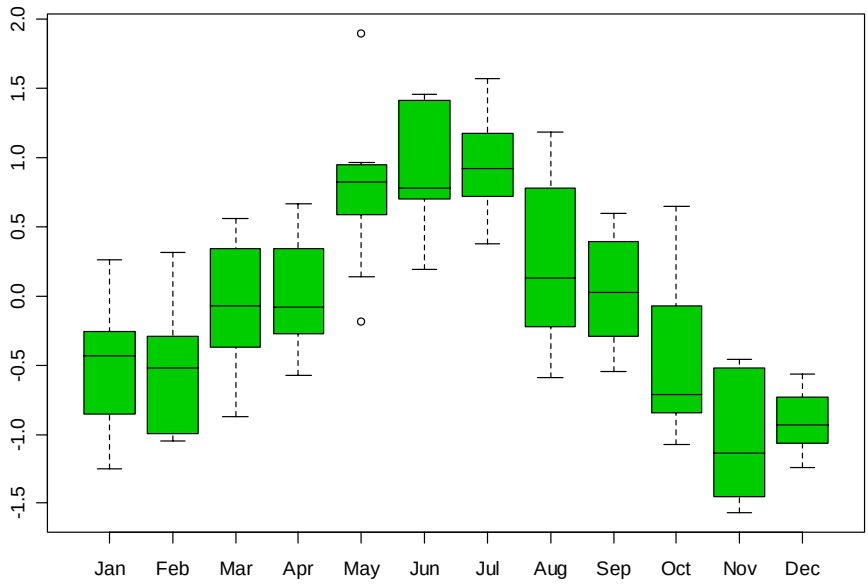
La fonction ***time()*** permet de reconstituer le vecteur temps pour la série (dans le cas présent, 1 mois est représenté par 1/12 d'unité –année–, c'est-à-dire, 0.083):

```
> time(tser)
          Jan      Feb      Mar      Apr      May      Jun
1998                                 1998.250 1998.333 1998.417
1999 1999.000 1999.083 1999.167 1999.250 1999.333 1999.417
2000 2000.000 2000.083 2000.167 2000.250 2000.333 2000.417
2001 2001.000 2001.083 2001.167 2001.250 2001.333 2001.417
2002 2002.000 2002.083 2002.167 2002.250 2002.333 2002.417
2003 2003.000 2003.083 2003.167 2003.250 2003.333 2003.417
2004 2004.000 2004.083 2004.167 2004.250 2004.333 2004.417
2005 2005.000 2005.083 2005.167 2005.250 2005.333 2005.417
2006 2006.000 2006.083 2006.167 2006.250 2006.333 2006.417
          Jul      Aug      Sep      Oct      Nov      Dec
1998 1998.500 1998.583 1998.667 1998.750 1998.833 1998.917
1999 1999.500 1999.583 1999.667 1999.750 1999.833 1999.917
2000 2000.500 2000.583 2000.667 2000.750 2000.833 2000.917
2001 2001.500 2001.583 2001.667 2001.750 2001.833 2001.917
2002 2002.500 2002.583 2002.667 2002.750 2002.833 2002.917
2003 2003.500 2003.583 2003.667 2003.750 2003.833 2003.917
2004 2004.500 2004.583 2004.667 2004.750 2004.833 2004.917
2005 2005.500 2005.583 2005.667 2005.750 2005.833 2005.917
2006 2006.500
```

La fonction ***cycle()*** indique l'appartenance de chaque donnée de la série à un cycle. Elle permet, par exemple de séparer les données par mois si la base de temps est l'année à l'aide de la fonction ***split()***. Ensuite, il est possible de traiter ou de représenter séparément les statistiques mois par mois:

```
> tser.cycle <- cycle(tser)
> tser.cycle
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1998                   4   5   6   7   8   9  10  11  12
1999   1   2   3   4   5   6   7   8   9  10  11  12
2000   1   2   3   4   5   6   7   8   9  10  11  12
2001   1   2   3   4   5   6   7   8   9  10  11  12
2002   1   2   3   4   5   6   7   8   9  10  11  12
2003   1   2   3   4   5   6   7   8   9  10  11  12
2004   1   2   3   4   5   6   7   8   9  10  11  12
2005   1   2   3   4   5   6   7   8   9  10  11  12
2006   1   2   3   4   5   6   7
```

```
> boxplot(split(tser, tser.cycle), names=month.abb, col=3)
```



La manipulation des séries temporelles se fait à l'aide de fonctions spéciales. ***aggregate()*** permet de réduire le nombre d'observations en diminuant la fréquence. Par exemple pour transformer la série ***tser*** qui a un pas de temps de 1 mois en une série de pas de temps de 1 trimestre, on utilise:

```
> aggregate(tser, 4, mean)
              Qtr1          Qtr2          Qtr3          Qtr4
1998                1.1000044734  0.1504423387 -0.8913504215
1999 -0.2573558940  0.6128746007  0.3136619780 -0.9834511978
2000 -0.1530408722  0.6418754169  0.3880346344 -0.4096905154
2001 -0.6122945593  0.8700708699  0.4578901932 -0.1139283508
2002 -0.8307082548  0.4476199503  0.5177380843 -0.9065081419
2003 -0.6072019277  0.7830329961  0.3878893980 -1.3146687739
2004 -0.5770798534  0.4102855075  0.3565191320 -0.2434950292
2005  0.0006754724  0.6708568830  0.1191228232 -0.8253291831
2006 -0.7665030326  0.9781256775
```

A noter que l'intitulé des colonnes est différent sous S+: elles sont simplement numérotées de 1 à 4. En fait, R crée des intitulés particuliers pour des séries de fréquence égale à 4 (Qtr1 -> Qtr4) ou à 12 (intitulé des mois en abbrégé), alors que S+ se contente simplement de numéroter les colonnes en absence de plus d'information sur leur nature.

window() extrait au contraire une sous-série contenue dans la fenêtre de temps indiquée sans modifier la fréquence des observations. Par exemple, pour extraire les années 1999 à 2001 complètes de ***tser***, on utilisera:

```
> window(tser, start=c(1999, 1), end=c(2001, 12))
             Jan         Feb         Mar         Apr
1999 -0.67748804 -0.88088289  0.78630325  0.13217563
2000 -0.62371255 -0.66990826  0.83449819  0.79783790
2001 -2.12436689 -0.52581156  0.81329477  1.06435920
             May         Jun         Jul         Aug
1999  0.46350286  1.24294531 -0.07297615  0.71385640
2000  0.68925141  0.43853694  0.16452186  0.06254603
2001  0.97296652  0.57288689  2.00658200  0.26111531
```

```
             Sep         Oct         Nov         Dec
1999  0.30010569 -0.95651594 -1.10585682 -0.88798084
2000  0.93703601 -0.48558293 -0.25026491 -0.49322370
2001 -0.89402673  0.28222829 -0.79804378  0.17403044
```

lag() permet de décaler la série en arrière dans le temps (ou en avant si on fournit une valeur de décalage négative); nous l'avons déjà utilisée plus haut. ***diff()*** calcule la différence entre une série et elle-même décalée dans le temps de *k* observations. ***ts.intersect()*** et ***ts.union()*** permettent de réunir deux ou plusieurs séries au sein d'une même matrice multivariée, avec une échelle de temps commune. Les fréquences respectives des séries à assembler doivent être identiques. ***ts.intersect()*** retient uniquement l'intervalle de temps commun à toutes les séries, tandis que ***ts.union()*** considère l'ensemble de l'intervalle temporel, toutes séries confondues.

Référence:

Venables, W.N. & B.D. Ripley, 2002. Modern applied statistics with S-plus (4th ed.). *Springer-Verlag, New York*. 495 pp.

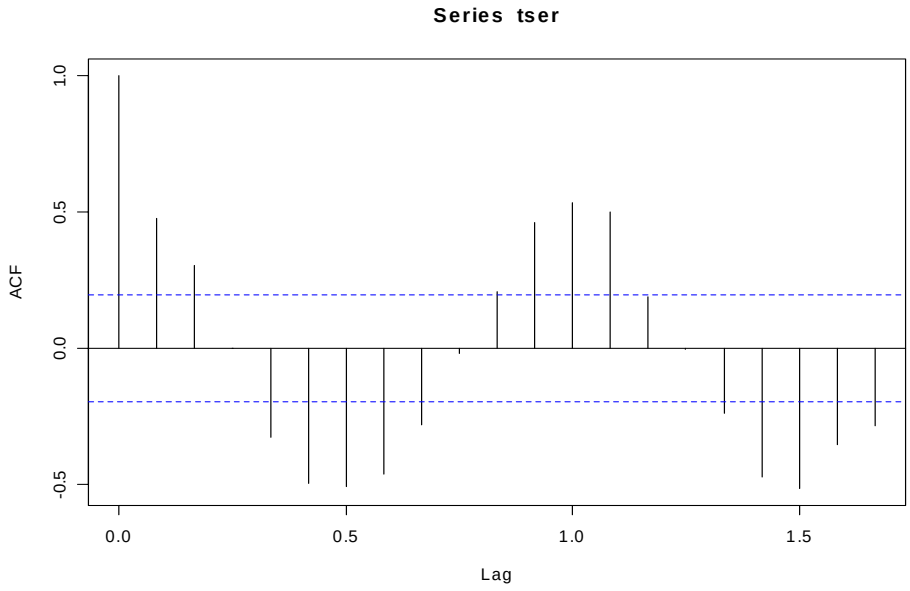
Analyse de séries spatio-temporelles

Les différentes méthodes présentées dans ce chapitre permettent de caractériser des séries spatio-temporelles, que ce soit en analysant l'autocorrélation ou la cross-corrélation, la quantité d'information (points de retournement, tournogramme), en effectuant une analyse spectrale, ou en déterminant l'existence d'une tendance générale ou locale. Le chapitre suivant traitera plus particulièrement de la décomposition de séries en plusieurs composantes (tendance, cycle saisonnier, résidus par exemple). Bien que la décomposition et l'analyse soient des phases du traitement des séries spatio-temporelles qui ne se font pas l'une sans l'autre, nous les traitons dans des chapitres différents parce qu'elles renvoient des résultats différents. Alors que les techniques d'analyse ne modifient pas la série de départ et renvoient seulement les résultats statistiques de l'analyse, les décompositions renvoient des objets plus complexes qui contiennent aussi les composantes issues du traitement, composantes qui peuvent être transformées à nouveau en objet 'time series' pour être traitées ou analysées en cascade éventuellement. Ce processus complexe, lié à l'organisation objet du code de la librairie PASTECS, sera donc traité intégralement dans le chapitre suivant, et ne sera pas du tout abordé dans celui-ci.

Autocorrélation, autocovariance, cross-corrélation et cross-covariance

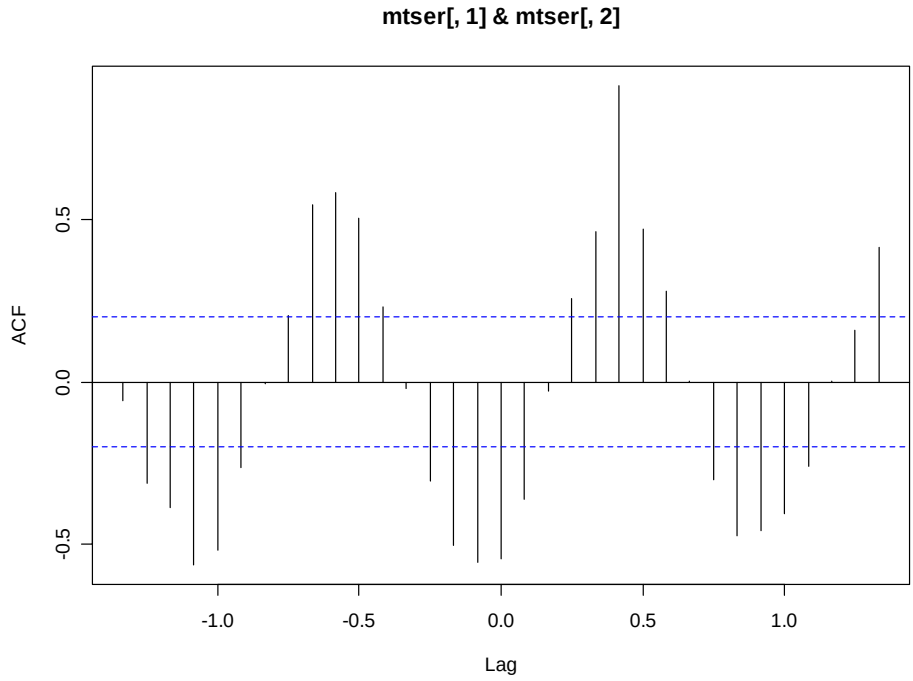
Une série spatio-temporelle est caractérisée principalement par une autocorrélation non nulle, ce qui signifie que les différentes observations dans la série ne sont pas indépendantes les unes des autres. Cette autocorrélation s'étudie en considérant le profil de la fonction d'autocorrélation obtenu en décalant la série d'une valeur progressivement croissante et en comparant la série initiale et la série décalée. Le coefficient de corrélation obtenu sera d'autant plus grand que les deux séries (initiale et décalée) sont similaires. A l'inverse, si les séries sont opposées, la corrélation sera négative. Une autocorrélation nulle indique qu'il n'y a pas de corrélation entre la série initiale et la série décalée au pas considéré. Le graphique de l'autocorrélation (caractéristique d'une série périodique) pour ***tser*** (une série artificielle silimaire à celle étudiée dans le chapitre précédent) est obtenu par la fonction ***acf()***:

```
> tser <- ts(sin((1:100)/6*pi)+rnorm(100, sd=0.5), start=c(1998,
+ 4), frequency=12)
> acf(tser)
```



De même, on peut tracer l'autocovariance (la covariance est calculée entre la série de départ et la série décalée à la place de la corrélation) en précisant l'argument **type = "covariance"** à l'appel de ***acf()***. On a aussi la possibilité de calculer une autocorrélation partielle avec **type = "partial"**. En outre, R fournit aussi ***ccf()***, fonction qui calcule la cross-corrélation ou la cross-covariance entre deux séries:

```
> mtser <- ts.intersect(tser, lag(tser, 5))
> ccf(mtser[,1], mtser[,2])   # Sous R seulement
```



On retrouve bien une cross-corrélation maximale pour un décalage de 5 entre la série de départ (***mtser[, 1]***) et la série décalée de 5 observations vers la gauche (***mtser[, 2]***).

Référence:

Venables, W.N. & B.D. Ripley, 2002. Modern applied statistics with S-plus (4th ed.). *Springer-Verlag, New York*. 495 pp.

Autocorrelation multiple (autoD2, crossD2 et D2 au centre)

Considérons le cas où on souhaiterait connaître l'échelle de temps la plus pertinente pour décrire une situation climatique donnée, ou que l'on veuille connaître l'échelle temporelle des changements hydrologiques en un point. Ces situations sont décrites non par un descripteur mais un ensemble (pression atmosphérique, pluviosité, température, etc.). Il faudrait donc définir une fonction semblable à l'autocorrélation ou au variogramme (voir plus loin dans ce paragraphe), mais étendue au cas multivarié.

Pour limiter le nombre de descripteurs on peut préalablement remplacer la série multivariable par les premières composantes principales, et se retrouver dans le cas univarié, mais les nouvelles variables (composantes) n'auront pas forcément d'interprétation satisfaisante.

Il n'est pas possible non plus de calculer une autocorrélation multiple en prenant des coefficients de corrélation multiples décalés de *k*. Car, en effet, le coefficient de corrélation multiple se réfère à un modèle d'ajustement linéaire entre une variable dépendante et une série de variables indépendantes. On pourrait à la rigueur prendre le coefficient de corrélation multiple total R^2_{mT} défini par:

$$R^2_{mT} = \sum_{i=1}^{n} R^2_{mi}$$

pour *n* descripteurs et où R^2_{mi} est le coefficient de corrélation multiple par rapport au descripteur *i* considéré comme variable dépendante.

Nous avons proposé une autre technique basée sur la distance généralisée (D2 de Mahalanobis), appelée l'**autoD2** (Ibanez, 1975). La distance généralisée de Mahalanobis tient compte à la fois des différences d'amplitudes et des corrélations entre les descripteurs.

Si on considère deux populations définies par *p* caractères communs *x*, le D2 de Mahalanobis (distance entre les centroïdes) sera:

$$\mathrm{D}\,2 = (\overline{x}_1 - \overline{x}_2)'.S^{-1}.(\overline{x}_1 - \overline{x}_2)$$

où $\overline{x}_1$ et $\overline{x}_2$ sont les vecteurs moyens respectifs des groupes 1 et 2 et S^{-1} est l'inverse de la matrice somme des matrices de variance-covariance intra-groupe:

$$S = \frac{A_1 + A_2}{n_1 + n_2 - 2}$$

A_1 et A_2 étant les matrices variance-covariance des groupes 1 et 2, n_1 et n_2 étant les nombres d'individus respectifs des deux groupes. Ensuite, on calcule les termes successifs d'une fonction, l'autoD2 par rapport au décalage *k*. Soit une série multivariable x_t avec t = 1, 2, …, *n*. La fonction sera donnée par:

$$\mathrm{autoD}\,2(k) = \left[\frac{1}{n-k}\left(\sum_{t=1}^{n-k} x_t - \sum_{t=k+1}^{n} x_t\right)\right]'.S^{-1}.\left[\frac{1}{n-k}\left(\sum_{t=1}^{n-k} x_t - \sum_{t=k+1}^{n} x_t\right)\right]$$

où: $\frac{1}{n-k}\sum_{t=1}^{n-k} x_t$ correspond au vecteur des moyennes d'un premier groupe d'observations allant de t à n-k, $\frac{1}{n-k}\sum_{t=k+1}^{n} x_t$ correspond au vecteur des moyennes d'un second groupe d'observations allant de k+1 à n et S^{-1} est l'inverse de la matrice somme des deux matrices de variance-covariance des deux groupes d'observation, l'un de t à n-k, l'autre de k+1 à n.

Si on considère les valeurs **réduites** des descripteurs (écart type = 1), alors S^{-1} correspond à la matrice de corrélation intra-groupe. Pour $k = 0$, la valeur de l'autoD2 est nulle. Si la fonction diminue au bout d'un décalage k, cela signifiera que la série multivariable est proche de la série de départ, non décalée, à cette **échelle** d'observation.

De la même manière, il est possible de calculer des fonctions croisées (**crossD2**) qui indiqueront au bout de quel intervalle de temps deux séries multivariables peuvent éventuellement être plus semblables. Naturellement le nombre de descripteurs des deux groupes doit être identique. Les formules sont symétriques par rapport à l'autoD2. Pour le calcul de la fonction avec un groupe $G1$ en avance sur un groupe $G2$, on considère les différences entre les moyennes de $G1$ entre 1 et n-k et celles de $G2$ entre k+1 et n. Pour calculer la fonction avec $G2$ en avance sur $G1$, on intervertit la position des groupes.

> *Compte tenu de l'inversion matricielle, l'autoD2 aura des résultats biaisés si les données sont trop hétérogènes (données erratiques de biologie), car l'hétérogénéité locale entraîne souvent la singularité de la matrice S.*

Le **D2 au centre** (Ibanez, 1981) est un indice qui permet de reconnaître les changements temporels d'une série multivariée de p descripteurs. Alors que l'autoD2 indiquait l'échelle des changements, cet indice indique précisément les changements entre les amplitudes et les corrélations des descripteurs en prenant en compte la distance généralisée (D2 de Mahalanobis).

Si on considère des séries standardisées (x_t, moyenne nulle et variance unité), afin de stabiliser les échelles de variation, la formule de la distance D2 de Mahalanobis devient:

$$D2 = x_t' . R^{-1} . x_t$$

où R représente la matrice de corrélation entre les descripteurs.

Cooley & Lohnes (1962) ont montré que ce D2 est équivalent à un χ^2 avec p degrés de liberté. Si la valeur D2 dépasse la valeur du χ^2 pour p degrés de liberté, au seuil 5%, cela signifiera que l'observation a moins de 5% de chances d'appartenir à la population. Ainsi on pourra reconnaître les anomalies de la série multivariée grâce à ce test statistique. Le principe de recherche est progressif:

1. On définit une fenêtre de départ contenant un certain nombre d'observations p. Si la fenêtre est trop petite, un très grand nombre d'anomalies vont apparaître. Si elle est trop grande au contraire, il y en aura extrêmement peu. La taille de cette fenêtre peut être définie en regardant au préalable à quelle échelle se manifeste la plus grande hétérogénéité de la série par l'autoD2.

2. On calcule toutes les D2 des points de cette fenêtre.

3. On décale la fenêtre d'une observation, en éliminant la première et en faisant entrer l'observation p+1.

4. On calcule le D2 pour cette observation p+1.

5. On revient à 3, et on effectue ainsi les calculs pour toutes les observations.

Cette technique peut donc s'appliquer également en temps réel. Imaginons une sonde multiparamétrique tractée par un navire. Au bout d'une centaine d'observations, par exemple, on peut saisir immédiatement qu'un changement s'est produit au moment même où les données sont enregistrées.

> *A noter : la même remarque que pour l'autoD2, concernant les biais introduits par l'hétérogénéité trop grande des données, reste valable dans le cas du D2 au centre.*

Références:

Cooley, W.W. & P.R. Lohnes, 1962. Multivariate procedures for the behavioural sciences. *Whiley & sons*.

Dagnélie, P., 1975. Analyse statistique à plusieurs variables. *Presse agronomique de Gembloux*.

Ibanez, F., 1975. Contribution à l'analyse mathématique des évènements en écologie planctonique: optimisations méthodologiques; étude expérimentale en continu à petite échelle du plancton côtier. *Thèse d'état, Paris VI*.

Ibanez, F., 1976. Contribution à l'analyse mathématique des évènements en écologie planctonique. Optimisations méthodologiques. *Bull. Inst. Océanogr. Monaco*, 72:1-96.

Ibanez, F., 1981. Immediate detection of heterogeneities in continuous multivariate oceanographic recordings. Application to time series analysis of changes in the bay of Villefranche sur mer. *Limnol. Oceanogr.*, 26:336-349.

Ibanez, F., 1991. Treatment of the data deriving from the COST 647 project on coastal benthic ecology: The within-site analysis. *In*: B. Keegan (ed), *Space and time series data analysis in coastal benthic ecology*, p 5-43.

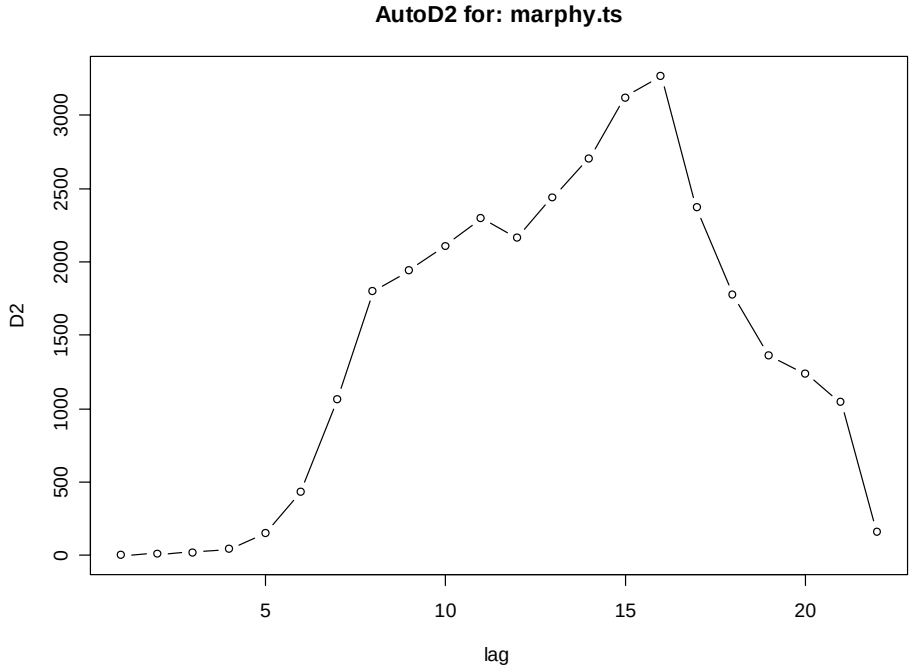
Exemples:

Nous allons étudier le jeu de données ***marphy*** de la librairie PASTECS qui correspond aux données physico-chimiques associées aux mêmes stations que les données de plancton ***marbio***. Le data frame ***marphy*** contient 4 colonnes, respectivement: ***Temperature***, ***Salinity***, ***Fluorescence***, ***Density***. La densité étant redondante avec la salinité, nous la laisserons tomber, pour nous concentrer sur les 3 premières variables environmentales que nous transformerons en séries temporelles régulières:

```
> data(marphy)
> names(marphy)
[1] "Temperature"  "Salinity"     "Fluorescence" "Density"
> marphy.ts <- as.ts(as.matrix(marphy[, 1:3]))
> plot(marphy.ts)   # Graphe de séries multiples sous R seulement
```

On peut étudier l'autocorrelation multiple de cet échantillon grâce à l'autoD2:

```
> AutoD2(marphy.ts)
$lag
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[21] 21 22

$D2
 [1]    4.22573   10.58976   23.47267   43.38305  153.44283
 [6]  433.54182 1064.54132 1802.43083 1940.63240 2111.62737
[11] 2298.56609 2166.24216 2443.59445 2708.61304 3121.54365
[16] 3267.95116 2370.99649 1772.94559 1366.08938 1239.05153
[21] 1045.82545  163.43575

$call
AutoD2(series = marphy.ts)

$data
[1] "marphy.ts"

$type
[1] "AutoD2"

$unit.text
[1] ""

attr(,"class")
[1] "D2"
```

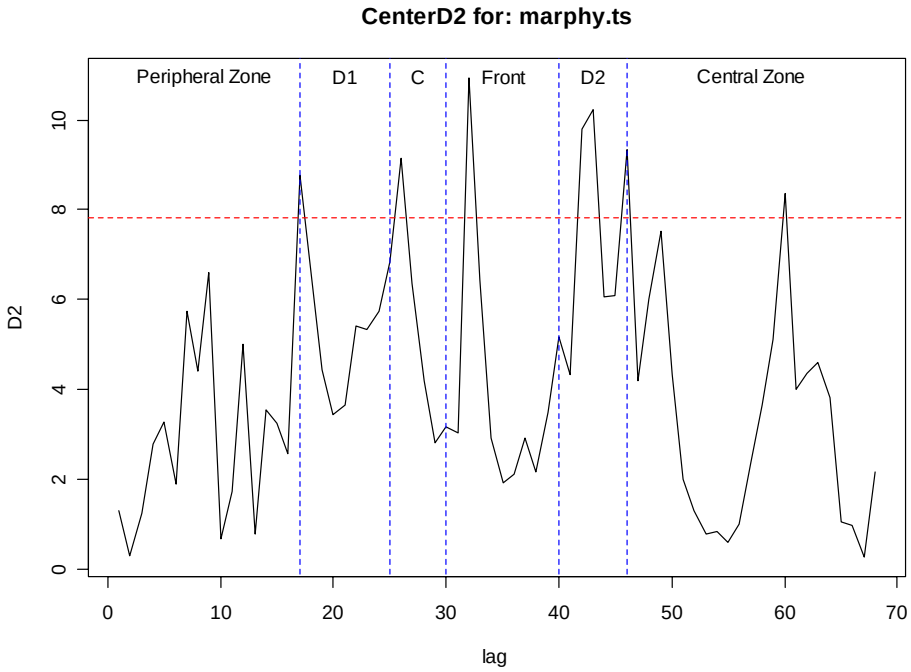


On constate que la distance de Malahanobis est la plus forte pour un décalage de 16. Nous conserverons donc cette valeur comme fenêtre pour le D2 au centre (voir ci-dessous) qui va nous indiquer la position de changements dans la composition des masses d'eau étudiées dans ce transect. Remarquons que si nous possédions deux ou plusieurs échantillons (par exemple, plusieurs transects, ou le même transect effectué à des moments différents, nous pourrions comparer leurs autocorrelations multiples à l'aide de la fonction ***CrossD2()***. Le résultat aurait été analysé de façon similaire à celui de l'***AutoD2()***, mais il aurait fallu distinguer deux crossD2: échantillon 1 en retard sur échantillon 2 et échantillon 2 en retard sur échantillon 1. Avec le même échantillon, on obtient évidemment un résultat identique à l'AutoD2 (non représenté):

```
> CrossD2(marphy.ts, marphy.ts)
```

Calculons maintenant le D2 au centre pour ce même échantillon:

```
> marphy.d2 <- CenterD2(marphy.ts, window=16)
> lines(c(17,17), c(-1,15), col=4, lty=2)    # Séparations
> lines(c(25,25), c(-1,15), col=4, lty=2)
> lines(c(30,30), c(-1,15), col=4, lty=2)
> lines(c(41,41), c(-1,15), col=4, lty=2)
> lines(c(46,46), c(-1,15), col=4, lty=2)
> text(c(8.5,21,27.5,35,43.5,57), 11, labels=c("Peripheral Zone",
+ "D1", "C", "Front", "D2", "Central Zone")) # Labels
> time(marphy.ts)[marphy.d2$D2 > marphy.d2$chisq]
[1] 17 26 32 42 43 46 60
```



La méthode du D2 au centre identifie les transitions qu'elle matérialise sous forme d'un pic unique par transition. Ainsi entre les stations 42 et 43, le D2 identifie 2 transitions. Mis à part la transition au point 60, les différentes zones préalablement identifiées le long du transect (traits verticaux bleu pointillé) sont retrouvées par la méthode, à plus ou moins une ou deux stations près. D'ailleurs, il faudrait même probablement reconsidérer l'identification préalable des zones sur base de la présente méthode qui donne un critère plus objectif (et multivarié!) pour l'identification des transitions entre les zones.

Analyse harmonique et analyse spectrale

L'analyse harmonique ou analyse en série de Fourier consiste à évaluer les composantes sinusoïdales d'un signal dont les fréquences sont des multiples de la fréquence fondamentale, si on appelle fréquence fondamentale la plus grande fréquence pouvant exister dans une chronique (ou série). Comment varient la période et la fréquence dans un développement en série de Fourier?

Etant donné la longueur totale de la série, soit n, par quel entier t faut-il multiplier la fréquence angulaire pour obtenir la plus grande fréquence? On a les égalités:

$$\omega.t = \frac{2\pi.t}{n} = \pi \Rightarrow t = \frac{n}{2}$$

La valeur de l'entier t tel que la fréquence soit la plus petite se déduit de la même façon:

$$\omega.t = \frac{2\pi.t}{n} = \frac{2\pi}{n} \Rightarrow t = 1$$

Le maximum en fréquence (associé au minimum en période), est égal à $1/2.\Delta t$. La représentation en série de Fourier s'écrira:

$$x_t = \overline{x} + \sum_{i=1}^{n/2} C_i \cos(i.\omega t - \phi_i)$$

où $\overline{x}$ est la moyenne du processus x_t, C_i est l'**amplitude** de l'harmonique i, $\omega_i = 2\pi.i/n$ correspond à la **vitesse angulaire**, ϕ_i est la **phase à l'origine**, n est le nombre d'observations. Cette équation peut aussi s'écrire:

$$x_t = \overline{x} + \sum_{i=1}^{n/2} A_i \sin i\omega t + B_i \cos i\omega t$$

Et si la série est préalablement centrée:

$$x_t = \sum_{i=1}^{n/2} A_i \sin i\omega t + B_i \cos i\omega t$$

Si on prend $i = 0$, le premier élément sera constant et égal à $\overline{x}$. En reprenant la dernière équation et en résolvant par les moindres carrés (cf. régression sinusoïdale), on a les formules:

$$A_i = \frac{2}{n}\sum_{t=1}^{n}(x_t.\sin i\omega t) \quad (A_i = C_i \sin\phi_i)$$

$$B_i = \frac{2}{n}\sum_{t=1}^{n}(x_t.\cos i\omega t) \quad (B_i = C_i \cos\phi_i)$$

On a d'autre part: $C_i = \sqrt{A_i^2 + B_i^2}$ et $\phi_i = \arctan\frac{A_i}{B_i}$

Les estimations de A_i et B_i sont indépendantes d'une harmonique à l'autre, et on peut estimer la variance de chacune par:

$$\sigma_i^2 = \frac{C_i^2}{2} \Rightarrow i < \frac{n}{2};\ \sigma_i^2 = C_i^2 \Rightarrow i = \frac{n}{2}$$

Leurs pourcentages de variance respectifs par rapport à la variance totale du processus σ_T^2 est:

$$\frac{C_i^{\,2}}{2\sigma_T^{\,2}} \Rightarrow i < \frac{n}{2};\ \frac{C_i^{\,2}}{\sigma_T^{\,2}} \Rightarrow i = \frac{n}{2}$$

Ces valeurs permettent de reconnaître et de sélectionner les *m* harmoniques les plus remarquables. Par soustraction vis-à-vis de la série originale, on obtiendra une série résiduelle où les oscillations dominantes sont absentes:

$$x_t'' = x_t - x_t'$$

La grandeur $C_i^2 = A_i^2 + B_i^2$ définit une fonction discrète dite **périodogramme**. Par construction, on sait que:

$$\sum_{i=0}^{n/2} C_i^{\,2} = \sum_{t=1}^{n} x_t^{\,2}$$

Le périodogramme définit pour chaque valeur de *i*, la contribution qu'apporte la composante sinusoïdale ayant la période n/i à la somme des carrés des x_t. On peut tester la signification de chaque harmonique en fonction du pourcentage de variance expliquée (Anderson, 1971). Avec *n* observations, *q* la plus grande période harmonique calculée (normalement $q = n/2$), et à un seuil de probabilité α (5% ou 1%), on a le pourcentage:

$$\frac{2}{n}\log_e(1-\sqrt[q]{1-\alpha}).100$$

Si le pourcentage de variance associé à une harmonique est supérieur à cette valeur, sa période est significativement différente de zéro au seuil α.

> *L'estimation fournie par le périodogramme donne un diagramme très irrégulier. Pour y remédier, on divise l'intervalle des fréquences en bandes centrées autour de* p *fréquences équidistantes. Ensuite, on calcule une moyenne mobile pondérée associée à chaque bande de fréquence. Les poids les plus élevés correspondent aux fréquences voisines de celles autour de laquelle est centrée la bande (fenêtre spectrale). Le lissage du périodogramme correspond à une estimation du pouvoir spectral du processus. Cependant, en pratique, on calculera le pouvoir spectral directement à partir de la fonction d'autocovariance présentée plus haut.*

La **convolution** se définit comme suit: soit deux fonctions *f* et *g* de deux variables, on appelle la convolée de *f* et *g* ($f \times g$) la fonction $h(x)$, d'une seule variable définie par:

$$h(x) = \int_{-\infty}^{+\infty} f(x-y).g(y).dy = \int_{-\infty}^{+\infty} g(x-y).f(y).dy$$

Dans le cas discret, si on considère des séries et non des fonctions, on définit la convolée par:

$$z_i = \sum_{j=-\infty}^{+\infty} x_j.y_{i-j} = \sum_{j=-\infty}^{+\infty} x_{i-j}.y_j$$

- si y_j est positif ou nul lorsque *j* varie de $-J$ à $+J$ et si la somme des y_j vaut 1, la convolution des x_{i-j} par les y_j équivaut à une moyenne mobile.

- si une série y_j est nulle pour les indices positifs, on a: $z_i = \sum_{j=-J}^{0} x_{i-j}.y_J$

Une fonction qui subit une convolution passe à travers un filtre linéaire. Le signal de sortie est donné par la convolution du signal d'entrée avec une fonction qui caractérise le filtre.

Soit $x(t)$ une fonction (et non une série) qui n'est pas forcément périodique, on appelle **transformée de Fourier**:

$$tx(f) = \int_{-\infty}^{+\infty} e^{-2\pi if}.x(t).dt \quad \text{et transformée inverse} \quad rx(t) = \int_{-\infty}^{+\infty} e^{2\pi if}.tx(f).df$$

Si $x(t)$ est réelle, $tx(j)$ est paire et on ne considère que des fréquences $f > 0$.

La convolution et la transformée de Fourier ont les propriétés suivantes. Soient $x(t)$ et $g(t)$ et $tx(f)$ et $tg(f)$. La transformée de Fourier fait correspondre à la convolution des deux fonctions la multiplication de leurs transformées. Réciproquement, la transformée de la fonction $h(t)$ obtenue en multipliant $g(t)$ par $x(t)$ sera donnée par la convolution d'une fonction $tg(f)$, appelée **fonction de transfert**, par $tx(f)$: $th(f) = tx(f) \; . \; tg(f)$.

Utiliser un filtre linéaire $g(t)$, c'est opérer une convolution. La transformée du filtre est sa fonction de transfert $tg(f)$. C'est en général une fonction complexe pouvant être décomposée en module et phase. Le module $|tg(f)| = G(f)$ est appelé **gain du filtre**. On a l'**amplitude du spectre** (filtre linéaire):

$$G(\omega) = \left| \sum_{t=0}^{T} a(t).e^{i\omega t} \right|^2$$

Par exemple, dans le cas d'une moyenne mobile sur un intervalle $t-T$, $t+T$, la fonction de transfert du filtre correspondant est:

$$tm(f) = \frac{\sin 2\pi ft}{2\pi t}$$

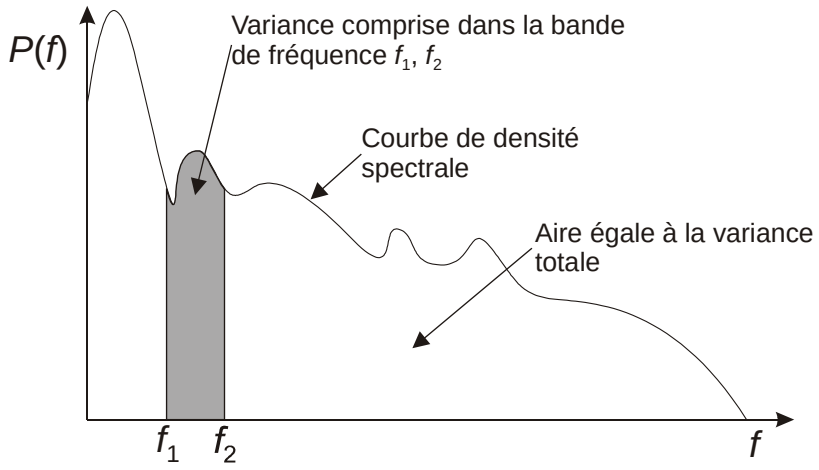
La formule de la transformée de Fourier rappelle la décomposition de Fourier avec T tendant vers l'infini. Le coefficient de la composante $e^{2\pi ift}$ est $tx(f)$; ainsi, plus la transformée décroît en module et plus les composantes de hautes fréquences diminuent.

Le **spectre** d'un processus correspond à une décomposition de la variance sur l'ensemble des fréquences. Il est obtenu en effectuant la transformée de Fourier de la fonction d'autocovariance ou d'autocorrélation (dans ce cas, la variance du processus est normée à un). Le **pouvoir spectral** est donné par:

$$P(f) = S_{xx}(f) = \sum_{\theta=-N}^{+N} C_\theta.e^{-i\omega\theta} = \sum_{\theta=-N}^{+N} C_\theta.\cos\omega\theta \quad \text{avec} \quad \theta = 1,2,...,n/4$$

où θ est le décalage choisi (lag, en anglais). La variance totale est définie par:

$$\sigma_x^{\,2} = \int_0^\pi P(f).df \quad \text{avec} \quad f = 1,2,...,n/4$$



Représentation du spectre d'une série stationnaire: analyse de variance selon les fréquences f.

La courbe obtenue ressemblera à celle d'un périodogramme car on dispose de peu de valeurs de θ. Pour obtenir une courbe lissée, on calculera une moyenne mobile pondérée associée à une bande de fréquences. Les poids les plus élevés correspondront aux fréquences voisines de celle autour de laquelle est centrée la bande. On effectuera une pondération de la fonction d'autocovariance initiale (fenêtre spectrale):

$$S_{xx}(f) = \sum_{\theta=-N}^{+N} k_\theta C_\theta \, \mathrm{e}^{-i\omega\theta}$$

où k_θ est la fonction de pondération choisie. Plus simplement, on peut lisser directement le périodogramme, par exemple:

$$\frac{S_{xx}(f_{j-1}) + S_{xx}(f_j) + S_{xx}(f_{j+1})}{3}$$

L'estimation lissée à la fréquence f_j d'un spectre, notée $\bar{S}_{xx}(f)$ s'obtient par:

$$\bar{S}_{xx}(f) = \int_{-\infty}^{+\infty} \omega_f(g).S_{xx}(f-g).dg \quad \text{avec} \quad \int_{-\infty}^{+\infty} \omega_f(g).dg = l$$

Mais convoler $S_{xx}(f)$ qui est égal à la transformée de Fourier de $C_{xx}(l)$ avec la fonction ω_f, correspond à la multiplication de C_{xx} par ω_t dont ω_f est la transformée. Par conséquent, $\bar{S}_{xx}(f)$ correspond à la transformée de Fourier directe du produit de convolution de C_{xx} par ω_t. Les fonctions ω_f et ω_t sont appelées respectivement, **fenêtre spectrale** (« spectral window ») et **fenêtre des décalages** (« lag window »).

Multiplier C_{xx} par ω_t a pour effet d'accélérer la décroissance en valeur absolue de la fonction d'autocovariance. Ceci implique que ω_t soit une fonction du décalage l nulle assez vite au-delà d'un décalage limite M. M est l'**ouverture de la fenêtre**. Plus M est faible, plus le lissage est grand. La plus simple des fonctions (rectangulaire) correspond à $\omega_t(l) = 1 \Rightarrow l < M$; $\omega_t(l) = 0 \Rightarrow l > M$. L'inconvénient est que sa transformée n'est pas toujours positive. De nombreux auteurs ont proposé des lissages divers (Daniell, Bartlett, Tuckey, Parzen,…). Le choix de l'intervalle de lissage M est aussi prépondérant que le choix de la fenêtre. Tant S+ que R proposent de telles méthodes via des fonctions de lissage très souples, telles que le « kernel smoothing » (voir la documentation des logiciels respectifs pour plus d'information).

Références:

Kendall, M., 1976. Time-series. *Charles Griffin & Co Ltd*. 197 pp.

Legendre, L. & P. Legendre, 1984. Ecologie numérique. Tome 2: La structure des données écologiques. *Masson, Paris*. 335 pp.

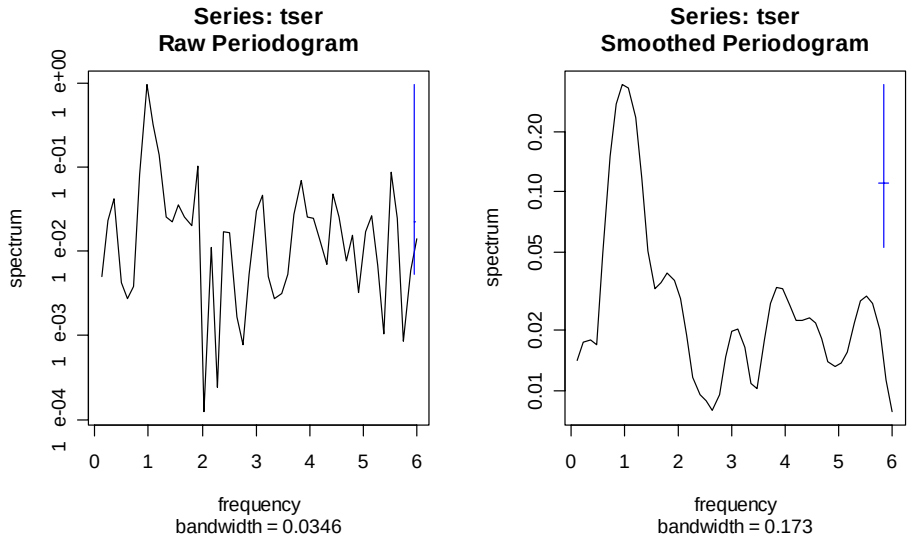
Malinvaud, E., 1978. Méthodes statistiques de l'économétrie. *Dunod, Paris*. 846 pp.

Philips, L. & R. Blomme, 1973. Analyse chronologique. *Université catholique de Louvain, Vander Ed.* 339 pp.

Exemple:

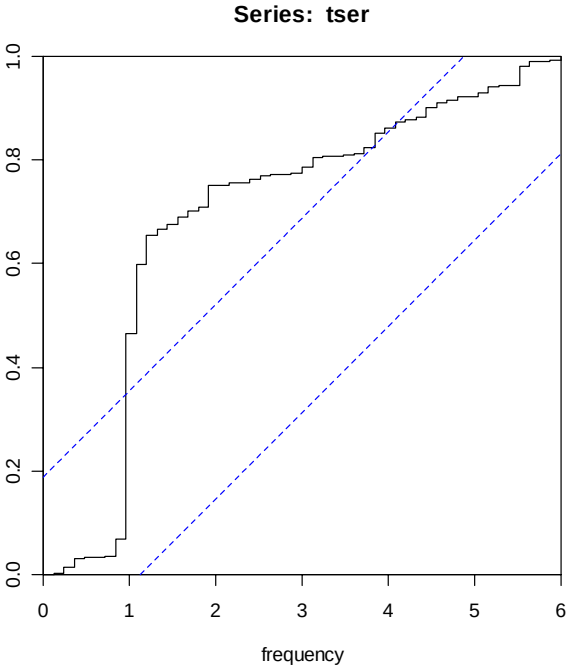
Sous S+ et R, la fonction ***spectrum()*** trace à la fois le périodogramme brut et le périodogramme lissé. Ainsi, les spectres (périodogrammes) bruts et lissés de ***tser*** sont obtenus côte à côte par:

```
> par(mfrow=c(1,2))              # Place 2 graphes côte à côte
> spectrum(tser)                 # Périodogramme brut
> spectrum(tser, spans=c(3,5))   # Périodogramme lissé
> par(mfrow=c(1,1))              # Valeur par défaut pour mfrow
```



Ils sont le pendant de l'analyse harmonique et de l'analyse spectrale proposées dans PASSTEC 2000 (respectivement ***Cycles -> Analyse harmonique*** et ***Cycles -> Analyse Spectrale*** en mode ***Univarié***), même si les algorithmes (utilisation d'un « kernel smoother » pour lisser le périodogramme sous S+ ou R) et les choix d'échelles pour les graphes ne donnent pas des résultats tout à fait similaires entre les deux programmes. S+ et R offrent aussi la possibilité de représenter le **périodogramme cumulé** qui permet souvent d'identifier plus clairement les fréquences significatives (ici, 1). Sous S+, il faut préalablement prendre soin de charger la librairie ***'MASS'***. Sous R, ***cpgram()*** est disponible dans la librairie ***'stats'***.

```
> library(MASS)                  # Seulement dans S+
> cpgram(tser)
```


Series: tser
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
frequency

Analyse spectrale croisée

L'analyse spectrale peut être étendue au cas des signaux bivariés. On pourra détecter les oscillations qui sont importantes simultanément dans deux séries, tester la corrélation entre les mouvements de même période, et estimer le déphasage éventuel entre ces oscillations.

On définit la corrélation avec retard entre deux signaux: $C_{xy}(\theta)$ et $C_{yx}(\theta)$. Les formules sont comparables à celles de l'autocorrélation, mais au lieu de considérer au numérateur le produit de valeurs réduites sans décalage et après décalage, on considère leur produit décalé pour deux descripteurs (cross-corrélation). Il y a donc deux fonctions de corrélation avec retard selon que l'on choisit tel ou tel descripteur en avance ou en retard sur l'autre.

Les fonctions de corrélation croisées ne sont pas paires: $\tilde{C}_{xy}(\theta) \neq \tilde{C}_{yx}(\theta)$. On calculera un spectre croisé qui correspondra à:

$$\tilde{S}_{xy}(f) = \tilde{p}_{xy}(f) + i.\tilde{q}_{xy}(f)$$

avec $\tilde{p}_{xy}(f)$: **cospectre réel** et $i.\tilde{q}_{xy}(f)$: **spectre de quadrature**. Ce spectre montre les liens entre les processus *x* et *y*, fréquence par fréquence. S'il n'y a pas de déphasage, $\tilde{q}_{xy}(f)$ est nul. C'est parce que la fonction d'autocovariance n'est pas paire que le spectre est un nombre complexe. On peut écrire aussi:

$$\tilde{S}_{xy}(f) = \tilde{\rho}_{xy}(f).\mathrm{e}^{i.\varphi_{xy}(f)}$$

avec $\tilde{\rho}_{xy}(f)$: l'amplitude et $\varphi_{xy}(f)$: la phase en radians (l'argument du nombre complexe du spectre croisé). Si $\varphi = p$ on a opposition complète de phase.

$$\tilde{\rho}^2_{xy}(f) = \tilde{p}^2_{xy}(f) + i.\tilde{q}^2_{xy}(f) \quad \text{et} \quad \tan\tilde{\varphi}_{xy}(f) = \frac{\tilde{q}_{xy}(f)}{\tilde{p}_{xy}(f)}$$

Le coefficient de **cohérence** teste l'intensité de la liaison linéaire entre deux processus (corrélation entre deux signaux pour chaque fréquence). Il varie entre 0 et 1. La cohérence au carré entre deux oscillations de fréquence *f* est de la forme:

$$\tilde{Co}^2_{xy}(f) = \frac{\tilde{\rho}^2_{xy}(f)}{\tilde{S}_{xx}(f).\tilde{S}_{yy}(f)} = \frac{\tilde{p}^2_{xy}(f) + i.\tilde{q}^2_{xy}(f)}{\tilde{S}_{xx}(f).\tilde{S}_{yy}(f)} = \frac{\left|\tilde{S}_{xy}(f)\right|^2}{\tilde{S}_{xx}(f).\tilde{S}_{yy}(f)}$$

Références:

Jenkins, G.M. & D.G. Watts, 1968. Spectral analysis and its applications. *Holden-Day, San Francisco*. 500 pp.

Kendall, M., 1976. Time-series. *Charles Griffin & Co Ltd*, 197 pp.

Legendre, L. et P. Legendre, 1984. Ecologie numérique. Tome 2: La structure des données écologiques. *Masson, Paris.* 335 pp.

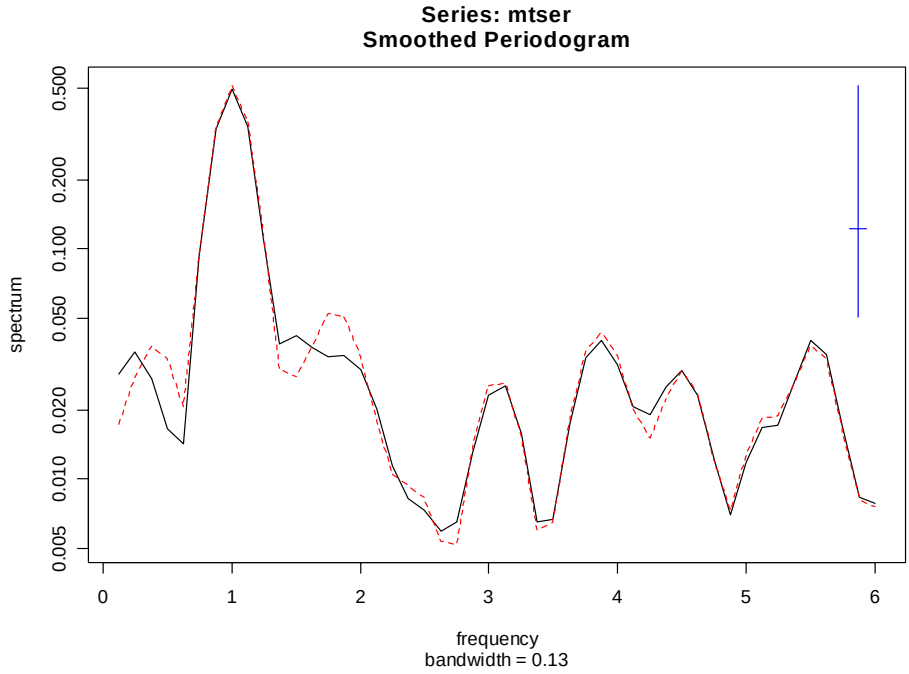
Malinvaud, E., 1978. Méthodes statistiques de l'économétrie. *Dunod, Paris.* 846 pp.

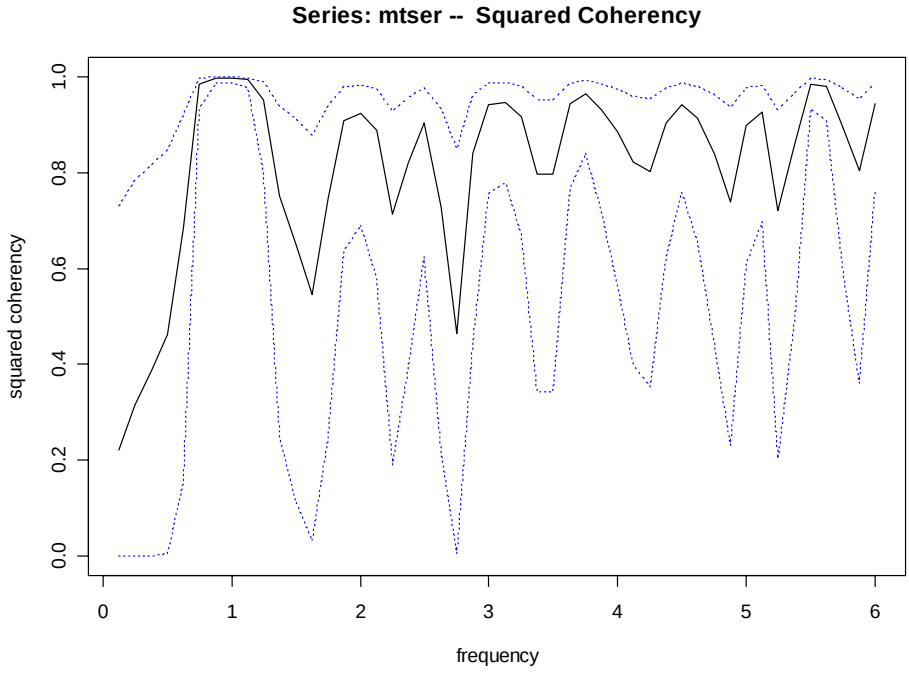
Philips, L. & R. Blomme, 1973. Analyse chronologique. *Université Catholique de Louvain. Vander Ed.* 339 pp.

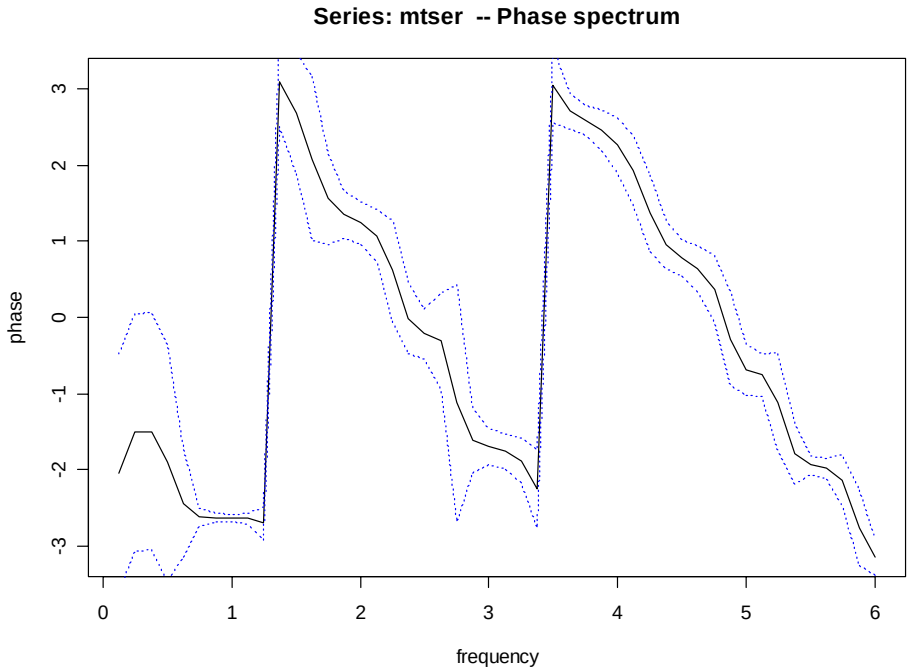
Exemple:

En mode multivarié, S+ et R proposent aussi des outils pour mener à bien une analyse spectrale croisée. La fonction ***spectrum()*** effectue un traitement similaire à ***Cycles -> Analyse Spectrale Croisée*** en mode ***Multivarié*** sous PASSTEC 2000. Sous R, la méthode renvoie un objet *'spec'* qui possède une méthode ***plot()*** permettant de tracé le périodogramme (par défaut), le graphe de cohérence au carré (**plot.type = "coherency"**) ou le graphe des phases (**plot.type = "phase"**). Tous ces graphes sont accompagnés d'une indication de l'intervalle de confiance:

```
> mtser.spc <- spectrum(mtser, spans=c(3,3))
> plot(mtser.spc, plot.type="c")        # Seulement sous R
> plot(mtser.spc, plot.type="p")        # Idem
```







Avec quelques lignes de code supplémentaires on peut obtenir un résultat similaire sous S+:

```
> mtser.spc <- spectrum(mtser, spans=c(3,3))
> plot(mtser.spc$freq, mtser.spc$coh, type="l", ylim=c(0,1),
+ xlab="frequency", ylab="squared coherency",
+ main="Series: mtser - Squared Coherency")
> gg <- 2/mtser.spc$df
> se <- sqrt(gg/2)
> coh <- sqrt(mtser.spc$coh)
> lines(mtser.spc$freq, (tanh(atanh(coh) + 1.96*se))^2, lty=2,
+ col=2)
> lines(mtser.spc$freq, pmax(0, tanh(atanh(coh) - 1.96*se))^2,
+ lty=2, col=2)
> phase <- (mtser.spc$phase + pi)%%(2*pi) - pi
> plot(mtser.spc$freq, phase, type="l", ylim=c(-pi,pi),
+ xlab="frequency", ylab="phase",
+ main="Series: mtser - Phase Spectrum")
> cl <- asin(pmin(0.9999, qt(0.95, 2/gg-2)*
+ sqrt(gg*(coh^{-2} - 1)/(2*(1-gg)))))
> lines(mtser.spc$freq, phase + cl, lty=2, col=2)
> lines(mtser.spc$freq, phase - cl, lty=2, col=2)
```

Points de retournement

Il s'agit ici de tester si la série étudiée correspond ou non à une suite d'événements purement aléatoires: Kendall (1976) a proposé une série de tests de vérification. De plus, des méthodes graphiques utilisant le positionnement des points de retournement pour tracer automatiquement des enveloppes autour des données existent.

Un **point de retournement** (PR; « turning point », en anglais), pic ou creux, est défini par une ou plusieurs valeurs encadrées par des valeurs plus faibles ou plus fortes. Il faut au moins trois observations successives pour définir un pic ou un creux: x_{t-1}, x_t, x_{t+1}. Si la série est purement aléatoire, les trois valeurs peuvent apparaître dans n'importe quel ordre (de six façons différentes). Un point de retournement ne peut apparaître que 4 fois sur 6, soit avec une probabilité de 2/3.

Considérons une série x_t contenant n observations et définissons une variable U par t = 2,…, n-1 telle que $U_t = 1$ si $x_{t-1} < x_t > x_{t+1}$ (pic) ou si $x_{t-1} > x_t < x_{t+1}$ (creux) et $U_t = 0$ dans tous les autres cas. Le nombre p de points de retournement est donné par:

$$p = \sum_{t=2}^{n-1} U_t$$

L'espérance de ce nombre p vaut:

$$\mathrm{E}(p) = \frac{2}{3}(n-2)$$

Soit, au hasard, il apparaît un point de retournement après une observation et demie. La variance de p est une fonction de n:

$$\sigma^2(p) = \frac{16n-29}{90}$$

A partir de cette formule, on peut tester une valeur observée par rapport à une valeur normale ayant cette variance. Ibanez (1982) a calculé la probabilité qu'apparaisse un point de retournement au hasard, dans une série de n nombres, à une position donnée. La formule générale donnant la probabilité $P(t)$ de trouver au hasard un point à la $t^{\text{ème}}$ place parmi n observations est:

$$\mathrm{P}(t) = 2.\frac{1}{n(t-1)!(n-t)!}$$

La **quantité d'information** I associée est donnée par la formule de Shannon:

$$I = -\log_2 \mathrm{P}(t)$$

Ainsi, on peut définir pour une série sa quantité d'information en interpolant linéairement les valeurs intermédiaires entre points de retournement.

Considérant les seuils d'information statistiques (5% soit 4,3 bits; 10% soit 3,32 bits), on peut reconnaître les évènements majeurs. Cette notion d'information, qui est dépendante des valeurs absolues du signal, peut servir de critère pour l'enregistrement de données en continu, pour la sélection des descripteurs utiles dans un traitement numérique, et comme test de reconnaissance de l'échelle optimale de l'échantillonnage d'un descripteur (celle qui révèle des évènements les plus distincts du hasard). Cette mesure peut aussi servir à découper des enregistrements.

Référence:

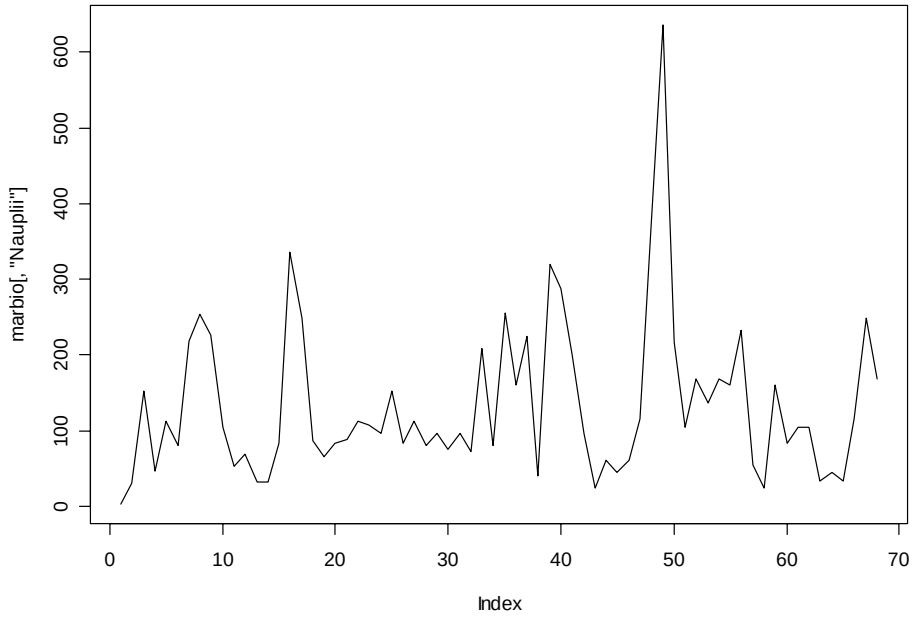
Ibanez, F., 1982. Sur une nouvelle application de la théorie de l'information à la description des séries chronologiques plantoniques. *J. Exp. Mar. Biol. Ecol.*, 4:619-632.

Kendall, M.G., 1976. Time-series, 2nd ed. *Charles Griffin & Co, London.*

Exemple:

Etudions les extrema de la série ***Nauplii*** du data frame ***marbio***, dont voici le graphe des données brutes:

```
> data(marbio)
> plot(marbio[, "Nauplii"], type="l")
```



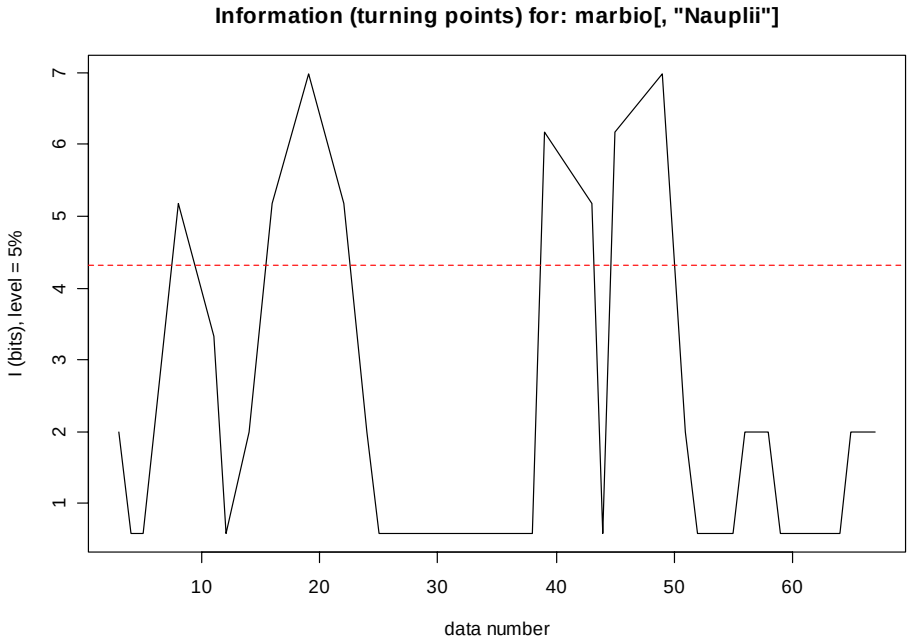
Les points de retournements s'obtiennent en appliquant la fonction ***turnpoints()***:

```
> Nauplii.tp <- turnpoints(marbio[, "Nauplii"])
> summary(Nauplii.tp)
Turning points for: marbio[, "Nauplii"]

nbr observations  : 68
nbr ex-aequos     : 2
nbr turning points: 45 (first point is a peak)
E(p) = 44 Var(p) = 11.76667 (theoretical)

   point type       proba      info
1      3 peak 0.250000000 2.0000000
2      4  pit 0.666666667 0.5849625
3      5 peak 0.666666667 0.5849625
4      6  pit 0.250000000 2.0000000
5      8 peak 0.027777778 5.1699250
6     11  pit 0.100000000 3.3219281
7     12 peak 0.666666667 0.5849625
```

```
8     14  pit 0.250000000 2.0000000
9     16 peak 0.027777778 5.1699250
10    19  pit 0.007936508 6.9772799
11    22 peak 0.027777778 5.1699250
12    24  pit 0.250000000 2.0000000
13    25 peak 0.666666667 0.5849625
14    26  pit 0.666666667 0.5849625
15    27 peak 0.666666667 0.5849625
16    28  pit 0.666666667 0.5849625
17    29 peak 0.666666667 0.5849625
18    30  pit 0.666666667 0.5849625
19    31 peak 0.666666667 0.5849625
20    32  pit 0.666666667 0.5849625
21    33 peak 0.666666667 0.5849625
22    34  pit 0.666666667 0.5849625
23    35 peak 0.666666667 0.5849625
24    36  pit 0.666666667 0.5849625
25    37 peak 0.666666667 0.5849625
26    38  pit 0.666666667 0.5849625
27    39 peak 0.013888889 6.1699250
28    43  pit 0.027777778 5.1699250
29    44 peak 0.666666667 0.5849625
30    45  pit 0.013888889 6.1699250
31    49 peak 0.007936508 6.9772799
32    51  pit 0.250000000 2.0000000
33    52 peak 0.666666667 0.5849625
34    53  pit 0.666666667 0.5849625
35    54 peak 0.666666667 0.5849625
36    55  pit 0.666666667 0.5849625
37    56 peak 0.250000000 2.0000000
38    58  pit 0.250000000 2.0000000
39    59 peak 0.666666667 0.5849625
40    60  pit 0.666666667 0.5849625
41    62 peak 0.666666667 0.5849625
42    63  pit 0.666666667 0.5849625
43    64 peak 0.666666667 0.5849625
44    65  pit 0.250000000 2.0000000
45    67 peak 0.250000000 2.0000000
```

On observe donc 45 points de retournement dans la série, alors que l'espérance théorique du nombre de points de retournement pour une série purement aléatoire, E(*p*), est de 44. La série possède donc un grand nombre de fluctuations aléatoires peu porteuses d'information. Si le nombre total de points de retournement est proche de celui d'une série aléatoire, il se peu que leur répartition ne soit pas homogène partout. Ouvrons une nouvelle fenêtre graphique et traçons le graphe de l'information liée à ces points de retournement:

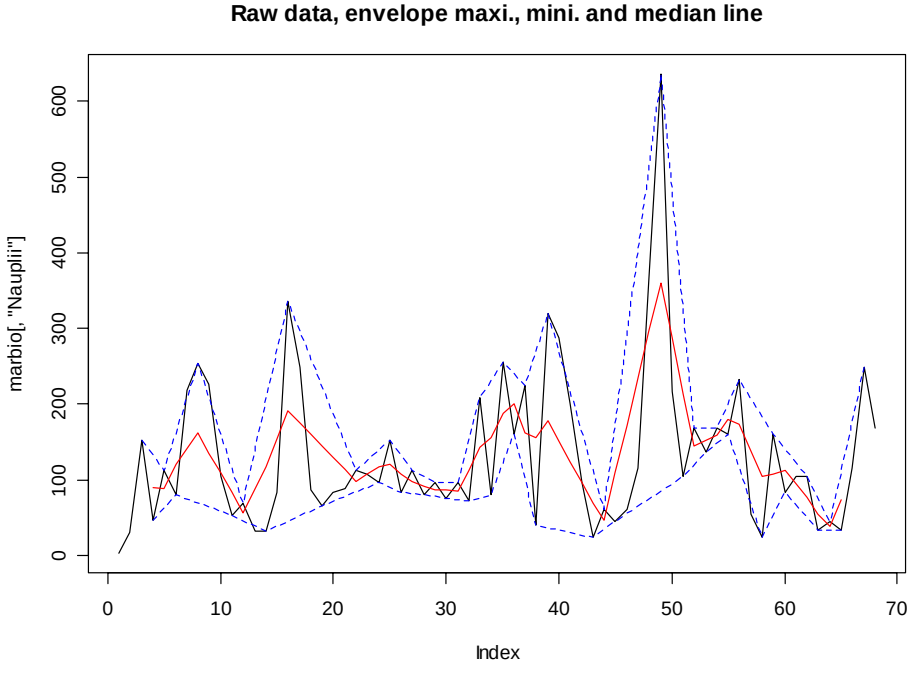
```
> plot(Nauplii.tp)
```



On constate qu'effectivement, si la série est peu porteuse d'information (succession erratique de pics et de creux) entre les observations 25 et 38 et aux alentours des observations 54 et 62, elle porte une information significative (au seuil p = 5%) à la 9ème observation et aux alentours de la 19ème, ainsi qu'entre la 38ème et la 52ème observation. Comme le graphe de la série brute est encore ouvert dans une autre fenêtre, il est possible d'afficher la série simultanément pour visualiser les zones à faible et à forte information.

On peut aussi tracer les enveloppes maximum et minimum (interpolation linéaire entre les pics et les creux, respectivement), ainsi que les points médians (points situés à mi-chemin entre les deux enveloppes) sur le graphique des données brutes par:

```
> plot(marbio[, "Nauplii"], type="l")
> lines(Nauplii.tp)
> title("Raw data, envelope maxi., mini. and median line")
```



On voit mieux ici que les zones porteuses d'informations correspondent à un ensemble de segments successifs tous croissants ou décroissants, par exemple aux alentours de la 19ème observation. Les enveloppes s'éloignent donc quelque peu des données à ces endroits. Par contre, les zones à faible information caractérisées par une succession rapide de segments croissants et décroissants, comme entre les observations 25 et 38 voient presque tous les points reliés soit par l'enveloppe supérieure, soit par l'enveloppe inférieure.

On peut extraire un vecteur représentant la position des extrema dans la série initiale avec ***extract()*** en choisissant librement la valeur à attribuer aux pics (**peak = …**), aux creux (**pit = …**) et aux autres points (**no.tp = …**). En reprenant la série exemple traitée dans le chapitre régularisation (***releve***, série ***Melosul***), on peut obtenir un vecteur de pondération qui donne plus d'importance aux extrema par:

```
> data(releve)
> weight.tp <- extract(turnpoints(releve[,"Melosul"]), no.tp=1,
peak=5, pit=5)
> weight.tp
 [1] 1 5 1 1 5 1 5 1 5 5 1 1 1 1 1 5 5 5 1 5 1 5 5 5 5 5 5 1 5 5 5
[32] 1 5 1 1 5 1 5 5 1 5 1 1 1 1 5 5 5 1 5 5 5 1 5 5 1 5 5 1 1 1
```

Ce vecteur peut être utilisé de la même manière que le vecteur ***weight*** obtenu précédemment dans le chapitre régularisation. Dans le cas présent, une pondération de 5 est attribuée aux points jugés importants dans la série (les extrema), alors qu'une pondération de 1 seulement est attribuée à tous les autres points. Cette pondération permet de calculer le meilleur pas de temps lors de la régularisation, pas qui non seulement contient un maximum de points de la série originelle, mais aussi et surtout, un maximum de ses extrema (voir l'exemple dans le chapitre régularisation pour la procédure à suivre).

Tournogramme

Le **tournogramme** est une généralisation des notions d'information liées aux points de retournement. Plus la série comporte de pics ou de creux erratiques, alternant avec une haute fréquence, et plus elle sera difficile à interpréter par l'écologiste. Un critère de qualité de la représentation d'une série de mesures d'une variable quantitative doit tenir compte du nombre de valeurs successives croissantes (ou décroissantes) intervenant en moyenne.

L'**indice de monotonie** retenu ici, est la quantité d'information associée à la probabilité d'obtenir par hasard le nombre d'extrema observés, la série étant supposée être aléatoire. Le tournogramme est la courbe de variation de cet indice, en fonction des pas de temps sur lesquels soit les premières observations ont été extraites, soit des moyennes ont été calculées sur les intervalles, puis les points de retournement comptabilisés. Il peut aider à orienter le choix de l'échelle, ou des échelles, de représentation des processus. La distribution de fréquence de la longueur (en unité de temps) des séquences monotones, pour un pas de temps donné, peut aider à l'interprétation de changements progressifs.

Définition de la **distribution de Gleissberg**: Gleissberg a proposé un algorithme de calcul de la distribution exacte du nombre d'extrema dans le cas aléatoire. Le nombre $d_{n,p}$ de séries différentes de n valeurs ayant p extrema est donné par:

$$d_{n,p} = (p+1).d_{n-1,p} + 2.d_{n-1,p-1} + (n-p-1).d_{n,p-1}$$

avec:

$$\begin{aligned} d_{n,1} &= 2.d_{n-1,1+2.d_{n-1,0}} \\ d_{n-1,1} &= 0 \qquad \text{pour } n \le 3 \\ d_{n,0} &= 2 \qquad \text{pour } n \ge 2 \end{aligned}$$

D'où la probabilité d'observer au hasard p points de retournement dans une série de n valeurs:

$$\mathrm{P}(p) = \frac{d_{n,p}}{n!}$$

La distribution de Gleissberg est asymptotiquement normale. Ainsi, lorsque n est grand ($n > 50$), on pourra aussi tester le nombre p de points de retournement par référence à la probabilité associée à une loi normale de moyenne:

$$\overline{p} = \frac{2}{3}(n-2)$$

et de variance (comme nous l'avons vu au paragraphe 'points de retournement'):

$$\sigma^2(p) = \frac{16n-29}{90}$$

Pour une série observée, nous pouvons nous demander si le nombre d'extrema obtenus correspond à celui d'une série aléatoire. Un test statistique est envisageable avec H_0 = la série est aléatoire. Selon que l'on désire savoir si la série est ou non aléatoire (test bilatéral), ou seulement si elle est plus ou moins monotone qu'une série aléatoire (test unilatéral), on définira H_1 comme suit:

- test bilatéral, la série n'est pas aléatoire, H_1: $p(|K - \mu_K| \ge |k - \mu_K|)$
- test unilatéral à gauche, la série est plus monotone, H_1: $p(K - \mu_K \le k - \mu_K)$

- test unilatéral à droite, la série est moins monotone, H_1: $p(K - \mu_K \geq k - \mu_K)$

avec *K*, l'ensemble des extrema possibles allant de 0 à $n - 2$; μ_K, l'espérance du nombre d'extrema lorsque la série est purement aléatoire (nous avons vu à la section précédente que $\mu_K = 2[n - 2]/3$) et *k*, le nombre d'extrema denombrés dans la série observée. La probabilité *p* est calculée par rapport soit à la distribution de Gleissberg exacte pour $n \leq 50$, soit à une distribution normale qui l'approxime pour $n > 50$. En pratique, seuls le test bilatéral (qui indique si la série est aléatoire ou non) et le test unilatéral à gauche (qui permet de savoir si la série est moins monotone qu'une série aléatoire, c'est-à-dire, qui maximise l'information en éliminant au maximum les fluctuations aléatoires) sont intéressants dans le cadre du tournogramme.

L'**indice de monotonie** proposé ici pour une série est donc la quantité d'information:

$$I = -\log_2(p)$$

p étant l'une des probabilités pour les tests précédentes. Dans le cas du test bilatéral, la quantité *I* sera négative par convention lorsque la série est moins monotone qu'une série aléatoire.

Le **tournogramme** (Dallot & Etienne, 1990) est donc la fonction décrivant la variation de *I* en fonction d'échelles d'observation différentes: soit on conservera de la série 1 valeur sur 2, sur 3, et ainsi de suite,… , soit on considèrera des moyennes pour des intervalles de temps *u* de plus en plus grands. Pour une fenêtre de *u* points sélectionnés, on peut obtenir des résultats différents selon que la fenêtre est positionnée à partir du premier point ou des suivants (*u*-1 estimations possibles). Lorsque le tournogramme est calculé uniquement à partir du premier point, il est dit tournogramme simple. Lorsqu'il est calculé pour toutes les valuers de départ possible, il est dit tournogramme complet. Dans ce dernier cas, les résultats sont résumés par 3 courbes: l'information moyenne, encadrée par les infomations minimales et maximales observées pour les différents points de départ.

Un problème d'estimation non négligeable se rencontre pour des valeurs faibles de *u* et lorsque la série comporte un grand nombre d'ex aequo nuls. Dans ce cas, on condense en une seule les valeurs nulles successives. Cette technique est évidemment inacceptable pour des enregistrements trop lacunaires.

Le **nombre de phases**, ou séquences monotones, *N* de longueur *d* séparant deux points de retournement dans une série purement aléatoire comportant *n* observations est donné par (Wallis & Moore, 1941):

$$N_d = \frac{2(n-d-2)(d^2+3d+1)}{(d+3)!}$$

Le nombre total de phases (entre 1 et *n*-3) est égal à:

$$N_t = \frac{1}{3}(2n-7)$$

Pour tester l'écart par rapport à la fluctuation aléatoire, on considère la distribution observée des phases pour trois longueurs différentes: 1, 2 et supérieure ou égale à 3. On compare avec le nombre de phases théoriques, en calculant le test χ^2 proposé par Wallis & Moore (1941) pour tenir compte de la non-indépendance des longueurs de phases. Le nombre de degrés de liberté à associer au χ^2 est de 2,5 pour des valeurs de χ^2 calculées supérieures ou égales à 6,3. Il est de 2 pour des valeurs de 6 $\chi^2/7$ inférieures à 6,3. Le calcul du nombre de phases et le test χ^2 associé ne sont pas encore inclus dans PASTECS.

L'application la plus évidente du tournogramme est dans le traitement de données enregistrées en continu. En tant qu'analyse préliminaire des échelles pertinentes d'observation, elle peut soit aider à la planification ultérieure, soit à l'interprétation. Elle

permet aussi de définir la meilleure représentation graphique d'une série (celle donnant le maximum d'information). Une comparaison avec d'autres fonctions telles que le variogramme ou le spectre peut être très profitable.

Références:

Dallot, S. & M. Etienne, 1990. Une méthode non paramétrique d'analyse des séries en océanographie biologique: les tournogrammes. *Biométrie et océanographie – Société de biométrie, 6, Lille, 26-28 mai 1986. IFREMER, Actes de colloques,* 10:13-31.

Johnson, N.L. & Kotz, S., 1969. Discrete distributions. *J. Wiley & sons, New York*, 328 pp.

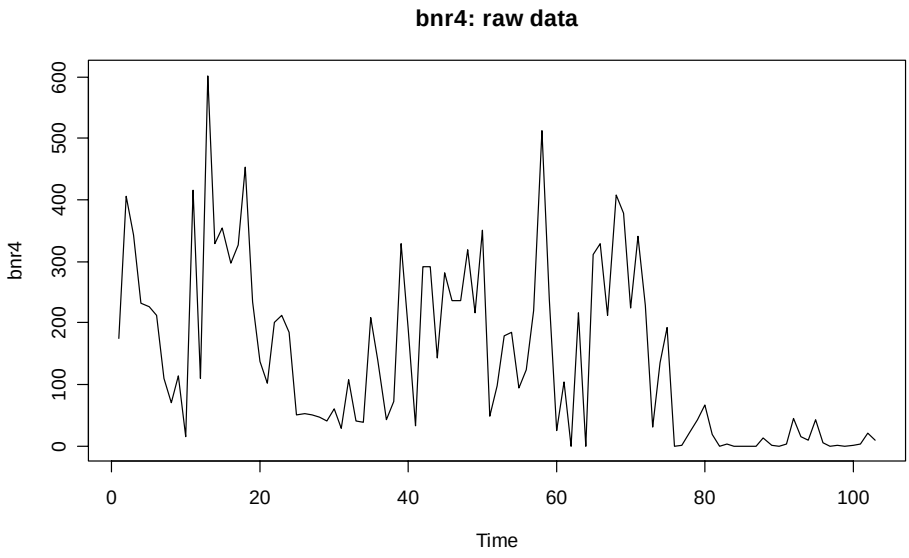
Kendall, M.G., 1976. Time-series, 2nd ed. *Charles Griffin & co, London.*

Wallis, W.A. & G.H. Moore, 1941. A significance test for time series. *National Bureau of Economic Research, tech. Paper n°1.*

Exemple:

On peut extraire la série 4 du jeu de données ***bnr*** de la librairie PASTECS et la transformer en une série temporelle régulière:

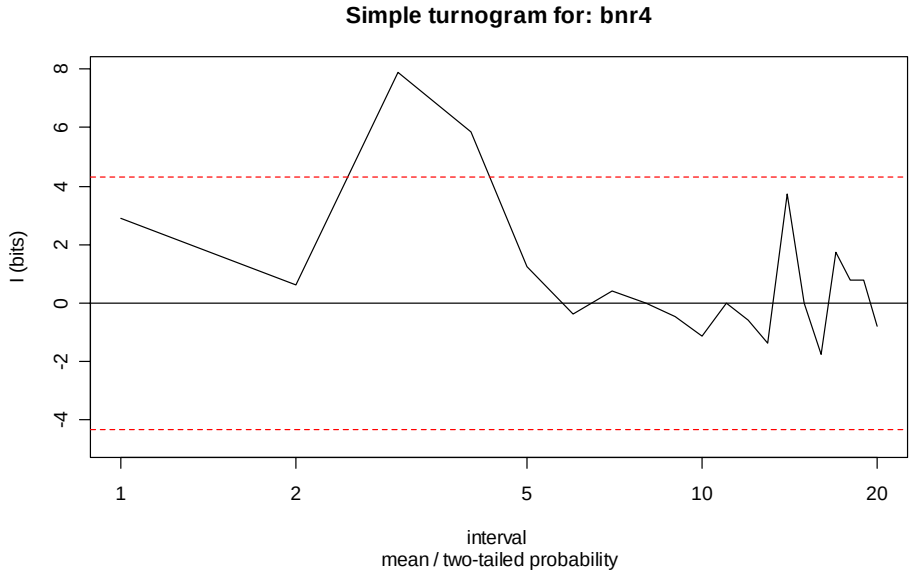
```
> data(bnr)
> bnr4 <- as.ts(bnr[, 4])
> plot(bnr4, type="l", main="bnr4: raw data", xlab="Time")
```



Le tournogramme simple, pour des moyennes sur les intervalles et pour un test bilatéral est obtenu par:

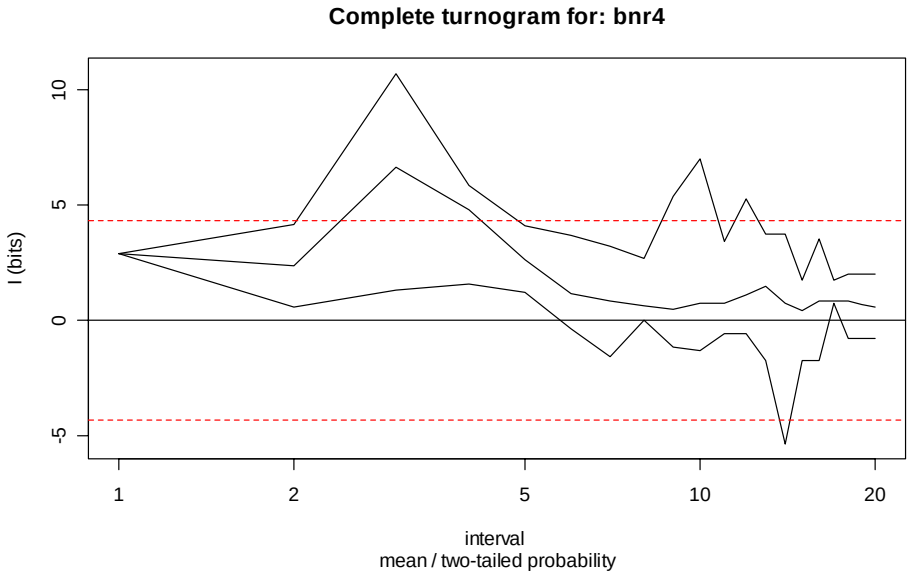
```
> bnr4.turno <- turnogram(bnr4)
```

```
> summary(bnr4.turno)

Simple turnogram for: bnr4

options      : mean / two-tailed probability
call         : turnogram(series = bnr4)
max. info.   : 7.903453 at interval 3 (P = 0.004176608: 15 turning
                                       points for 35 observations
extract level: 3


   interval   n turns       info
1         1 103    61  2.8849788
2         2  52    32  0.6097183
3         3  35    15  7.9034527
4         4  26    11  5.8490835
5         5  21    11  1.2432696
6         6  18    11 -0.3733604
7         7  15     8  0.4169020
8         8  13     7  0.0000000
9         9  12     7 -0.4792596
10       10  11     7 -1.1157070
11       11  10     5  0.0000000
12       12   9     5 -0.5790350
13       13   8     5 -1.3917540
14       14   8     2  3.7388752
15       15   7     3  0.0000000
16       16   7     4 -1.7427020
17       17   7     2  1.7427020
18       18   6     2  0.7776076
19       19   6     2  0.7776076
20       20   6     3 -0.7776076
```



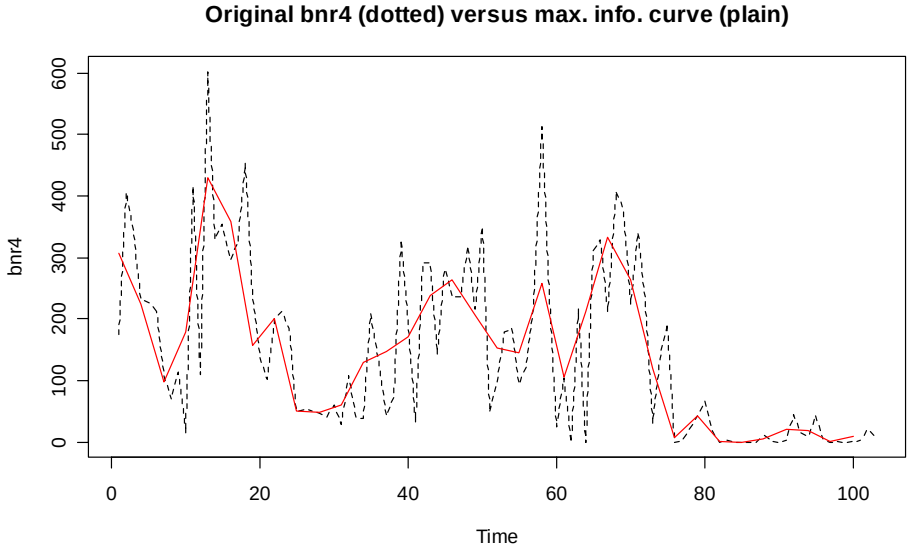
Dans le cas présent, la meilleure échelle d'observation semble être toutes les 3 observations actuelles (**extract level = 3**; valeur la plus élevée dans le tournogramme et significativement plus monotone qu'une série aléatoire au seuil 5%). On peut aussi tracer le tournogramme complet avec:

```
> turnogram(bnr4, complete=TRUE)

Complete turnogram for: bnr4

options      : mean / two-tailed probability
intervals    : 1 .. 20  / step = 1
nbr of obs.  : 103 .. 6
max. info.   : 6.64499 at interval 3 (P = 0.009992147: 15.66667
                          turning points for 35 observations)
extract level: 3
```



Ce qui confirme l'échelle 3 pour les valeurs maximales et moyennes (les deux courbes supérieures), mais pas pour les valeurs minimales. Cela signifie qu'avec un décalage donné, la courbe avec une échelle d'observation toutes les 3 valeurs n'apparaît pas forcément comme celle qui contient le plus d'information. Quoi qu'il en soit, le tournogramme simple est en accord avec la courbe maximale et moyenne. On peut donc extraire la série recalculée toutes les 3 observations à partir de la première (**drop = 0**, valeur par défaut dans ***extract()***), et la superposer sur le graphique de la série:

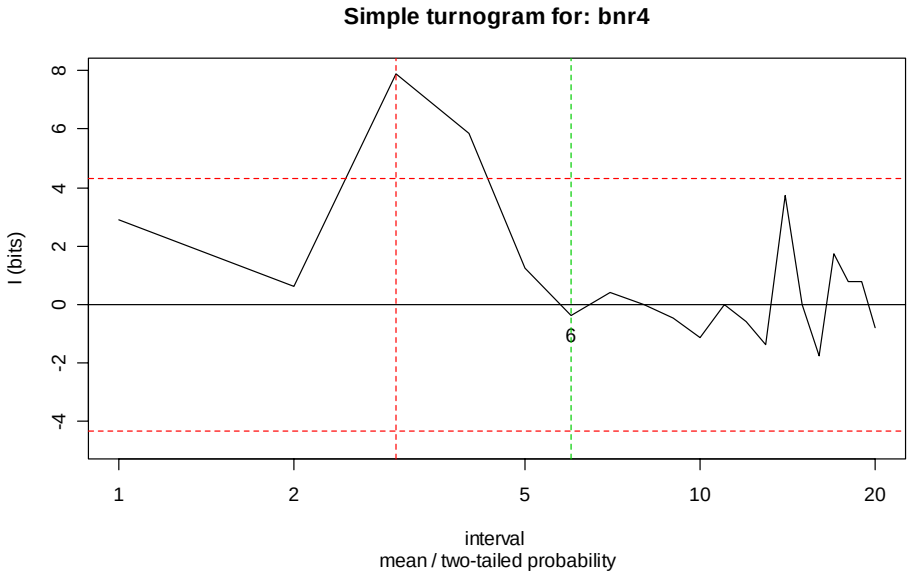
```
> bnr4.interv3 <- extract(bnr4.turno)
> plot(bnr4, type="l", lty=2, xlab="Time")
> lines(bnr4.interv3, col=2)
> title("Original bnr4 (dotted) versus max. info. curve (plain)")
```



D'après la théorie de l'information, la courbe en rouge contient le moins possible d'oscillations aléatoires et contient donc le maximum d'information sur la série, en particulier par rapport à la courbe originale en pointillés noirs. Le graphe de la fonction rouge est donc plus « parlant » que celui de la série originelle. Dans le cas où ces données correspondraient à une « prémanip », cette analyse suggère d'adopter un pas d'échantillonnage trois fois plus grand pour optimiser l'information obtenue par rapport à l'effort d'échantillonnage dans les « manips » ultérieures.

On peut choisir un autre intervalle pour l'extraction de la série que celui calculé par défaut de deux façons. Soit on peut le redéfinir interactivement sur le graphe, à l'aide de la fonction ***identify()***:

```
> plot(bnr4.turno)
> bnr4.turno$level <- identify(bnr4.turno, col=3)
Level      : 6
Information: -0.3733604
Probability: 0.7719823
Nbr of obs.: 18
Turnpoints : 11
```



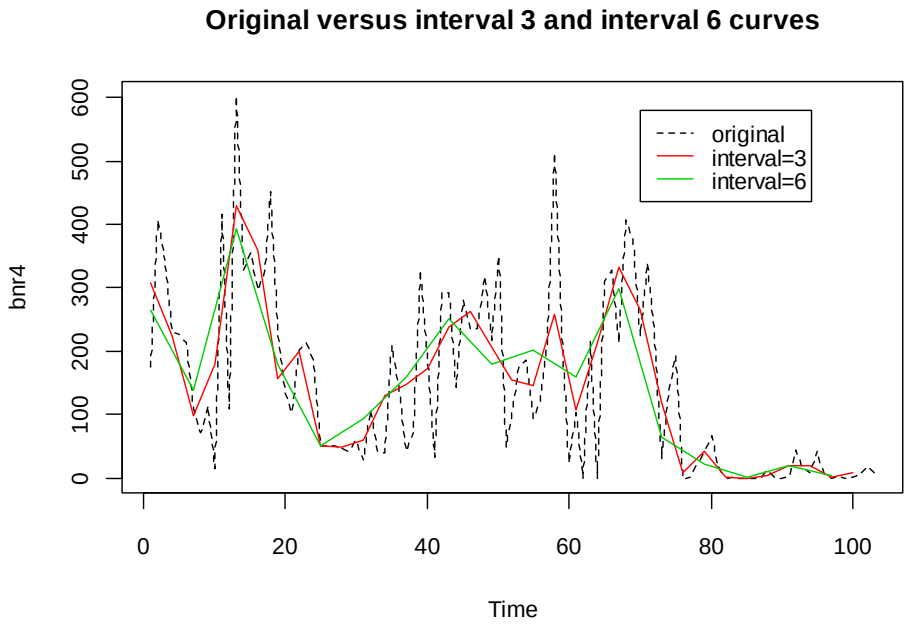
Sur le graphe, la ligne pointillée verticale rouge correspond à la valeur de ***level*** initiale calculée automatiquement. Le trait vertical pointillé vert est la nouvelle valeur de ***level*** obtenu après avoir entré la commande ***identify(…)*** et avoir cliqué sur le graphe au niveau du point marqué "6" (ce dernier n'apparaît pas, normalement, sur le graphe).

On peut encore rentrer directement la nouvelle valeur désirée de ***level*** dans l'objet *'turnogram'*:

```
> bnr4.turno$level <- 6
```

Comparons maintenant l'allure des 3 séries: la série originale, la série extraite avec le meilleur intervalle d'après le tournogramme (3), et la série extraite avec un intervalle double (6):

```
> bnr4.interv6 <- extract(bnr4.turno)
> plot(bnr4, type="l", lty=2, xlab="Time")
> lines(bnr4.interv3, col=2)
> lines(bnr4.interv6, col=3)
> legend(70, 580, c("original", "interval=3", "interval=6"),
+ col=1:3, lty=c(2, 1, 1))
> title("Original versus interval 3 and interval 6 curves")
```



S'il est assez évident que la série originale contient beaucoup de fluctuations aléatoires parasites, il est quasiment impossible de discerner à l'oeil quel est le meilleur choix de l'intervalle entre la courbe rouge et la courbe verte. Le tournogramme nous éclaire sur ce point, puisque la courbe rouge est significativement moins monotone qu'une courbe aléatoire alors que la courbe verte ne l'est pas. Dans le cas de la courbe verte, on a perdu tellement d'information que l'on ne peut plus discerner si les fluctuations sont aléatoires (succcession erratique de pics et de creux) ou non.

Variogramme

Matheron (1971) désigne par variable régionalisée $Z(x)$, la réalisation unique d'une certaine fonction aléatoire. La plus connue est la fonction aléatoire stationnaire. Par définition, elle possède une loi de probabilité invariante par translation. L'hypothèse intrinsèque de stationnarité développée par Matheron ignore la propriété de stationnarité de la fonction et s'intéresse uniquement à la stationnarité des accroissements. Elle suppose que les accroissements sont de moyenne nulle et que leur variance ne dépend que de la distance entre les observations et non de leur position:

$$E\left[Z(x)-Z(x+h)\right]=0$$
$$E\left[Z(x)-Z(x+h)\right]^2=2.\gamma(h)$$

La fonction $\gamma(h)$ est le **semi-variogramme** (souvent appelé pour simplifier variogramme).

En océanographie, les données ont une capacité de dispersion pratiquement illimitée et on ne peut leur attribuer de variance *a priori*. La variance sera fonction de la maille, de la densité des organismes, du volume des prélèvements. Pour ce type de données, Matheron a proposé une hypothèse encore moins restrictive, dite **hypothèse intrinsèque**: même si la variance est *a priori* infinie, il peut se confirmer que les accroissements de la variable aient une variance finie. L'exposé qui suit emprunte des informations tirées de Matheron (1971), David (1977) et Delhomme (1978).

L'équation du semi-variogramme empirique s'écrit:

$$\gamma^*(h)=\frac{1}{2.N(h)}\sum_i\left[Z(x_i)-Z(x_i+h)\right]^2$$

avec $N(h)$ correspondant au nombre de points distants de h, h étant une distance égale à un multiple de l'intervalle d'échantillonnage. Si les données sont irrégulièrement espacées, on considèrera des classes d'intervalles de temps qui ne se chevauchent pas. La forme du variogramme indique les propriétés structurales de la série et l'échelle des changements.

Références:

David, M., 1977. Developments in geomathematics. Tome 2: Geostatistical or reserve estimation. *Elsevier Scientific, Amsterdam*. 364 pp.

Delhomme, J.P., 1978. Applications de la théorie des variables régionalisées dans les sciences de l'eau. *Bull. BRGM*, section 3 n°4:341-375.

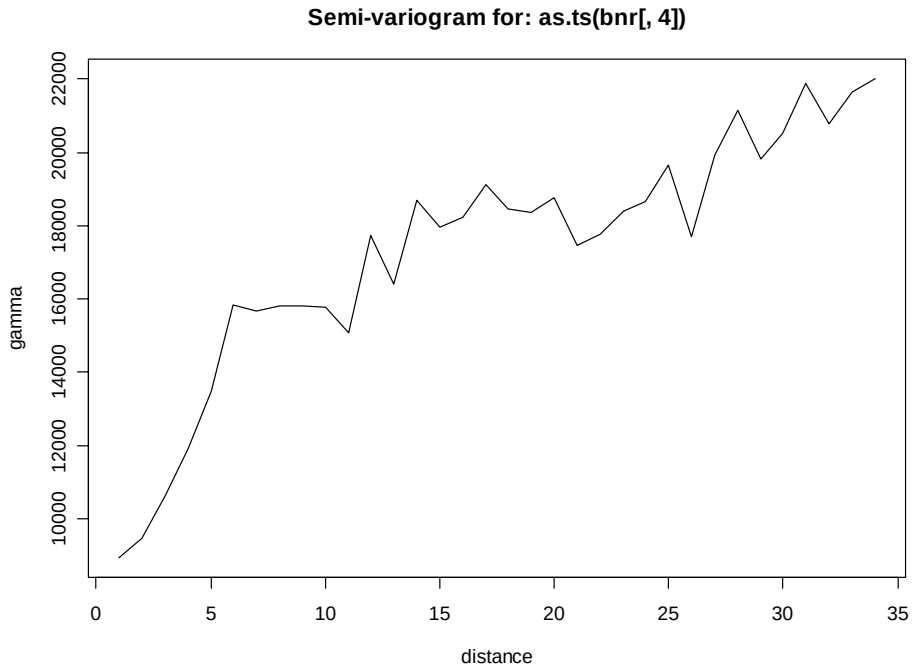
Matheron, G., 1971. La théorie des variables régionalisées et ses applications. *Cahiers du Centre de Morphologie Mathématique de Fontainebleau*. Fasc. 5 ENSMP, Paris. 212 pp.

Exemple:

En reprenant le même exemple que pour le tournogramme (série 4 de ***bnr***), on obtient son semi-variogramme par:

```
> data(bnr)
> vario(bnr[,4])
   distance semivario
1         1  8933.284
```

```
2          2  9450.129
3          3 10609.430
4          4 11919.288
5          5 13466.546
6          6 15853.098
7          7 15666.052
8          8 15818.568
9          9 15811.266
10        10 15756.645
11        11 15066.837
12        12 17747.786
13        13 16389.750
14        14 18679.669
15        15 17961.426
16        16 18236.943
17        17 19139.413
18        18 18463.482
19        19 18376.732
20        20 18767.747
21        21 17480.317
22        22 17754.858
23        23 18407.075
24        24 18675.184
25        25 19656.635
26        26 17710.909
27        27 19933.375
28        28 21158.673
29        29 19820.446
30        30 20525.432
31        31 21885.569
32        32 20792.761
33        33 21654.493
34        34 22013.464
```



Distogramme

L'échelle des changements d'une série multivariable peut être définie par une extension du variogramme: le **distogramme**. Il s'agit d'une fonction qui correspond à la distance moyenne entre les observations distantes de *h*. Pour chaque intervalle *h*, temporel et/ou spatial, on estimera une matrice des distances entre les stations, distance prenant en compte les valeurs des descripteurs (abondances ou biomasses d'espèces, mesures physiques, etc…).

Cette distance peut être choisie euclidienne, mais il est souvent plus avantageux de considérer la distance de corde pour chaque couple d'observations. Cette distance est une distance euclidienne normée à 1 pour chaque échantillon. Si *x* représente la densité de l'espèce *i* à la station *j*, on doit avoir:

$$\sum_{i=1}^{n} x_{ij}^2 = 1 \quad \text{avec} \quad j = 1, 2, ..., p$$

Tous les vecteurs échantillons *j* ont donc un module unitaire, bien que leurs directions soient inchangées. La distance de corde est égale à:

$$\delta_{j,k}^2 = \sum_{i=1}^{n} (x_{ij} - x_{ik})^2$$

Elle est nulle si deux échantillons possèdent les mêmes espèces dans les mêmes proportions. Elle est maximale, et égale à 2 quand les espèces diffèrent totalement d'un prélèvement à l'autre. Elle permet donc de comparer la composition faunistique selon l'échelle considérée sans subir l'influence (qui pourrait être considérable) de la minorité des espèces très abondantes. La formule du distogramme peut s'écrire:

$$\overline{\delta_{(h)}^2} = \frac{1}{2N_{(h)}} \sum_{j} \delta_{j,j+h}^2$$

Le distogramme permet aussi de décrire les changements de répartition des différentes classes d'âge d'une population.

Références:

Dauvin, J.C. & F. Ibanez, 1986. Variations à long-terme (1977-1985) du peuplement des sables fins de la Pierre Noire (baie de Morlaix, Manche Occidentale): analyse statistique de l'évolution structurale. *Hydrobiologia*, 142:171-186.

Ibanez, F. & J.C. Dauvin, 1988. Long-term changes (1977-1987) in a muddy fine sand *Abra alba – Melinna palmate* community from the Western English Channel: multivariate time-series analysis. *Mar. Ecol. Prog. Ser.*, 49:65-81.

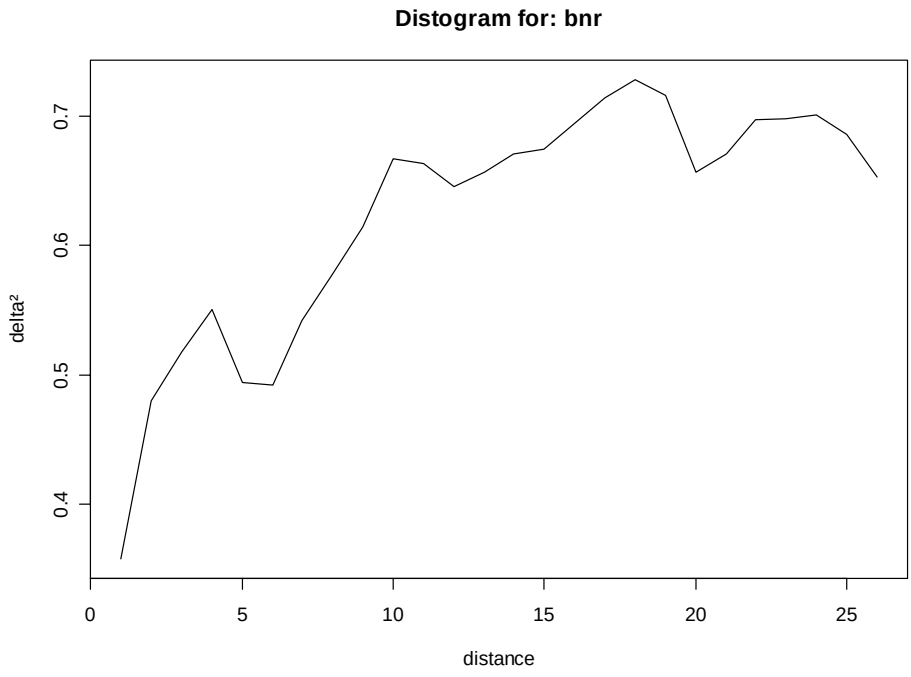
Mackas, D.L., 1984. Spatial autocorrelation of plankton community composition in a continental shelf ecosystem. *Limnol. Ecol.*, 20:451-471.

Exemple:

Le distogramme pour l'ensemble du tableau ***bnr*** est obtenu par:

```
> data(bnr)
> disto(bnr)
   distance distogram
1         1 0.3581454
```

```
2          2 0.4797966
3          3 0.5173366
4          4 0.5501900
5          5 0.4942032
6          6 0.4922207
7          7 0.5422460
8          8 0.5774747
9          9 0.6148662
10        10 0.6674661
11        11 0.6634647
12        12 0.6457200
13        13 0.6567708
14        14 0.6706163
15        15 0.6743301
16        16 0.6943444
17        17 0.7143640
18        18 0.7279900
19        19 0.7162906
20        20 0.6569722
21        21 0.6708284
22        22 0.6974970
23        23 0.6976051
24        24 0.7008844
25        25 0.6854326
26        26 0.6526971
```



Tendance générale

La **tendance** générale (trend, en anglais) représente une variation lente dans un sens déterminé. Cependant, si le plus souvent la tendance correspond à une fonction plus ou moins monotone croissante ou décroissante, dans d'autres cas la tendance générale est figurée par le cycle annuel par exemple. Dans ce cas, on parle de **tendance cyclique**.

Selon l'objet de l'étude entreprise, il peut être tout à la fois aussi important de l'estimer que de l'éliminer. L'estimer, car elle représente la trace la plus évidente et interprétable de l'évolution chronologique. L'ôter, car la part de variance qui lui est attachée est souvent si forte qu'elle peut masquer les autres composantes du signal: cycle long, saison, phénomène de haute fréquence ou bien, lorsque c'est la tendance qui est cyclique, l'évolution croissante ou décroissante de la série. Les méthodes qui servent à l'estimer ou à l'éliminer font partie des méthodes de décomposition des séries spatio-temporelles qui seront étudiées dans le chapitre suivant. Ici, nous nous contenterons seulement de tester son existence.

Il est possible de tester de manière non paramétrique l'existence d'une tendance. Si on effectue une corrélation linéaire entre les valeurs successives d'un processus et les valeurs des dates d'observations (définies par une suite d'entiers croissants), on va pouvoir tester si la tendance générale est significativement linéairement croissante ou décroissante. Nous renvoyons le lecteur aux modalités classiques du test de linéarité de la régression dans les ouvrages de statistique. Cependant la tendance générale peut être présente et ne pas correspondre à une droite. C'est pourquoi, pour tester l'existence d'une tendance de forme quelconque, on va remplacer les valeurs observées du processus par leurs rangs, puis calculer leur corrélation non paramétrique de Spearman *rs* avec le temps.

Soit une série de *n* observations. Si on appelle $\overline{R}$ le rang moyen, R_x le rang de la valeur de l'observation *x*, R_t le rang de la valeur de l'abscisse temporelle correspondante, *ex* le nombre d'ex aequos, la formule s'écrit:

$$rs = \frac{\sum_{i=1}^{n}(R_x-\overline{R})^2 + \sum_{i=1}^{n}(R_t-\overline{R})^2 - \sum_{i=1}^{n}\frac{ex^3-ex}{12} - \sum_{i=1}^{n}(R_x-R_t)^2}{2\sqrt{\sum_{i=1}^{n}(R_x-\overline{R})^2\left[\sum_{i=1}^{n}(R_t-\overline{R})^2 - \sum_{i=1}^{n}\frac{ex^3-ex}{12}\right]}}$$

Les valeurs du coefficient de Spearman sont comprises entre –1 et +1. Si le processus est purement aléatoire, la moyenne de *rs* est égale à 0 et sa variance est égale à 1/(n-1). La distribution des *rs* est normale pour $n > 50$. On calcule ici la quantité:

$$rs\sqrt{\frac{n-2}{1-rs^2}}$$

qui suit une loi *t* de Student, avec *n*-2 degrés de liberté, valable pour tout *n*.

> ***Attention!*** *On constate que lorsque le nombre d'ex aequos correspondant aux valeurs nulles est supérieur à 80%, la valeur de ce coefficient est totalement biaisée.*

Référence:

Siegel, S. & N.J. Castellan, 1988. Non-parametric statistics. *McGraw-Hill, New York.* 399 pp.

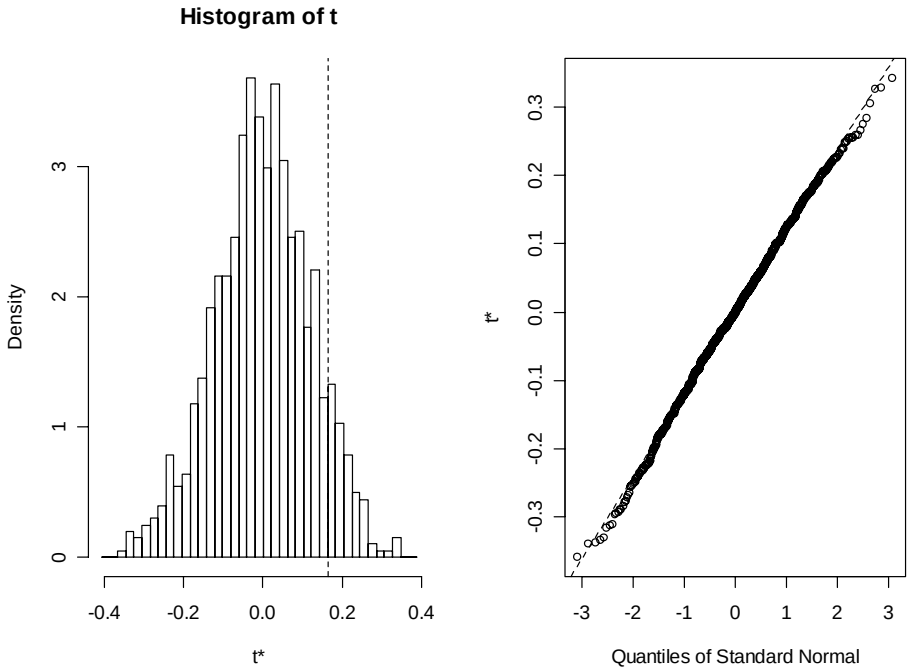
Exemple:

On peut estimer si la série 8 du tableau ***marbio*** contient une tendance significative comme suit:

```
> data(marbio)
> trend.test(marbio[,8])
        Spearman's rank correlation rho

data:  marbio[, 8] and time(marbio[, 8])
S = 43853, p-value = 0.1836
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.1630113

Warning message:
p-values may be incorrect due to ties in: cor.test.default(x,
time(x), alternative = "two.sided", method = "spearman")
```

Notez que le message d'avertissement concernant les ex-aequos (ties) n'apparaît pas dans S+. Il est possible de faire un bootstrap sur ce test pour déterminer de manière plus précise (c'est-à-dire par rapport à la distribution intrinsèque) la significativité du test (attention: l'exécution du test est nettement ralentie, parce qu'il est répété R fois; de plus, les informations fournies diffèrent entre R et S+):

```
> marbio8.trend.test <- trend.test(marbio[,8], R=999)
> marbio8.trend.test      # Sortie légèrement différente sous S+

BLOCK BOOTSTRAP FOR TIME SERIES

Fixed Block Length of 1

Call:
tsboot(tseries = x, statistic = trend.test, R = R, l = 1, sim =
"fixed")

Bootstrap Statistics :
     original     bias    std. error
t1* 0.1630113 -0.1642905   0.1199864

> plot(marbio8.trend.test)      # Graphique différent sous S+
> # Dans S+: qqnorm(marbio8.trend.test) pour le second graphe
```




```
> boot.ci(marbio8.trend.test, conf=c(0.95, 0.99),
+ type="norm")                  # Seulement sous R
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = marbio8.trend.test, conf = c(0.95, 0.99),
    type = "norm")

Intervals :
Level      Normal
95%   ( 0.0921,  0.5625 )
99%   ( 0.0182,  0.6364 )
Calculations and Intervals on Original Scale

> marbio8.trend.test$p.value
[1] 0.09
```

Les résultats peuvent varier quelque peu d'un bootstrap à l'autre. Dans les deux cas, on en conclut que la série n'a pas de tendance significative au seuil p = 5%, mais pas au seuil p = 10% dans le cas du bootstrap test seulement.

Tendance locale (méthode des sommes cumulées)

Une méthode simple dite des **sommes cumulées**, bien que peu utilisée, permet de reconnaître les **changements de tendance** d'une série. Elle permet:

- de détecter les changements survenant dans le niveau moyen de la série,
- de déterminer la date d'apparition de ces changements,
- d'estimer la valeur moyenne d'intervalles homogènes.

Soit une série échantillonnée régulièrement à pas constant x_t, t variant entre 1 et n. Choisissons une valeur de référence r (par exemple la moyenne). On retire cette valeur r de toutes les estimations de la série, puis on effectue le cumul des valeurs successives:

$$S_1 = (x_1 - r)$$
$$S_2 = (x_1 - r) + (x_2 - r) = S_1 + (x_2 - r) = x_1 + x_2 - 2r$$
$$\ldots$$

Donc:

$$S_q = \sum_{i=1}^{q} x_i - q.r$$

Cette somme cumulée est très sensible au changement de la valeur moyenne d'une série. Une propriété de ce type de graphique est que toute moyenne locale se déduit immédiatement de la pente. Soit deux points x_i et x_j limites inférieure et supérieure d'une séquence relativement monotone (constante, croissante ou décroissante). La pente p entre ces deux valeurs séparées par k intervalles de temps ($j - i = k$), vaudra $p = (x_j - x_i)/k$. En développant:

$$p = \frac{x_j - x_i}{k} = \frac{\left(\sum_{l=1}^{j} x_l - j.r\right) - \left(\sum_{l=1}^{i} x_l - i.r\right)}{k} = \frac{\sum_{l=i+1}^{j} x_l}{k} - r$$

d'où:

$$\overline{x}_{ij} = p + r$$

La moyenne locale entre deux points distants de k est égale à la pente du graphique des sommes cumulées plus la valeur de référence r choisie.

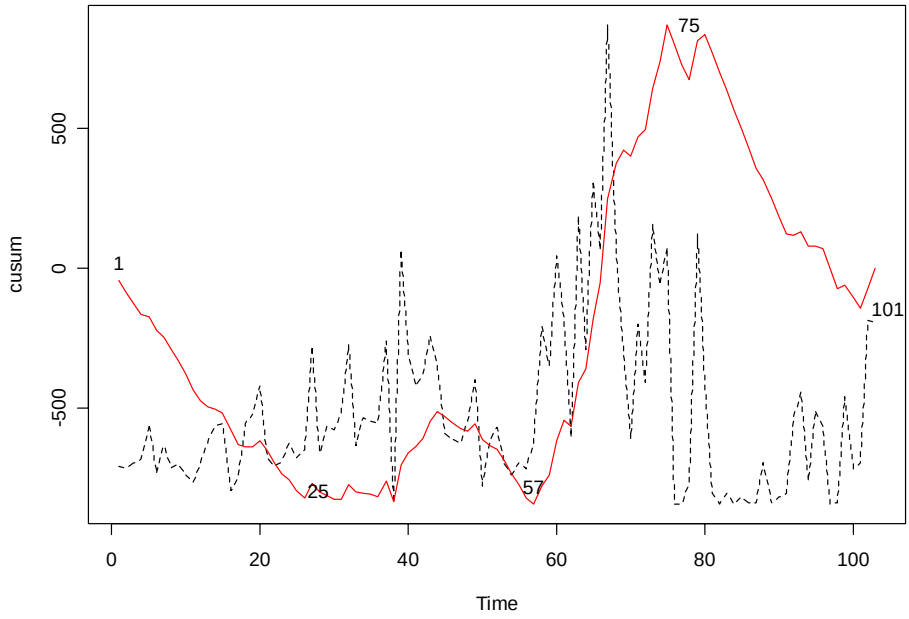
Référence:

Ibanez, F., J.M. Fromentin & J. Castel, 1993. Application de la méthode des sommes cumulées à l'analyse des séries chronologiques océanographiques. *C. R. Acad. Sci. Paris, Life Sciences,* 316:745-748.

Exemple:

Pour déterminer si les valeurs moyennes de la huitième série du tableau ***bnr*** restent constante ou non en fonction du temps, on entrera:

```
> data(bnr)
> bnr8.lt <- local.trend(bnr[,8])
```



La courbe cusum (en rouge par défaut) est superposée à la courbe initiale (en pointillé noir par défaut, et mis à la même échelle que la courbe cusum) amplifie les changements de valeur moyenne au cours du temps. Ainsi, des segments croissants ou décroissants dans la courbe cusum mettent en évidence des valeurs moyennes différentes dans la série. En utilisant la function ***identify()***, on peut cliquer autant de points que désirés sur la courbe cusum et obtenir le calcul des valeurs moyennes entre ces points (sur le graphe précédent, nous avons cliqué successivement les points 1, 25, 57, 75 et 101):

```
> identify(bnr8.lt)
$pos
[1]   1  25  57  75 101

$trends
[1]  40.04167  69.71875 166.27778  32.34615

$k
[1] 71.26214
```

Les tendances moyennes entre ces points sont renvoyées dans ***$trends***. La valeur moyenne sur l'ensemble de la série (par défaut, ou la valeur de référence rentrée dans ***local.trend(series, k = …)***) est reprise dans ***$k***. Remarquons que les points 25 et 57 ont pratiquement même ordonnée (on pourrait les relier avec un segment de droite horizontal), et donc, la moyenne locale entre ces deux points est proche de la valeur de référence ***$k***. Par contre, entre les points 1 et 25, la courbe cusum est décroissante, ce qui exprime une valeur moyenne inférieure à la valeur de référence (40.0 < 71.3). A l'opposé, la courbe cusum est croissante entre les points 57 et 75, ce qui indique une valeur moyenne entre ces points supérieure à la valeur de référence (166.3 > 71.3).

Décomposition de séries spatio-temporelles

Toutes les techniques présentées dans cette partie ont pour but de décomposer des séries spatio-temporelles régulières (donc, à pas de temps constant et sans trous) en deux ou plusieurs composantes. On extrait soit une composante issue d'un filtrage ou d'un lissage, ou alors correspondant à un modèle (variation cyclique, tendance linéaire, polynomiale, exponentielle, etc, …). Quel que soit le nombre de composantes obtenues, la dernière est toujours le résidu (**residuals**), c'est-à-dire, la fraction de la série initiale (**series**) qui ne se retrouve dans aucune des autres composantes (**components**).

Pour un modèle **additif**, on aura:

$$\mathbf{residuals} = \mathbf{series} - \Sigma\,\mathbf{components}$$

Un tel modèle est également appelé **linéaire** lorsque chaque composante est une combinaison linéaire de la ou des variables dépendantes, en l'occurrence pour les séries spatio-temporelles, le temps ou l'espace, ou les deux ensemble.

Un modèle **multiplicatif** se définira selon un schéma équivalent, mais dans ce cas, c'est le produit des composantes qui donnera la série. Une des composantes (la série filtrée, la série désaisonnalisée ou la tendance générale, selon le contexte), est alors exprimée dans les mêmes unités que la série de départ, alors que les autres sont des facteurs multiplicatifs adimensionnels. Il est possible de transformer un modèle multiplicatif en un modèle additif en passant aux logarithmes, puisque le log d'un produit est égal à la somme des log des facteurs. Si l'opération donne une combinaison linéaire des log des facteurs, on dit que l'on a **linéarisé** le modèle par la transformation logarithmique.

On pourrait encore définir des modèles **mixtes** où certaines composantes sont multiplicatives alors que d'autres sont additives, mais il n'y a alors plus moyen de les linéariser. Pour cette raison, de tels modèles sont très peu répandus. La nature de la ou des composantes dépend du traitement effectué. Par exemple pour un **filtrage**, il s'agit de la série filtrée; pour une **stationnarisation**, il s'agit de la tendance générale extraite, pour une **désaisonnalisation**, il s'agit du cycle saisonnier, etc.

Fonction générale de décomposition tsd()

Dans PASTECS, toutes les techniques de décomposition de série fonctionnent selon le même canevas et renvoient toutes un objet *'tsd'* ('time series decomposition'). Cet objet contient à la fois les composantes de la ou des séries qui ont été décomposées, des informations sur la méthode utilisée, ainsi que des données supplémentaires qui servent au diagnostic du traitement réalisé.

La fonction centrale qui effectue le traitement sur une ou plusieurs séries et renvoie un objet *'tsd'* est ***tsd()***. Son argument **method** permet de sélectionner une méthode de traitement à appliquer à toutes les séries à décomposer. Son argument **type** indique si le modèle est additif ou multiplicatif. Cette fonction appelle d'autres fonctions plus simples ***decxxx()*** où ***xxx*** est le nom de la méthode (ex: ***decmedian()*** pour la méthode des médianes mobiles ***"median"***, ***decevf()*** pour la méthode de filtrage par les vecteurs propres ***"evf"***, etc…). Les fonctions ***decxxx()*** ne peuvent traiter qu'une série à la fois, et ne sont normalement pas appelées directement par l'utilisateur. Toutefois, étant donné qu'elles renvoient aussi des objets *'tsd'*, l'utilisateur avancé peut les préférer pour des questions de performance ou de simplicité dans ses propres scripts.

Une fois qu'une décomposition est réalisée, il est possible d'extraire les différentes composantes (grâce à ***tseries()***) ou une partie d'entre elles (en utilisant la méthode ***extract()***) et de les retransformer en objet 'time series' régulières (*'rts'* sous S+ ou *'ts'* sous R). On peut ainsi appliquer de nouvelles méthodes de décomposition sur un ou plusieurs composants et approfondir l'analyse des séries par des traitements de

décomposition en cascade jusqu'à ce qu'on ait extrait et analysé toute l'information pertinente contenue dans ces séries.

Exemple:

On peut régulariser les séries 3 à 8 du jeu de données ***releve*** à l'aide de la méthode linéaire (voir chapitre régularisation) comme suit:

```
> data(releve)
> rel.regy <- regul(releve$Day, releve[3:8], xmin=6, n=87,
+ units="daystoyears", frequency=24, tol=2.2, methods="linear",
+ datemin="21/03/1989", dateformat="d/m/Y")
```

Ensuite, on va transformer les séries régularisées en objets 'time series' utilisables par ***tsd()***:

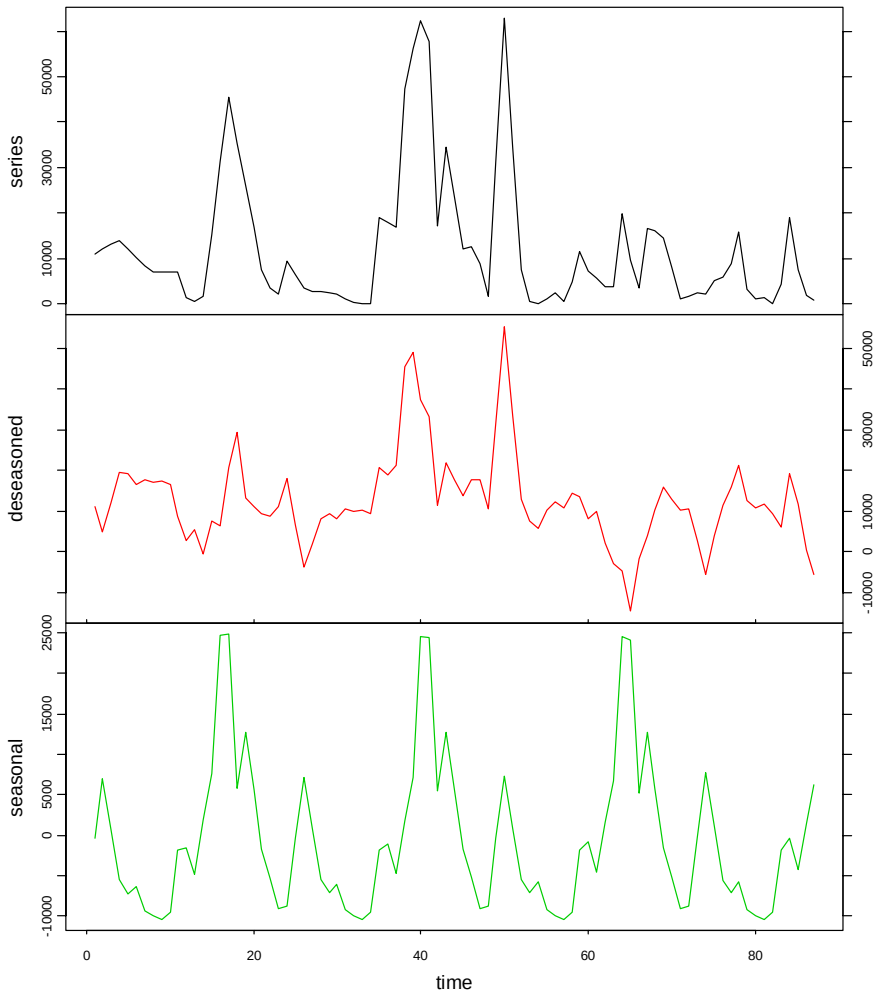
```
> rel.ts <- tseries(rel.regy)
```

Nous pouvons à présent effectuer simultanément une décomposition de toutes les séries contenues dans ***rel.ts***. Par exemple, pour extraire un cycle saisonnier par la méthode LOESS, nous entrons:

```
> rel.dec <- tsd(rel.ts, method="loess", s.window=13, trend=FALSE)
> rel.dec
 Call:
 tsd(x = rel.ts, method = "loess", s.window = 13, trend = FALSE)

Series:
[1] "Astegla" "Chae"    "Dity"    "Gymn"    "Melosul" "Navi"

Components for Astegla
[1] "seasonal"  "residuals"

Components for Chae
[1] "seasonal"  "residuals"

Components for Dity
[1] "seasonal"  "residuals"

Components for Gymn
[1] "seasonal"  "residuals"

Components for Melosul
[1] "seasonal"  "residuals"

Components for Navi
[1] "seasonal"  "residuals"
```

On peut maintenant afficher le graphe correspondant à la décomposition de l'une des séries, par exemple ***Melosul***:

```
> plot(rel.dec, series=5, col=1:3)
```



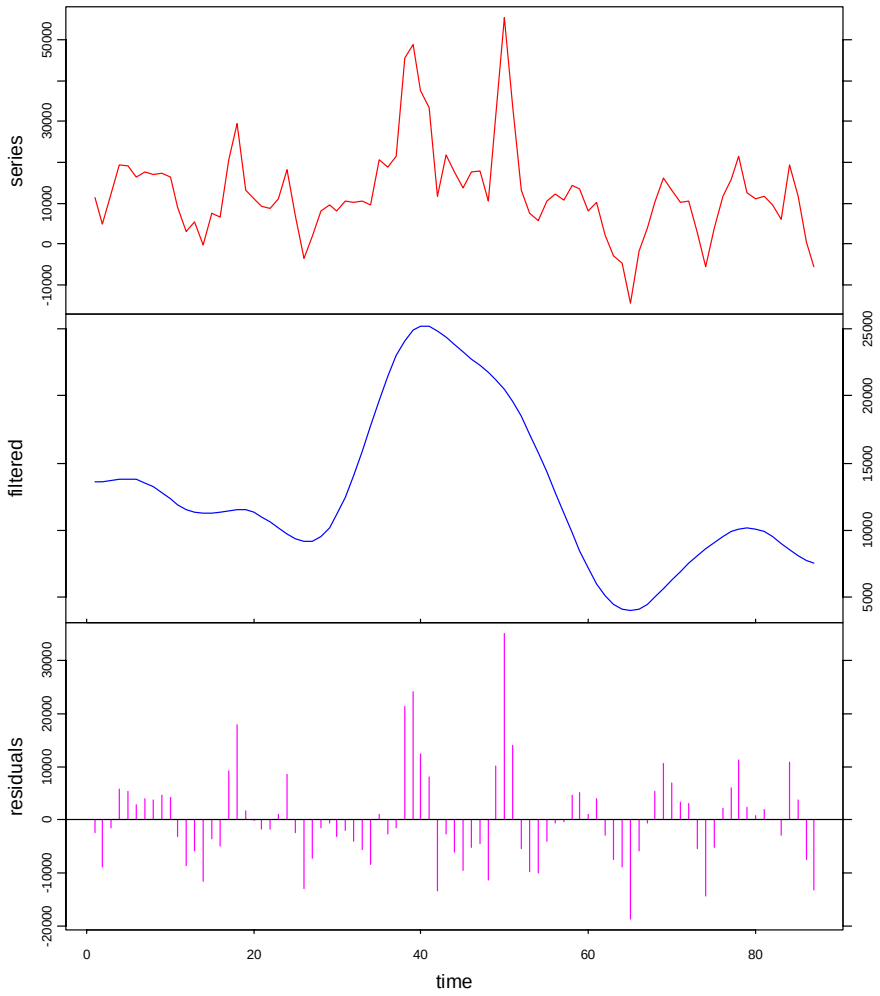
Enfin, l'extraction de composantes de certaines séries et leur transformation en nouveaux objets *'rts'* ou *'ts'* se fait comme suit (par exemple, on extrait uniquement la composante ***deseasoned*** des séries 3 à 6, et puis on affiche les 10 premiers éléments contenus dans les 'time series' extraites):

```
> rel.des <- extract(rel.dec, series=3:6, components="deseasoned")
> rel.des[1:10,]
    Dity.deseasoned Gymn.deseasoned Melosul.deseasoned Navi.deseasoned
 [1,]  -1856.450218       728.08507          11232.346        1263.204
 [2,]   -326.203029       925.87962           4857.551        2193.775
 [3,]     -5.397257       958.98367          12218.422        2141.072
 [4,]    305.799403       725.75255          19485.965        2042.397
 [5,]    510.738352       389.66511          19241.367        2524.486
 [6,]    673.469410        46.52693          16485.644        2800.179
 [7,]    666.811431      -205.96157          17601.801        2505.898
 [8,]    668.430789       928.90368          16997.344        2273.421
 [9,]    694.149206     -1949.28870          17367.641        1813.558
[10,]    696.564421     -1890.15039          16520.938        1603.852
```

Comme loess est implémenté légèrement différemment dans S+, les résultats ne sont pas strictement identiques, mais comparables. Ces objets peuvent ensuite être analysés avec les outils destinés aux séries régulières (autocorrélation, analyse spectrale, etc., voir chapitre analyse des séries spatio-temporelles) ou être décomposés à leur tour par un

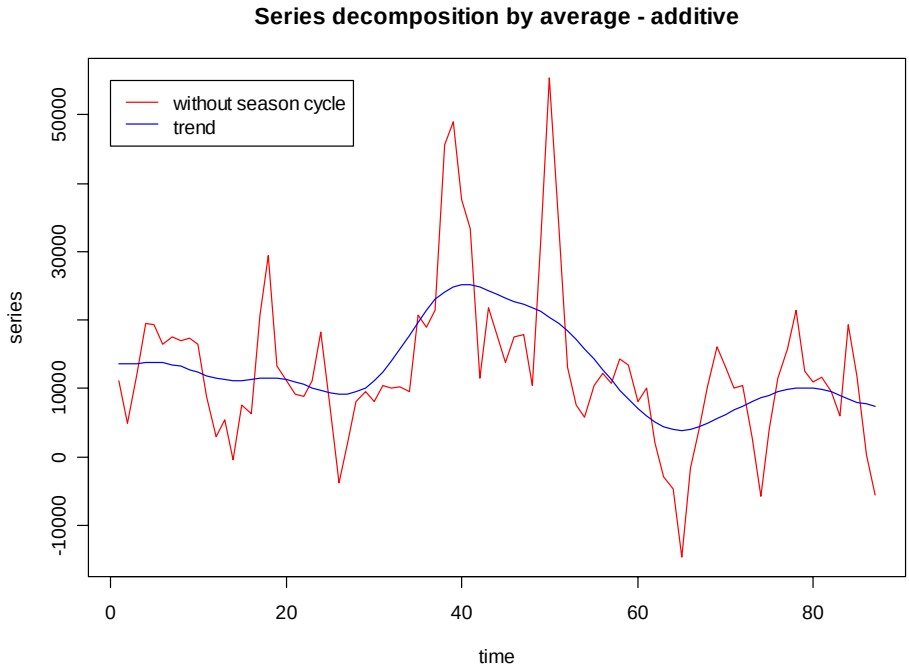
nouvel appel à ***tsd()***. Par exemple, nous pouvons éliminer la tendance générale présente dans les résidus (**stationnarisation**) par la méthode des moyennes mobiles, en leur appliquant une nouvelle fois ***tsd()***, avec l'argument **method = "average"** (notez que la série ***Melosul*** est en position 3 maintenant):

```
> rel.des.dec <- tsd(rel.des, method="average", order=2, times=10)
> plot(rel.des.dec, series=3, col=c(2,4,6))
```



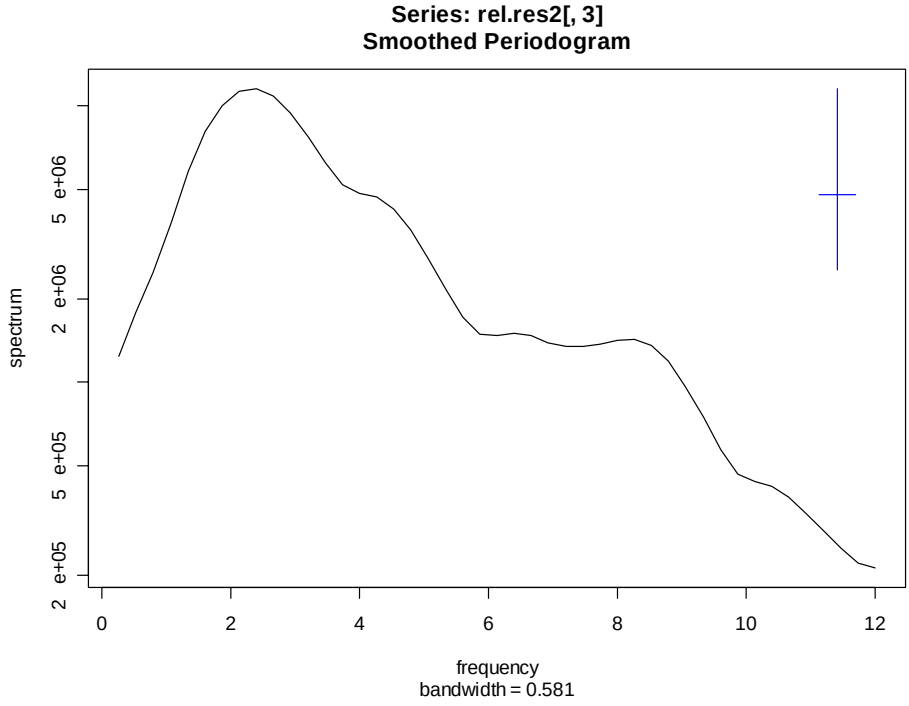
Dans ce dernier graphe, ***series*** correspond donc à la composante ***deseasoned*** issue du premier traitement, ***filtered*** correspond à la tendance générale présente dans cette série désaisonnalisée et ***residuals*** correspond à la partie (pseudo-)stationnaire qu'il reste de la série après élimination d'un cycle saisonnier (premier traitement) et d'une tendance générale (second traitement) selon un modèle additif (la somme de toutes les composantes redonne la série de départ). Pour le graphe, on aurait donc pu préciser **labels = c("deseasoned", "trend", "residuals")** pour être plus clair. Notons au passage qu'il est possible de faire d'autres types de graphiques à l'aide de la méthode ***plot()***. Par exemple, pour représenter de manière superposée la série désaisonnalisée et la tendance qui en est extraite (ce qui est peut-être plus parlant dans ce cas), nous pourrions entrer:

```
> plot(rel.des.dec, series=3, col=c(2,4), stack=FALSE,
+ resid=FALSE, labels=c("without season cycle", "trend"),
+ lpos=c(0,55000))
```



On peut continuer l'analyse en approfondissant de plus en plus les décompositions et les analyses des composantes jusqu'à ce que l'on considère avoir extrait toute l'information contenue dans la ou les séries initiales. Par exemple, on pourrait effectuer une analyse spectrale des résidus après élimination du cycle saisonnier et de la tendance générale:

```
> rel.res2 <- extract(rel.des.dec, components="residuals")
> spectrum(rel.res2[,3], spans=c(5,7))
```



A l'évidence, il y a encore un signal cyclique de fréquence 2,5-3 dans les résidus qui n'a pas encore été extrait.

*Les sept fonctions suivantes (**decdiff()**, **decaverage()**, **decmedian()**, **decevf()**, **decreg()**, **decensus()** et **decloess()**) sont utilisées par **tsd()**, mais sont aussi fournies séparément pour permettre à l'utilisateur avancé de réaliser ses propres routines de décomposition indépendamment de la fonction **tsd()**.*

Filtrage d'une série spatio-temporelle

Le filtrage d'une série ou chronique consiste à remplacer chaque valeur de cette série par une combinaison de ses diverses valeurs. Un filtrage est dit **linéaire** lorsqu'il utilise une combinaison linéaire des valeurs de cette série:

$$Y_t = \sum_{i=0}^{k} \alpha_i . X_{t-i}$$

où X_t est la série de départ, Y_t est la série filtrée. Les méthodes des différences ou la méthode des moyennes mobiles sont des filtres linéaires, alors que la médiane mobile, par exemple, est un filtre non linéaire. Lorsque le filtre conduit à réduire la variance en éliminant notamment les très hautes fréquences, on parle aussi de **lissage** du signal (smoothing en anglais).

Filtrage linéaire par la méthode des différences decdiff()

La méthode des différences a pour but d'éliminer la tendance. Ce n'est valable que si la série a une tendance monotone et non en « dents de scie ». Autrement dit, cela suppose que la série est autocorrélée positivement. Pour décrire la méthode, définissons d'abord la notion **d'opérateur de retard**:

$$L^k X_t = X_{t-k} \quad k = 0,1,2,...$$

Si k = 0, la série est inchangée. L'intérêt de ce formalisme est que L peut être traité comme une variable. Ainsi, on aura l'équivalence:

$$L^k L^s X_t = L^{k+s} X_t$$

En effet, appelons $Y_t = L^s X_t = X_{t-s}$. Comme $L^k Y_t = Y_{t-k}$, on a bien $L^k L^s X_t = X_{t-k-s}$, qui correspond à l'expression ci-dessus.

Considérons l'opérateur polynomial $\nabla = L^0 - L^1$. Cet opérateur est la différence première:

$$\nabla X_t = (L^0 - L^1) X_t = X_t - X_{t-1}$$

Les différences d'ordre r (successives) sont définies par:

$$\nabla^r X_t = (L^0 - L^1)^r X_t = \sum_{i=0}^{r} (-1)^{r-i} C_r^i L^i X_t$$

où C_r^i désigne les combinaisons simples de i termes pris r à r. Par exemple pour les différences secondes:

$$\nabla^2 X_t = (L^0 - L^1)^2 X_t = (L^0 + L^2 - 2L^1 L^0) X_t = X_t + X_{t-2} - 2X_{t-1}$$

La transformation de X_t en $\nabla^2 X_t$ élimine totalement ou en partie la tendance. On sait en effet qu'un polynôme de degré p présente la propriété que sa $p^{\text{ème}}$ différence finie est une constante et que les différences d'ordre $p + 1$, $p + 2$, … sont nulles.

La méthode des différences ne permet donc pas d'évaluer la tendance mais de l'éliminer; elle est extrêmement courante lorsqu'on désire se rapprocher de façon rapide et simple de la stationnarité, en répétant le procédé si nécessaire. Elle est ainsi employée très souvent préalablement à l'analyse spectrale.

La méthode des différences, dans une version légèrement différente, est également efficace pour éliminer une tendance sinusoïdale. Si on dispose par exemple de séries pluriannuelles avec un pas d'observation mensuel, dans la mesure où on considère que la variabilité saisonnière peut être modélisée par une sinusoïde (ce qui n'est pas le cas en général, car souvent les cycles annuels biologiques sont « télescopés »: la période de reproduction printanière est souvent décalée d'une année sur l'autre…), alors on remplace les données X_t par les écarts aux moyennes des mois respectifs. Si la série était une sinusoïde, un tel filtrage aurait pour effet de la transformer en une droite. Supposons que l'on ait n années, la valeur désaisonnalisée Y_t au mois i s'écrira:

$$Y_t = X_t - \frac{\sum_{i=1}^{n} X_i}{n}$$

où X_i sont les valeurs respectives rencontrées au mois i pendant les n années considérées. Cette méthode n'est pas implémentée dans ***decdiff()***, mais elle est disponible dans ***decloess()*** avec les arguments **order = "periodic"** et **trend = FALSE** (voir description de la fonction).

Références:

Kendall, M., 1976. Time-series. *Charles Griffin & Co Ltd*. 197 pp

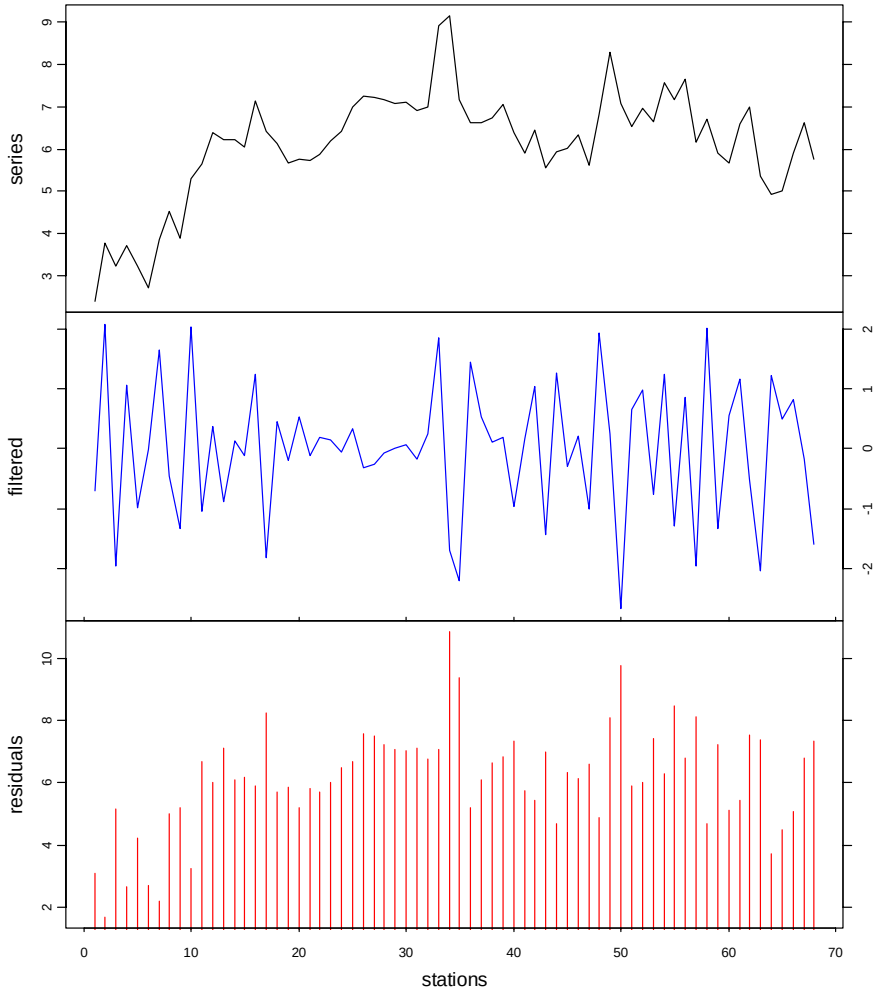
Laloire, J.C., 1972. Méthodes du traitement des chroniques. *Dunod, Paris*, 194 pp.

Philips, L. & R. Blomme, 1973. Analyse chronologique. *Université Catholique de Louvain. Vander ed*. 339 pp.

Exemple:

On peut calculer la différence du second ordre (**lag = 1**) pour la série ***ClausocalanusB*** en log du jeu de données ***marbio***, puis tracer le graphe de la filtration par:

```
> data(marbio)
> ClausoB.ts <- ts(log(marbio$ClausocalanusB + 1))
> ClausoB.dec <- decdiff(ClausoB.ts, lag=1, order=2, ends="fill")
> plot(ClausoB.dec, col=c(1,4,2), xlab="stations")
```



Filtrage linéaire par les moyennes mobiles decaverage()

La moyenne mobile (MB) remplace chaque observation par une moyenne pondérée sur les termes situés de part et d'autre de cette observation. Par exemple, si la MB est calculée sur trois termes, on aura:

$$M_t = b_{-1}.X_{t-1} + b_0.X_t + b_{+1}.X_{t+1}$$
$$M_{t+1} = b_{-1}.X_t + b_0.X_{t+1} + b_{+1}.X_{t+2}$$
etc.

La somme de ces poids égale l'unité. On a donc les relations:

$$M_t = \sum_{\tau=-k}^{k} b_\tau.X_{t+\tau} \quad \text{avec} \quad \sum_{\tau=-k}^{k} b_\tau = 1$$

Lorsque tous les coefficients de pondération sont égaux, la MB est dite **simple**. La constante b_τ ne dépend plus du rang τ, et on a:

$$M_t = \sum_{\tau=-k}^{k} \frac{X_{t+\tau}}{2k+1}$$

La valeur $2k + 1$ est appelée **bande de lissage** de la MB **d'ordre** k.

Appliquer successivement un filtrage par les MB revient à appliquer un filtrage par une MB unique. On démontre que si, après avoir effectué une MB sur $2k + 1$ termes, on applique une seconde MB sur $2l + 1$ termes, la série finale peut être obtenue directement à partir de la série initiale par une MB à $2(k + l) + 1$ termes:

$$N_t = \sum_{\tau=-(k+l)}^{(k+l)} d_\tau.X_{t+\tau} = \sum_{\tau=-l}^{l} c_\tau.M_{t+\tau} \quad \text{avec} \quad M_t = \sum_{\tau=-k}^{k} b_\tau.X_{t+\tau}$$

On peut définir les poids d_τ de cette MB unique à partir des poids b_τ et c_τ des deux MB successives:

- Poids de MB1: $B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{2k+1} \end{bmatrix}$
- Poids de MB2: $C = \begin{bmatrix} c_1 & c_2 & \cdots & c_{2l+1} \end{bmatrix}$
- Poids de MB12: $D = BC = \begin{bmatrix} b_1c_1 & b_1c_2 & \cdots & b_1c_{2l+1} \\ b_2c_1 & b_2c_2 & & b_2c_{2l+1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{2k+1}c_1 & b_{2k+1}c_2 & \cdots & b_{2k+1}c_{2l+1} \end{bmatrix}$

Le produit matriciel BC est une matrice qui contient autant de diagonales qu'il y a de coefficients à calculer, soit $2(k + l) + 1$.

La **variance** de la série filtrée vaut:

$$\sigma_M^2 = \sigma_\varepsilon^2.\sum_\tau b_\tau^2$$

Ainsi, une MB simple de 13 termes, appliquée à une série aléatoire donne une série de même moyenne, mais dont la variance est réduite de 1/13. Si une MB est soumise à la restriction que la somme de ses poids vaut 1, la somme des carrés des poids sera minimale

lorsque ceux-ci seront égaux à 1. C'est donc une MB simple qui réduit le plus efficacement la variabilité d'une série.

Lorsque l'on effectue une MB, on se trouve confronté à un problème d'estimation des termes extrêmes de la série lissée. L'estimation peut se faire en effectuant une MB progressivement moins étendue. Les deux valeurs les plus imprécises, le premier et le dernier point faisant l'objet d'une moyenne à seulement k +1 termes au lieu de $2k$ + 1.

Une technique plus précise consiste à rajouter des valeurs fictives avant de passer au lissage. Supposons que l'on désire effectuer une MB d'ordre 2 sur une série à n termes. Deux valeurs au début, et deux valeurs à la fin ne peuvent être estimées. On considère alors la moyenne des deux premiers termes de la série que l'on va placer deux fois au début de la série. De même, on calcule la moyenne des deux derniers termes, et on rajoute cette valeur au bout de la série. La série avant le filtrage MB est alors à n + 4 termes. Elle va donc permettre une estimation, imparfaite, mais suffisante en pratique.

Une MB calculée sur $2k$ + 1 termes a la propriété **d'éliminer le cycle de même période**. La **fréquence de coupure du spectre** (voir chapitre analyse des séries spatio-temporelles, paragraphe analyse spectrale) est en effet égal à:

$$f_c = \frac{1}{2k+1}$$

Ainsi, si on dispose d'une série pluriannuelle avec des observations mensuelles, une MB simple centrée de 12 mois élimine la variation saisonnière. Cependant, on ne peut pas centrer les observations dans une fenêtre contenant un nombre pair de termes. C'est pourquoi on va considérer un lissage avec une fenêtre à 13 termes. Le filtre de **désaisonnalisation** s'écrira:

$$M_t = \frac{1}{12}\sum_{\tau=-5}^{5} X_{t+\tau} + \frac{X_{t-6} + X_{t+6}}{24}$$

Comme il faut considérer le poids de 12 valeurs et non de 13, les observations extrêmes dans la fenêtre comptent seulement pour moitié. Naturellement, 6 valeurs au début et 6 valeurs à la fin de la série ne pourront être estimées par cet algorithme. La méthode d'addition des valeurs au début et à la fin de la série ne s'applique pas ici si on a à estimer 6 mois successifs au début et à la fin d'une série pluriannuelle. Si la série comporte un grand nombre d'années, on pourra préalablement rajouter les valeurs des 6 derniers mois de la première année au début et les valeurs des 6 premiers mois de la dernière année à la fin.

Quand on effectue une MB d'ordre 6 pour éliminer un cycle annuel, la variance de la série lissée est pondérée d'un facteur égal à:

$$\sum_{\tau} b_\tau^2 = 11.\left(\frac{1}{12}\right)^2 + 2.\left(\frac{1}{24}\right)^2 = 0.08$$

Cela signifie que la variance est réduite de 28% ($\sqrt{0.08}$). Cette MB a la propriété d'éliminer les fonctions périodiques de période égale à 12, 6, 4, 3, 12/5 et 2. Elle épargne les fonctions linéaires.

L'application du filtre des MB induit un mouvement périodique appelé **effet Slutsky-Yule**. La série engendrée par une MB sur un processus purement aléatoire ne l'est plus. La période approximative T de ce mouvement est déterminé par la corrélation entre les termes successifs et peut se calculer par la relation $\cos(T) = r(1)$. Si par exemple le coefficient d'autocorrélation d'ordre 1, $r(1)$, d'une MB à 13 termes est 12/13, ce filtre engendre des oscillations parasites qui obéissent à $\cos(T)$ = 12/13, soit $T \approx 22.6°$ ou en mois, $T \approx 360/22.6 \approx 16$ mois. L'application répétée infinie de MB conduit à une sinusoïde pure.

A noter que sous S+/R, on dispose d'autres fonctions très puissantes de lissage, notamment le ***kernel smoothing*** et le ***supersmoother*** de Friedmann. Toutes les techniques de décompositions non prévues dans ***tsd()*** peuvent toutefois être utilisées à l'aide de la méthode "**reg**", puisque cette dernière demande la série initiale et la série traitée (sans se préoccuper du traitement effectué par ailleurs) pour créer l'objet *'tsd'*.

Références:

Kendall, M., 1976. Time-series. *Charles Griffin & Co Ltd*. 197 pp

Laloire, J.C., 1972. Méthodes du traitement des chroniques. *Dunod, Paris*, 194 pp.

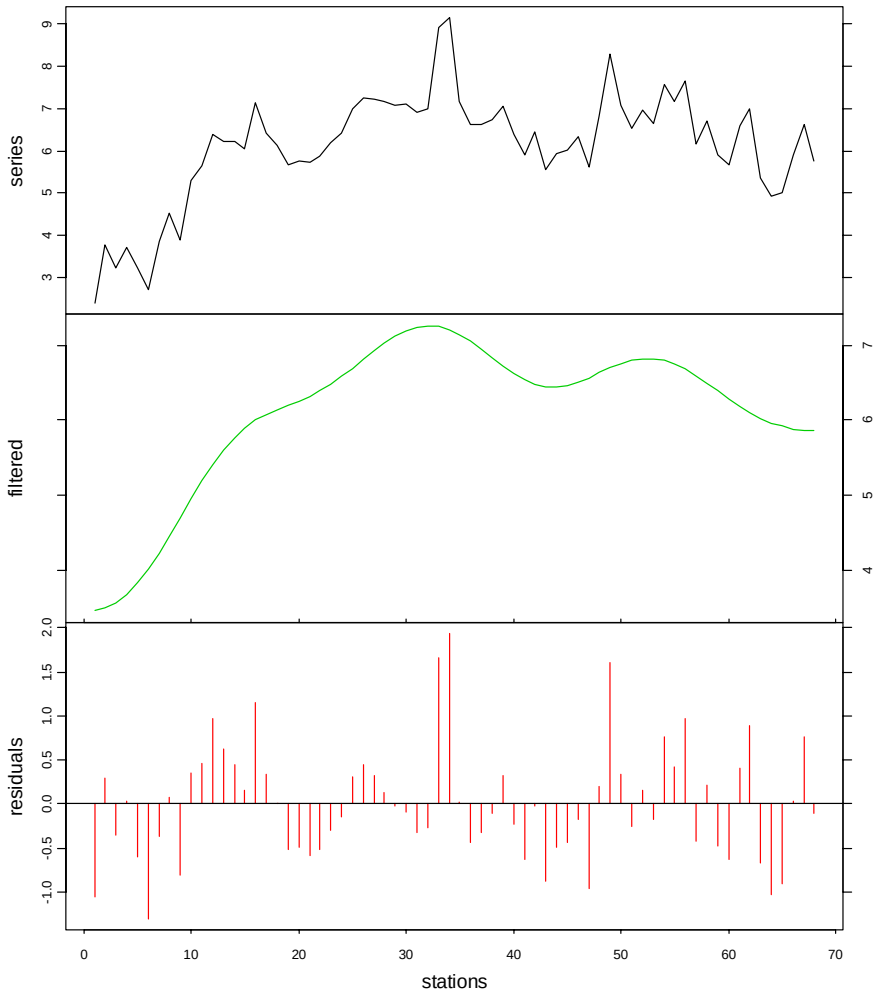
Malinvaud, E., 1978. Méthodes statistiques de l'économétrie. *Dunod, Paris*. 846 pp.

Philips, L. & R. Blomme, 1973. Analyse chronologique. *Université Catholique de Louvain. Vander ed*. 339 pp.

Exemple:

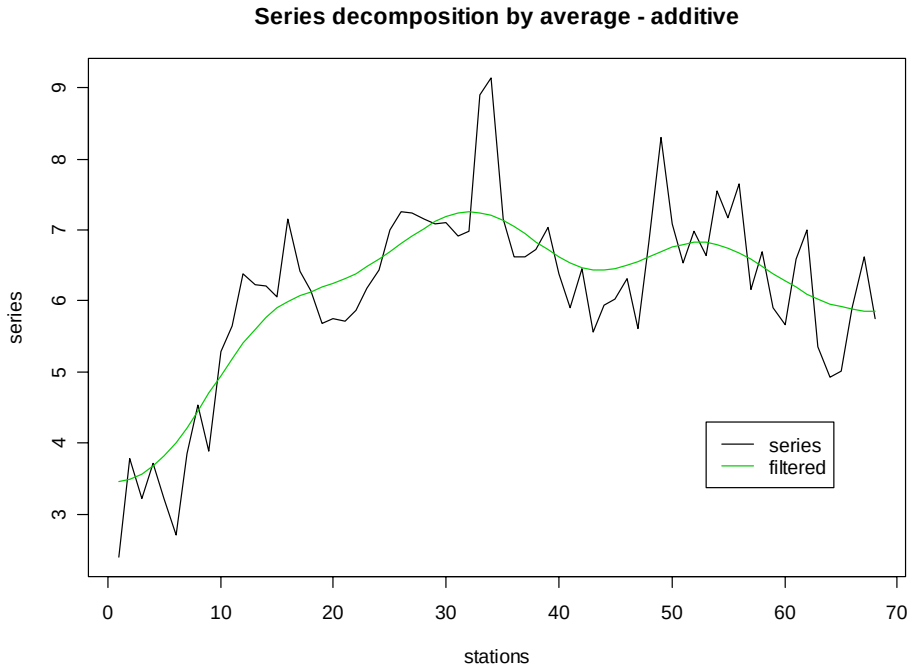
On peut filtrer 10 fois de suite la série ***ClausocalanusB*** (en log) du jeu de données ***marbio*** par une moyenne mobile simple et centrée d'ordre 2, puis tracer le graphe de la filtration par:

```
> data(marbio)
> ClausoB.ts <- ts(log(marbio$ClausocalanusB + 1))
> ClausoB.dec <- decaverage(ClausoB.ts, order=2, times=10,
+ sides=2, ends="fill")
> plot(ClausoB.dec, col=c(1,3,2), xlab="stations")
```



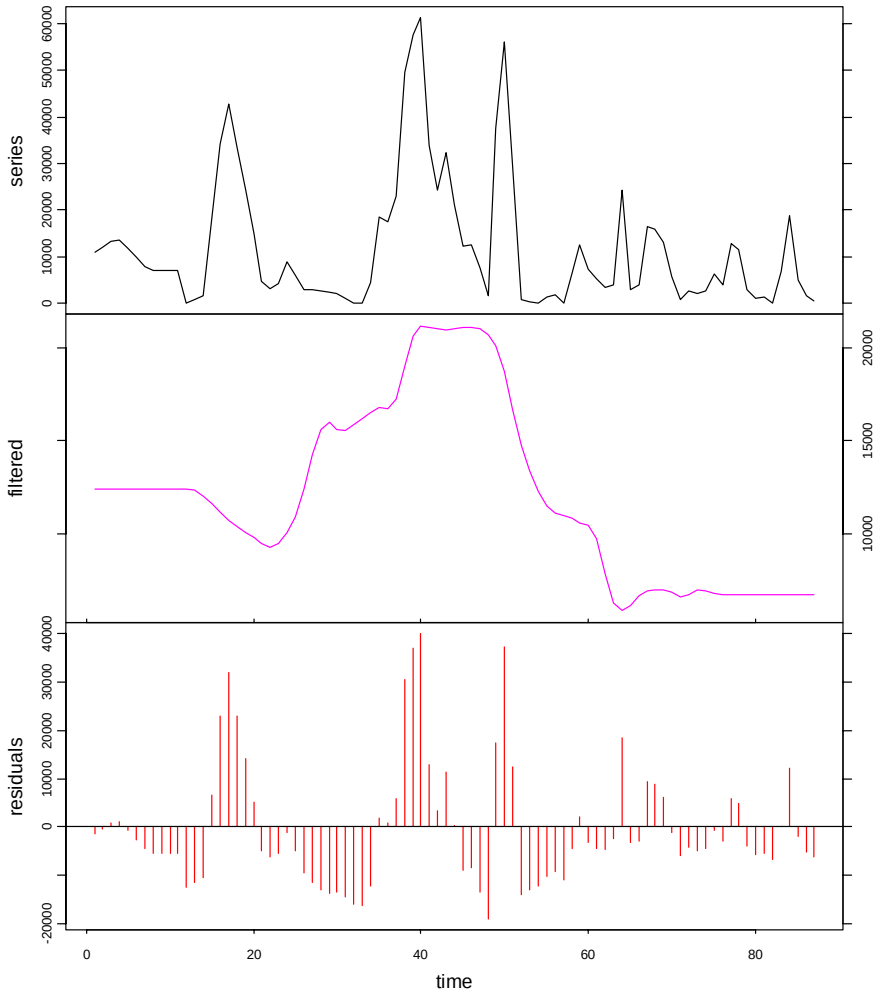
Ici, on voit mieux le résultat sur un graphique superposé sur lequel les résidus ne sont pas représentés:

```
> plot(ClausoB.dec, col=c(1,3), xlab="stations", stack=FALSE,
+ resid=FALSE, lpos=c(53,4.3))
```



On peut aussi désaisonnaliser la série ***Melosul*** du jeu de données ***releve*** (après régularisation comme expliqué au chapitre régularisation) par une moyenne mobile, bien que la méthode LOESS soit plus efficace à ce niveau:

```
> data(releve)
> melo.regy <- regul(releve$Day, releve$Melosul, xmin=9, n=87,
+ units="daystoyears", frequency=24, tol=2.2, methods="linear",
+ datemin="21/03/1989", dateformat="d/m/Y")
> melo.ts <- tseries(melo.regy)
> melo.dec <- decaverage(melo.ts, order="periodic", times=1,
+ sides=2, ends="periodic")
> plot(melo.dec, col=c(1,6,2))
```



Comme on peut le constater, le « cycle saisonnier » est de forme quelconque dans ***residuals***. Il s'agit en fait de toutes les oscillations de fréquences de 12 mois, ainsi que les plus petites harmoniques qui ont été éliminées, pour ne laisser que les oscillations plus longues dans ***filtered***. Attention toutefois à l'effet Slutsky-Yule si le filtre est appliqué plusieurs fois.

Filtrage non linéaire par les médianes mobiles decmedian()

Au lieu d'utiliser des moyennes mobiles, on peut appliquer des médianes mobiles. Dans ce cas, on considèrera également une fenêtre impaire à k termes et on calculera sa médiane:

$$Z_t = \text{médiane}(X_{t-k}, \ldots, X_t, \ldots, X_{t+k})$$

On démontre qu'une expression comme $\sum|Z_t - \theta|$ est minimisée quand θ est égal à la médiane de Z_t.

Le lissage (ou filtrage) des médianes mobiles peut être itérativement répété. Ce procédé dit des « running medians » (R) converge rapidement vers une courbe invariante. La notation conventionnelle est la suivante:

- 3RSR: lissage à 3 termes jusqu'à obtention de la convergence. Une médiane 3RSR correspond aux valeurs qui minimisent l'expression

$$S = \sum_{i=-1}^{1}|Z_{t+i} - \theta| \quad \text{avec} \quad t = 2,\ldots,n-1$$

- 53RSR: lissage double, soit un de 3 termes, et ensuite un de 5 termes jusqu'à convergence. Néanmoins, la convergence n'est pas absolument garantie si la série est longue.

- 4(3RSR)2H: ce lissage est préconisé par Tukey (1977). Il est généralement appliqué deux fois: on calcule les résidus d'un premier lissage, on effectue un second lissage du même type sur ces résidus, et on additionne les deux séries lissées ensemble.

Ce filtre peut être appliqué soit pour lisser, soit surtout pour segmenter une série. La courbe obtenue est voisine de celle de départ (différent d'avec les moyennes mobiles), sauf en ce qui concerne les grands pics et les grands creux de la série. Ceux-ci sont ignorés et on constate un lissage en forme de plateaux successifs. En effet, si on prend trois points successifs, on obtiendra fatalement un plateau en conservant la valeur centrale. Si on répète ce procédé, on tend à créer des plateaux de plus de deux valeurs. Ce lissage peut aussi servir à détecter un petit nombre de classes et à reconnaître des discontinuités ponctuelles.

Références:

Gebski, V.J., 1985. Some properties of splicing when applied to non-linear smoothers. *Comp. Stat. Data Anal.*, 3:151-157.

Philips, L. & R. Blomme, 1973. Analyse chronologique. *Université Catholique de Louvain. Vander ed.* 339 pp.

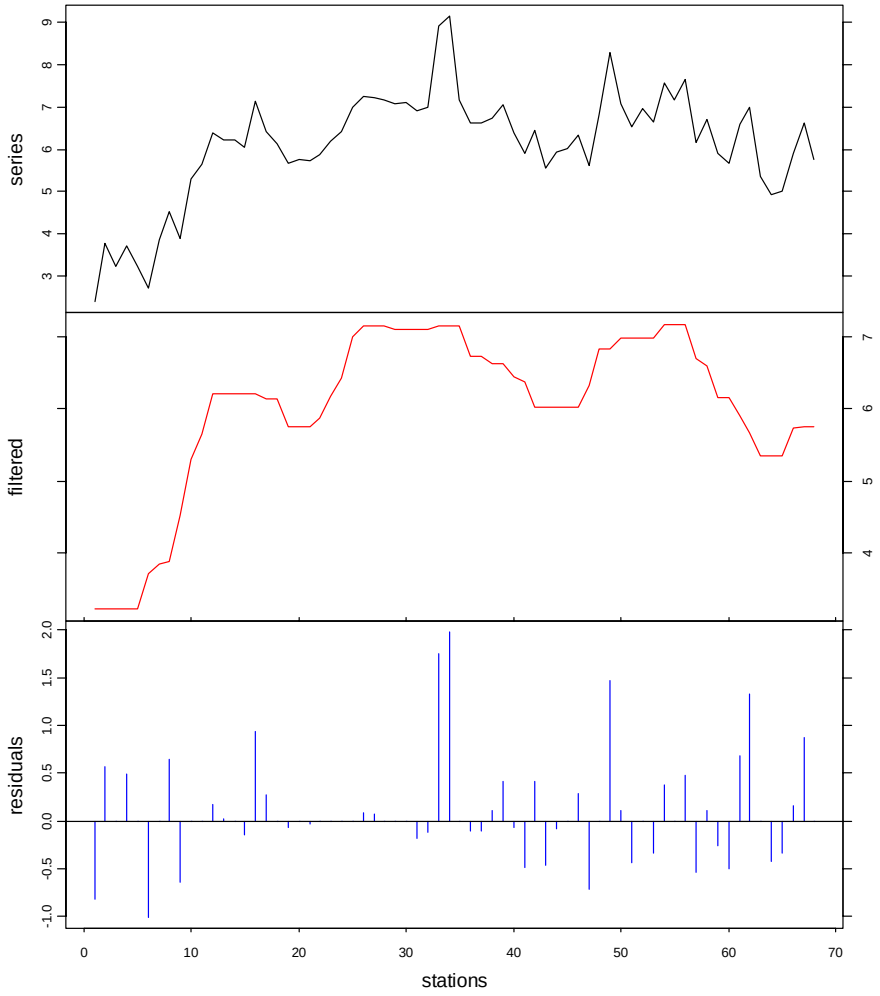
Tukey, J.W., 1977. Exploratory Data Analysis. *Reading Massachusetts: Addison-Wesley.*

Exemple:

On peut filtrer 10 fois de suite (bien plus que ce qu'il ne faut pour atteindre la stabilisation) la série ***ClausocalanusB*** en log du jeu de données ***marbio***, par une médiane mobile d'ordre 2, puis tracer le graphe de la filtration par:

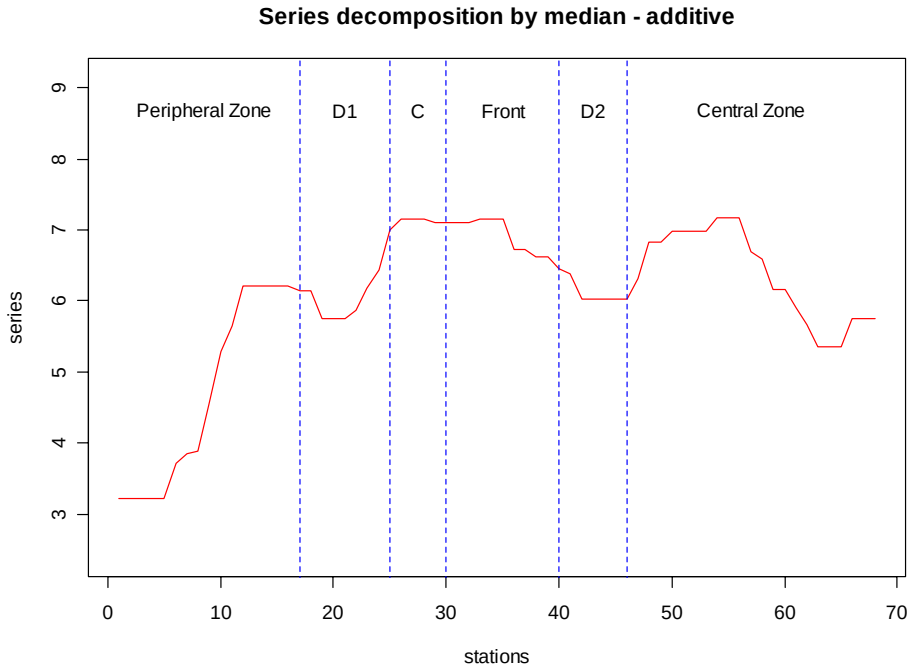
```
> data(marbio)
> ClausoB.ts <- ts(log(marbio$ClausocalanusB + 1))
```

```
> ClausoB.dec <- decmedian(ClausoB.ts, order=2, times=10,
+ ends="fill")
> plot(ClausoB.dec, col=c(1,4,2), xlab="stations")
```



Les différents « paliers » sont ici mis en évidence. Un graphique superposé (mais sur lequel on ne trace ni les résidus **resid = FALSE**, ni la courbe d'origine **col = 0**) et sur lequel on représente aussi les séparations des masses d'eau, identifiées de manière indépendante, fait apparaître une certaine relation:

```
> plot(ClausoB.dec, col=c(0,2), xlab="stations",
+ stack=FALSE, resid=FALSE)
> lines(c(17,17), c(0,10), col=4, lty=2)
> lines(c(25,25), c(0,10), col=4, lty=2)
> lines(c(30,30), c(0,10), col=4, lty=2)
> lines(c(41,41), c(0,10), col=4, lty=2)
> lines(c(46,46), c(0,10), col=4, lty=2)
> text(c(8.5,21,27.5,35,43.5,57), 8.7, labels=c("Peripheral Zone",
+ "D1", "C", "Front", "D2", "Central Zone"))
```


Series decomposition by median - additive
Peripheral Zone
D1
C
Front
D2
Central Zone
series
stations
3
4
5
6
7
8
9
0
10
20
30
40
50
60
70

Filtrage par les vecteurs propres decevf()

Une méthode de décomposition générale des séries correspond au filtrage dit des vecteurs propres (en anglais: *EigenVectors Filtering* ou EVF, parfois appelée *Analyse en Composantes Principales de Processus* ou ACPP). Les différentes composantes reflèterons la tendance générale, puis successivement des mouvements de variance de moins en moins importante. Comme l'extraction des composantes suit le pourcentage de variance qui lui est associé, la première extraite peut être quelconque, pseudo-périodique, voire périodique. Ainsi dans le cas d'une série pluriannuelle, il est possible que la tendance générale ne corresponde pas à un mouvement croissant ou décroissant mais à la périodicité saisonnière.

Le filtrage par les vecteurs propres se réalise de la façon suivante. Soit une série X_t, t variant de 1 à n, et un décalage temporel τ. On construit d'abord une matrice X correspondant à r séries décalées d'un multiple de τ:

$$X = \begin{bmatrix} X_{(r-1)\tau+1} & \cdots & X_{2\tau+1} & X_{\tau+1} & X_1 \\ X_{(r-1)\tau+2} & & X_{2\tau+2} & X_{\tau+2} & X_2 \\ X_{(r-1)\tau+3} & & X_{2\tau+3} & X_{\tau+3} & X_3 \\ \vdots & \ddots & & & \vdots \\ X_n & \cdots & X_{n-(r-3)\tau} & X_{n-(r-2)\tau} & X_{n-(r-1)\tau} \end{bmatrix}$$

Les résultats du filtrage dépendront de τ et du décalage maximum entre les séries extrêmes, soit: $\Delta t = (r-1)\tau$. Après la construction de X, on calcule la matrice de covariance entre les r descripteurs, matrice équivalente à une matrice d'autocovariance classique dans le cas où τ est égal à 1. Ensuite, on extrait les valeurs et vecteurs propres de cette matrice. Les valeurs propres indiqueront la part de variance associée aux éléments de la décomposition. Chaque estimation, tendance, cycle long, cycle annuel, variabilité résiduelle sera en effet obtenue par les valeurs de X en tenant compte successivement des premiers axes extraits. Si on multiplie un vecteur propre (le premier par exemple) par la composante correspondante, on définit une matrice X prédite qui représente une structure particulière. Pour illustrer cela, considérons une matrice X de départ très simple, avec $n = 10$, $r = 4$ et $\tau = 1$.

$$X = \begin{bmatrix} X_4 & X_3 & X_2 & X_1 \\ X_5 & X_4 & X_3 & X_2 \\ X_6 & X_5 & X_4 & X_3 \\ X_7 & X_6 & X_5 & X_4 \\ \vdots & \vdots & \vdots & \vdots \\ X_{10} & X_9 & X_8 & X_7 \end{bmatrix}$$

Si on remarque que les valeurs de la plupart des observations apparaissent r fois (donc quatre fois pour les observations comprises entre 4 et 7), certaines autres apparaissent trois fois (pour les observations 3 et 8), deux fois (pour 2 et 9) et une fois (pour 1 et 10). Une manière d'estimer les valeurs prédites des observations consiste donc à faire des moyennes, r fois des termes homologues de la matrice X prédite. Comme la multiplication matricielle (vecteurs propres – composantes) suivie des moyennes en diagonales des valeurs de X prédite revient à définir une pondération par moyenne mobile de la série originale, les poids consistant en une combinaison linéaire des éléments des vecteurs propres (voir Ibanez & Etienne, 1992) ou obtient un filtrage de la série.

La difficulté de la décomposition se résume dans le choix judicieux du décalage maximum Δt, entre les séries différenciées, issues de la série originale. Si Δt est trop petit, la série prédite à partir du premier axe factoriel suivra presque identiquement la courbe originale. Les autres axes n'étant l'expression que de fluctuations aléatoires. Si le décalage est trop grand, elle va à la limite, se confondre avec une droite de tendance. Or la décomposition doit exprimer, au travers du premier composant, la tendance générale qui

correspond au facteur de variation de plus grande variance et de forme quelconque. Comme le nombre de données est souvent faible en écologie marine, il est malheureusement exclu de faire une estimation de la variabilité à partir d'une analyse spectrale.

Le choix du décalage maximum s'effectuera en considérant le profil de la fonction d'autocorrélation du descripteur. Cette fonction indique le décalage qui correspond au premier passage à 0 (c'est-à-dire l'échelle à partir de laquelle les séries décalées sont indépendantes entre elles, intervalle parfois appelé **échelle caractéristique du processus**). Lorsqu'on ne dispose d'aucune information sur les cycles précédents dans la série, on peut avancer que l'échelle caractéristique correspond à la période d'une oscillation déphasée de $\pi/4$ (corrélation nulle) par rapport à la moyenne des périodicités les plus énergétiques: pour une échelle caractéristique de 20 intervalles d'échantillonnage par exemple, les cycles majeurs seront situés en moyenne aux alentours de 160 intervalles.

Le décalage correspondant au maximum de remontée de la fonction représente la périodicité moyenne de plus forte variance, tous cycles confondus. Si cette remontée est faible, et si le passage à 0 intervient après un long décalage, alors la série exprime une très forte tendance non cyclique. Si la fonction décroît vers 0 exponentiellement, le processus est stationnaire, donc sans tendance. Si la fonction oscille dès le premier décalage autour de 0, la série correspond à un bruit blanc (ni tendance, ni cycle). C'est donc le décalage qui donne l'information nécessaire au choix de Δt .

Ce filtrage correspondant à un processus moyenne mobile, il est possible d'estimer la fenêtre spectrale théorique équivalente (dans le cas où on prend $\tau = 1$). La **fonction de gain** exprime, pour une gamme de fréquences, le rapport entre le spectre de la série originale et de la série lissée (sous condition de stationnarité, voir chapitre analyse des séries spatio-temporelles, analyse spectrale):

$$F_z(\omega) = G(\omega).F_x(\omega)$$

où ω est un découpage en fréquences, F_z correspond au spectre de la série lissée, F_x au spectre de la série originale et G à la fonction de gain. La fonction G peut se calculer indépendamment de toute évaluation de spectre à partir des poids de la moyenne mobile:

$$G(\omega) = \left| \sum_{t=0}^{h} P_t . \mathrm{e}^{i\omega t} \right|^2$$

où les P_t représentent les poids successifs de la moyenne mobile (voir Ibanez & Etienne, 1992). La fonction de gain donne une bonne approximation des périodes conservées après filtrage.

Références:

Colebrook, J.M., 1978. Continuous plankton records: zooplankton and environment, North-East Atlantic and North Sea 1948-1975. *Oceanologica Acta*, 1:9-23.

Ibanez, F. & J.C. Dauvin, 1988. Long-term changes (1977-1987) on a muddy fine sand *Abra alba – Melinna palmate* population community from the Western English Channel. *J. Mar. Prog. Ser.*, 49:65-81.

Ibanez, F., 1991. Treatment of data deriving from the COST 647 project on coastal benthic ecology: The within-site analysis. *In:* B. Keegan (ed.) *Space and time series data analysis in coastal benthic ecology*. Pp. 5-43.

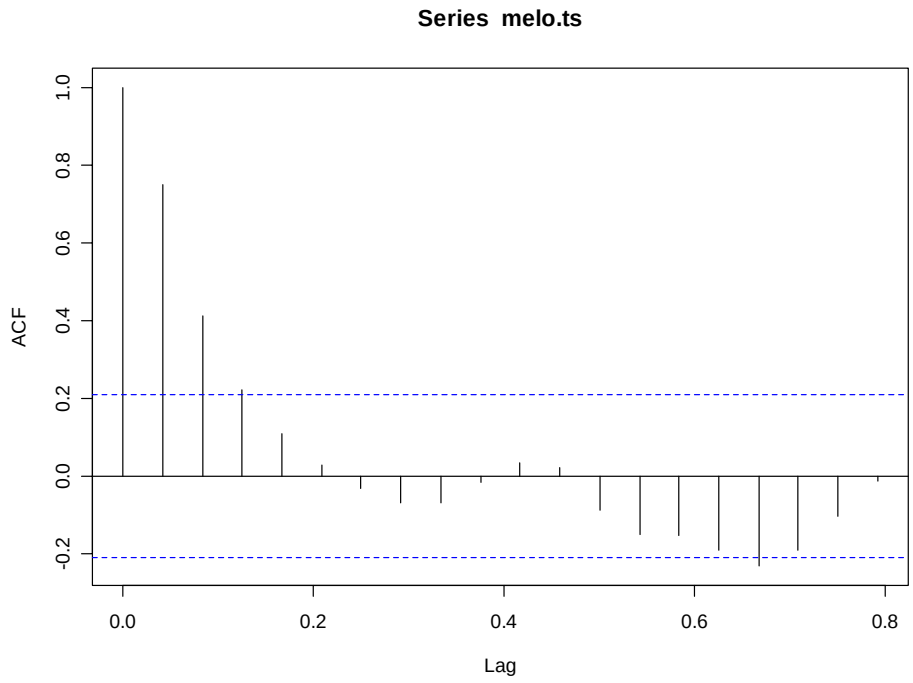
Ibanez, F. & M. Etienne, 1992. Le filtrage des séries chronologiques par l'analyse en composantes principales de processus (ACPP). *J. Rech. Océanogr.*, 16:27-33.

Ibanez, F., J.C. Dauvin & M. Etienne, 1993. Comparaison des évolutions à long-terme (1977-1990) de deux peuplements macrobenthiques de la Baie de Morlaix (Manche Occidentale): relations avec les facteurs hydroclimatiques. *J. Exp. Mar. Biol. Ecol.*, 169:181-214.

Exemple:

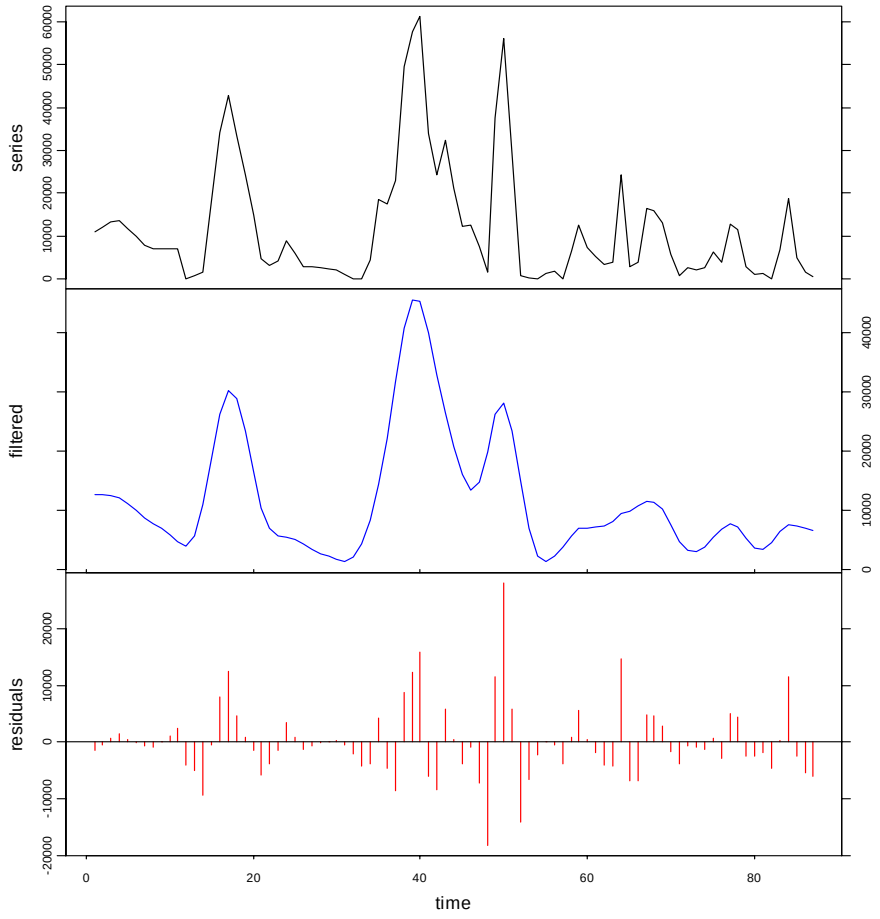
Si l'on reprend la série ***Melosul*** du jeu de données ***releve***, régularisée et transformée en un objet 'time series', le graphe de l'autocorrelation donne:

```
> data(releve)
> melo.regy <- regul(releve$Day, releve$Melosul, xmin=9, n=87,
+ units="daystoyears", frequency=24, tol=2.2, methods="linear",
+ datemin="21/03/1989", dateformat="d/m/Y")
> melo.ts <- tseries(melo.regy)
> acf(melo.ts)
```



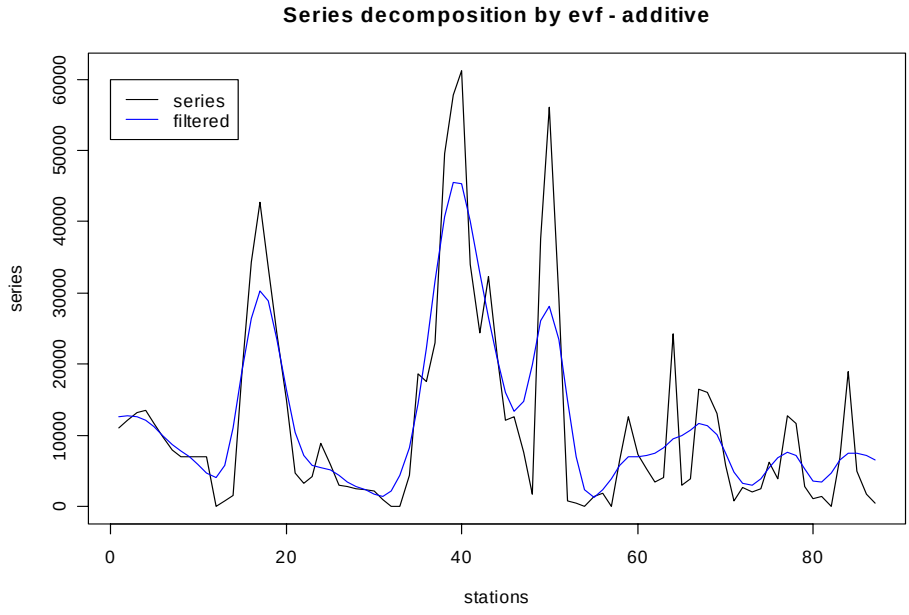
L'autocorrelation n'est plus significative après un lag de 0.16 ans. Comme la fréquence d'échantillonnage dans cette série est de 24 par an, cela donne un lag de 0.16 * 24 ≈ 4 observations. Nous utiliserons donc cette valeur comme argument **lag** dans la décomposition par les vecteurs propres. Nous choisirons aussi de ne conserver que le premier axe. Cela donne:

```
> melo.evf <- decevf(melo.ts, lag=4, axes=1)
> plot(melo.evf, col=c(1,4,2))
```



Si l'on superpose la série filtrée à la série originale, cela donne:

```
> plot(melo.evf, col=c(1,4), xlab="stations", stack=FALSE,
+ resid=FALSE, lpos=c(0, 60000))
```



Estimation de la tendance par régression decreg()

Les méthodes de filtrage présentées ci-dessus permettent d'extraire ou d'éliminer une tendance de forme quelconque (on n'a pas d'idée *a priori* de sa forme; on ne cherche pas à la modéliser). Les méthodes d'estimation par régression, au contraire, font appel à des hypothèses très fortes quant à sa nature: on cherche à ajuster une ou plusieurs équations fonctionnelles à la tendance de la série.

L'idée simple pour estimer une tendance générale est de vérifier son ajustement par une droite, une parabole, un polynôme d'ordre plus élevé simulant un mouvement pseudo-cyclique de forte période. Ces techniques reposent sur l'algorithme des moindres carrés: on minimise les carrés d'écarts entre les données observées et un polynôme de degré fixé à l'avance. L'estimation des paramètres se fait en considérant un système d'équations partielles.

Pour un processus X_t, la régression simple en fonction du temps est de forme: $X_t = a.\ t + b$, pour un polynôme d'ordre 2: $X_t = a_1.t^2 + a_2.t + b$, et ainsi de suite. Une ambiguïté vient de ce que, même si un ajustement par un polynôme d'ordre 5 par exemple, semble très bien décrire visuellement la tendance générale, l'ajustement simple peut être lui aussi déjà hautement significatif. Pour savoir si un degré supérieur est nécessaire, il faut tester si le coefficient de régression partielle attaché à ce degré est significativement différent de 0. Ajoutons qu'un ajustement n'est valable théoriquement que si les erreurs sont indépendantes entre elles et possèdent une distribution normale, cas malheureusement peu fréquent avec des séries spatio-temporelles!

Quoi qu'il en soit, ce ne sont pas les résultats statistiques qui serviront de base à la définition de la tendance générale: l'écologiste, suivant le problème à résoudre, le fait de disposer d'un modèle de variation d'une forme donnée au départ, en fonction de l'échelle de l'étude ou selon sa propre expérience, reste le seul maître de la décision.

On pourrait imaginer bien d'autres formes de tendance générale (exponentielle, logistique, etc.). S+ et R offrent énormément de possibilités pour la définition et l'ajustement de modèles linéaires et non-linéaires. Certains de ces modèles prennent en compte une autocorrélation entre les observations, aspect fondamental dans le cadre de l'analyse de séries spatio-temporelles. Nous renvoyons le lecteur intéressé à la documentation en ligne de ces logiciels respectifs pour plus d'informations à ce sujet.

La tendance générale est parfois liée à l'évidence à un cycle (annuel, lunaire, etc.). La régression sinusoïdale permet de tester statistiquement, et de définir un modèle de variation rigoureusement périodique. Naturellement, ce modèle doit être utilisé avec discernement. Ainsi, même si la variabilité saisonnière existe pour des abondances d'espèces, elle ne s'ajuste pas forcément à une sinusoïde si les amplitudes maximums sont décalées d'une année à l'autre, d'un mois ou plus. Et si alors on étudie les écarts entre les valeurs observés et le modèle, ceux-ci ne correspondent plus à une variabilité indépendante de la variation saisonnière. Ils ne représentent que l'inadéquation du modèle choisi, qui lui, est strictement périodique.

L'équation fonctionnelle sera donnée par:

$$Y = c.\cos(\omega t + \phi) = a.\cos\omega t + b.\sin\omega t$$

En effectuant le changement de variable suivant:

$$X_1 = \cos\frac{2\pi i}{T} \quad \text{et} \quad X_2 = \sin\frac{2\pi i}{T}$$

on se ramène au système linéaire:

$$Y_i = a.X_{1i} + b.X_{2i} + \varepsilon_i$$

identique au cas de la multirégression (système d'équations normales à résoudre).

On doit aussi estimer le déphasage, c'est-à-dire trouver la valeur de ϕ, soit par:

$$Y = c.\cos(\omega t + \phi) = c.\cos\phi.\cos\omega t - c.\sin\phi.\sin\omega t$$

On en déduit:

$$a = c.\cos\phi \quad \text{et} \quad b = -c.\sin\phi; \quad \text{d'où:} \quad \phi = \arctan\frac{-b}{a}$$

Avec le calcul des résidus, on peut tester l'ajustement par analyse de la variance. En appelant variation la somme des carrés des écarts, le modèle se décompose en variation sinusoïdale + variation résiduelle. Sous réserve de normalité, on pourra tester l'ajustement par un test *F* de Fisher (tableau de l'ANOVA; rapport de la variance expliquée par la régression sur la variance résiduelle). Sinon, des tests non paramétriques faciliteront l'interprétation (analyse de variance non paramétrique, corrélation de rangs).

Références:

Frontier, S., 1981. Méthodes statistiques. *Masson, Paris*. 246 pp.

Kendall, M., 1976. Time-series. *Charles Griffin & Co Ltd*. 197 pp.

Legendre, L. & P. Legendre, 1984. Ecologie numérique. Tome 2: La structure des données écologiques. *Masson, Paris*. 335 pp.

Malinvaud, E., 1978. Méthodes statistiques de l'économétrie. *Dunod, Paris*. 846 pp.

Sokal, R.R. & F.J. Rohlf, 1981. Biometry. *Freeman & Co, San Francisco*. 860 pp.

Exemples:

Modèle linéaire de la tendance

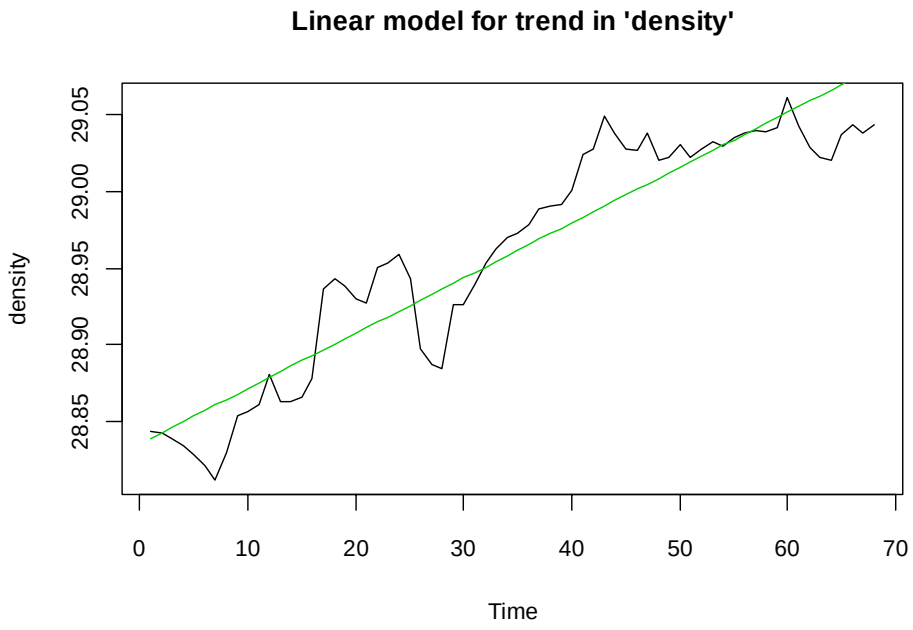
On peut effectuer une régression linéaire sur la série ***Density*** du jeu de données ***marphy*** afin de l'utiliser comme estimation de la tendance générale dans une décomposition de cette série spatio-temporelle. La variable indépendante (x) est le temps que l'on reconstitue à partir de la série spatio-temporelle à l'aide de la fonction ***time()***. Ensuite, on effectue la régression linéaire à l'aide de la fonction ***lm()*** qui demande de spécifier le modèle sous forme **Y ~ X**, avec **Y** étant la variable dépendante et **X** étant la variable indépendante. La méthode ***summary()*** retourne un rapport détaillé sur cette régression:

```
> data(marphy)
> density <- ts(marphy[, "Density"])
> Time <- time(density)
> density.lin <- lm(density ~ Time)
> summary(density.lin)

Call:
lm(formula = density ~ Time)

Residuals:
      Min        1Q    Median        3Q       Max
-0.052226 -0.018014  0.001945  0.017672  0.058895

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.884e+01  6.585e-03 4379.27   <2e-16 ***
Time        3.605e-03  1.659e-04   21.73   <2e-16 ***
```

```
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.02685 on 66 degrees of freedom
Multiple R-Squared: 0.8774,     Adjusted R-squared: 0.8755
F-statistic: 472.3 on 1 and 66 DF,  p-value:     0
```

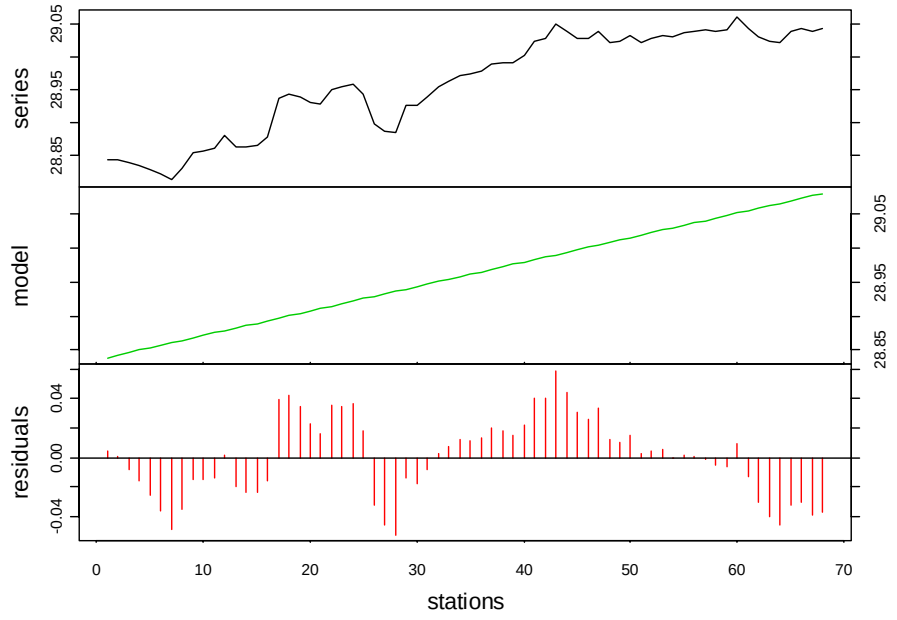
Notons que le tableau de l'ANOVA indique qu'aussi bien la pente **Time** que l'ordonnée à l'origine **(Intercept)** sont très hautement significatifs. Mais rappelons que nous violons ici une hypothèse de base de la régression linéaire puisque les termes d'erreurs présentent très probablement une certaine autocorrélation et ne sont donc pas totalement indépendants entre eux! On peut cependant considérer les résultats de l'ANOVA comme indicatifs, tout en gardant à l'esprit qu'ils sont biaisés. Les valeurs de densité correspondant à ce modèle sont récupérées à l'aide de la méthode ***predict()***, et nous pouvons maintenant superposer le modèle au graphe de la série originale.

```
> xreg <- predict(density.lin)
> plot(density)
> lines(xreg, col=3)
> title("Linear model for trend in 'density'")
```



La décomposition de la série ***density*** est réalisée grâce à ***decreg()*** comme suit. Nous obtenons un objet *'tsd'* qui nous est maintenant familier, ainsi que le type de représentation graphique à l'aide de la méthode ***plot()*** qui lui est associé:

```
> density.dec <- decreg(density, xreg)
> plot(density.dec, col=c(1, 3, 2), xlab="stations")
```



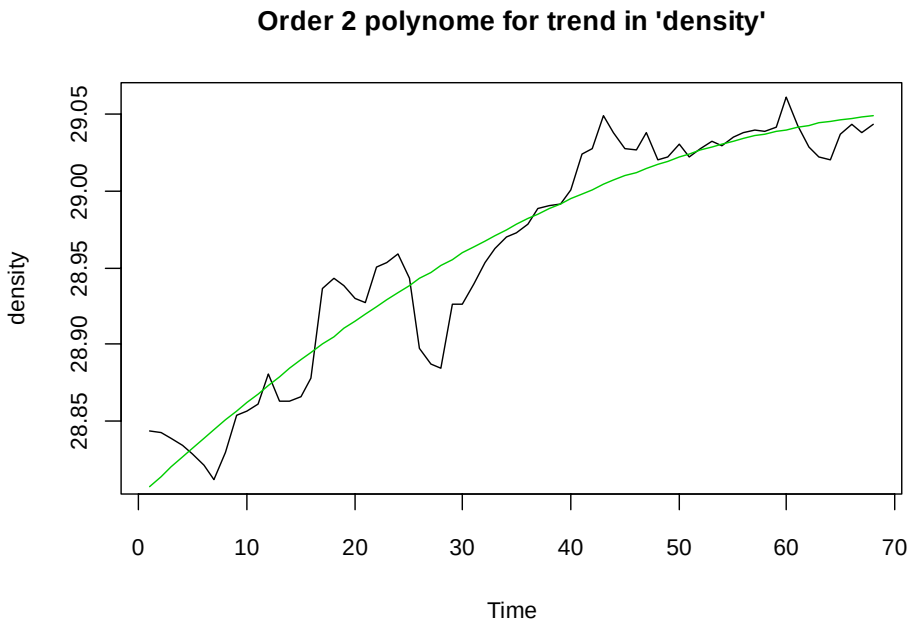
Modèle polynomial de la tendance

Nous pouvons bien entendu nous demander si une droite est le meilleur modèle qui soit pour représenter la tendance générale dans cette série. Ainsi, par exemple, on aurait pu également choisir un polynôme d'ordre 2 pour représenter cette tendance, ce qui donne:

```
> density.poly <- lm(density ~ Time + I(Time^2))
> summary(density.poly)

Call:
lm(formula = density ~ Time + I(Time^2))

Residuals:
      Min        1Q    Median        3Q       Max
-0.067079 -0.010665  0.001498  0.014777  0.045341

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept)  2.880e+01  8.374e-03 3439.404  < 2e-16 ***
Time         6.593e-03  5.600e-04   11.773  < 2e-16 ***
I(Time^2)   -4.330e-05  7.866e-06   -5.505 6.73e-07 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.02234 on 65 degrees of freedom
Multiple R-Squared: 0.9164,     Adjusted R-squared: 0.9138
F-statistic: 356.2 on 2 and 65 DF,  p-value:     0
```

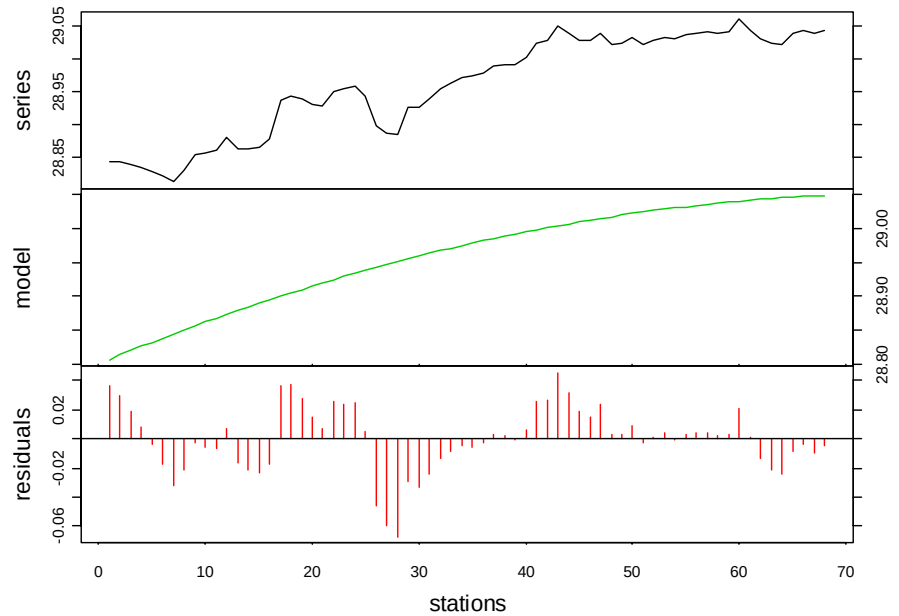
Une régression polynômiale s'obtient en effectuant une régression linéaire sur des termes successifs x, x^2, x^3, x^4,… Pour un polynôme d'ordre 2, on s'arrête au second terme. La formule à utiliser est donc du type: **Y ~ X + I(X^2)** (nous sommes obligés d'utiliser la fonction ***I()*** pour « protéger » noter terme **X^2**, étant donné qu'il ne s'agit plus simplement d'une variable, mais d'une expression (nous renvoyons le lecteur intéressé par cela à l'aide en ligne de la fonction ***lm()***). Avec toutes les précautions que nous avons déjà évoquées pour l'interprétation du tableau de l'ANOVA, nous observons que ce terme supplémentaire **I(Time^2)** est lui aussi très fortement significatif. Un polynôme d'ordre 2 semble donc parfaitement indiqué pour représenter la tendance générale dans cette série. Voyons ce que cela donne graphiquement:

```
> xreg2 <- predict(density.poly)
> plot(density)
> lines(xreg2, col=3)
> title("Order 2 polynome for trend in 'density'")
```



C'est effectivement beaucoup mieux que le modèle linéaire. La décomposition de la série à l'aide de ***decreg()*** se fait comme précédemment:

```
> density.dec2 <- decreg(density, xreg2)
> plot(density.dec2, col=c(1, 3, 2), xlab="stations")
```



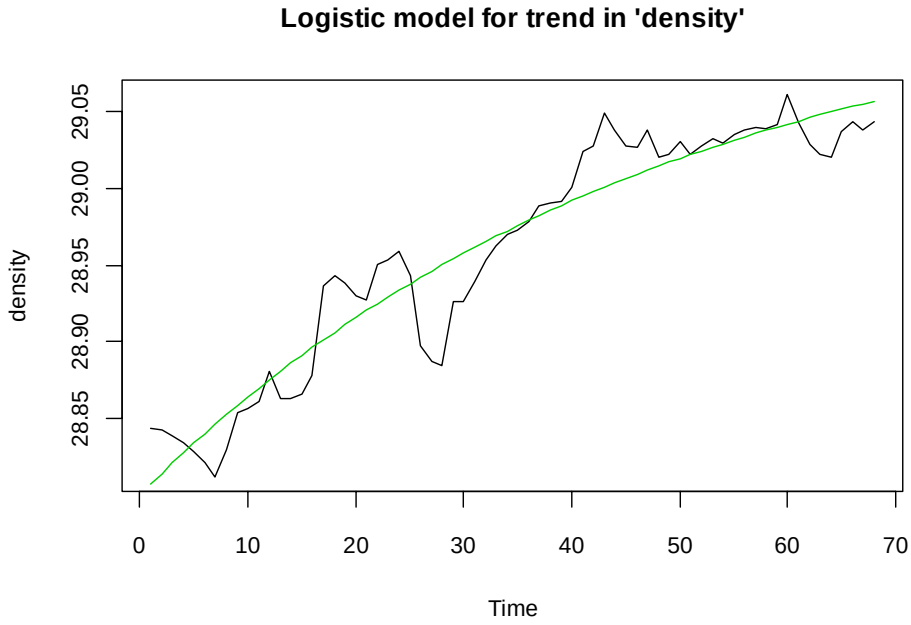
Modèle non linéaire de la tendance

Nous pourrions bien entendu tester également des polynômes d'ordre 3, 4,…, mais nous nous arrêterons-là dans notre exemple pour tester une autres possibilité: l'utilisation d'un modèle non linéaire pour représenter la tendance générale dans cette série, par exemple, une courbe logistique:

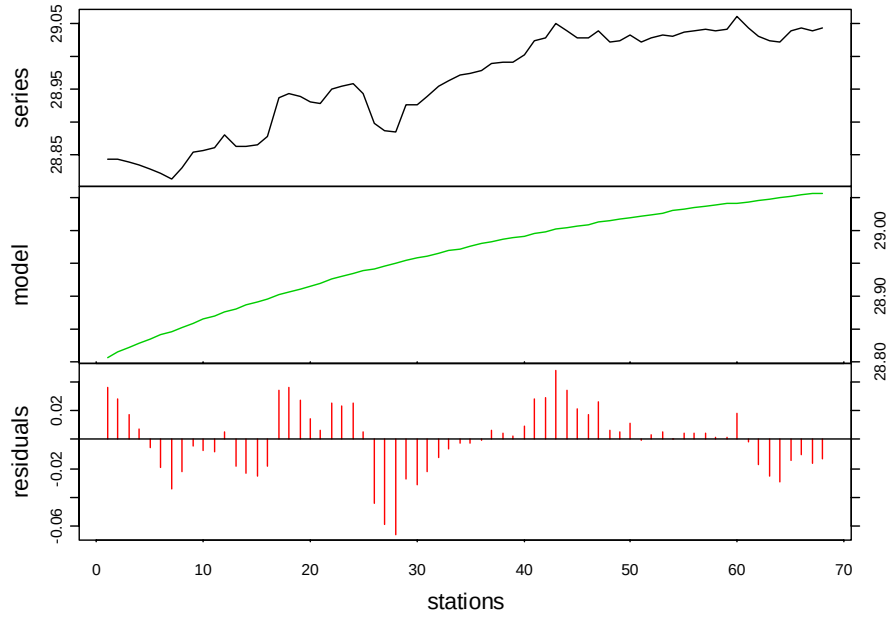
```
> library(stats) # library(nls) in R < 1.9.0
> density.logis <- nls(density ~ SSlogis(Time, Asym, xmid, scal))
> summary(density.logis)

Formula: density ~ SSlogis(Time, Asym, xmid, scal)

Parameters:
       Estimate Std. Error t value Pr(>|t|)
Asym   29.13767    0.03855 755.908  < 2e-16 ***
xmid -210.29714   41.78382  -5.033 4.07e-06 ***
scal   47.28910   10.33741   4.575 2.20e-05 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.023 on 65 degrees of freedom

Correlation of Parameter Estimates:
        Asym   xmid
xmid -0.9722
scal  0.9811 -0.999
```

NNous utilisons maintenant la fonction ***nls()*** en lieu et place de la fonction ***lm()***. La définition du modèle est quelque peu différente: **Y ~ model(X, params,…)**, avec **model()**, une fonction non linéaire. Lorsque la fonction est capable de calculer elle-même ses paramètres de départ, ces derniers ne doivent pas êtres fournis et l'utilisation de la fonction ***nls()*** est alors presque aussi simple que celle de ***lm()***, mais il n'en va pas toujours ainsi. Voyez l'aide en ligne de ***nls()*** pour plus de renseignements sur l'utilisation de la régression non linéaire dans S+ ou R. Nous observons que, ici encore tous les paramètres semblent très hautement significatifs dans le modèle. Nous ne commenterons pas plus loin les résultats renvoyés par la méthode ***summary()***. L'extraction des valeurs prédites par le modèle, le graphe superposé de la fonction de départ et la décomposition de la série s'obtiennent comme avec un modèle linéaire:

```
> xregl <- predict(density.logis)
> plot(density)
> lines(xregl, col=3)
> title("Logistic model for trend in 'density'")
```




```
> density.decl <- decreg(density, xregl)
> plot(density.decl, col=c(1, 3, 2), xlab="stations")
```



Modèle sinusoidal de tendance cyclique

Pour terminer, nous allons illustrer l'utilisation d'un modèle sinusoidal pour représenter une composante saisonnière dans une série. Pour ce faire, nous utiliserons une série artificelle périodique:

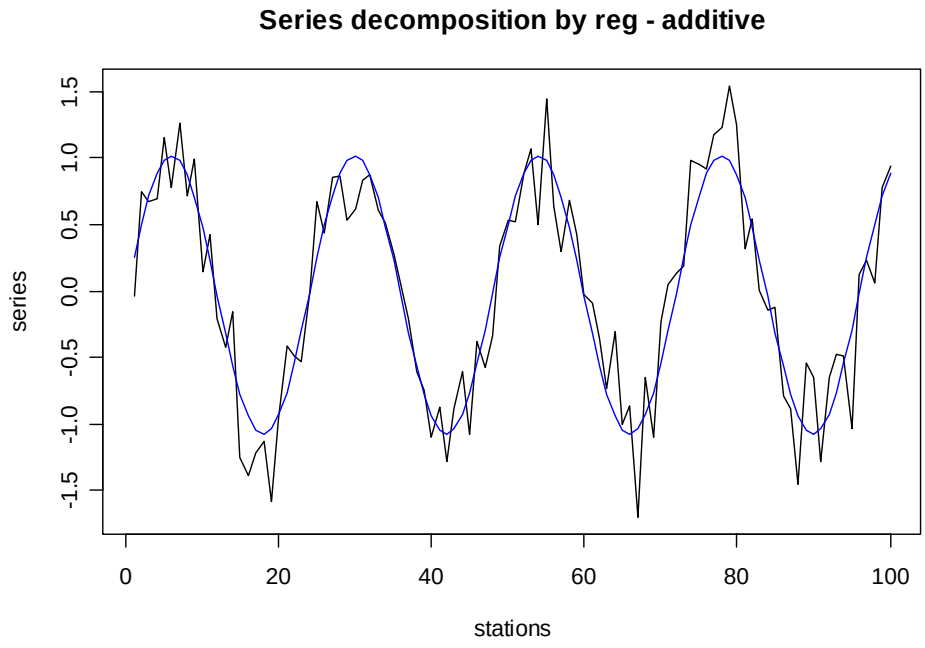
```
> tser <- ts(sin((1:100)/12*pi)+rnorm(100, sd=0.3), start=c(1998,
+ 4), frequency=24)
```

Comme nous l'avons vu dans l'introduction théorique, un changement de variable nous permet de linéariser un modèle sinusoïdal, de sorte que nous pouvons utiliser la fonction ***lm()***, plus simple d'utilisation que la fonction ***nls()***:

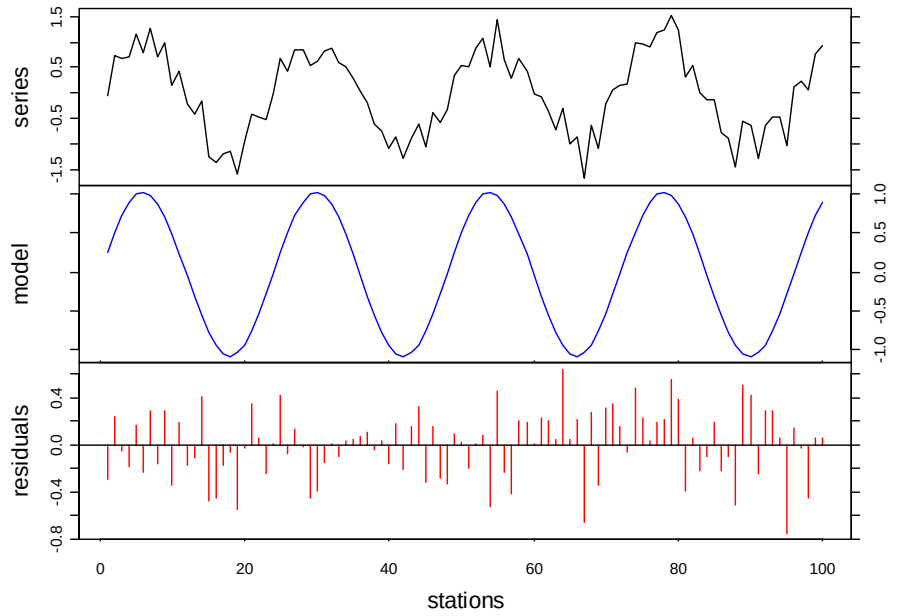
```
> Time <- time(tser)
> tser.sin <- lm(tser ~ I(cos(2*pi*Time)) + I(sin(2*pi*Time)))
> summary(tser.sin)

Call:
lm(formula = tser ~ I(cos(2 * pi * Time)) + I(sin(2 * pi * Time)))

Residuals:
     Min       1Q   Median       3Q      Max
-0.74682 -0.19634  0.03170  0.19409  0.64224

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          -0.02944    0.02875  -1.024    0.308
I(cos(2 * pi * Time)) -0.51424    0.04111 -12.508   <2e-16 ***
I(sin(2 * pi * Time))  0.91312    0.04019  22.720   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.2871 on 97 degrees of freedom
Multiple R-Squared: 0.8719,     Adjusted R-squared: 0.8692
F-statistic: 330.1 on 2 and 97 DF,  p-value:     0
```

Conformément à notre construction de la série aléatoire, les paramètres **cos(…)** et **sin(…)** sont tous deux très significatifs, mais pas l'ordonnée à l'origine **(Intercept)**. Le tracé des graphes et la décomposition de la série à l'aide de ce modèle se font de manière similaire aux exemples précédents:

```
> tser.reg <- predict(tser.sin)
> tser.dec <- decreg(tser, tser.reg)
> plot(tser.dec, col=c(1, 4), xlab="stations", stack=FALSE,
+ resid=FALSE, lpos=c(0, 4))
```



```
> plot(tser.dec, col=c(1, 4, 2), xlab="stations")
```



Décomposition par CENSUS II deccensus()

La méthode CENSUS II (Shiskin & Eisenpress, 1957) souvent utilisée en économétrie, permet de décomposer une série pluriannuelle d'observations mensuelles en trois composantes principales:

1. une composante générale dite trend cyclique C_t résultant de deux mouvements: la tendance évolutive ou mouvement fondamental du phénomène, et les variations cycliques de longue période,

2. une composante saisonnière S_t, dont la courbe lissée présente une reproductibilité plus ou moins accentuée d'une année à l'autre,

3. une composante aléatoire I_t, dont le profil est irrégulier et qui représente l'apparition d'événements occasionnels. Cette composante est théoriquement distribuée normalement et d'autocorrélation nulle.

Le modèle CENSUS II est multiplicatif, on admet que les observations originales X_t sont le produit des 3 composantes précédentes:

$$X_t = C_t \,.\, S_t \,.\, I_t$$

L'influence saisonnière est supposée être proportionnelle à C_t. Equation que l'on peut écrire sous la forme:

$$X_t = A_t \,.\, S_t \quad \text{avec} \quad A_t = C_t \,.\, I_t$$

où A_t désigne la série désaisonnalisée. Dans ce modèle, la composante trend cyclique C_t et la série originale X_t sont exprimées dans les mêmes unités, alors que le facteur saisonnier S_t et la composante aléatoire I_t sont exprimés sous forme de coefficients adimensionnels.

Les composantes C_t, S_t et I_t sont estimées par un processus itératif basé sur l'application d'une succession de moyennes mobiles sur une table de Buys-Ballot. Une propriété intéressante de cette décomposition est qu'on n'observe pratiquement pas de cohérence siginificative entre A_t et S_t aux fréquences saisonnières (voir analyse des séries spatio-temporelles, paragraphe analyse spectrale croisée pour la définition de la cohérence). Les composantes C_t ou A_t peuvent être traitées par l'analyse spectrale (dans la mesure où il n'y a pas de gradient).

Références:

Béthoux, N., M. Etienne, F. Ibanez & J.L. Rapaire, 1980. Spécificités hydrologiques des zones littorales. Analyse chronologique par la méthode CENSUS II et estimation des échanges océan-atmosphère appliqués à la baie de Villefranche sur mer. *Ann. Inst. Océanogr. Paris*, 56:81-95.

Fromentin, J.M. & F. Ibanez, 1994. Year to year changes in meteorological features on the French coast area during the last half-century. Examples of two biological responses. *Oceanologica Acta*, 17:285-296.

Institut National de Statistique de Belgique, 1965. Décomposition des séries chronologiques en leurs composantes suivant différentes méthodes. Etudes statistiques et économétriques. *Bull. Stat. INS*, 10:1449-1524.

Philips, J. & R. Blomme, 1973. Analyse chronologique. *Université Catholique de Louvain, Vander ed.* 339 pp.

Rosenblatt, H.M., 1968. Spectral evaluation of BLS and CENSUS revised seasonal adjustment procedures. *J. Amer. Stat. Assoc.*, 68:472-501.

Shiskin, J. & H. Eisenpress, 1957. Seasonal adjustment by electronic computer methods. *J. Amer. Stat. Assoc.*, 52:415-449.

Exemple:

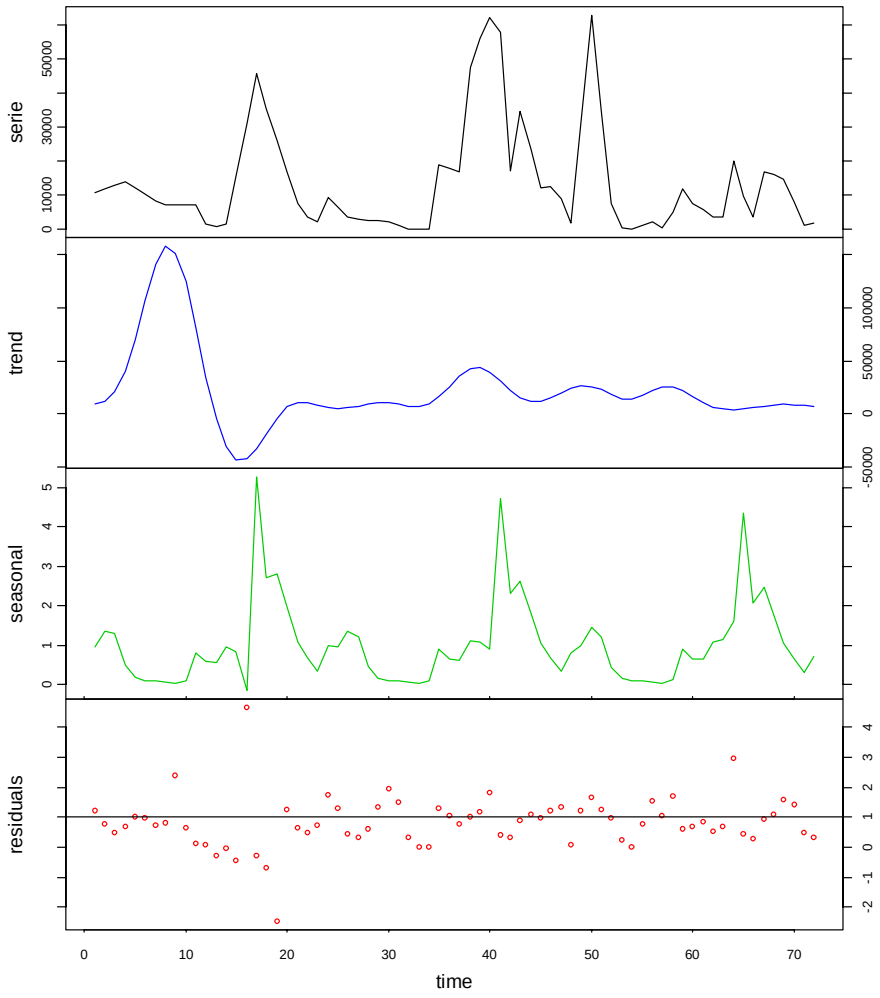
Remarque : cette fonction n'est disponible que dans la version 1.2-0 de PASTECS sous R. Elle n'existe pas dans la version 1.0-1 sous S+ 2000!

Si l'on part du tableau ***releve***, régularisé comme suit, on obtient une série qui s'étale sur à peu près 3 ans et demi. Comme la méthode CENSUS II demande des périodes entières, nous allons enlever les quelques mois de la dernière année incomplète et travailler uniquement sur les 3 premières années complètes (***rel.ts2***):

```
> # Seulement sous R (pas disponible dans S+)
> data(releve)
> rel.regy <- regul(releve$Day, releve[2:7], xmin=6, n=87,
+ units="daystoyears", frequency=24, tol=2.2, methods="linear",
+ datemin="21/03/1989", dateformat="d/m/Y")
> rel.ts <- tseries(rel.regy)
> start(rel.ts)
[1] 1989    6
> end(rel.ts)
[1] 1992   20
> rel.ts2 <- window(rel.ts, end=c(1992,5))
```

La décomposition CENSUS II donne:

```
> rel.dec2 <- deccensus(rel.ts2[, "Melosul"], trend=TRUE)
> plot(rel.dec2, col=c(1,4,3,2))
```



A noter que ce modèle est multiplicatif. Sur le graphe, seule la composante ***trend*** a une échelle semblable à la série de départ. Les autres composantes (***seasonal*** et ***residuals***) sont des facteurs multiplicatifs qui fluctuent donc autour de 1.

Décomposition par LOESS decloess()

LOESS est une méthode de régression polynomiale locale. Pour chaque valeur X_t d'une série X, on va considérer les k voisins à gauche et à droite, éventuellement pondérés (par exemple par la distance les séparant de X_t). On effectue une régression par les moindres carrés d'un polynôme d'ordre p (habituellement p vaut 1 ou 2), et on récupère la valeur prédite au temps t (voir Cleveland *et al*, 1992). Si l'ordre p du polynôme vaut 0, on se ramène à une moyenne mobile.

Avec un choix judicieux de k, et éventuellement en réitérant le processus, on va pouvoir extraire la tendance générale. Comme avec la méthode des moyennes mobiles, on va aussi pouvoir désaisonnaliser une série en prenant une fenêtre égale à un an, donc, en considérant 12 termes pour des données mensuelles (13 en fait pour une fenêtre centrée sur les observations de la série, voir paragraphe moyenne mobile). Une combinaison de ces deux traitements va permettre de décomposer une série en tendance, cycle saisonnier et résidus selon un modèle additif. On peut aussi n'extraire que le cycle saisonnier et les résidus (surtout utile lorsque la tendance générale *est* périodique sur un an).

Comme les résidus ont rarement une distribution normale et ne sont souvent pas indépendants entre eux dans les séries, les hypothèses de base indispensables pour la régression à l'aide de la méthode des moindres carrés sont violées la plupart du temps. Ainsi, pour obtenir une estimation plus exacte des composantes, il est possible d'utiliser une méthode de régression plus robuste (*i.e.*, moins sensible à la violation de ces hypothèses de base). On peut par exemple décider de pondérer les valeurs de manière inversement proportionnelle à leur influence respective sur la régression. Ainsi, une valeur qui, à elle seule, aurait tendance à tirer la courbe vers elle (valeur isolée, très éloignée du nuage formé par toutes les autres) recevra une pondération très faible afin de minimiser son effet.

Références:

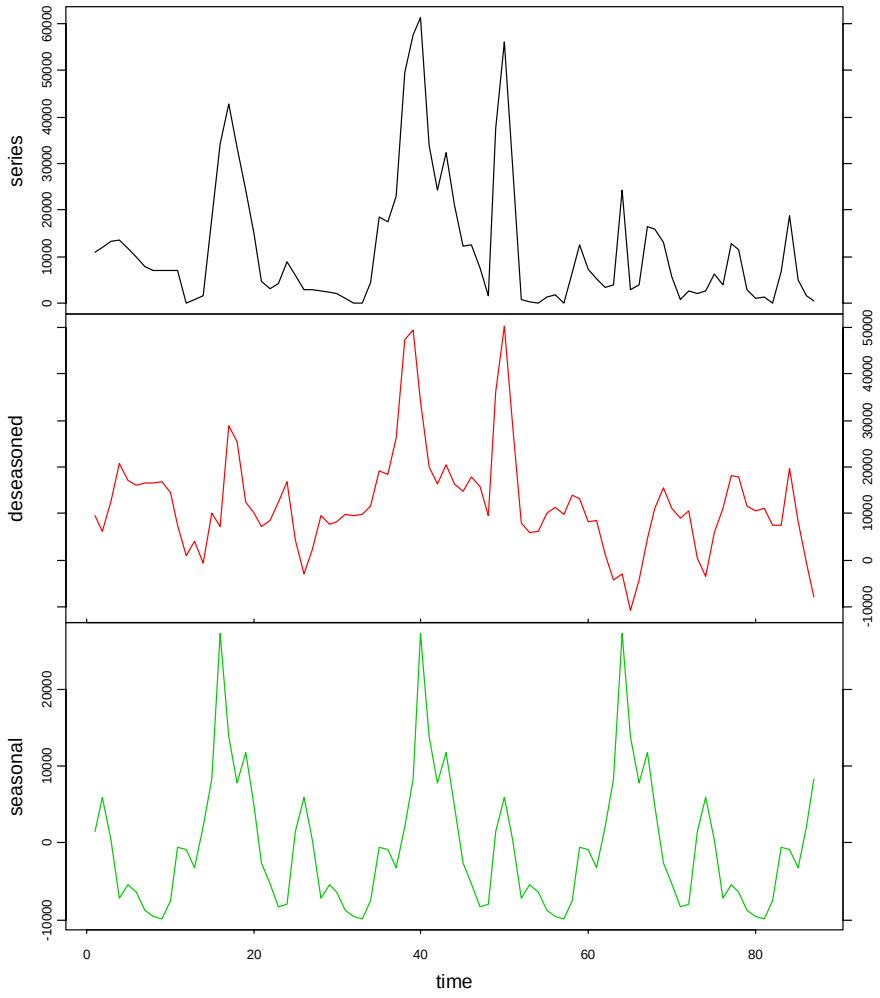
Cleveland, W.S., E. Grosse & W.M. Shyu, 1992. Local regression models. Chapter 8 of Statistical Models in S. J.M. Chambers & T.J. Hastie (eds). *Wadsworth & Brook/Cole*.

Cleveland, R.B., W.S. Cleveland, J.E. McRae, & I. Terpenning, 1990. STL: A seasonal-trend decomposition procedure based on loess. *J. Official Stat.*, 6:3-73.

Exemple:

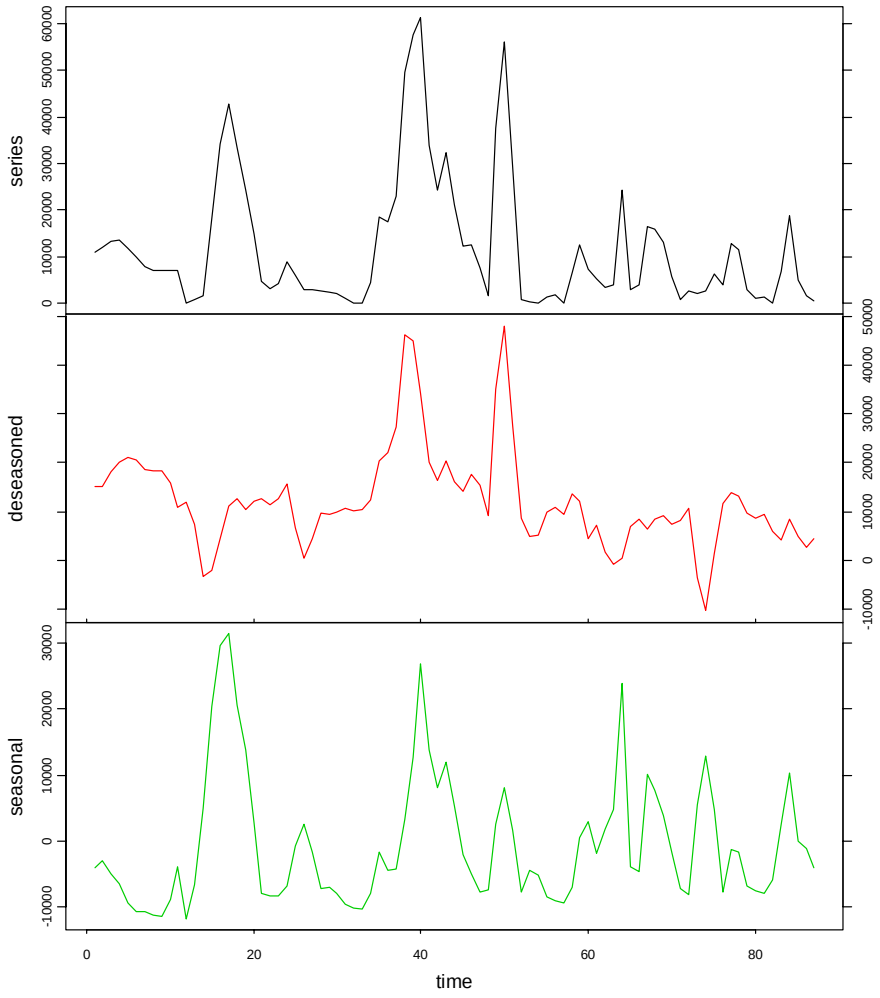
L'utilisation de ***decloess()*** pour extraire un cycle saisonnier à partir de la méthode des différences a déjà été présentée dans l'exemple de ***tsd()***. Nous le reprenons ici, et nous explorerons ensuite d'autres valeurs pour les paramètres, afin de voir leurs effets sur la décomposition:

```
> data(releve)
> melo.regy <- regul(releve$Day, releve$Melosul, xmin=9, n=87,
+ units="daystoyears", frequency=24, tol=2.2, methods="linear",
+ datemin="21/03/1989", dateformat="d/m/Y")
> melo.ts <- tseries(melo.regy)
> melo.dec <- decloess(melo.ts, s.window="periodic")
> plot(melo.dec, col=1:3)
```



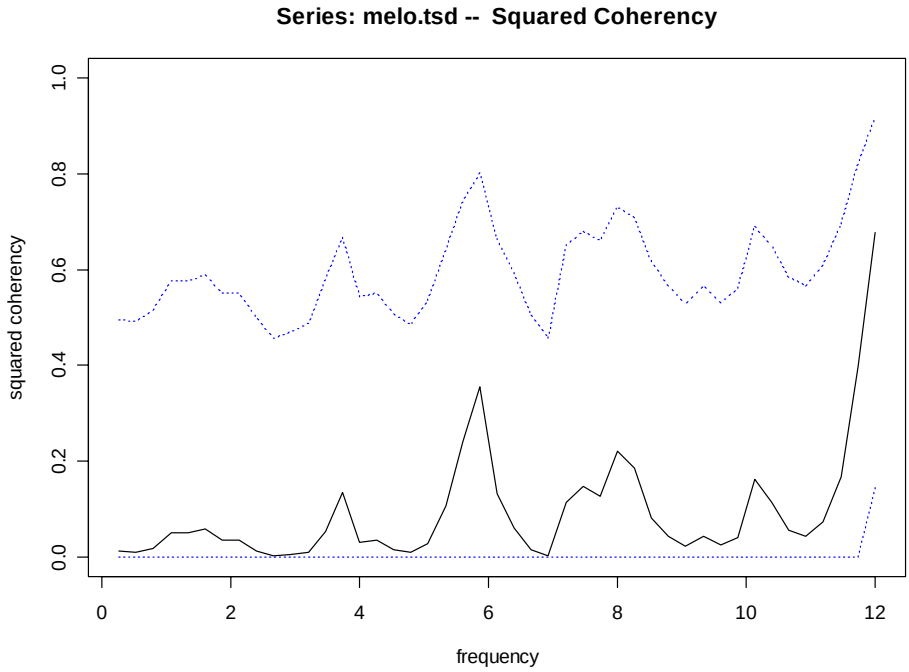
Avec l'option **s.window = "periodic"**, la forme de la composante cyclique saisonnière est quelconque, mais est reproduite à l'identique pour toutes les années (puisque la même moyenne mensuelle est utilisée pour chaque mois de chaque année). Avec l'option **s.window = 13** et **s.degree = 0**, on obtient pratiquement le même résultat. Par contre, avec **s.degree = 1**, on va permettre une variation de la composante cyclique saisonnière d'une année sur l'autre:

```
> melo.dec2 <- decloess(melo.ts, s.window=13, s.degree=1)
> plot(melo.dec2, col=1:3)
```



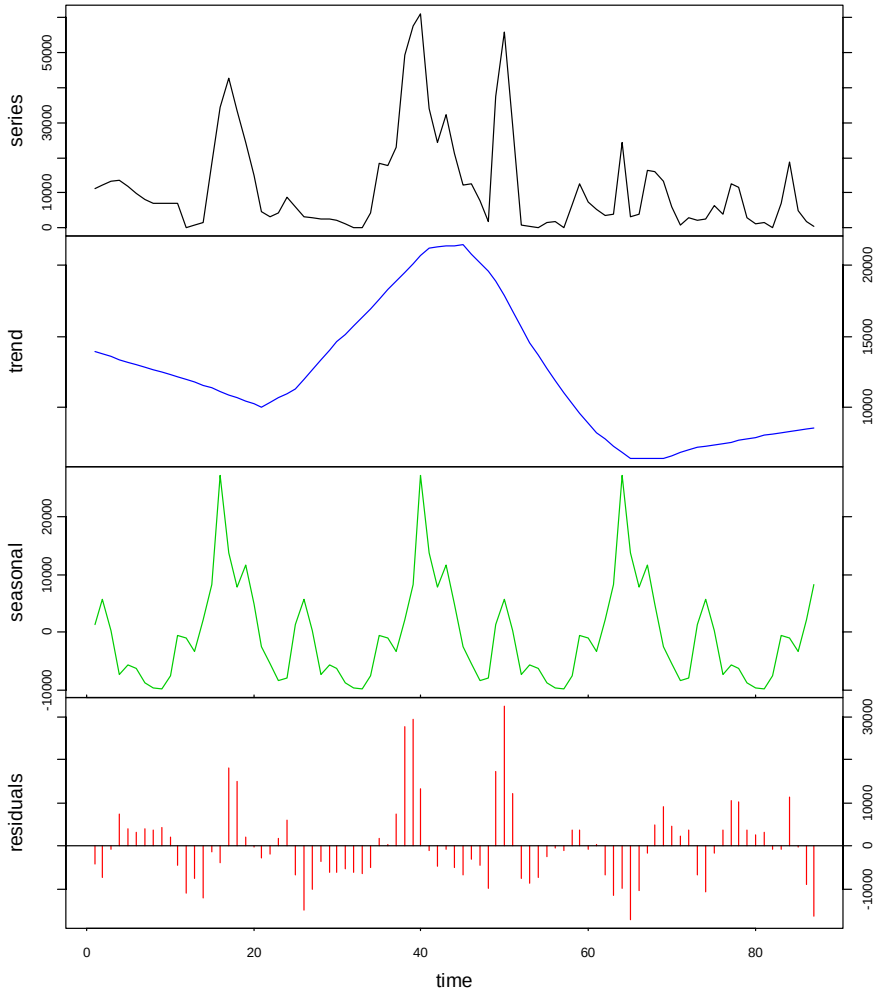
Une étude de la cohérence (voir analyse des séries spatio-temporelle, paragraphe analyse spectrale croisée) entre les deux composantes ***deseasoned*** et ***seasonal*** sous R donne:

```
> melo.tsd <- tseries(melo.dec2)
> melo.spc <- spectrum(melo.tsd, spans=c(3,3), plot=FALSE)
> plot(melo.spc, plot.type="c")         # Sous R uniquement
```



On peut voir que la cohérence est très faible entre les deux séries, c'est le principe même de la décomposition saisonnière. Enfin, sous R uniquement, on peut également extraire la tendance (selon un modèle additif, voir introduction théorique), simultanément à un cycle saisonnier qui se répète d'année en année (**s.window = "periodic"**) ou qui évolue au cours du temps (**s.window ≈ frequency**), en précisant **trend = TRUE**:

```
> # The following command works only in R!
> melo.dec3 <- decloess(melo.ts, s.window="periodic", trend=TRUE)
> plot(melo.dec3, col=c(1,4,3,2))
```



Méthodes d'ordination et de classification de données multivariées

Les méthodes présentées dans ce chapitre ne sont pas spécifiques aux séries spatio-temporelles, mais peuvent aussi être utilisées en complément des techniques spécifiques présentées ci-avant. En outre, elles représentent une autre facette importante de l'analyse des données en écologie côtière. C'est pourquoi nous leurs consacrons un chapitre séparé. La plupart de ces méthodes existent déjà sous S+ ou R, que ce soit en standard ou dans des bibliothèques spécialisées telles que ***ade4*** ou ***vegan***. D'ailleurs, n'étant pas spécifiques au traitement des séries spatio-temporelles, et en vertu de l'organisation des différentes librairies dans S+ et R, elles ne sont pas incluses dans la librairie PASTECS.

La façon la plus simple d'étudier des données multivariées, est de les représenter graphiquement, par exemple à l'aide de la fonction ***pairs()*** présentée dans le chapitre fonctions de base du langage S. Cette fonction trace toutes les combinaisons possibles de nuages de points entre deux variables et les arrangent selon une matrice carrée. Ce type de présentation est utile pour un petit nombre de variables (4 ou 5 au maximum), mais on arrive très vite à une présentation difficile à décrypter au fur et à mesure que le nombre de variables augmente. **Xgobi** (http://www.research.att.com/areas/stat/xgobi/) est un programme gratuit spécialisé dans la visualisation de données multivariées et offre beaucoup plus de possibilités, y compris le MDS (voir cadrage multidimensionnel) pour réduire le nombre de dimensions. Xgobi est interfaçable directement avec S+ ou R mais il s'agit d'un logiciel tournant sous X11 (Unix). Une nouvelle version, dénommée **ggobi** fonctionne à la fois sous Unix et sous Windows et est interfaçable à partir de R (voir http://www.ggobi.org). Il n'y a, à notre connaissance, pas encore d'implémentation de ce programme pour Macintosh.

La plupart du temps, on cherche à réduire les dimensions de façon à ne conserver que les plus représentatives, c'est-à-dire, celles qui expriment la plus grande fraction de la variation totale de la matrice de données. Les méthodes d'**ordination** (ACP, AFC, ACP-VI, ACC,…) se chargent de cette opération. On peut également choisir de représenter les **similarités** ou **dissimilarités** entre les observations par un indice appelé **distance**. On peut ensuite effectuer une **classification** sur base de la **matrice des distances** obtenue, ce qui mène à générer un **dendrogramme**. On peut également représenter cette matrice de distance en 2 ou 3 dimensions grâce au **MDS**. A la fois les méthodes d'ordination et de classification permettent d'identifier des groupes distincts dans l'échantillon, sur base de critères multivariés. Ces deux types d'analyses sont habituellement utilisés de manière complémentaire. Ils sont présentés tour à tour dans ce chapitre, y compris des exemples pratiques réalisés dans S+/R.

Analyse en composantes principales (ACP)

L'**analyse en composantes principales** permet d'effectuer une ordination des descripteurs et des observations (voir par exemple Legendre & Legendre, 1984). PASSTEC 2000 utilise une méthode itérative d'extraction des vecteurs propres et des composantes: l'algorithme des moyennes pondérées réciproques (Hill, 1973) qui présente l'avantage de permettre l'accès aux ordinations sous contraintes (formes canoniques). Les méthodes d'ACP sous S+/R font, quant à elles, appel au calcul matriciel classique pour leur calcul, mais des librairies additionnelles proposent d'autres formes d'ACP également.

Dans ce type d'analyse on peut introduire les valeurs de variables externes et construire une ordination dans laquelle les axes extraits, indépendants, sont également les plus corrélés aux variables externes. La méthode d'estimation est alors modifiée en tenant compte d'ajustement avec un deuxième groupe de descripteurs à chaque pas d'itération. Les formes canoniques de l'ACP sont l'**analyse de redondance**, la méthode d'**analyse des corrélation canoniques**, l'**analyse des variables canoniques** (**analyse discriminante**). La synthèse de Jongman *et al* (1987) décrit toutes ces techniques. Un intérêt majeur des analyses avec contraintes est de visualiser les variables biologiques dans différents espaces: l'espace des **facteurs environnementaux** et/ou l'espace défini par les **covariables spatio-temporelles**.

Références:

Hill, M.O., 1973. Reciprocal averaging: an eigenvector method of ordination. *J. Ecol.*, 61:237-249.

Jongman, R.H.G., C.J.F. ter Braak & O.F.R. van Tongeren, 1987. Data analysis in community and landscape ecology. *Pudoc Wageningen ed.* 298 pp.

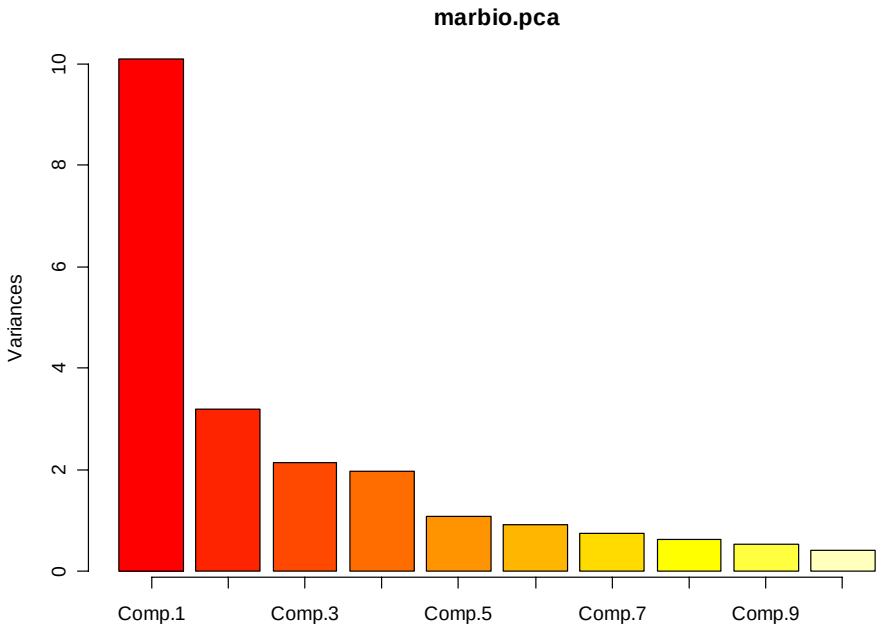
Legendre, L. & P. Legendre, 1984. Ecologie numérique. Tome 2: La structure des données écologiques. *Masson, Paris.* 335 pp.

Exemple:

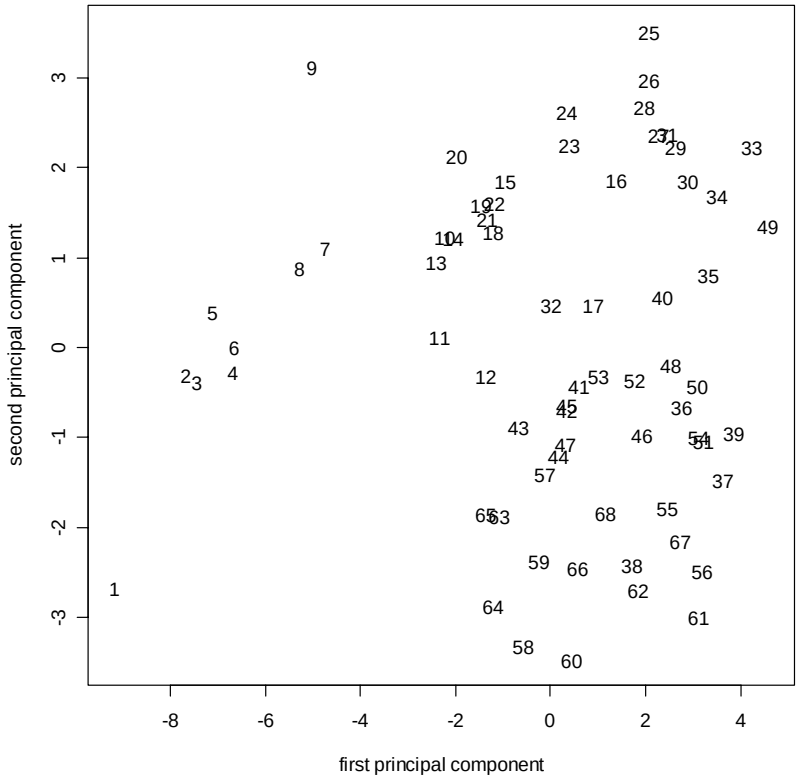
Voici une analyse en composantes principales classique réalisée dans S+ ou R:

```
> data(marbio)
> marbio.pca <- princomp(log(marbio+1), cor=TRUE)
> summary(marbio.pca)
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4
Standard deviation     3.1752223 1.7871712 1.4611111 1.4056765
Proportion of Variance 0.4200849 0.1330825 0.0889519 0.0823303
Cumulative Proportion  0.4200849 0.5531674 0.6421193 0.7244496
                          Comp.5    Comp.6    Comp.7    Comp.8
Standard deviation     1.0451577 0.9602461 0.8664106 0.7974705
Proportion of Variance 0.0455148 0.0384197 0.0312778 0.0264983
Cumulative Proportion  0.7699644 0.8083840 0.8396618 0.8661602
                          Comp.9   Comp.10   Comp.11   Comp.12
Standard deviation     0.7387467 0.6497202 0.6447846 0.5967232
Proportion of Variance 0.0227395 0.0175890 0.0173228 0.0148366
Cumulative Proportion  0.8888996 0.9064886 0.9238114 0.9386480
                         Comp.13   Comp.14   Comp.15   Comp.16
Standard deviation     0.5241174 0.5115779 0.4622373 0.3973671
Proportion of Variance 0.0114458 0.0109047 0.0089026 0.0065792
Cumulative Proportion  0.9500938 0.9609985 0.9699011 0.9764803
                         Comp.17   Comp.18   Comp.19   Comp.20
Standard deviation     0.3537922 0.3143593 0.3064875 0.2767779
Proportion of Variance 0.0052154 0.0041176 0.0039139 0.0031919
Cumulative Proportion  0.9816957 0.9858133 0.9897272 0.9929191
                         Comp.21   Comp.22   Comp.23   Comp.24
Standard deviation     0.2680745 0.2185241 0.1646432 0.1523722
Proportion of Variance 0.0029943 0.0019897 0.0011295 0.0009674
Cumulative Proportion  0.9959134 0.9979031 0.9990326 1.0000000
```

```
> plot(marbio.pca)
```




```
> marbio.pc <- predict(marbio.pca)
> plot(marbio.pc[,1], marbio.pc[,2], xlab="first principal component",
+ ylab="second principal component", type="n")
> text(marbio.pc[,1], marbio.pc[,2], labels=1:nrow(marbio.pc))
```



Analyse en composantes principales biplot

L'analyse en composante principale (ACP) dite **biplot**, combine la représentation factorielle et l'expression de la diversité estimée par l'indice de diversité de Simpson (ter Braak, 1983). L'indice de Simpson (1949) *SI* est souvent considéré comme moins instable que beaucoup d'autres indices de diversité. Si on appelle p_{ik} le pourcentage de l'espèce *k* à la station *i*:

$$SI = \|p_i\|^2 = \sum_{k=1}^{m} p_{ik}^2$$

avec: *k* = 1, 2, ..., *r* espèces et *i* = 1, 2, ..., *n* sites. Si on considère un très grand nombre *m* d'individus, la mesure sera sensible aux espèces rares. Au contraire pour un *m* petit, elle sera sensible aux espèces dominantes. L'indice de Simpson *SI*, peut être transformé en mesure de la diversité α soit par 1 - *SI* (Pielou, 1969) ou 1/*SI* (Hill, 1973).

La distance euclidienne au carré entre sites représentera une mesure de la diversité β:

$$d_{ij}^2 = \sum_{k=1}^{r} (p_{ik} - p_{jk})^2$$

L'ACP, sous certaines conditions, permettra de visualiser les diversités α et β. Soit un tableau de départ à *n* lignes observations (sites) et *m* colonnes descripteurs (proportions d'espèces). Par construction, la somme de chaque ligne est égale à 1 (somme des pourcentages d'espèces à chaque station *i*). L'indice de Simpson *SI* est simplement égal à la norme (carré de la longueur ou du module) du vecteur définissant chaque site. ter Braak (1983) présente deux ACP permettant une ordination de la diversité: une **ACP non centrée** et une **ACP centrée**.

ACP non centrée

Si l'ACP est effectuée sur des données non centrées, cela signifie que les axes principaux dans l'espace des espèces vont passer par l'origine des axes de départ. Si on considère les normes des *n* sites dans l'espace factoriel à *m* dimensions, on retrouve évidemment les normes antérieures. Cependant lorsqu'on ne conserve qu'un petit nombre d'axes, comme par construction ils concentrent la plus grande partie de la variance, la norme d'origine est assez bien prédite. Ainsi il suffit de visualiser la distance d'un site à l'origine des axes principaux 1 et 2, pour avoir une bonne appréciation de la diversité α. Si le site est loin de l'origine, *SI* fort, sa diversité est faible, et vice-versa. La diversité β également pourra être visualisée directement par les proximités relatives entre sites.

Il peut être intéressant d'effectuer une double représentation sites/espèces dans l'espace factoriel. Ce type de projection arbitraire n'est pas recommandé pour l'interprétation (Saporta, 1990), cependant il peut se justifier ici dans la mesure où par un certain algorithme on peut à partir des proximités sites/espèces retrouver les proportions de telle espèce *k* vis-à-vis de tel site *i*. Pour cela il suffit d'effectuer la représentation en considérant comme coordonnées pour les espèces les éléments des vecteurs propres normés à 1, et les composantes normées aux valeurs propres correspondantes. La proportion de l'espèce *k* au site *i* peut s'approximer par le produit de sa distance à l'origine par celle de la projection du site sur l'axe joignant l'espèce à l'origine.

ACP centrée

Dans le cas d'une ACP centrée ce ne sont plus les proportions des espèces qui forment le tableau de départ, mais leur écart à leur moyenne (p_{ik} - p_k avec p_k = 1/*n* . Σp_{ik}). L'hyperplan défini par les axes principaux ne passe plus par l'origine de départ, il est

placé au centroïde des sites. Pour interpréter l'ordination comme dans le cas de l'ACP non centrée, il est donc obligatoire de connaître la position exacte de l'origine de départ.

Si les éléments du $i^{ème}$ vecteur propre sont définis par v_{1i}, v_{2i}, ...,v_{mi} (avec $\Sigma v_{ki}^2 = 1$), la projection de l'origine *z* sur l'axe *i* est donnée par:

$$z_i = -(v_{1i}.p_1 + v_{2i}.p_2 + ... + v_{mi}.p_m)$$

Autrement dit, la formule classique pour définir la valeur de la composante *i* pour un point de coordonnées $\{-p_1, -p_2, ..., -p_m\}$ par rapport au centroïde. *z* peut se représenter comme un site. La distance au carré d'un site par rapport à l'origine *z* se calculera par (cas de deux axes retenus):

$$s^2 = \sum_{k=1}^{m} p_k^2 - (z_1^2 + z_2^2)$$

L'index *SI* pour un site sera approximé par s^2 qui représente la distance au carré par rapport à l'origine de départ. Les sites à haute diversité seront situés près de *z*, ceux à faible diversité, loin de *z*.

Références:

Ibanez, F., 2000. Diversité et ordination. *Océanis*, (sous presse).

Pielou, E.C., 1969. Ecological diversity. *Wiley and Sons, New York*.

Saporta, G., 1990. Probabilités, analyse des données et statistiques. *Technip, Paris*. 492 pp.

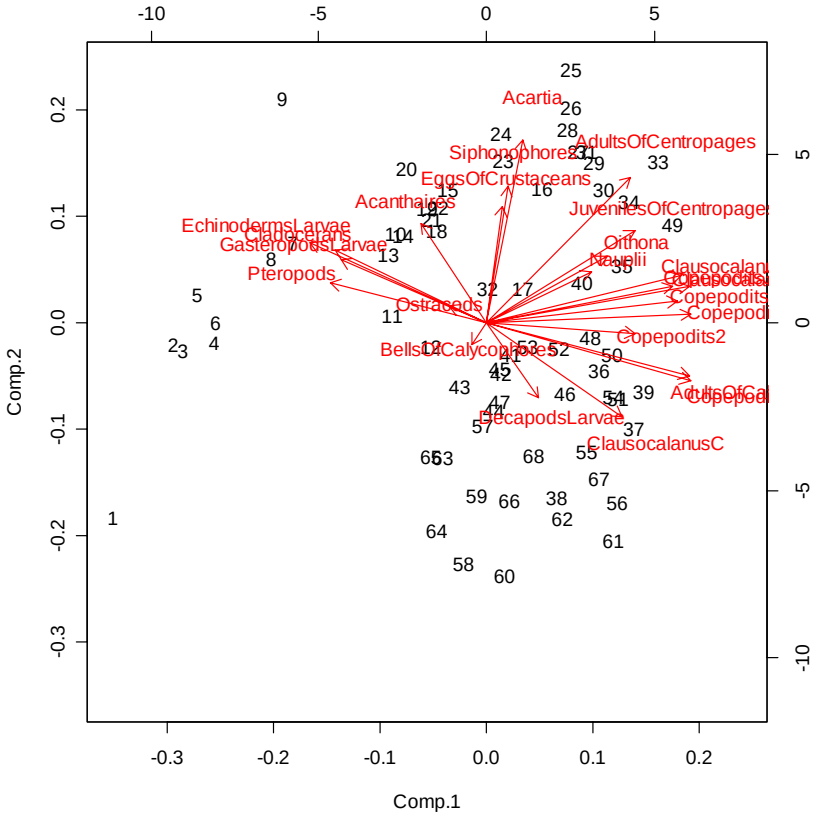
Simpson, E.H., 1949. Measurement of diversity. *Nature*, 163:688.

ter Braak, C.J.F., 1983. Principal component biplots and alpha and beta diversity. *Ecology*, 64:454-462.

Exemple:

En reprenant le même exemple que ci-dessus pour l'ACP, nous avons:

```
> data(marbio)
> marbio.pca <- princomp(log(marbio+1), cor=TRUE)
> biplot(marbio.pca)
```



A noter que les graphiques (tant dans l'exemple ACP que dans l'ACP biplot) peuvent apparaître en miroir, dans S+ notamment, suite au fait que le signe des composantes principales peut être inversé (cela arrive lorsque l'algorithme de calcul sous-jacent est différent), tout en donnant une solution également valable.

Autres méthodes d'ordination (AFC, ACP-VI, ACC, RDA,…)

Tout ce que nous venons de voir sur l'ACP n'illustre qu'une infime partie de ce qu'il est possible de faire avec le langage S en matière d'ordination. Nous invitons le lecteur intéressé par d'autres méthodes à consulter l'aide en ligne de S+ et de R, et notamment à lire l'aide relative aux fonctions ***ca()***, ***supplc()***, ***supplr()*** de la librairie ***multiv***, ***factanal()***, ***cancor()*** de la librairie ***stats*** et ***corresp()***, ***mca()*** de la librairie ***MASS***. De plus, des librairies de fonctions supplémentaires permettent des traitements plus poussés, ainsi que des représentations graphiques en 3D, ou bien encore mettant en évidence différents groupes (couleurs, forme et taille des symboles, envelope convexe autour des groupes – **convex hull**–). Voir notamment les librairies additionelles ***ade4***, ***CoCoAn*** ou ***vegan***. Comme pour tous les graphes sous S+ ou R, il est également possible de travailler de manière interactive sur certains graphes d'ordination. Donc, les possibilités sont énormes et débordent très nettement du cadre de ce document dédié plus spécifiquement à l'analyse des séries spatio-temporelles. Une introduction plus détaillée des méthodes multivariées en général se trouvera au chapitre 11 de Venables & Ripley (2002).

Référence:

Venables, W.N. & B.D. Ripley, 2002. Modern applied statistics with S-plus. 4th ed. Springer, New York. 495 pp.

Classification: matrice de distance & dendrogramme

Les méthodes présentées dans ce paragraphe font intervenir une **matrice de distances** (**similarité** ou **dissimilarité**). La matrice de distances est calculée dans un premier temps, et ensuite, elle est représentée sous forme d'un dendrogramme (ou grâce au MDS, voir cadrage multidimensionnel).

Le dendrogramme, arbre représentatif d'une classification n'est pas exclusivement destiné à l'exploitation de séries multivariées. Cependant, de même que les méthodes d'ordination, les techniques de classification sont souvent incontournables pour l'interprétation: reconnaissance des discontinuités, des intervalles de temps homogènes, des périodes de changement des amplitudes et des interrelations entre les descripteurs.

La méthode programmée renvoie aux techniques de classification ascendante hiérarchique, unifiées par l'algorithme de Williams et Lance (1967). L'ensemble de la théorie peut être trouvée dans l'ouvrage de Legendre & Legendre (1984).

Les distances disponibles sont au nombre de dix (du moins sous R):

1. Euclidienne
2. Euclidienne^2
3. maximale
4. binaire
5. de Manhattan
6. de Bray-Curtis
7. de Jaccard
8. de Kulczynski
9. de Camberra
10. de Gower

L'aide en ligne des fonctions ***dist()*** et ***vegdist()*** (sous R uniquement, packages ***stats*** et ***vegan***) donne plus d'informations sur ces distances.

Les procédures agglomératives pour le dendrogramme sont au nombre de sept:

1. Liens simples
2. Liens complets
3. Association moyenne
4. Groupement médian
5. Groupement centroïde
6. Association de Ward
7. Association de McQuitty

Voir l'aide en ligne de la fonction ***hclust()***.

Références:

Daget, J., 1976. Les modèles mathématiques en écologie. *Masson, Paris*.

Lance, G.N. & W.T. Williams, 1967. A general theory of classificatory sorting strategies. I Hierarchical systems. *Computer J.*, 9:373-380

Lance, G.N. & W.T. Williams, 1967. A general theory of classificatory sorting strategies. II Clustering systems. *Computer J.*, 10:271-277

Legendre, L. & P. Legendre, 1984. Ecologie numérique. Tome 2. La structure des données écologiques. *Masson*, 335 pp.

Legendre, P., 1986. Constrained clustering. *In*: "Developments in numerical Ecology". Eds. P. Legendre & L. Legendre. *NATO ASI Series. Series G: Ecological Sciences*, 14:289-309.

Monestiez, P., 1978. Méthodes de classification automatique sous contraintes spatiales. *Biométrie et Ecologie (Société française de Biométrie)*, 1:367-381

Dans S+ et R, la fonction centrale pour le calcul d'une matrice de distance (similarité ou dissimilarité) est la fonction ***dist()***, complétée par la fonction ***vegdist()*** du package ***vegan*** dans R. Une fois la matrice créée (en fait, il s'agit d'un vecteur contenant les valeurs correspondantes à la semi-matrice à l'exclusion de la diagonale dans S+, et d'un objet de classe *'dist'* dans R), on peut lui appliquer d'autres fonctions, en particulier ***hclust()*** qui calcule les données nécessaires à la réalisation d'un dendrogramme. La façon dont le dendrogramme lui-même est affiché diffère selon que l'on travaille dans S+ ou dans R. Sous S+, il faut appeler la fonction ***plclust()***. Sous R, étant donné qu'un objet de class *'hclust'* est créé, et que cet objet possède ses propres méthodes, il suffit d'appeler la méthode ***plot()*** pour obtenir le dendrogramme. Le nombre d'arguments possibles pour le calcul des clusters est également plus exhaustif sous R que sous S+ (voir description en ligne de ces fonctions respectives: `?function`).

Exemple:

La matrice de distance avec la métrique de Manhattan, pour les 5 premières espèces et pour les 10 premières observations du jeu de données ***marbio*** est obtenue comme suit (elle est stockée dans la variable ***obs.dist***):

```
> library(stats) # Uniquement sous R, library(mva) pour R < 1.9.0
> data(marbio)
> obs.dist <- dist(as.matrix(marbio[1:10, 1:5]), "man")
> obs.dist
      1    2    3    4    5    6   7   8   9
2    70
3   100   30
4    73   37   57
5    27   59   79   46
6    47   71   91   46   20
7   381  311  281  308  354  334
8   273  203  173  200  246  230 136
9  1127 1057 1027 1054 1100 1080 746 854
10  300  230  200  227  273  253 163  35 869
```

Etant donné que S+ ne crée pas de classe particulière pour la matrice de distance, la commande précédente lance un affichage « brut » de l'objet créé sous S+:

```
> obs.dist
 [1]   70  100   73   27   47  381  273 1127  300   30   37   59
[13]   71  311  203 1057  230   57   79   91  281  173 1027  200
[25]   46   46  308  200 1054  227   20  354  246 1100  273  334
[37]  230 1080  253  136  746  854   35  869
attr(, "Size"):
[1] 10
attr(, "method"):
[1] "manhattan"
```

Ce type d'affichage est bien sûr nettement moins clair que sous R. La matrice de distance est stockée de la même façon dans R, sous forme d'un vecteur linéaire. Pour s'en

convaincre, on peut « déclasser » l'objet avant son affichage par `unclass(obs.dist)`, ce qui donnera sensiblement le même résultat que sous S+. Dans la suite de ce chapitre, toutes les sorties correspondent à un traitement dans R. A chaque fois que des différences (autres que l'affichage de la matrice de distance, discutée ci-dessus) apparaissent, elles seront soulignées.

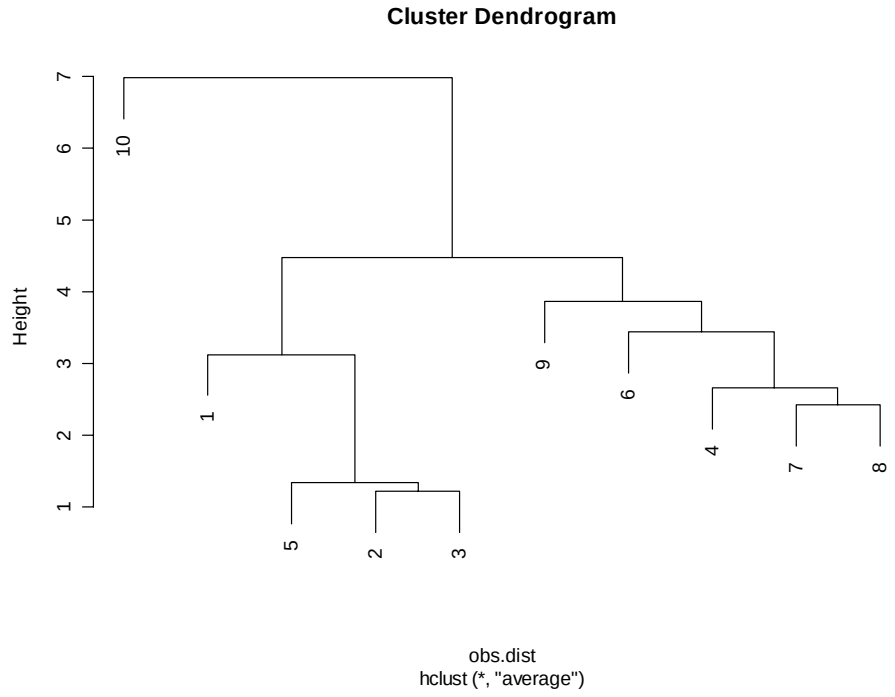
A noter que le calcul se fait sur les lignes par défaut (donc sur les observations dans le cas présent). Il est possible de classifier les colonnes simplement en transposant la matrice de départ à l'aide de la fonction ***t()***, comme illustré dans l'exemple suivant (cette fois-ci, une distance euclidienne est calculée après passage aux logs et la matrice est stockée dans la variable ***esp.dist***):

```
> esp.dist <- dist(t(log(marbio[1:10, 1:5]+1)), "eucl")
> esp.dist
                 Acartia AdultsOfCalanus Copepodits1 Copepodits2
AdultsOfCalanus 15.117477
Copepodits1     15.246813       0.6286087
Copepodits2     10.034657       7.3098618    7.422492
Copepodits3      7.512755       8.2050639    8.268037    4.211182
```

Outre la différence d'affichage de la matrice de distance (sous forme d'un vecteur linéaire), l'intitulé des lignes et des colonnes n'est pas repris sous S+. Cela ne change cependant rien aux calculs qu'il est possible de réaliser ultérieurement.

Reprenons l'exemple précédent, mais en calculant les distances pour les 10 premiers descripteurs. Le dendrogramme est calculé à l'aide de la fonction ***hclust()***. La méthode d'aggrégation des groupes doit être précisée (ici **"average"**). Sous R, le dendrogramme correspondant est obtenu à l'aide de la méthode ***plot()*** de l'objet *'hclust'* obtenu:

```
> obs.dist <- dist(log(marbio[1:10, 1:10]+1), "eucl")
> hc <- hclust(obs.dist, method="average")
> plot(hc)                                  # Sous R uniquement
```



Sous S+, il faut remplacer la dernière ligne de commande par:

```
> plclust(hc)                                  # Sous S+ uniquement
```

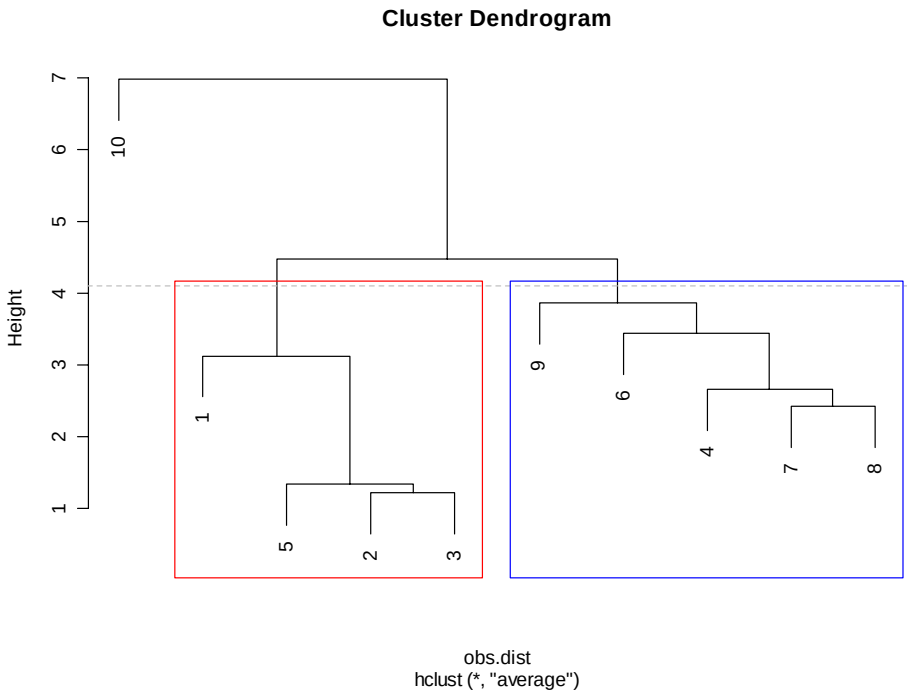
pour obtenir un résultat similaire.

Il est ensuite possible de représenter une coupure sur le dendrogramme. Pour cela, on peut facilement définir une nouvelle fonction ***lines.hclust()***, comme suit:

```
> lines.hclust <- function(hclust.obj, h, lty=2,...) {
+ lines(c(-1, length(hclust.obj$order)+1), c(h, h), lty=lty,...)}
```

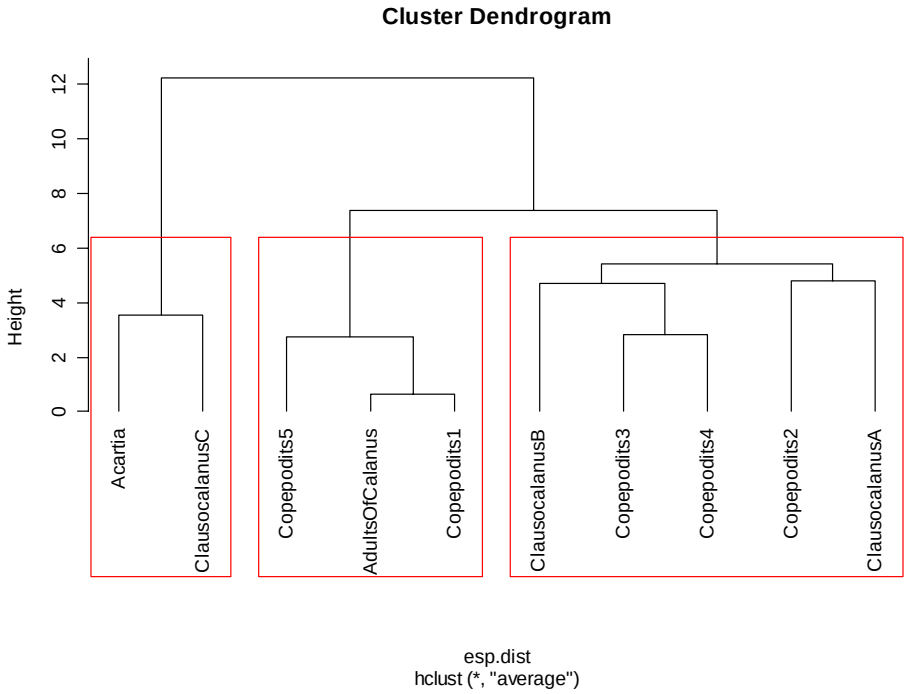
Utilisons cette fonction pour marquer une coupure sur le dendrogramme, ici, à la distance **h = 4.1**. Sous R, il est également possible de pointer la coupure de manière interactive sur le graphique à l'aide de ***identify.hclust()*** ou d'encadrer les groupes obtenus ou certains de ces groupes à l'aide de ***rect.hclust()*** au même niveau de coupure (le premier groupe contenant l'élément '10' n'est pas encadré dans l'exemple):

```
> lines.hclust(hc, 4.1, col="gray")        # utiliser col=2 sous S+
> rect.hclust(hc, h=4.1, which=c(2,3), border=c(2,4)) # Sous R
```



Un second exemple effectue la même analyse que ci-dessus, mais en effectuant une classification des espèces. Pour cela, il suffit de transposer la matrice de départ à l'aide de ***t()*** avant de calculer les distances. Sous R, le label des points dans le dendrogramme se fait automatiquement. Le dendrogramme est, cette fois-ci, dessiné de manière classique avec tous les nœuds qui se prolongent jusqu'à la ligne de base grâce à l'option **hang = -1**. Ensuite, on demande au programme de séparer 3 groupes et de les encadrer:

```
> esp.l <- t(log(marbio[1:10, 1:10]+1))  # Matrice log transposée
> esp.dist <- dist(esp.l, "eucl")        # Matrice distances
> hc <- hclust(esp.dist, "average")      # Construire dendro.
> plot(hc, hang=-1)                      # Afficher dendro.
> rect.hclust(hc, k=3)                   # Encadrer 3 groupes
```



On obtient le même résultat sous S+ (sauf l'encadrement des groupes qui n'y est pas implémenté) avec un peu plus de code car, d'une part, il faut ajouter les intitulés des éléments (noms des espèces) à la main, mais en plus, il faut ajuster les marge afin que les noms les plus longs puissent être affichés correctement. Tout ce traitement est réalisé automatiquement, en interne, sous R. Sous S+, on entrera donc:

```
> esp.l <- t(log(marbio[1:10, 1:10]+1))  # Matrice log transposée
> esp.dist <- distance(esp.l, "eucl")    # Matrice distances
> hc <- hclust(esp.dist, "average")      # Construire dendro.
> par(mar=c(6.2,4,4,2)+.1)               # Modifier les marges
> plclust(hc, hang=-1, labels=labels(esp.l)[[1]])  # Créer dendro.
> par(mar=c(5,4,4,2)+.1)            # Restaurer marges par défaut
```

Sous R, il est possible de sauvegarder une liste des différents éléments appartenant à chacun des groupes en attribuant le résultat de ***rect.hclust()*** à une variable. Toutefois, dans les deux logiciels, il est aussi possible de séparer les groupes dans la matrice ou d'ajouter une colonne ***group*** à la matrice de départ, à l'aide de la fonction ***cutree()*** qui est plus indiquée dans ce dernier cas:

```
> group <- cutree(hc, k=3)           # Crée 3 groupes
> group                              # Les affiche à l'écran
       Acartia AdultsOfCalanus     Copepodits1     Copepodits2
             1               2               2               3
   Copepodits3     Copepodits4     Copepodits5  ClausocalanusA
             3               3               2               3
 ClausocalanusB  ClausocalanusC
             3               1
> group <- as.factor(group)          # Les transforment en facteurs
> esp.l <- cbind(esp.l, group)       # et les ajoutent à la matrice
```

Encore une fois, étant donné les différences d'implémentation des fonctions sous R et sous S+, les groupes ne sont pas affichés de la même façon:

```
> group <- cutree(hc, k=3)             # Crée 3 groupes
```

```
> group                                  # Les affiche sous S+
[1] 2 3 3 1 1 1 3 1 1 2
attr(, "height"):
[1] 5.414361 3.535559 2.715499
```

On note en outre que les numéros attribués aux différents groupes ne sont pas les mêmes. R attribue le numéro 1 au groupe contenant le premier élément, alors que S+ attribue le numéro 1 au groupe qui se détache le plus haut dans l'histogramme. Mis à part ces quelques différences, les 3 groupes obtenus sont identiques dans les deux logiciels pour l'exemple considéré.

> *Ce chapitre ne présente qu'une petite partie des possibilités de S+ ou de R en matière de classification. Le **K-means clustering** est également disponible dans S+ et dans la bibliothèque **stats** de R. Les bibliothèques **ccluster** et **mcluster** de R vont beaucoup plus loin encore et offrent d'autres méthodes de **clustering convexe** ou de **clustering selon un modèle**. De même, sous R, il y a d'autres méthodes comme **bea() Bond Energy Algorithm** dans le package **multiv**, ou encore, le partitionnement récursif dans **rpart**, y compris des méthodes plus avancées faisant intervenir une classification moyenne calculée à partir de plusieurs arbres comme dans les librairies R **randomForest** ou **ipred**. Consultez l'aide en ligne de ces fonctions et librairies pour plus d'informations.*

Cadrage multidimensionnel (MDS)

Le cadrage multidimensionnel cherche à minimiser, dans un espace dont le nombre de dimensions est fixé au départ par l'utilisateur, la différence entre distance écologique entre observations et distances sur un graphique. A l'origine la méthode MDS (nonmetric MultiDimensional Scaling) a été proposée par Kruskal (1964).

Etant donné une matrice des distances d_{ij} quelconque entre observations, le principe est de rechercher une configuration de ses n points dans un espace euclidien de dimension fixée (représentation graphique), telle que les distances δ_{ij} entre ces points respectent au mieux l'ordre défini par d: si $d_{ij} < d_{ik}$, on cherche à avoir $\delta_{ij} < \delta_{ik}$ pour le maximum de points. Dans l'algorithme on minimisera une quantité appelée **stress**:

$$\text{stress} = \sqrt{\sum_{i,j} (d_{ij} - \delta_{ij})^2 / \sum_{i,j} d_{ij}^2}$$

La méthode est la suivante: on part d'une configuration euclidienne donnée en considérant par exemple les distances entre points dans le premier plan propre d'une analyse en composantes principales (ACP), et on estime une régression entre ces distances δ et les distances écologiques ordonnées d. On en déduit la valeur du stress correspondant. Cette première configuration est affinée par des itérations en déplaçant légèrement les positions des points selon une méthode de gradient pour diminuer le stress. On repasse à la phase de régression, etc., jusqu'à convergence à un seuil acceptable choisi par l'utilisateur pour les différences entre les stress de deux itérations successives (10^{-3} ou 10^{-5} par exemple). De fait le choix des δ est de peu d'importance et l'utilisation des distances entre points correspondant aux deux premières composantes est très adéquat.

Pour juger de l'ajustement opéré en comparant distances écologiques originales et distances graphiques obtenues à partir des coordonnées des axes de la MDS, on figure les relations entre les deux dans un **diagramme de Shepard**. Plus les positions des points s'alignent sur la bissectrice et plus l'ajustement est bon. En fait dans le calcul c'est cette droite de régression qui est optimisée, cependant selon le seuil choisi pour la fin des itérations, il peut encore y avoir des déviations importantes, et dans ce cas il faut recommencer avec un seuil plus petit.

La différence avec l'ACP est que la dimension de l'espace est fixée à l'avance et que les solutions ne sont pas emboîtées: la meilleure représentation à trois dimensions ne se déduit pas de la meilleure représentation à deux dimensions en rajoutant un troisième axe.

L'intérêt majeur de cette méthode est d'obtenir une visualisation très précise des groupes que l'on aurait pu déceler par une classification, alors que pour le même résultat il aurait fallu un très grand nombre de composantes d'une ACP. Comme on optimise l'ordre des distances graphiques par rapport à l'ordre des distances écologiques, tout coefficient de distance écologique peut être pris en compte, ce qui n'est pas vrai avec les techniques dérivées de l'ACP (distances euclidiennes obligatoirement).

Références:

Kruskal, J.B., 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1-27.

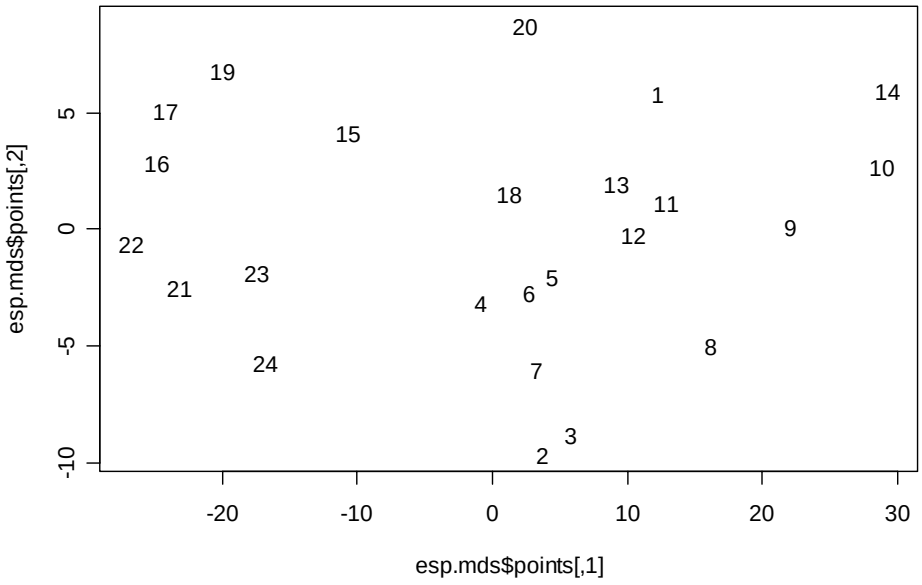
Legendre, P. & L. Legendre., 1998. Numerical Ecology. *Elsevier*. 853 pp.

Saporta, G., 1990. Probabilités, analyse des données et statistiques. *Technip, Paris*. 492 pp.

Exemple:

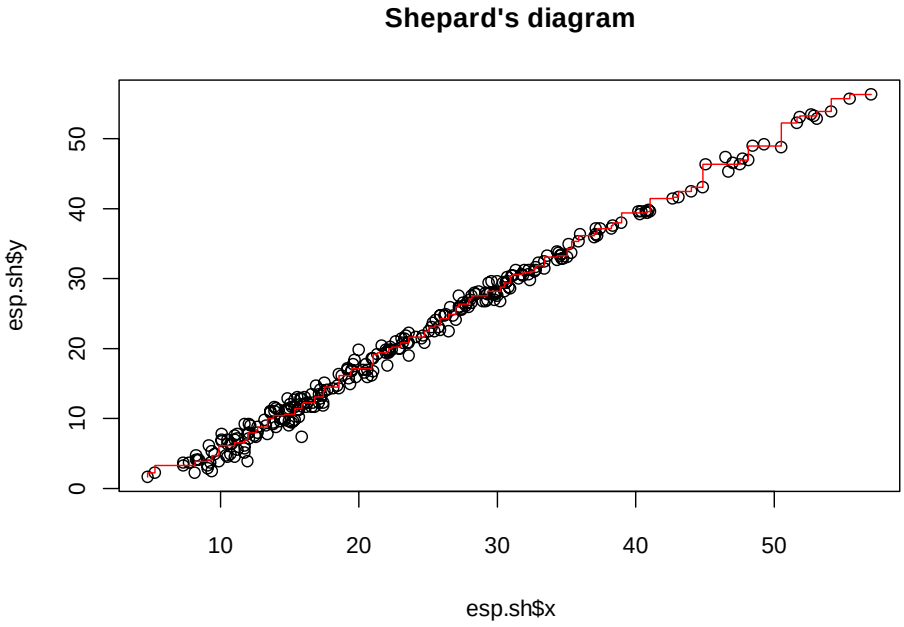
Analysons le jeu de données ***marbio*** à l'aide d'un cadrage multidimensionnel:

```
> library(MASS)                  # Sous R < 1.9.0 seulement
> data(marbio)                   # Sous R uniquement
> esp.dist <- dist(t(log(marbio+1)), "eucl")
> esp.mds <- isoMDS(esp.dist)
initial  value 5.255671
iter   5 value 3.746465
iter  10 value 3.322370
iter  15 value 3.280309
iter  20 value 3.260275
iter  20 value 3.257681
iter  20 value 3.257557
final  value 3.260275
converged

> plot(esp.mds$points, type="n")
> text(esp.mds$points, labels=as.character(1:ncol(marbio)))
```



Et le diagramme de Shepard est obtenu comme suit:

```
> esp.sh <- Shepard(esp.dist, esp.mds$points)
> plot(esp.sh)
> lines(esp.sh$x, esp.sh$yf, type="S", col=2)
> title ("Shepard's diagram")
```


Shepard's diagram
esp.sh$y
0
10
20
30
40
50
10
20
30
40
50
esp.sh$x

Partie II: fonctions de PASTECS

Cette partie présente les différentes fonctions de la librairie PASTECS par ordre alphabétique et explique leurs différents arguments en détails.

Description des fonctions par ordre alphabétique

abund(x, f = 0.2)

Description:

> *Trie les descripteurs (habituellement les espèces dans une matrice espèce x stations) en fonction de leur abondance soit en nombre de valeurs non-nulles, soit en nombre d'individus (en log). Le coefficient* **f** *donne un poids plus ou moins important à l'un ou à l'autre de ces deux critères.*

Arguments:

- **x:** un data frame contenant les différentes variables à classer selon leur abondance, en colonnes (généralement, des dénombrements d'espèces).

- **f:** le poids donné, lors du tri, au critère d'abondance en terme de nombre d'individus dénombrés (strictement compris entre 0 et 1; le poids donné au nombre de valeurs non nulles étant **1 - f**). La valeur par défaut 0.2 donne suffisamment d'importance au nombre de zéros pour récupérer les espèces abondantes du point de vue de ce critère, tout en permettant de récupérer à l'autre extrémité les espèces rares, mais localement abondantes.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

print.abund(), summary.abund(), plot.abund(), lines.abund(), identify.abund(), extract.abund()

Exemple:

Voir partie I.

AutoD2(series, lags = c(1, nrow(series)/3), step = 1, plotit = TRUE, add = FALSE,...)

Description:

> *Calcule l' autoD2 pour un ensemble de séries spatio-temporelles régulières et en trace le graphe.*

Arguments:

- **series:** des séries régulières de type *'rts'* sous S+ ou *'ts'* sous R.
- **lags:** les valeurs minimale et maximale à prendre en compte pour les décalages. Par défault, la valeur minimale est 1 et la valeur maximale est le tiers du nombre d'observations dans la série.
- **step:** le pas à utiliser pour incrémenter les décalages entre la valeur minimale et la valeurs maximale renseignée dans **lags**. Par défaut: 1.
- **plotit:** si **TRUE** (par défaut), le graphe correspondant est tracé.
- **add:** si **TRUE**, le graphe de l'autoD2 est ajouté au graphe actif courant au lieu de créer un nouveau graphe. **FALSE** par défaut.

Correspondance dans PASSTEC 2000:

Correspond à ***Cycles -> Méthode D2*** en mode ***Multivarié*** sous PASSTEC 2000.

Voir aussi:

CrossD2(), CenterD2()

Exemple:

Voir partie I.

buysbal(x, y = NULL, frequency = 12, units = "years", datemin = NULL, dateformat = "m/d/Y", count = FALSE)

Description:

> *Crée un tableau de Buys-Ballot à partir d'une série régulière dont l'unité temporelle est l'année ("years") ou le jour ("days"), ou à partir d'un vecteur temps et un vecteur observations (éventuellement irrégulièrement espacés ou avec trous). Si* ***count*** *=* ***TRUE****, la fonction ne crée pas la table, mais détermine le nombre d'observations qui serviront à calculer chaque case de cette table.*

Arguments:

- **x:** soit un vecteur temps (dans ce cas **y** doit contenir les observations correspondantes de la série), soit un objet série spatio-temporelle régulière *'rts'* ou *'ts'* (dans ce cas, **y** doit être nul).
- **y:** les données de la série, échantillonnées aux temps indiqués dans **x**, ou NULL (par défaut) si **x** est une série temporelle régulière.
- **frequency:** la fréquence à utiliser pour le tableau de Buys-Ballot. Par défaut, 12, ce qui correspond à des données mensuelles pour une unité temporelle **"years"** (une colonne est créée pour chaque mois).
- **units:** unité temporelle pour **x**. Soit **"years"** (par défaut) auquel cas aucune transformation d'échelle ne sera opérée, soit **"days"**. Dans ce dernier cas, **x** sera transformé en une échelle temporelle **"years"** à l'aide de la fonction ***daystoyears()***.
- **datemin:** une chaîne de caractères représentant la date qui correspond à la valeur minimale de **x**, au format indiqué dans **dateformat**. Par exemple, si on garde la valeur par défaut de **dateformat, datemin** = **"04/23/1998"** signifie que l'on veut utiliser le 23 avril 1998 comme première date. Sous R, on peut aussi utiliser des dates au format POSIXt. Par défaut, **datemin** = **NULL**, auquel cas la date minimale correspond à **min(x)**. Si **units** = **"years"**, aucune transformation n'est opérée et cet argument est ignoré.
- **dateformat:** indiquez le format de la date. Par exemple: **"d/m/Y"** ou **"m/d/Y"** (par défaut). Notez la différence, dans R, entre **"y"** qui définit les années sur 2 chiffres (par exemple 87 pour 1987) et **"Y"** qui définit les années sur 4 chiffres (1987). **Veuillez à ne pas vous tromper à ce niveau!**
- **count:** par défaut **FALSE**, le tableau de Buys-Ballot est construit. Si **TRUE**, la fonction ne renvoie pas le tableau de Buys-Ballot, mais une table indiquant le nombre d'observations qui serviront à calculer chaque case du tableau de Buys-Ballot (voir exemple).

Correspondance dans PASSTEC 2000:

Première partie du traitement lancé par ***Filtrage -> Buys-Ballot + Census*** en mode ***Univarié***.

Voir aussi:

daystoyears(), tsd()

Exemple:

Voir partie I

CenterD2(series, window = nrow(series)/5, plotit = TRUE, add = FALSE, type = "l", level = 0.05, lhorz = TRUE, lcol = 2, llty = 2, ...)

Description:

> *Calcule le D2 au centre pour un ensemble de séries spatio-temporelles régulières avec une fenêtre donnée (***window***) et en trace le graphe.*

Arguments:

- **series:** des séries régulières de type *'rts'* sous S+ ou *'ts'* sous R.
- **window:** la longueur de la fenêtre à prendre en compte pour le calcul du D2 au centre. Plus la fenêtre est longue, plus le nombre de changements détectés dans la série sera faible. Un bon compromis entre fenêtre courte, mais beaucoup de changements détectés et fenêtre longue, mais très peu de changements détectés, est d'utiliser la valeur maximale renvoyée par l'AutoD2 pour les mêmes séries.
- **plotit:** si **TRUE** (par défaut), le graphe correspondant est tracé.
- **add:** si **TRUE**, le graphe du D2 au centre est ajouté au graphe actif courant au lieu de créer un nouveau graphe. **FALSE** par défaut.
- **type:** le type de graphe. Par défaut **"l"**, seule la ligne brisée du D2 est tracée. Une autre option intéressante est **"b"**, les points et la ligne sont tracés.
- **level:** le seuil statistique à utiliser pour le test du χ^2. Par défaut **0.05** (p = 5%).
- **lhorz:** si **TRUE** (par défaut), une ligne horizontale est dessinée sur le graphe, qui matérialise le seuil de significativité du test au niveau de **level**.
- **lcol:** la couleur à utiliser pour le trait horizontal de significativité. Par défaut, la couleur **2** est utilisée.
- **llty:** le style de trait pour la ligne horizontale de seuil de significativité. Par défaut, le style **2** (trait pointillé) est utilisé.
- **…:** des paramètres optionnels supplémentaires pour le graphe du D2 au centre.

Correspondance dans PASSTEC 2000:

Correspond à ***Cycles -> D2 au centre*** en mode ***Multivarié*** sous PASSTEC 2000.

Voir aussi:

AutoD2(), CrossD2()

Exemple:

Voir partie I.

CrossD2(series1, series2, lags = c(1, nrow(series1)/3), step = 1, plotit = TRUE, add = FALSE, ...)

Description:

> *Calcule le crossD2 pour deux ensembles de séries spatio-temporelles régulières et en trace le graphe.*

Arguments:

- **series1:** premier ensemble de séries régulières de type *'rts'* sous S+ ou *'ts'* sous R.
- **series2:** second ensemble de séries régulières de type *'rts'* sous S+ ou *'ts'* sous R. La série 2 sera calculée en retard sur la série 1. Inverser les deux arguments pour avoir 1 en retard sur 2.
- **lags:** les valeurs minimale et maximale à prendre en compte pour les décalages. Par défault, la valeur minimale est 1 et la valeur maximale est du tiers du nombre d'observations dans la série.
- **step:** le pas à utiliser pour incrémenter les décalages entre la valeur minimale et la valeurs maximale renseignée dans **lags**. Par défaut: **1**.
- **plotit:** si **TRUE** (par défaut), le graphe correspondant est tracé.
- **add:** si **TRUE**, le graphe du crossD2 est ajouté au graphe actif courant au lieu de créer un nouveau graphe. **FALSE** par défaut.

Correspondance dans PASSTEC 2000:

Correspond à ***Cycles -> Méthode D2*** en mode ***Multivarié*** sous PASSTEC 2000.

Voir aussi:

AutoD2(), CenterD2()

Exemple:

Voir partie I.

daystoyears(x, datemin = NULL, dateformat = "m/d/Y")

Description:

> *Convertit les données temporelles contenues dans* **x** *de l'échelle* **"days"** *(une unité = un jour) à l'échelle* **"years"** *(une unité = un an). Voir aussi le chapitre régularisation pour plus de détails sur la représentation du temps dans S+ et R.*

Arguments:

- **x:** un vecteur représentant le temps à l'échelle **"days"**.
- **datemin:** une chaîne de caractères représentant la date qui correspond à la valeur minimale de **x**, au format indiqué dans **dateformat**. Par exemple, si on garde la valeur par défaut de **dateformat, datemin = "04/23/1998"** signifie que l'on veut utiliser le 23 avril 1998 comme première date. Dans R, on peut aussi utiliser une date au format POSIXt. Par défaut, vaut **NULL** auquel cas la date minimale correspond à **min(x)**.
- **dateformat:** indiquez le format de la date. Par exemple: **"d/m/Y"** ou **"m/d/Y"** (par défaut). Notez la différence, dans R, entre **"y"** qui définit les années sur 2 chiffres (par exemple 87 pour 1987) et **"Y"** qui définit les années sur 4 chiffres (1987). **Veuillez à ne pas vous tromper à ce niveau!**

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

yearstodays(), buysbal()

Exemple:

L'instruction suivante change l'échelle temporelle de **"days"** à **"years"** pour la colonne ***Day*** du data frame ***releve*** en prenant comme date de départ le 5 octobre 2000:

```
> data(releve)       # Seulement dans R
> rel.time <- daystoyears(releve$Day, datemin = "10/05/2000")
```

decaverage(x, type = "additive", order = 1, times = 1, sides = 2, ends = "fill", weights = NULL)

Description:

> *Filtre une série spatio-temporelle régulière par la méthode des moyennes mobiles. Renvoie un objet 'tsd'. Cette fonction n'est normalement pas utilisée seule, mais est appelée par* ***tsd()****, avec l'argument* **method** = "**average**".

Arguments:

- **x:** un objet '*rts*' sous S+ ou '*ts*' sous R contenant une seule série régulière (***decaverage()*** n'est pas capable de traiter plusieurs séries simultanément).
- **type:** le type de modèle: ou bien additif ("**additive**", par défaut), ou bien multiplicatif ("**multiplicative**").
- **order:** l'ordre de la moyenne mobile simple à utiliser (la fenêtre étant de 2 x **order** + 1, centrée sur l'observation courante ou à gauche de celle-ci en fonction de l'argument **sides**, et tous les points sont de poids égaux). Par défaut: **1**. Si **weights** est précisé, il est prioritaire sur **order**. On peut aussi préciser **order** = "**periodic**". Dans ce cas, un filtre de désaisonnalisation est calculé sur **frequency** de la série (+ 1 terme si **frequency** est pair).
- **times:** le nombre de fois que le filtre est appliqué, par défaut, **1** fois.
- **sides:** si **2**, la fenêtre est centrée sur l'observation courante (par défaut). Si **1**, la fenêtre est à gauche de l'observation courante (et inclus celle-ci).
- **ends:** soit "**NAs**" (remplir les premières et dernières valeurs non calculables par NA, soit "**fill**" (les remplir par la moyenne des valeurs proches avant filtration, par défaut), soit "**circular**" (utiliser les valeurs de la fin pour le début et vice versa) soit "**periodic**" (utiliser les périodes entières de cycle immédiatement contigües, voir filtrage linéaire par les moyennes mobiles, désaisonnalisation).
- **weights:** un vecteur qui précise le poids des différentes observations dans la fenêtre de moyenne mobile. Cet argument est prioritaire sur **order**. Par défaut, **NULL**, **order** est utilisé.

Correspondance dans PASSTEC 2000:

Cette fonction correspond à ***Filtrage -> Elimination de tendance ou de cycle*** en mode ***Univarié***, option ***Moyennes mobiles*** dans la boite de dialogue et à ***Tendance -> Estimation statistique*** toujours en mode ***Univarié***, option ***Moyenne mobile*** dans la boite de dialogue. Dans PASSTEC 2000, on n'a pas le choix du mode de calcul pour les extrémités.

Voir aussi:

tsd(), extract.tsd(), tseries(), decdiff(), decmedian(), decevf(), decreg(), decloess(), deccensus()

Exemple:

Voir partie I

deccensus(x, type = "multiplicative", trend = FALSE)

Description:

> *Effectue la décomposition d'une série spatio-temporelle régulière selon la méthode CENSUS II, selon un modèle multiplicatif. Renvoie un objet 'tsd'. Cette fonction n'est normalement pas utilisée seule, mais est appelée par* ***tsd()****, avec l'argument* **method = "census"**. *Pour une décomposition saisonnière selon un modèle additif, utilisez plutôt* ***decloess()***. ***Cette fonction n'est pas disponible sous S+.***

Arguments:

- **x:** un objet *'rts'* sous R contenant une seule série régulière (***deccensus()*** n'est pas capable de traiter plusieurs séries simultanément). L'unité temporelle doit être l'année (**"years"**). Le nombre de cycles doit être complet (années entières) et au moins égal à 3.

- **type:** le type de modèle (pour des raisons de compatibilité avec les autres fonctions **decxxx()**. Seul le modèle multiplicatif (**"multiplicative"**) est accepté ici.

- **trend:** si **TRUE**, une tendance est également extraite, en plus d'un cycle saisonnier. On obtient alors les composantes suivantes: ***"trend"***, ***"seasonal"*** et ***"residuals"***. Si **FALSE** (par défaut), uniquement un cycle saisonnier est extrait (composantes ***"seasonal"*** et ***"deseasoned"***).

Correspondance dans PASSTEC 2000:

Cette fonction correspond à ***Filtrage -> Census II*** en mode ***Univarié***.

Voir aussi:

tsd(), extract.tsd(), tseries(), decdiff(), decaverage(), decmedian(), decevf(), decreg(), decloess()

Exemple:

Voir partie I.

decdiff(x, type = "additive", lag = 1, order = 1, ends = "fill")

Description:

> *Filtre une série spatio-temporelle régulière par la méthode des différences (élimination de la tendance générale, monotone croissante ou décroissante). Renvoie un objet 'tsd'. Cette fonction n'est normalement pas utilisée seule, mais est appelée par* ***tsd()****, avec l'argument* **method = "diff"**.

Arguments:

- **x:** un objet *'rts'* sous S+ ou *'ts'* sous R contenant une seule série régulière (***decdiff()*** n'est pas capable de traiter plusieurs séries simultanément).
- **type:** le type de modèle: ou bien additif (**"additive"**, par défaut), ou bien multiplicatif (**"multiplicative"**).
- **lag:** le décalage à prendre en compte à chaque calcul de différence (par exemple, pour une différence d'ordre 1, on a: $X_t - X_{t-lag}$. La valeur par défaut de **lag** est **1**.
- **order:** l'ordre de la différentiation (combien de fois le calcul est effectué successivement), par défaut: **1**.
- **ends:** soit **"NAs"** (remplir les premières valeurs non calculables par NA), soit **"fill"** (les remplir par la moyenne des valeurs suivantes avant différentiation, par défaut), soit **"drop"** (ne pas les remplir). Si **ends = "drop"**, la série filtrée sera plus courte au début de **lag** x **order**. Dans les autres cas, la longueur de la série est préservée.

Remarque:

La méthode des différences pour obtenir un cycle saisonnier est implémentée dans ***decloess()*** avec l'argument **order = "periodic"**. Voir description de cette fonction pour plus de détails.

Correspondance dans PASSTEC 2000:

Cette fonction correspond à ***Filtrage -> Elimination de tendance ou de cycle*** en mode ***Univarié***, option ***Différences*** dans la boite de dialogue. Dans PASSTEC 2000, **lag** vaut toujours **1**. Ici, il peut être précisé.

Voir aussi:

tsd(), extract.tsd(), tseries(), decaverage(), decmedian(), decevf(), decreg(), decloess(), deccensus()

Exemple:

voir partie I

decevf(x, type = "additive", lag = 5, axes = 1:2)

Description:

> *Filtre une série spatio-temporelle régulière par les vecteurs propres. Renvoie un objet 'tsd'. Cette fonction n'est normalement pas utilisée seule, mais est appelée par* ***tsd()****, avec l'argument* **method = "evf"**.

Arguments:

- **x:** un objet *'rts'* sous S+ ou *'ts'* sous R contenant une seule série régulière (***decevf()*** n'est pas capable de traiter plusieurs séries simultanément).
- **type:** le type de modèle: ou bien additif (**"additive"**, par défaut), ou bien multiplicatif (**"multiplicative"**).
- **lag:** le décalage à prendre en compte (une ACP sur les vecteurs allant d'un décalage = 0 à un décalage = **lag** est effectuée, et les axes repris dans **axes** sont extraits). La valeur par défaut de **lag** est **5**.
- **axes:** les axes à prendre en compte. Normalement ce sont les premiers axes. Par exemple, pour prendre en compte les 3 premiers axes, utiliser **axes = 1:3**. Il est possible de spécifier d'autres axes. Par exemple, pour prendre en compte les axes 2 et 4, on entrera: **axes = c(2,4)**. Par défaut, les 2 premiers axes sont pris en compte (**axes=1:2**).

Correspondance dans PASSTEC 2000:

Cette fonction correspond à ***Filtrage -> Filtrage par les vecteurs propres*** en mode ***Univarié***, dans PASSTEC 2000.

Voir aussi:

tsd(), extract.tsd(), tseries(), decaverage(), decdiff(), decmedian(), decreg(), decloess(), deccensus()

Exemple:

Voir partie I.

decloess(x, type = "additive", s.window, s.degree = 0, t.window = NULL, t.degree = 2, robust = FALSE, trend = FALSE)

Description:

> *Effectue la décomposition d'une série spatio-temporelle régulière selon la méthode LOESS (régression polynomiale locale). Renvoie un objet 'tsd'. Cette fonction n'est normalement pas utilisée seule, mais est appelée par* ***tsd()****, avec l'argument* **method = "loess"**.

Arguments:

- **x:** un objet '*rts*' sous S+ ou '*ts*' sous R contenant une seule série régulière (***decloess()*** n'est pas capable de traiter plusieurs séries simultanément).
- **type:** le type de modèle (pour des raisons de compatibilité avec les autres fonctions **decxxx()**. Seul le modèle additif (**"additive"**) est accepté ici.
- **s.window:** la largeur de la fenêtre pour extraire le cycle saisonnier (indiquez un nombre impair immédiatement égal ou supérieur au nombre de données annuelles pour extraire un cycle saisonnier. Utilisez d'autres largeurs de fenêtre pour d'autres cycles (circadien, lunaire, etc…). Ce paramètre doit être obligatoirement renseigné.
- **s.degree:** l'ordre du polynôme à utiliser pour l'extraction du cycle saisonnier (**0** ou **1**). Par défaut, **s.degree = 0**.
- **t.window:** la largeur de la fenêtre pour extraire la tendance (indiquez un nombre impair) lorsque **trend = TRUE**. Si ce paramètre n'est pas fournit, une valeur raisonnable est estimée, puis utilisée.
- **t.degree:** l'ordre du polynôme à utiliser pour l'extraction de la tendance (**0**, **1** ou **2**). Par défaut, **t.degree = 2**.
- **robust:** si **TRUE**, une méthode de régression robuste est utilisée (voir description de la méthode). Si **FALSE** (par défaut), la méthode classique des moindres carrés est utilisée pour estimer les différentes courbes polynomiales.
- **trend:** si **TRUE**, une tendance est également extraite, en plus d'un cycle saisonnier (uniquement sous R). On obtient alors les composantes suivantes: ***"trend"***, ***"seasonal"*** et ***"residuals"***. Si **FALSE** (par défaut), uniquement un cycle saisonnier est extrait (composantes ***"seasonal"*** et ***"deseasoned"***).

Correspondance dans PASSTEC 2000:

Cette fonction, avec les options **s.window = "periodic"** et **trend = FALSE** correspond à ***Filtrage -> Elimination de tendance ou de cycle*** en mode ***Univarié***, option ***Ecarts saisonniers*** dans la boite de dialogue. Dans ce cas, le graphique correspond à un graphe superposé de la série initiale et des résidus obtenu sous PASTECS à l'aide de la commande: `plot(series.dec, col=c(4,0,3), stack=FALSE, leg=FALSE)`.

Voir aussi:

tsd(), extract.tsd(), tseries(), decdiff(), decaverage(), decmedian(), decevf(), decreg(), deccensus()

Exemple:

Voir partie I.

decmedian(x, type="additive", order = 1, times = 1, ends = "fill")

Description:

> *Filtre une série spatio-temporelle régulière par la méthode des médianes mobiles. Renvoie un objet 'tsd'. Cette fonction n'est normalement pas utilisée seule, mais est appelée par* ***tsd()****, avec l'argument* **method = "median"**.

Arguments:

- **x:** un objet *'rts'* sous S+ ou *'ts'* sous R contenant une seule série régulière (***decmedian()*** n'est pas capable de traiter plusieurs séries simultanément).
- **type:** le type de modèle: ou bien additif (**"additive"**, par défaut), ou bien multiplicatif (**"multiplicative"**).
- **order:** l'ordre de la médiane mobile à utiliser (le nombre de termes utilisé est de 2 x **order** + 1, et la fenêtre est centrée sur l'observation courante). Par défaut: **1**.
- **times:** le nombre de fois que le filtre est appliqué, par défaut, **1** fois.
- **ends:** soit **"NAs"** (remplir les premières et dernières valeurs non calculables par NA), soit **"fill"** (les remplir par la médiane des valeurs proches avant filtration, par défaut).

Correspondance dans PASSTEC 2000:

Cette fonction correspond à ***Tendance -> Estimation statistique*** en mode ***Univarié***, option ***Médianes mobiles*** dans la boite de dialogue. Dans PASSTEC 2000, on n'a pas le choix du mode de calcul pour les extrémités.

Voir aussi:

tsd(), extract.tsd(), tseries(), decdiff(), decaverage(), decevf(), decreg(), decloess(), deccensus()

Exemple:

Voir partie I.

decreg(x, xreg, type = "additive")

Description:

> *Décompose une série spatio-temporelle régulière selon un modèle de regression. Renvoie un objet 'tsd'. Cette fonction n'est normalement pas utilisée seule, mais est appelée par* ***tsd()****, avec l'argument* **method = "reg"**.

Arguments:

- **x:** un objet *'rts'* sous S+ ou *'ts'* sous R contenant une seule série régulière (***decreg()*** n'est pas capable de traiter plusieurs séries simultanément).

- **xreg:** un vecteur contenant les valeurs renvoyées par un modèle de regression aux même temps que les observations de **x**. Ce modèle peut être linéraire, non-linéaire, autorégressif, etc… Toutes les fonctions de régression de S+ ou de R sont disponibles pour élaborer un modèle à partir de la série de départ. La fonction ***decreg()*** ne fait que transformer ce modèle en un objet *'tsd'* comme ceux renvoyés par les autres fonctions ***decXXX()*** ainsi que par ***tsd()***.

- **type:** le type de modèle: ou bien additif (**"additive"**, par défaut), ou bien multiplicatif (**"multiplicative"**).

Correspondance dans PASSTEC 2000:

Cette fonction correspond à ***Tendance -> Estimation statistique*** en mode ***Univarié***, option ***Régression Polynômiale*** ou ***Régression Sinusoïdale*** dans la boite de dialogue. Dans PASSTEC 2000, on est limité à ces deux possibilités.

Voir aussi:

tsd(), extract.tsd(), tseries(), decdiff(), decaverage(), decevf(), decmedian(), decloess(), deccensus()

Exemple:

Voir partie I

disjoin(x)

Description:

> *Effectue le codage disjonctif complet d'un vecteur de type facteur (classe 'factor' sous R), obtenu par exemple à partir de la function* ***cut()****.*

Arguments:

- **x:** un vecteur de type *'factor'*.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

buysbal()

Exemple:

Voir partie I.

disto(x, max.dist = nrow(x)/4, plotit = TRUE, disto.data = NULL)

Description:

> *Calcule et trace le distogramme pour une matrice de type descripteurs x stations ou pour des séries spatio-temporelles régulières.*

Arguments:

- **x:** une matrice, un data frame ou une série spatio-temporelle multiple.
- **max.dist:** la distance maximale à calculer. Par défaut, il s'agit du quart du nombre d'observations (c'est-à-dire, du nombre de lignes dans la matrice).
- **plotit:** si **TRUE**, le graphe du distogramme est tracé.
- **disto.data:** des données issues d'un appel précédent de la fonction, fournies afin d'en tracer le graphe. Par défaut, **NULL**, les données du distogramme sont (re)calculées.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, cette fonction est accessible par le menu ***Description*** -> ***Distogramme*** en mode ***Multivarié***.

Voir aussi:

vario()

Exemple:

Voir partie I.

escouf(x, level = 1, verbose = TRUE)

Description:

> *Calcule les variables équivalentes selon la méthode d'Escoufier sur un tableau multivarié (data frame)* **x**. *Il est possible de limiter le calcul à une corrélation inférieure à 1 (***level***), et donc, de ne pas incorporer les variables qui ont une influence très faible dans le calcul. Ceci est d'autant plus utile que la méthode est lente. La valeur par défaut de* **level** *(**1**) demande toutefois le calcul de l'ensemble des variables. Il est à noter que* **level** *peut être reprécisé dans les méthodes* ***plot()***, ***lines()*** *et* ***extract()*** *pour tracer le seuil ou n'extraire les variables qu'à concurrence d'un seuil, et ce, indépendamment de la valeur fournie ici. Enfin,* ***identify()*** *permet de déterminer le seuil de façon interactive en cliquant sur le graphique.*

Arguments:

- **x:** un data frame contenant les différentes variables à classer selon la méthode d'Escoufier.

- **level:** le niveau de 'corrélation' désiré avant d'arrêter le calcul (compris entre 0 et 1. La valeur par défaut **1** effectue le calcul jusqu'à ce que la corrélation maximale soit atteinte, c'est-à-dire lorsque toutes les variables ont été incorporées. Si une valeur est fournie, il est conseillé de rester dans les limites **0.95 < level < 1**, ce qui aura pour effet d'arrêter le calcul sans tenir compte de toutes les variables de moindre importance. L'extraction à un seuil plus bas sera toujours possible en reprécisant **level** à une valeur plus faible dans la méthode ***extract()***.

- **verbose:** si **TRUE** (par défaut), indique les variables incorporées à chaque étape pendant le calcul.

Correspondance dans PASSTEC 2000:

Menu ***Description -> Méthode d'Escoufier*** en mode ***Multivarié***. Dans PASSTEC 2000, on peut cocher l'option 'Passage en logarithme'. Dans PASTECS, ceci doit être réalisé préalablement, par exemple par la commande `datalog <- log(data+1)`. Ne pas oublier d'ajouter 1 pour éviter les erreurs sur les valeurs nulles; S+ retournerait alors –Inf.

Les autres options ('Sorties statistiques', 'Affichage du graphe' et 'Sauvegarde des résultats' correspondent respectivement aux fonctions ***summary()***, ***plot()*** et ***escouf.obj$vr*** qui peuvent être appelée ultérieurement si on a pris soin de sauvegarder les résultats de l'analyse dans une nouvelle variable (comme ***marbio.esc*** dans l'exemple ci-dessous). L'organisation objet de la méthode Escoufier, et son calcul par étape sous PASTECS permet plus de flexibilité, puisque le seuil (**level**) peut-être redéfini en cours d'analyse, ce qui n'est pas possible sous PASSTEC 2000. A noter également que PASSTEC 2000 ne possède pas de fonction équivalente à ***extract()***. L'extraction du tableau final à un seuil donné doit donc y être réalisée manuellement contrairement à PASTECS!

Voir aussi:

print.escouf(), summary.escouf(), plot.escouf(), lines.escouf(), identify.escouf(), extract.escouf()

Exemple:

Voir partie I.

extract.abund(e, n = e$n, left = TRUE)

Description:

> *Méthode* ***extract()*** *des objets de classe 'abund'. Extrait un tableau contenant les variables par ordre d'importance selon le tri par abondance, et ce, à concurrence de* ***n*** *variables, ou bien rejette ces variables et retient les autres si* **left = FALSE**. ***Attention! Etant donné que la méthode extract() relit la matrice de départ, il faut absolument éviter de la modifier entre les appels de abund() et extract(), sinon l'extraction risque de ne pas se faire correctement ou alors de ne pas correspondre à l'analyse réalisée!***

Arguments:

- **e:** un objet de classe *'abund'* issu d'une analyse lancée par la fonction ***abund()***.
- **n:** le nombre de variables à extraire (ou à délaisser si **left = FALSE**). Par défaut, lit la valeur contenue dans **e$n**.
- **left:** si **TRUE** (par défaut), récupère les variables à gauche de la séparation **n**. Si **FALSE**, récupère les variables à droite de cette séparation **n**.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

abund(), print.abund(), summary.abund(), plot.abund(), lines.abund(), identify.abund()

Exemple:

Voir partie I.

extract.escouf(e, n, level = e$level)

Description:

> *Méthode* ***extract()*** *des objets de classe 'escouf'. Extrait un tableau contenant les variables par ordre d'importance selon l'analyse d'Escoufier, et ce, à concurrence de* **n** *variables ou jusqu'au seuil* **level**. ***Attention! Etant donné que la méthode extract() relit la matrice de départ, il faut absolument éviter de la modifier entre les appels de escouf() et extract(), sinon l'extraction risque de ne pas se faire correctement ou alors de ne pas correspondre à l'analyse réalisée!***

Arguments:

- **e:** un objet de classe *'escouf'* issu d'une analyse lancée par la fonction ***escouf()***.
- **n:** le nombre de variables à extraire. Si **n** est précisé, il est utilisé en priorité à **level**.
- **level:** la valeur du seuil (entre 0 et 1). Seules les variables dont le RV est inférieur à ce seuil seront extraites, les variables de moindre importance étant ignorées. Par défaut (donc, si ni **n**, ni **level** ne sont précisés), **e$level** est utilisé. S'il n'existe pas, la fonction s'arrête avec une erreur indiquant qu'il faut préciser level.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

escouf(), print.escouf(), summary.escouf(), plot.escouf(), lines.escouf(), identify.escouf()

Exemple:

Voir partie I.

extract.regul(e, n = ncol(e$y), series = NULL)

Description:

> *Extrait une ou plusieurs 'time series' régulière(s) d'un objet 'regul' et les placent dans un objet 'rts' sous S+, ou 'ts' sous R. Toutes les méthodes des objets 'rts' ou 'ts' sont alors disponibles pour traiter ces données.*

Arguments:

- **e:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.
- **n:** le nombre de séries à extraire (de la première à la n^{ème}, dans l'ordre). Par défaut, **n** est égal au nombre de séries présentes dans l'objet *'regul'*. Dans ce cas, ***extract()*** a exactement le même effet que ***tseries()***.
- **series:** la liste des noms ou des index des séries à extraire. Si **series** est précisé, **n** est ignoré. Par défaut, **series = NULL**. Dans ce cas, c'est **n** qui est pris en compte.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), tseries(), is.tseries()

Exemple:

Voir partie I.

extract.tsd(e, n = length(e$ts), series = NULL, components = NULL)

Description:

> *Extrait une ou plusieurs composantes d'une ou plusieurs séries contenues dans un objet 'tsd' et les reconverti en un objet 'rts' sous S+, ou 'ts' sous R. Toutes les méthodes des objets 'rts' ou 'ts' sont alors disponibles pour traiter ces données. Pour extraire toutes les composantes de toutes les séries, on peut aussi utiliser* ***tseries()****.*

Arguments:

- **e:** un objet *'tsd'* obtenu à partir de la fonction ***tsd()***.
- **n:** le nombre de séries à extraire (de la première à la $n^{ème}$, dans l'ordre). Par défaut, **n** est égal au nombre de séries présentes dans l'objet *'tsd'*. Si les deux arguments **series** et **components** sont nuls, alors ***extract()*** a exactement le même effet que ***tseries()***.
- **series:** la liste des noms ou des index des séries à extraire. Si **series** est précisé, **n** est ignoré. Par défaut, **series = NULL**. Dans ce cas, c'est **n** qui est pris en compte. On peut aussi préciser des valeurs négatives pour extraire toutes les séries *sauf* celles indiquées.
- **components:** la liste des noms ou des index des composants à extraire. Si **components = NULL** (par défaut), alors toutes les composantes des séries sélectionnées sont extraites. On peut aussi préciser des valeurs négatives pour extraire toutes les composantes *sauf* celles indiquées.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

tsd(), print.tsd(), summary.tsd(), specs.tsd(), plot.tsd()

Exemple:

Voir partie I.

extract.turnogram(e, n, level = e$level, FUN = e$fun, drop = 0)

Description:

> *Extrait une 'time series' régulière d'un objet 'turnogram', recalculée tous les* **level** *intervalles et à l'aide de la function d'aggrégation* **FUN**. *Lorsque cet intervalle correspond au maximum de l'indice de monotonie I sur le graphe du tournogramme, la série extraite maximise l'information (c'est-à-dire, contient un maximum de variations non aléatoires et un minimum de fluctuations aléatoires).*

Arguments:

- **e:** un objet *'turnogram'* obtenu à partir de la fonction ***turnogram()***.
- **n:** le nombre d'observations à prendre en compte dans la série initiale. Préciser **n = NULL** (par défaut) pour utiliser toutes les observations de la série initiale.
- **level:** le *niveau d'extraction*, c'est-à-dire, l'intervalle à retenir pour recalculer la série. Par défaut, il s'agit de la valeur enregistrée dans l'objet *'turnogram'*, que ce soit la valeur calculée automatiquement, ou la valeur modifiée par ***identify.turnogram()***.
- **FUN:** la function d'aggrégation à utiliser sur l'intervalle (***first()***, ***last()***, ***mean()***, ***median()***, ***sum()***, …). Par défaut, il s'agit de la même fonction qui a servi à calculer le tournogramme. Changer cette valeur n'a que peu de sens, mais l'option permet d'effectuer des extractions indépendamment des résultats du tournogramme, par exemple, pour comparer l'effet de la fonction d'aggrégation sur l'allure de la série recalculée.
- **drop:** le nombre d'observations initiales à éliminer (point de départ décalé), avant d'extraire la série recalculée. Par défaut, **drop = 0**, l'extraction se fait à partir de la première observation.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

turnogram(), print.turnogram(), summary.turnogram(), plot.turnogram(), identify.turnogram(), pgleissberg(), first(), last()

Exemple:

Voir partie I.

extract.turnpoints(e, no.tp = 0, peak = 1, pit = -1)

Description:

> *Extrait un vecteur représentant la position des pics et des creux dans la série originelle à partir d'un objet 'turnpoints'.*

Arguments:

- **e:** un objet *'turnpoints'* obtenu à partir de la fonction ***turnpoints()***.
- **n:** le nombre de points à extraire. Par défaut, **n = length(e)**, tous les points (correspondant à des extrema) sont extraits.
- **no.tp:** le code à utiliser pour représenter une observation qui n'est pas un point de retournement (ni pic, ni creux). Par défaut, **0**.
- **peak:** le code à utiliser pour représenter un pic. Par défaut, **1**.
- **pit:** le code à utiliser pour représenter un creux. Par défaut, **-1**.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

turnpoints(), print.turnpoints(), summary.turnpoints(), plot.turnpoints(), lines.turnpoints()

Exemple:

Voir partie I.

first(x, na.rm = FALSE)

Description:

> *Extrait le premier élément d'un vecteur. Cette fonction est principalement utile dans le cadre du tournogramme pour n'extraire que le premier élément sur chaque intervalle.*

Arguments:

- **x:** un vecteur.
- **na.rm:** si **TRUE**, les valeurs de type **NA** sont d'abord éliminées, et le premier élément non **NA** est extrait.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

last(), turnogram()

Exemple:

```
> a <- c(NA, 1, 2, NA, 3, 4, NA)
> first(a, na.rm=TRUE)
[1] 1
```

GetUnitText(series)

Description:

> *Extrait les unités associées avec une ou plusieurs séries spatio-temporelles et les formattent pour un affichage optimal en légende d'un axe de graphe. Cette fonction est normalement utilisée en interne par d'autres, comme* ***AutoD2()***, ***CrossD2()***, ***turnogram()***,…

Arguments:

- **series:** une ou plusieurs séries spatio-temporelles régulières de type *'rts'* sous S+ ou *'ts'* sous R.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

AutoD2(), CrossD2(), turnogram()

Exemple:

```
> timeser <- ts(1:24, frequency=12)   # 12 observations per year
> if (exists("is.R") && is.function(is.R) && is.R()) {
+     attr(timeser, "units") <- "years"
+ } else {
+     attr(attr(timeser, "tspar"), "units") <- "years"
+ }
> GetUnitText(timeser)

[1] "months"
```

hist.regul(x, nclass=30, col=c(4, 5, 2), xlab, ylab, main, plotit=TRUE, …)

Description:

> *Trace un histogramme des variables qui coïncident entre la série initiale et la série finale régularisée, en fonction de la distance dans le temps qui sépare ces paires respectives. La première barre représente le nombre de données qui coïncident exactement. Les barres suivantes représentent le nombre de données initiales situées dans la fenêtre de tolérances, mais de plus en plus éloignées des points régularisés. La dernière barre représente le nombre de points qui ont dû être interpolés parce qu'aucune donnée initiale ne se trouvait dans la fenêtre de tolérance définie (trous dans la série). Renvoie aussi un résultat chiffré.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.
- **nclass:** paramètre de l'histogramme. Valeur initiale du nombre de classes à réaliser. Ce nombre est recalculé au mieux par le programme, mais il arrive qu'il ne puisse obtenir une séparation en classes adéquate avec le nombre fournit. Essayer alors une valeur supérieure. Par défaut, **nclass = 30**.
- **col:** les couleurs à utiliser pour remplir respectivement la première barre (coïncidence exacte), les suivantes (coïncidence dans la fenêtre de tolérance), et la dernière barre (valeurs interpolées). Par défaut, **col = c(4, 5, 2)**.
- **xlab:** intitulé de l'axe x. Par défaut: **"Time distance in <unit>"**, où **<unit>** est l'unité qui a été définie dans l'objet *'regul'*.
- **ylab:** intitulé de l'axe y. Par défaut: **"Frequency, tol = <tol>"** où **<tol>** est la valeur de tolérance choisie lors du calcul par ***regul()***.
- **main:** titre principal du graphique. Par défaut: **"Number of matching observations"**.
- **plotit:** indique si le graphe doit être tracé ou non, en plus du renvoi des valeurs chiffrées. Par défaut, oui (**TRUE**).
- **…:** paramètres optionnels complémentaires pour les graphiques.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), print.regul(), summary.regul(), specs.regul(), plot.regul(), lines.regul(), identify.regul(), extract.regul(), regul.screen(), regul.adj()

Exemple:

Résultat similaire à ***regul.adj()***, mais utilisable en diagnostic post-régularisation sur un objet *'regul'*, voir partie I pour un exemple de sortie de ***regul.adj()***.

identify.abund(x, label.pts = FALSE, lvert = TRUE, lvars = TRUE, col = 2, lty = 2, …)

Description:

> *Méthode **identify()** des objets de classe 'abund'. Permet d'identifier un point sur l'une des trois courbes du graphe d'abondance de manière interactive (à la souris). Ce point sépare les variables retenues de celles qui ne le sont pas lors de l'extraction. Le tracé du repère vertical, ainsi que la coloration différentielle des indications de variables se font comme pour la méthode **lines()**. Cette méthode retourne le nombre de variables extraites. Cette valeur peut être enregistrée de manière persistante dans la variable objet 'abund' en effectuant une assignation **x$n <- identify(x)** (voir exemple dans la partie I). Si **label.pts = TRUE**, permet de rajouter le label d'un ou plusieurs points sur le graphe au lieu d'identifier la séparation. La méthode **plot()** doit impérativement être appelée avant **identify()**, afin que le graphique soit déjà tracé.*

Arguments:

- **x:** un objet de classe *'abund'* issu d'une analyse lancée par la fonction ***abund()***.
- **label.pts:** si **FALSE** (par défaut), identifie la séparation entre les variables à retenir et celles qui sont à écarter. Si **label.pts = TRUE**, permet d'apposer le label de certains points sur le graphique de manière interactive à la souris.
- **lvert:** si **TRUE** (par défaut), un repère vertical sépare les variables reprises et celles qui sont éliminées.
- **lvars:** si **TRUE** (par défaut), les variables reprises sont repérées dans la même couleur que les traits de repère.
- **col:** couleur à utiliser pour les repères. Par défaut, la couleur **2** est utilisée.
- **lty:** style de trait pour les repères. Par défaut, le style pointillé (**2**) est utilisé.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

abund(), print.abund(), summary.abund(), plot.abund(), lines.abund(), extract.abund()

Exemple:

Voir partie I

identify.escouf(x, lhorz = TRUE, lvert = TRUE, lvars = TRUE, col = 2, lty = 2, …)

Description:

> *Méthode **identify()** des objets de classe 'escouf'. Permet d'identifier un point sur la courbe RV ou RV' de manière interactive (à la souris), et calcule le seuil correspondant à une cassure à ce niveau. Le tracé des repères horizontaux et/ou verticaux, ainsi que la coloration différentielle des indications de variables se font comme pour la méthode **lines()**. Cette méthode retourne la valeur du seuil calculé. Celle-ci peut être enregistrée de manière persistante dans la variable objet 'escouf' en effectuant une assignation **x$level <- identify(x)** (voir exemple dans la partie I). La méthode **plot()** doit impérativement être appelée avant **identify()** , afin que le graphique soit déjà tracé.*

Arguments:

- **x:** un objet de classe '*escouf*' issu d'une analyse lancée par la fonction ***escouf()***.
- **lhorz:** si **TRUE** (par défaut), un repère horizontal est tracé au niveau du seuil.
- **lvert:** si **TRUE** (par défaut), un repère vertical sépare les variables reprises et celles qui sont éliminées au seuil donné.
- **lvars:** si **TRUE** (par défaut), les variables reprises au seuil donné sont repérées dans la même couleur que les traits de repère.
- **col:** couleur à utiliser pour les repères. Par défaut, la couleur **2** est utilisée.
- **lty:** style de trait pour les repères. Par défaut, le style pointillé (**2**) est utilisé.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

escouf(), print.escouf(), summary.escouf(), plot.escouf(), lines.escouf(), extract.escouf()

Exemple:

Voir partie I.

identify.local.trend(x, …)

Description:

> *Identifie des points sur un graphique de la tendance locale obtenu à l'aide de **local.trend()** et calcule les moyennes locales entre ces points. Le graphe doit déjà être tracé avant l'appel de cette fonction.*

Arguments:

- **x:** un objet *'local.trend'* obtenu à partir de la fonction ***local.trend()***.
- **…:** des paramètres optionnels à passer à la fonction ***identify()*** (voir ***?identify.default***).

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

local.trend()

Exemple:

Voir partie I.

identify.regul(x, series = 1, col = 3, label = "#")

Description:

> *Identifie des points sur un graphique obtenu à partir de **plot.regul()** qui doit déjà être tracé avant l'appel de cette fonction.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.
- **series:** le numéro de la série à identifier. Par défaut, **1**, soit la seule série s'il n'y en a qu'une ou la première série d'un groupe s'il y en a plusieurs.
- **col:** la couleur à utiliser pour tracer la nouvelle série régularisée. Par défaut, **col = 3**.
- **label:** le symbole à utiliser pour marquer les points identifiés (par défaut: **#**).

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), print.regul(), summary.regul(), specs.regul(), plot.regul(), hist.regul(), extract.regul(), regul.screen(), regul.adj()

Exemple:

Voir partie I.

identify.turnogram(x, lvert = TRUE, col = 2, lty = 2, ...)

Description:

> *Identifie de manière interactive sur le graphe du tournogramme un seuil d'extraction de la série pour un autre intervalle que celui calculé automatiquement par* ***turnogram()****. Le graphe du tournogramme, obtenu par la méthode* ***plot()*** *appliquée à l'objet* **x** *(qui doit être de type 'turnogram') doit déjà être tracé avant l'appel de cette fonction.*

Arguments:

- **x:** un objet *'turnogram'* obtenu à partir de la fonction ***turnogram()***.
- **lvert:** si **TRUE** (par défaut), une ligne verticale est tracée sur le graphe, indiquant le nouvel intervalle auquel l'extraction de la série se fera.
- **col:** la couleur à utiliser pour tracer la ligne verticale. Par défaut, **col = 2**.
- **lty:** le style à utiliser pour tracer la ligne verticale. Par défaut, **lty = 2**, correspondant à une ligne pointillée.
- **…**: paramètres additionnels pour le traçage de la ligne verticale.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

turnogram(), print.turnogram(), summary.turnogram(), plot.turnogram(), extract.turnogram(), pgleissberg()

Exemple:

Voir partie I.

is.tseries(x)

Description:

> *Détermine si une variable est un objet 'time series' (voir chapitre manipulations des séries régulières des objets 'rts' et 'ts').*

Arguments:

- **x:** une variable quelconque à tester.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

tseries()

Exemples:

Voir chapitres régularisation et manipulation de base des séries régulières *'rts'* et *'ts'* dans la partie I.

last(x, na.rm = FALSE)

Description:

> *Extrait le dernier élément d'un vecteur. Cette fonction est principalement utile dans le cadre du tournogramme pour n'extraire que le dernier élément sur chaque intervalle.*

Arguments:

- **x:** un vecteur.
- **na.rm:** si **TRUE**, les valeurs de type **NA** sont d'abord éliminées, et le dernier élément non **NA** est extrait.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

first(), turnogram()

Exemple:

```
> a <- c(NA, 1, 2, NA, 3, 4, NA)
> last(a, na.rm=TRUE)
[1] 4
```

lines.abund(x, n = x$n, lvert = TRUE, lvars = TRUE, col = 2, lty = 2, …)

Description:

> *Méthode **lines()** des objets de classe 'abund'. Trace un repère vertical et colore les indications des variables afin de différentier celles qui sont reprises de celles qui sont éliminées lors de l'extraction. Cette méthode doit donc obligatoirement être appelée après **plot()**.*

Arguments:

- **x:** un objet de classe *'abund'* issu d'une analyse lancée par la fonction ***abund()***.
- **n:** le nombre de variables reprises.
- **lvert:** si **TRUE** (par défaut), un repère vertical sépare les variables reprises et celles qui sont éliminées.
- **lvars:** si **TRUE** (par défaut), les variables reprises sont repérées dans la même couleur que les traits de repère.
- **col:** couleur à utiliser pour les repères. Par défaut, la couleur **2** est utilisée.
- **lty:** style de trait pour les repères. Par défaut, le style pointillé (**2**) est utilisé.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

abund(), print.abund(), summary.abund(), plot.abund(), identify.abund(), extract.abund()

Exemple:

Voir partie I

lines.escouf(x, level = x$level, lhorz = TRUE, lvert = TRUE, lvars = TRUE, col = 2, lty = 2, …)

Description:

> *Méthode **lines()** des objets de classe 'escouf'. Trace les repères (horizontaux et/ou verticaux) et colore les indications des variables afin de différencier celles qui sont reprises de celles qui sont éliminées à un seuil donné sur le graphique de la méthode d'Escoufier. Cette méthode doit donc obligatoirement être appelée après **plot()**.*

Arguments:

- **x:** un objet de classe '*escouf*' issu d'une analyse lancée par la fonction ***escouf()***.
- **level:** la valeur du seuil (entre 0 et 1) où les repères seront tracés. Par défaut, la valeur de seuil de **x$level** est utilisée.
- **lhorz:** si **TRUE** (par défaut), un repère horizontal est tracé au niveau du seuil.
- **lvert:** si **TRUE** (par défaut), un repère vertical sépare les variables reprises et celles qui sont éliminées au seuil donné.
- **lvars:** si **TRUE** (par défaut), les variables reprises au seuil donné sont repérées dans la même couleur que les traits de repère.
- **col:** couleur à utiliser pour les repères. Par défaut, la couleur **2** est utilisée.
- **lty:** style de trait pour les repères. Par défaut, le style pointillé (**2**) est utilisé.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

escouf(), print.escouf(), summary.escouf(), plot.escouf(), identify.escouf(), extract.escouf()

Exemple:

Voir partie I

lines.regul(x, series = 1, col = 3, lty = 1, plot.pts = TRUE, …)

Description:

> *Complète un graphique de représentation d'une régularisation obtenue à partir de* **plot.regul()** *(objet 'regul'). Permet de superposer une nouvelle série régularisée sur le graphe à des fins de comparaison des méthodes de régularisation respectives.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.
- **series:** le numéro de la série à tracer. Par défaut, **1**, soit la seule série s'il n'y en a qu'une ou la première série d'un groupe s'il y en a plusieurs.
- **col:** la couleur à utiliser pour tracer la nouvelle série régularisée. Par défaut, **col = 3**.
- **lty:** le style de trait (plein, pointillé, …) à utiliser pour la nouvelle série. Par défaut le trait plein est sélectionné (valeur **1**).
- **plot.pts:** si **TRUE** (par défaut), trace les points de la série régularisé (+). Ceux qui coïncident avec la série de départ sont entourés d'un cercle.
- **…:** paramètres optionnels complémentaires pour les graphiques.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), print.regul(), summary.regul(), specs.regul(), plot.regul(), identify.regul(), hist.regul(), extract.regul()

Exemple:

Voir partie I

lines.stat.slide(x, stat = "mean", col = 3, lty = 1, ...)

Description:

> *Méthode **lines()** des objets de classe 'stat.slide'. Trace une statistique supplémentaire sur un graphique obtenu à l'aide de **plot()** appliqué à un objet 'stat.slide'. Cette méthode doit donc obligatoirement être appelée après **plot()**.*

Arguments:

- **x:** un objet de classe *'stat.slide'* issu d'un traitement par la fonction ***stat.slide()***.
- **stat:** la statistique à représenter. Vous avez le choix entre **"min"**, **"max"**, **"median"**, **"mean"** (par défaut), **"pos.median"**, **"pos.mean"**, **"geo.mean"** et **"pen.mean"**. Les autres statistiques ne sont pas représentables sur le graphe.
- **col:** les couleurs à utiliser pour la nouvelle statistique. Par défaut: **col = 3**.
- **lty:** le style de trait à utiliser pour le tracé de la nouvelle statistique. Par défaut, le trait plein (**lty = 1**).
- **…:** paramètres optionnels supplémentaires du graphe.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

stat.slide(), print.stat.slide(), plot.stat.slide()

Exemple:

Voir partie I.

lines.turnpoints(x, max = TRUE, min = TRUE, median = TRUE, col = c(4, 4, 2), lty = c(2, 2, 1), ...)

Description:

> *Trace les enveloppes maximum et minimum (reliant les pics et les creux, respectivement), ainsi que les points médians entre ces deux enveloppes sur le graphe de la série originelle. Il faut donc tracer le graphe de la série de départ (à l'aide de **plot()**) avant d'utiliser cette fonction.*

Arguments:

- **x:** un objet *'turnpoints'* obtenu à partir de la fonction ***turnpoints()***.
- **max:** si **TRUE** (par défaut), l'enveloppe maximum est tracée.
- **min:** si **TRUE** (par défaut), l'enveloppe minimum est tracée.
- **median:** si **TRUE** (par défaut), la ligne reliant les points médians est tracée.
- **col:** la couleur à utiliser pour respectivement l'enveloppe maximum, l'enveloppe minimum et les points médians. Par défaut, les couleurs **4**, **4** et **2** sont utilisées respectivement.
- **lty:** le style de trait à utiliser pour les enveloppes et les points médians. Par défaut, les styles pointillés (**2**) sont utilisés pour les enveloppes et le style plein (**1**) est utilisé pour la courbe reliant les points médians.
- **…:** paramètres optionnels complémentaires pour les graphiques.

Correspondance dans PASSTEC 2000:

Cette fonction trace les enveloppes et les points médians sur le graphe de la série originelle comme dans PASSTEC 2000.

Voir aussi:

turnpoints(), print.turnpoints(), summary.turnpoints(), plot.turnpoints(), extract.turnpoints()

Exemple:

Voir partie I.

local.trend(x, k = mean(x), plotit = TRUE, type="l", cols = 1:2, ltys = 2:1, xlab="Time", ylab = "cusum", ...)

Description:

> *Détermine s'il y a des tendances locales dans la série (valeurs moyennes changeantes sur différents intervalles de temps) par la méthode des sommes cumulées. Rem: sous R, voir aussi le package* ***strucchange****.*

Arguments:

- **x:** une série régulière de type *'rts'* sous S+ ou *'ts'* sous R.
- **k:** la valeur moyenne de référence à soustraire des sommes cumulées. Par défaut, il s'agit de la moyenne de toutes les observations dans la série.
- **plotit:** si **TRUE** (par défaut), le graphique superposé de la courbe cusum et de la série originelle sont tracés. Si **FALSE**, aucun graphe n'est dessiné.
- **type:** le type de trait à utiliser pour représenter la série d'origine. Par défaut, **type = "l"**, un trait continu, sans indication des points est utilisé.
- **cols:** les couleurs à utiliser pour respectivement la série originelle et la courbe cusum. Par défaut, il s'agit des couleurs **1** et **2**.
- **ltys:** les styles de trait à utiliser pour la série originelle et la courbe cusum. Par défaut, les styles **2** et **1**, respectivement (trait pointillé et trait plein).
- **xlab:** l'intitulé de l'axe x. Par défaut: **Time**.
- **ylab:** l'intitulé de l'axe y. Par défaut: **cusum**.
- **…**: des paramètres optionels à passer à la fonction de traçage du graphe.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, il s'agit de: ***Tendance -> Tendance locale*** en mode ***Univarié***.

Voir aussi:

identify.local.trend(), trend.test(), stat.slide()

Exemple:

Voir partie I.

match.tol(x, table, nomatch = NA, tol.type = "both", tol = 0)

Description:

> *Détermine les points de **x** qui se trouvent dans **table**, à une marge de tolérance près. Cette fonction est utilisée en interne par les routines **regul()**, **regul.screen()** et **regul.adj()**.*

Arguments:

- **x:** un vecteur contenant les valeurs de temps d'une série régulière.
- **table:** un vecteur contenant les valeurs à rechercher. Dans le cas présent, il s'agit des valeurs de temps d'une série irrégulière à comparer.
- **nomatch:** le symbole à utiliser pour indiquer qu'aucune correspondance n'a été trouvée. Par défaut, **nomatch = NA**.
- **tol.type:** le type d'ajustement pour la tolérance, soit **"left"**, **"both"** (par défaut), **"right"** ou **"none"**. Si **tol.type = "left"**, les valeurs de **x** qui correspondent aux valeurs régularisées sont cherchées dans une fenêtre $]\mathbf{x_{régul} - tol, x_{régul}}]$. Si **tol.type = "both"**, ces valeurs sont recherchées dans la fenêtre $]\mathbf{x_{régul} - tol, x_{régul} + tol}[$. Si **tol.type = "right"**, les fenêtres $[\mathbf{x_{régul}, x_{régul} + tol}[$ sont scrutées. Au cas où plusieurs valeurs seraient présentes dans la fenêtre, celle qui est la plus proche dans le temps de la valeur régularisée $\mathbf{x_{régul}}$ est conservée. Enfin, si **tol.type = "none"**, seules les valeurs qui coïncident exactement sont recherchées.
- **tol:** la taille de la fenêtre de tolérance. Doit être une fraction entière de l'intervalle entre les valeurs de **x**, et ne peut lui être supérieur.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), regul.adj(), hist.regul()

Exemple:

Aucun (utilisé en interne par d'autres fonctions de PASTECS).

pennington(x, calc = "all", na.rm = FALSE)

Description:

> *Calcule les estimateurs de Pennington (moyenne, variance, variance de la moyenne pour des données d'abondance issues d'un échantillonnage en aveugle, et contenant beaucoup de valeurs nulles.*

Arguments:

- **x:** un vecteur numérique contenant les abondances échantillonnées aux différentes stations

- **calc:** indique quel(s) estimateur(s) doivent être calculé(s). Les différentes possibilités sont: **"mean"**, calcul de la moyenne uniquement; **"var"**, calcul de la variance; **"mean.var"**, calcul de la variance de la moyenne; **"all"** (valeur par défaut), calcul des trois précédents simultanément et renvoi de leurs valeurs dans un vecteur **"mean", "var", "mean.var"**.

- **na.rm: TRUE** s'il faut ne pas tenir compte des valeurs manquantes ou **FALSE** (par défaut) s'il faut en tenir compte. Dans ce dernier cas, la présence d'au moins une valeur manquante (**NA**) dans le vecteur x empêche le calcul et la fonction renvoie également **NA**.

Correspondance dans PASSTEC 2000:

Aucune. Cette fonction est appelée par ***stat.pen()*** en interne. Elle est également disponible directement pour pouvoir effectuer les calculs basiques des estimateurs de Pennington dans un programme utilisateur ou à partir de la ligne de commande.

Voir aussi:

stat.pen(), stat.slide()

Exemple:

Voir partie I.

pgleissberg(n, k, lower.tail = TRUE, two.tailed = FALSE)

Description:

> *Retourne la probabilité associée à une distribution de Gleissberg (exacte si* **n** *< 50, ou approximée par une distribution normale si* **n** *≥ 50. La distribution de Gleissberg indique la probabilité d'une série qu'elle soit aléatoire, en fonction du nombre d'observations (***n***) et du nombre d'extremums (***k***). Cette distribution est utilisée pour calculer le tournogramme.*

Arguments:

- **n:** le nombre d'observations dans la série.
- **k:** le nombre d'extremums dans la série, comme calculé par ***turnpoints()***.
- **lower.tail:** si **TRUE** (par défaut) et **two.tailed = FALSE**, la probabilité à gauche est renvoyée, sinon, la probabilité à droite est renvoyée.
- **two.tailed:** si **TRUE**, la probabilité associée à un test bilatéral est renvoyée, sinon, celle correspondant à un test unilatéral (par défaut).

Correspondance dans PASSTEC 2000:

Aucune. Cette fonction est appelée par ***turnogram()*** en interne. Elle est également disponible directement pour pouvoir tester si une série donnée est ou non aléatoire selon la théorie de l'information.

Voir aussi:

turnogram()

Exemple:

```
> pgleissberg(20, 8, lower.tail=TRUE, two.tailed=FALSE)

[1] 0.02548283

> pgleissberg(20, 10, lower.tail=TRUE, two.tailed=FALSE)

[1] 0.2011741
```

Dans le premier cas (**n = 20** et **k = 8**), la série est significativement plus monotone qu'une série aléatoire (5% < *p* < 1%). Dans le second cas (**n = 20**, **k = 10**) elle est probablement purement aléatoire (*p* >> 5%), à moins que l'intervalle d'observation ne soit trop large que pour pouvoir discerner entre pics et creux erratiques et fluctuations non aléatoires.

plot.abund(x, n = x$n, lvert = TRUE, lvars = TRUE, lcol = 2, llty = 2, all = TRUE, dlab = c("cumsum", "% log(ind.)", "% non-zero"), dcol = c(1,2,4), dlty, dpos, …)

Description:

> *Méthode **plot()** des objets de classe 'abund'. Génère un graphique représentant à la fois l'abondance en nombre d'individus et en nombre de valeurs non nulles, pour les descripteurs classés selon **abund()**. La somme cumulée des différences entre ces deux courbes aide à identifier les transitions entre les groupes. Un repère indiquant les variables à extraire peut être tracé directement ou ultérieurement à l'aide de **lines()** ou **identify()**). Toutes les options de la méthode **plot()** sont disponibles.*

Arguments:

- **x:** un objet de classe *'abund'* issu d'une analyse lancée par la fonction ***abund()***.
- **n:** le nombre de variables à extraire. Par défaut, la valeur contenue dans **x$n** est utilisée. Précisez **NULL** pour ne pas tracer de repères.
- **lvert:** si **TRUE** (et **n** précisé, par défaut), un repère vertical sépare les variables reprises et celle qui sont éliminées de l'extraction.
- **lvars:** si **TRUE** (et **level** précisé, par défaut), les variables reprises sont repérées dans la même couleur que les traits de repère.
- **lcol:** couleur à utiliser pour les repères. Par défault, la couleur **2** est utilisée.
- **llty:** style de trait pour les repères. Par défaut, le style pointillé (**2**) est utilisé.
- **all:** si **TRUE** (par défaut), toutes les lignes (*cumsum*, %*log(ind.)* et %*non-zero*) sont tracées. Si **FALSE**, uniquement *cumsum* est tracé.
- **dlab:** les labels pour la légende. Par défaut, il s'agit respectivement de: **"cumsum"**, **"% log(ind.)"** et **"% non-zero"** pour les 3 courbes.
- **dcol:** couleurs à utiliser pour tracer les trois courbes. Par défaut, les couleurs **1**, **2** et **4** sont utilisées respectivement.
- **dlty:** style de trait à utiliser pour les trois courbes. Par défaut, le trait plein est utilisé.
- **dpos:** position horizontale et verticale du cadre de légende. Par défaut, en bas à gauche du graphique. Notez que la légende n'est tracée que si **all** = **TRUE**.
- **…:** paramètres optionnels supplémentaires pour le graphe

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

abund(), print.abund(), summary.abund(), lines.abund(), identify.abund(), extract.abund()

Exemple:

Voir partie I.

plot.escouf(x, level = x$level, lhorz = TRUE, lvert = TRUE, lvars = TRUE, lcol = 2, llty = 2, diff = TRUE, dlab = "RV' (units not shown)", dcol = 4, dlty, dpos = 0.8, …)

Description:

> *Méthode **plot()** des objets de classe 'escouf'. Génère un graphique représentant l'augmentation de RV au fur et à mesure que les variables sont ajoutées dans l'ordre obtenu par l'analyse. La variation de RV' peut-être également représentée sur le graphique, ainsi que des repères indiquant la position du seuil (mais ceux-ci peuvent être tracés aussi ultérieurement à l'aide de **lines()** ou **identify()**). Toutes les options de la méthode **plot()** sont disponibles. En particulier, pour représenter RV à l'aide d'une ligne brisée (comme dans PASSTEC 2000) au lieu d'un escalier, préciser l'option* **type = 'l'**.

Arguments:

- **x:** un objet de classe '*escouf*' issu d'une analyse lancée par la fonction ***escouf()***.
- **level:** la valeur du seuil (entre 0 et 1) où les repères seront tracés. Par défaut, la valeur de seuil de **x$level** est utilisée. Précisez **NULL** pour ne pas tracer de repères.
- **lhorz:** si **TRUE** (et **level** précisé, par défaut), un repère horizontal est tracé au niveau du seuil.
- **lvert:** si **TRUE** (et **level** précisé, par défaut), un repère vertical sépare les variables reprises et celle qui sont éliminées au seuil donné.
- **lvars:** si **TRUE** (et **level** précisé, par défaut), les variables reprises au seuil donné sont repérées dans la même couleur que les traits de repère.
- **lcol:** couleur à utiliser pour les repères. Par défaut, la couleur **2** est utilisée.
- **llty:** style de trait pour les repères. Par défaut, le style pointillé (**2**) est utilisé.
- **diff:** si **TRUE** (par défaut), la courbe RV' est également tracée, sinon pas.
- **dlab:** le label de la courbe RV' (par défaut **"RV' (units not shown)"**). Précisez par exemple **"RV' (unités non indiquées)"** pour un graphique en français. Vous pouvez faire de même avec le titre principal via l'option **main**.
- **dcol:** couleur à utiliser pour la courbe RV'. Par défaut, la couleur **4** est utilisée.
- **dlty:** style de trait pour la courbe RV'. Par défaut, le trait plein est utilisé.
- **dpos:** position horizontale relative du label pour la courbe RV'. La valeur par défaut de **0.8** correspond à 80%, c'est-à-dire, aux 4/5 à l'horizontale du graphique. La position verticale est déterminée automatiquement de façon à ce que le label soit situé au-dessus du point correspondant de la courbe.

Correspondance dans PASSTEC 2000:

Correspond à l'option **"Affichage du graphe"** dans la boite de dialogue **frmEscoufier** de PASSTEC 2000 (mais avec moins d'options).

Voir aussi:

escouf(), print.escouf(), summary.escouf(), lines.escouf(), identify.escouf(), extract.escouf()

Exemple:

Voir partie I.

plot.regul(x, series = 1, col = c(1,2), lty, plot.pts = TRUE, leg = FALSE, llab, lpos, xlab, ylab, main, …)

Description:

> *Trace un graphique superposé de la série initiale et de la série régularisée à partir d'un objet 'regul' obtenu à l'aide de la fonction* ***regul()****. Si* **plot.pts = TRUE***, les points de la série régularisée sont identifiés par des croix s'ils sont interpolés, ou par des croix entourées de cercle s'ils correspondent à ceux de la série initiale. Des traits verticaux indiquent le début et la fin de chacune des deux séries. Une légende peut être tracée en option.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.
- **series:** le numéro de la série à tracer. Par défaut, **1**, soit la seule série s'il n'y en a qu'une ou la première série d'un groupe s'il y en a plusieurs.
- **col:** les couleurs à utiliser pour tracer respectivement la série initiale et la série lissée. Par défaut, **col = c(1,2)**.
- **lty:** le style de trait (plein, pointillé, …) à utiliser pour les deux séries. Si cet argument n'est pas précisé, il prendra le style par défaut définit dans l'environnement en cours.
- **plot.pts:** si **TRUE** (par défaut), trace les points de la série régularisé (+). Ceux qui coïncident avec la série de départ sont entourés d'un cercle.
- **leg:** une légende doit-elle être ajoutée au graphique. Par défaut, non (**FALSE**).
- **llab:** les intitulés pour les entrées de la légende. Par défaut, **"initial"** pour la première courbe et le nom de la méthode de régularisation utilisée pour la seconde.
- **lpos:** la position de la légende {x, y}, dans les unités du graphique.
- **xlab:** intitulé de l'axe x. Par défaut: **"Time (<unit>)"**, où **<unit>** est l'unité qui a été définie dans l'objet *'regul'*.
- **ylab:** intitulé de l'axe y. Par défaut: **"Series"**.
- **main:** titre principal du graphique. Par défaut: **"Regulation of Series"** si une seule série a été traitée, ou **"Regulation of <sername>"** ou **<sername>** est le nom de la série correspondante, s'il y a plusieurs séries dans l'objet **x**.
- **…:** paramètres optionnels complémentaires pour les graphiques.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, des graphes similaires sont accessibles à partir des boites de dialogue des méthodes 'spline' et 'area' (***Régulation -> Spline*** et ***Régulation -> Méthode des aires*** en mode ***Univarié***).

Voir aussi:

regul(), print.regul(), summary.regul(), specs.regul(), lines.regul(), identify.regul(), hist.regul(), extract.regul()

Exemple:

Voir partie I.

plot.stat.slide(x, stat = "mean", col = c(1,2), lty, leg = FALSE, llab = c("series", stat), lpos = c(1.5, 10), xlab = "time", ylab = "y", main = paste("Sliding statistics"), ...)

Description:

> *Méthode **plot()** des objets de classe 'stat.slide'. Génère un graphique représentant la série de départ ainsi que la statistique glissante choisie. Il est possible de superposer d'autres statistiques à l'aide de **lines()**.*

Arguments:

- **x:** un objet de classe *'stat.slide'* issu d'un traitement lancé par la fonction ***stat.slide()***.
- **stat:** la statistique à représenter. Vous avez le choix entre **"min"**, **"max"**, **"median"**, **"mean"** (par défaut), **"pos.median"**, **"pos.mean"**, **"geo.mean"** et **"pen.mean"**. Les autres statistiques ne sont pas représentables sur le graphe.
- **col:** les couleurs à utiliser respectivement pour la série initiale et pour la statistique. Par défaut: **col = c(1,2)**.
- **lty:** le style de trait à utiliser respectivement pour le tracé de la série de départ et pour la statistique. Le style de trait par défaut (généralement un trait plein) est utilisé en l'absence de toute précision supplémentaire.
- **leg:** si **TRUE** (**FALSE** par défaut), une légende est également tracée.
- **llab:** les intitulés à utiliser dans le cadre de légende. Par défaut, il s'agit de **"series"** et de l'intitulé de la statistique **stat**.
- **lpos:** la position du cadre de légende sur le graphique (coordonnées {x, y} du coin supérieur gauche du cadre).
- **xlab:** intitulé de l'axe x. Par défaut: **"time"**.
- **ylab:** intitulé de l'axe y. Par défaut: **"y"**.
- **main:** titre du graphique. Par défaut: **"Sliding statistics"**.
- **…:** paramètres optionnels supplémentaires du graphe.

Correspondance dans PASSTEC 2000:

Correspond à l'option 'Affichage du graphe' dans la boite de dialogue des statistiques glissantes de PASSTEC 2000, mais on ne peut y représenter que la moyenne.

Voir aussi:

stat.slide(), print.stat.slide(), lines.stat.slide()

Exemple:

Voir partie I.

plot.tsd(x, series = 1, stack = TRUE, resid = TRUE, col, lty, labels=dimnames(X)[[2]], leg = TRUE, lpos = c(0,0), xlab = "time", ylab = "series", main, ...)

Description:

> *Trace un graphique combiné qui représente la série initiale et les différentes composantes obtenue par la décomposition les uns au dessus des autres pour un objet 'tsd' obtenu à l'aide de la fonction* ***tsd()***.

Arguments:

- **x:** un objet *'tsd'* obtenu à partir de la fonction ***tsd()***.
- **series:** le numéro de la série à tracer. Par défaut, **1**, soit la seule série s'il n'y en a qu'une ou la première série d'un groupe s'il y en a plusieurs.
- **stack:** les graphes de chaque composante doivent-ils être tracés les uns au-dessus des autres (**stack = TRUE**, par défaut), ou superposés sur un même graphique (**stack = FALSE**)?
- **resid:** la composante **"residuals"** doit-elle être tracée également (**TRUE**, par défaut) ou pas (**FALSE**)?
- **col:** les couleurs à utiliser pour tracer respectivement la série initiale et les différentes composantes. La couleur par défaut pour les graphiques est appliquée à tous les tracés si ce paramètre n'est pas précisé.
- **lty:** le style de trait (plein, pointillé, …) à utiliser pour respectivement la série initiale et les différentes composantes. Si cet argument n'est pas précisé, il prendra le style par défaut définit dans l'environnement en cours pour tous les tracés.
- **labels:** les titres à utiliser pour libeller les axes y de chaque graphe lorsque **stack = TRUE**. Les labels à utiliser pour la légende lorsque **stack = FALSE** et **leg = TRUE**. Par défaut, le nom de la composante (**"trend"**, **"seasonal"**, **"filtered"**, **"residuals"**,…) est utilisé.
- **leg:** lorsque **stack = FALSE** uniquement, faut-il tracer une légende (**TRUE**, par défaut) ou pas (**FALSE**)?
- **lpos:** position du coin supérieur gauche de la légende en coordonnées {x, y} du graphe. Par défaut, **lpos = c(0,0)**.
- **xlab:** le titre de l'axe x. Par défaut: **"time"**.
- **ylab:** le titre de l'axe y, uniquement lorsque **stack = FALSE** (dans le cas contraire, c'est **labels** qui est utilisé pour les axes y des différents graphes). Par défaut: **"series"**.
- **main:** le titre principal du graphe, uniquement lorsque **stack = FALSE**. Par défaut: **"Series decomposition by <method> - <type>"** où **<method>** est la méthode de décomposition utilisée (**"diff"**, **"average"**, **"median"**, **"evf"**,…) et **<type>** est le type de modèle (soit **"additive"**, soit **"multiplicative"**).
- **…:** paramètres optionnels complémentaires pour les graphiques.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, des graphes similaires sont accessibles à partir des boites de dialogue des différentes méthodes utilisées. Cependant, les graphiques sont automatiquement superposés sur les mêmes axes, au lieu d'être placés les uns au-dessus des autres, et les résidus ne sont pas tracés. Cela correspond aux options **stack = FALSE** et **resid = FALSE**. PASTECS offre donc des choix plus larges pour la présentation des décompositions.

Voir aussi:

tsd(), print.tsd(), summary.tsd(), specs.tsd(), extract.tsd()

Exemple:

Voir partie I.

plot.turnogram(x, level = 0.05, lhorz = TRUE, lvert = TRUE, lcol = 2, llty = 2, xlog = TRUE, xlab = paste("interval", x$unit.text), ylab = "I (bits)", main = paste(x$type, "turnogram for:",x$data), sub = paste(x$fun, "/", x$proba), ...)

Description:

> *Trace le graphique du tournogramme pour un objet 'turnogram' obtenu à l'aide de la fonction* **turnogram()**.

Arguments:

- **x:** un objet *'turnogram'* obtenu à partir de la fonction ***turnogram()***.
- **level:** le seuil de significativité à indiquer sur le graphe. Par défaut, **level=0.05** (p = 5%).Ce seuil est indiqué uniquement si **lhorz =TRUE**.
- **lhorz:** si une (test unilatéral) ou deux (test bilatéral) lignes horizontales doivent être dessinées sur le graphe pour indiquer le seuil de significativité du test à la valeur de **level**. Par défaut, **lhorz = TRUE**, la ou les lignes sont tracées.
- **lvert:** si une ligne verticale doit être indiquée sur le graphe du tournogramme, montrant l'intervalle correspondant à l'extraction (au départ, cela correspond au maximum d'information, mais cela peut être modifié par la suite, voir exemple). Par défaut, cette ligne est tracée (**lvert = TRUE**).
- **lcol:** la couleur à utiliser pour les lignes supplémentaires: seuil de significativité (**lhorz = TRUE**) et intervalle correspondant au maximum d'information (**lvert = TRUE**). Par défaut, la couleur **2** est utilisée.
- **llty:** le style de trait pour les lignes supplémentaires. Par défault, **llty = 2**, le trait est pointillé.
- **xlog:** si **TRUE** (par défaut), l'échelle des abscisses sur le graphe du tournogramme, c'est-à-dire la succession des intervalles testés, est exprimée en logarithmes.
- **xlab:** le titre de l'axe x. Par défaut: **"interval (unit)"**.
- **ylab:** le titre de l'axe y. Par défaut: **"I (bits)"**.
- **main:** le titre principal du graphe: **"Simple/Complete turnogram for: data"**.
- **sub:** le sous-titre à utiliser pour le graphe. Par défaut: **"function / one- or two-tailed probability"**, où **function** est la function utilisée pour aggréger les intervalles (argument **FUN=…** dans **turnogram()**).
- **…:** paramètres optionnels complémentaires pour les graphiques.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, le tournogramme simple pour l'aggrégation par la moyenne et pour un test bilatéral est tracé, mais sans représentation du seuil de significativité. PASTECS offre donc des choix plus larges pour le calcul et la présentation des tournogrammes.

Voir aussi:

turnogram(), print.turnogram(), summary.turnogram(), identify.turnogram(), extract.turnogram(), pgleissberg()

Exemple:

Voir partie I.

plot.turnpoints(x, level = 0.05, lhorz = TRUE, lcol = 2, llty = 2, type = "l", xlab = "observations", ylab = paste("I (bits), level = ", level*100, "%", sep=""), main = paste("Information (turning points) for:",x$data), ...)

Description:

> *Trace le graphique de la quantité d'information associée à chaque observation pour un objet 'turnpoints' obtenu à l'aide de la fonction* **turnpoints()**.

Arguments:

- **x:** un objet *'turnpoints'* obtenu à partir de la fonction ***turnpoints()***.
- **level:** le seuil de significativité à indiquer sur le graphe. Par défaut, **level=0.05** (p = 5%).Ce seuil est indiqué uniquement si **lhorz** =**TRUE**.
- **lhorz:** si un trait horizontal doit figurer le seuil de significativité sur le graphique. **TRUE** par défaut.
- **lcol:** la couleur à utiliser pour indiquer le seuil de significativité. Par défaut, la couleur **2** est utilisée.
- **llty:** le style de trait pour le seuil de significativité. Par défaut, **llty = 2**, le trait est pointillé.
- **type:** le type de graphe à tracer. Par défaut, **"l"**, les points sont reliés par des segments de droite.
- **xlab:** le titre de l'axe x. Par défaut: **"observations"**.
- **ylab:** le titre de l'axe y. Par défaut: **"I (bits), level = x%"**.
- **main:** le titre principal du graphe: **"Information (turning points) for: data"**.
- **…:** paramètres optionnels complémentaires pour le graphique.

Correspondance dans PASSTEC 2000:

Cette fonction trace le même graphique que le "graphe des informations" dans PASSTEC 2000.

Voir aussi:

turnpoints(), print.turnpoints(), summary.turnpoints(), lines.turnpoints(), extract.turnpoints()

Exemple:

Voir partie I.

print.abund(x)

Description:

> *Méthode **print()** des objets de classe 'abund'. Formate l'affichage de la variable. Renvoie des informations un peu moins détaillées que la méthode **summary()**.*

Arguments:

- **x:** un objet de classe *'abund'* issu d'une analyse lancée par la fonction ***abund()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

abund(), summary.abund(), plot.abund(), lines.abund(), identify.abund(), extract.abund()

Exemple:

Voir partie I.

print.escouf(x)

Description:

> *Méthode **print()** des objets de classe 'escouf'. Formate l'affichage de la variable (informations de base). Utiliser **summary()** pour obtenir une sortie plus détaillée.*

Arguments:

- **x:** un objet de classe '*escouf*' issu d'une analyse lancée par la fonction ***escouf()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

escouf(), summay.escouf(), plot.escouf(), lines.escouf(), identify.escouf(), extract.escouf()

Exemple:

Voir partie I.

print.regul(x)

Description:

> *Méthode d'impression des objets 'regul'. Appelée automatiquement à l'affichage de l'objet. Utiliser la méthode* ***summary()*** *pour obtenir un rapport plus détaillé.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), summary.regul(), plot.regul(), lines.regul(), identify.regul(), hist.regul()

Exemple:

Voir partie I.

print.stat.slide(x)

Description:

> *Méthode **print()** des objets de classe 'stat.slide'. Formate l'affichage des résultats. Pour des informations plus détaillées, utiliser la méthode **summary()**.*

Arguments:

- **x:** un objet de classe '*stat.slide*' issu d'un traitement obtenu par la fonction ***stat.slide()***.

Correspondance dans PASSTEC 2000:

Les résultats sont automatiquement affichés après calcul à l'aide de ***Description -> Statistiques glissantes*** en mode ***Univarié***. PASTECS offre toutefois plus de statistiques calculables.

Voir aussi:

stat.slide(), plot.stat.slide(), lines.stat.slide()

Exemple:

Voir partie I.

print.tsd(x)

Description:

> *Méthode d'impression des objets 'tsd'. Appelée automatiquement à l'affichage de l'objet. Utiliser la méthode* ***summary()*** *pour afficher un rapport plus détaillé.*

Arguments:

- **x:** un objet *'tsd'* obtenu à partir de la fonction ***tsd()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

tsd(), summary.tsd(), specs.tsd(), plot.tsd(), extract.tsd()

Exemple:

Voir partie I

print.turnogram(x)

Description:

> *Méthode d'impression des objets 'turnogram'. Appelée automatiquement à l'affichage de l'objet. Utiliser la méthode* ***summary()*** *pour obtenir plus d'informations.*

Arguments:

- **x:** un objet *'turnogram'* obtenu à partir de la fonction ***turnogram()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

turnogram(), summary.turnogram(), plot.turnogram(), identify.turnogram(), extract.turnogram(), pgleissberg()

Exemple:

Le résultat de cette fonction est un peu moins détaillé que celui de ***summary()***, mais y est très semblable (voir partie I).

print.turnpoints(x)

Description:

> *Méthode d'impression des objets 'turnpoints'. Appelée automatiquement à l'affichage de l'objet. Utiliser la méthode* ***summary()*** *pour obtenir une impression plus détaillée des résultats.*

Arguments:

- **x:** un objet *'turnpoints'* obtenu à partir de la fonction ***turnpoints()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

turnpoints(), summary.turnpoints(), plot.turnpoints(), lines.turnpoints(), extract.turnpoints()

Exemple:

Le résultat de cette fonction est un peu moins détaillé que celui de ***summary()***, mais y est très semblable (voir partie I).

regarea(x, y, xmin, n, deltat, rule = 1, window = deltat, interp = FALSE, split = 100)

Description:

> *Régularise une série (***x**, **y***) en commençant à* **xmin**, *avec un pas* **deltat** *et* **n** *données. La méthode des aires est utilisée, avec une fenêtre* **window**.

Arguments:

- **x:** un vecteur indiquant le temps auquel les valeurs irrégulières **y** ont été échantillonnées. Les valeurs manquantes sont permises. Si ce vecteur n'est pas ordonné en valeurs croissantes, il le sera avant le calcul de l'interpolation.

- **y:** un vecteur contenant les valeurs de la série à régulariser. Les valeurs manquantes sont autorisées.

- **xmin:** la valeur de temps minimale (pour la première observation) dans la série régularisée. Par défaut, la plus petite valeur de temps dans la série initiale.

- **n:** le nombre d'entrées à calculer dans la série régularisée. Par défaut, le même nombre de données que dans la série de départ.

- **deltat:** le pas de temps d'une entrée à l'autre dans la série régularisée. Par défaut, il est calculé de manière à ce que la plus grande valeur de temps dans la série régularisée coïncide avec la valeur correspondante dans la série de départ.

- **rule:** la règle à considérer pour les valeurs extrapolées (en dehors de la série échantillonnée) dans la série régularisée. Avec **rule = 1** (par défaut), ces entrées ne sont pas calculées et reçoivent **NA**; avec **rule = 2**, ces entrées sont extrapolées.

- **window:** taille de la fenêtre à utiliser pour réaliser l'interpolation (voir description de la méthode). Par défaut, des fenêtres successives adjacentes sont utilisées, comme dans PASSTEC 2000 (**window = deltat**).

- **interp:** indique si les point qui coïncident entre la série de départ et la série régularisée doivent être également interpolés (**interp = TRUE**) ou non (**interp = FALSE**, par défaut).

- **split:** pour optimiser le temps de calcul, et pour éviter de saturer la mémoire vive, les très longues séries extrapolées sont divisées en sous-séries lors du calcul (voir description de la méthode; mais ceci est transparent pour l'utilisateur). La valeur par défaut de **100** convient généralement très bien, et ne doit être diminuée que si le programme rapporte un problème de mémoire vive lors du calcul de la régularisation.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, cette méthode est accessible par: ***Régulation -> Méthode des aires*** en mode ***Unitivarié***. Toutefois, PASSTEC 2000 ne permet pas de définir la fenêtre d'interpolation indépendamment du pas de régularisation (fenêtres chevauchantes ou disjointes), contrairement à PASTECS.

Voir aussi:

regul(), regconst(), reglin(), respline()

Exemple:

Voir partie I.

regconst(x, y, xmin, n, deltat, rule = 1, f = 0)

Description:

> *Régularise une série (***x**, **y***) en commençant à* **xmin**, *avec un pas* **deltat** *et* **n** *données. La méthode des valeurs constantes est utilisée.*

Arguments:

- **x:** un vecteur indiquant le temps auquel les valeurs irrégulières **y** ont été échantillonnées. Les valeurs manquantes sont permises. Si ce vecteur n'est pas ordonné en valeurs croissantes, il le sera avant le calcul de l'interpolation.

- **y:** un vecteur contenant les valeurs de la série à régulariser. Les valeurs manquantes sont autorisées.

- **xmin:** la valeur de temps minimale (pour la première observation) dans la série régularisée. Par défaut, la plus petite valeur de temps dans la série initiale.

- **n:** le nombre d'entrées à calculer dans la série régularisée. Par défaut, le même nombre de données que dans la série de départ.

- **deltat:** le pas de temps d'une entrée à l'autre dans la série régularisée. Par défaut, il est calculé de manière à ce que la plus grande valeur de temps dans la série régularisée coïncide avec la valeur correspondante dans la série de départ.

- **rule:** la règle à considérer pour les valeurs extrapolées (en dehors de la série échantillonnée) dans la série régularisée. Avec **rule = 1** (par défaut), ces entrées ne sont pas calculées et reçoivent **NA**; avec **rule = 2**, ces entrées sont extrapolées.

- **f:** coefficient qui donne plus de poids à la valeur à gauche (**f = 0**, par défaut) à la valeur à droite (**f = 1**), ou à un intermédiaire entre les deux (**0 < f < 1**) lors du calcul de l'interpolation.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), reglin(), regspline(), regarea()

Exemple:

Voir partie I.

reglin(x, y, xmin, n, deltat, rule = 1)

Description:

> *Régularise une série (***x***,* **y***) en commençant à* **xmin***, avec un pas* **deltat** *et* **n** *données. La méthode d'interpolation linéaire entre les deux valeurs encadrantes est utilisée.*

Arguments:

- **x:** un vecteur indiquant le temps auquel les valeurs irrégulières **y** ont été échantillonnées. Les valeurs manquantes sont permises. Si ce vecteur n'est pas ordonné en valeurs croissantes, il le sera avant le calcul de l'interpolation.

- **y:** un vecteur contenant les valeurs de la série à régulariser. Les valeurs manquantes sont autorisées.

- **xmin:** la valeur de temps minimale (pour la première observation) dans la série régularisée. Par défaut, la plus petite valeur de temps dans la série initiale.

- **n:** le nombre d'entrées à calculer dans la série régularisée. Par défaut, le même nombre de données que dans la série de départ.

- **deltat:** le pas de temps d'une entrée à l'autre dans la série régularisée. Par défaut, il est calculé de manière à ce que la plus grande valeur de temps dans la série régularisée coïncide avec la valeur correspondante dans la série de départ.

- **rule:** la règle à considérer pour les valeurs extrapolées (en dehors de la série échantillonnée) dans la série régularisée. Avec **rule = 1** (par défaut), ces entrées ne sont pas calculées et reçoivent **NA**; avec **rule = 2**, ces entrées sont extrapolées.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), regconst(), regspline(), regarea()

Exemple:

Voir partie I.

regspline(x, y, xmin, n, deltat, rule = 1, periodic = FALSE)

Description:

> *Régularise une série (***x**, **y***) en commençant à* **xmin**, *avec un pas* **deltat** *et* **n** *données. La méthode des splines cubiques est utilisée.*

Arguments:

- **x:** un vecteur indiquant le temps auquel les valeurs irrégulières **y** ont été échantillonnées. Les valeurs manquantes sont permises. Si ce vecteur n'est pas ordonné en valeurs croissantes, il le sera avant le calcul de l'interpolation.

- **y:** un vecteur contenant les valeurs de la série à régulariser. Les valeurs manquantes sont autorisées.

- **xmin:** la valeur de temps minimale (pour la première observation) dans la série régularisée. Par défaut, la plus petite valeur de temps dans la série initiale.

- **n:** le nombre d'entrées à calculer dans la série régularisée. Par défaut, le même nombre de données que dans la série de départ.

- **deltat:** le pas de temps d'une entrée à l'autre dans la série régularisée. Par défaut, il est calculé de manière à ce que la plus grande valeur de temps dans la série régularisée coïncide avec la valeur correspondante dans la série de départ.

- **rule:** la règle à considérer pour les valeurs extrapolées (en dehors de la série échantillonnée) dans la série régularisée. Avec **rule = 1** (par défaut), ces entrées ne sont pas calculées et reçoivent **NA**; avec **rule = 2**, ces entrées sont extrapolées.

- **periodic:** indique si la série est considérée comme périodique (**periodic = TRUE**, la première valeur de **y** DOIT alors être égale à sa dernière valeur – interruption sous S+ ou « warning » sous R, dans le cas contraire–). Si tel est le cas, les dérivées de la première et de la dernière spline sont estimées à partir de l'autre extrémité de la série. Dans le cas contraire (**periodic = FALSE**, par défaut), les dérivées des premiers et derniers segments de spline sont alors considérées égales à zéro.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, cette méthode est accessible par: ***Régulation -> Spline*** en mode ***Univarié***.

Voir aussi:

regul(), regconst(), reglin(), regarea()

Exemple:

Voir partie I.

regul(x, y, xmin, n, units = "days", frequency, deltat = 1/frequency, datemin, dateformat="m/d/Y", tol, tol.type = "center", methods = "linear", rule = 1, f = 0, periodic = F, window, split = 100, specs = NULL)

Description:

> *Régularise des séries et renvoie un objet 'regul' qui permet à la fois le diagnostic du traitement de régularisation (**summary()**, **specs()**, **plot()**, **lines()**, **hist()**) et une extraction aisée d'objets de type 'ts' ou 'rts', time series régulières à l'aide de **extract()** ou **tseries()** (voir exemple).*

Arguments:

- **x:** un vecteur contenant les valeurs de temps des séries échantillonnées. Ce temps (sous forme décimale) peut être exprimé dans une unité arbitraire, à définir par **units** (voir plus loin) comme **"months"**, **"years"**, **"sec"**,… Il est plus généralement exprimé en **"days"** (fichiers importés de PASSTEC 2000 ou d'Excel, dates standard de S+ ou R), avec une éventuelle partie décimale qui représente l'heure de la journée.

- **y:** un vecteur (série unique) ou une matrice colonne représentant les mesures de plusieurs séries aux temps repris dans **x**

- **xmin:** la valeur de temps minimal désirée pour les séries régularisées, exprimée dans les même unités que **x**. La valeur par défaut est la même que la plus petite valeur de **x**

- **n:** le nombre de valeurs dans la série régularisée. Par défaut, le même nombre de valeurs que pour **x**

- **units:** précisez ici l'unité utilisée pour mesurer le temps (par défaut: **"days"**). Une valeur spécifique **"daystoyears"** indique que **x** est exprimé en jours, mais que l'on veut obtenir des séries régularisées dont l'unité de temps soit l'année (pour étudier les variations saisonnières dans des séries pluriannuelles). Dans ce dernier cas, **units** devient **"years"** dans l'objet *'regul'* créé, et le temps est mis à l'échelle de sorte que les parties entières représentent les années. Préciser **datemin** pour obtenir une mise à l'échelle correcte.

- **frequency:** la fréquence des données régularisées par rapport à l'unité de temps. Par exemple, une fréquence de 12 pour un temps dont l'unité est l'année, correspond à des données mensuelles. Attention! Lorsque **units** = **"daystoyears"**, **frequency** et **deltat** sont relatifs à l'année, contrairement à tous les autres arguments qui sont exprimés dans l'unité de départ, c'est-à-dire le jour! Dans ce cas précis, **frequency** est à utiliser de préférence puisqu'on veut généralement une valeur entière pour la fréquence (par exemple: 4, 12 ou 24; ce qui donne respectivement 1/4, 1/12 ou 1/24 pour **deltat**).

- **deltat:** l'intervalle entre deux mesures dans la série régularisée. C'est l'inverse de la fréquence. Un seul de ces deux arguments doit être précisé. Si les deux sont données, c'est la fréquence qui est utilisée en priorité.

- **datemin:** précisez ici la date absolue de la valeur minimale dans le vecteur **x**, au format **"m/d/Y"**. Par exemple, **"03/21/1998"** pour le 21 mars 1998. Surtout utile lorsque **units** = **"daystoyears"** pour obtenir une représentation correcte du temps après mise à l'échelle. D'autres représentations sont permises, ainsi que l'utilisation de dates au format POSIXt sous R (voir ***daystoyears()*** pour plus d'informations à ce sujet).

- **dateformat: :** indiquez le format de la date. Par exemple: **"d/m/Y"** ou **"m/d/Y"** (par défaut). Notez la différence, dans R, entre **"y"** qui définit les années sur 2

chiffres (par exemple 87 pour 1987) et **"Y"** qui définit les années sur 4 chiffres (1987). **Veuillez à ne pas vous tromper à ce niveau!**

- **tol:** la tolérance pour déterminer si une valeur mesurée correspond à une valeur régularisée. Si **tol = 0**, les temps dans la série initiale et dans la série régularisée doivent être strictement identiques pour que les valeurs correspondantes de **y** ne soient pas interpolées. **tol** doit être une fraction entière de **deltat** (**deltat**, **deltat/2**, **deltat/3**, etc…), et ne peut lui être supérieure. Dans le cas contraire, la valeur de **tol** sera automatiquement ajustée à la valeur permise la plus proche. Par défaut, **tol = NULL**, ce qui équivaut à **tol = 0**. Attention: dans le cas particulier où **units = "daystoyears"**, **tol** est exprimé en jours, alors que **deltat** est exprimé en fraction d'année!

- **tol.type:** le type d'ajustement pour la tolérance, soit **"left"**, **"both"** (par défaut), **"right"** ou **"none"**. Si **tol.type = "left"**, les valeurs de **x** qui correspondent aux valeurs régularisées sont cherchées dans une fenêtre $]\mathbf{x_{régul}} - \mathbf{tol}, \mathbf{x_{régul}}]$. Si **tol.type = "both"**, ces valeurs sont recherchées dans la fenêtre $]\mathbf{x_{régul}} - \mathbf{tol}, \mathbf{x_{régul}} + \mathbf{tol}[$. Si **tol.type = "right"**, les fenêtres $[\mathbf{x_{régul}}, \mathbf{x_{régul}} + \mathbf{tol}[$ sont scrutées. Au cas où plusieurs valeurs seraient présentes dans la fenêtre, celle qui est la plus proche dans le temps de la valeur régularisée $\mathbf{x_{régul}}$ est conservée. Enfin, si **tol.type = "none"**, toutes les valeurs de la série régularisée sont interpolées.

- **methods:** la ou les méthodes de régularisation à utiliser pour la ou les séries à traiter parmi **"constant"**, **"linear"**, **"spline"** ou **"area"**. Dans le cas où plusieurs séries seraient sélectionnées (**y** est une matrice avec les différentes séries en colonnes), il est possible de définir la méthode individuellement pour chaque série. Par exemple, **methods = c("linear","area", "spline")** signifie que la méthode linéaire est choisie pour la première série, la méthode des aires pour la seconde et la régularisation par courbes splines pour la troisième. Par défaut, la méthode linéaire est utilisée pour toutes les séries.

- **rule:** la règle à considérer pour les valeurs extrapolées (en dehors de la série échantillonnée) dans la série régularisée. Avec **rule = 1** (par défaut), ces entrées ne sont pas calculées et reçoivent **NA**; Avec **rule = 2**, ces entrées sont extrapolées (à ne pas utiliser, en principe!).

- **f:** paramètre de la méthode constante de régularisation. Coefficient qui donne plus de poids à la valeur à gauche (**f = 0**, par défaut) à la valeur à droite (**f = 1**), ou à un intermédiaire entre les deux (**0 < f < 1**) lors du calcul de l'interpolation.

- **periodic:** paramètre de la méthode spline. Indique si la série est considérée comme périodique (**periodic = TRUE**, la première valeur de **y** DOIT alors être égale à sa dernière valeur –interruption sous S+ ou « warning » sous R, dans le cas contraire–). Si tel est le cas, les dérivées de la première et de la dernière spline sont estimées à partir de l'autre extrémité de la série. Dans le cas contraire (**periodic = FALSE**, par défaut), les dérivées des premiers et derniers segments de spline sont alors considérés égales à zéro.

- **window:** paramètre de la méthode des aires. Taille de la fenêtre à utiliser pour réaliser l'interpolation (voir description de la méthode). Par défaut, l'intervalle moyen de la série initiale est utilisé. Préciser la même valeur que **deltat** pour travailler avec des fenêtres successives adjacentes comme dans PASSTEC 2000 (**window = deltat**).

- **split:** autre paramètre de la méthode des aires. Pour optimiser le temps de calcul, et pour éviter de saturer la mémoire vive, les très longues séries extrapolées sont divisées en sous-séries lors du calcul (voir description de la méthode; mais ceci est transparent pour l'utilisateur). La valeur par défaut de **100** convient généralement très bien, et ne doit être diminuée que si le programme rapporte un problème de mémoire vive lors du calcul de la régularisation.

- **specs:** un objet de type *'specs.regul'* retourné par la fonction ***specs()*** appliquée à une objet *'regul'*. Permet de récupérer les paramètres d'une précédente régularisation (voir description et exemple de ***specs.regul()***).

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, seules les méthodes 'spline' et 'area' sont implémentées. Elles sont accessibles respectivement par: ***Régulation -> Spline*** et ***Régulation -> Méthode des aires*** en mode ***Univarié***. La méthode des aires ne peut calculer que des fenêtres égales au pas de la régularisation dans PASSTEC 2000, alors que dans PASTECS, le pas et la taille de fenêtre peuvent être modifiés indépendamment. De plus, pour la méthode spline, PASTECS offre la possibilité de mieux calculer les premier et dernier segments, si l'on peut faire l'hypothèse que la série est périodique, et donc que la dernière valeur est égale à la première.

Voir aussi:

print.regul(), summary.regul(), specs.regul(), plot.regul(), lines.regul(), identify.regul(), hist.regul(), extract.regul(), regul.screen(), regul.adj(), tseries(), regconst(), reglin(), regspline(), regarea()

Exemple:

Voir partie I.

regul.adj(x, xmin, frequency, deltat, tol, tol.type = "both", nclass = 50, col = c(4, 5, 2), xlab, ylab, main, plotit = TRUE, …)

Description:

> *Trace un histogramme prédictif des variables qui coïncident entre la série initiale et la série finale régularisée, en fonction de la distance dans le temps qui sépare ces paires respectives. La première barre représente le nombre de données qui coïncident exactement. Les barres suivantes représentent le nombre de données initiales situées dans la fenêtre de tolérances, mais de plus en plus éloignées des points régularisés. La dernière barre représente le nombre de points qui doivent être interpolés parce qu'aucune donnée initiale ne se trouve dans la fenêtre de tolérance définie. Renvoie aussi un résultat chiffré. Contrairement à* ***hist.regul()****, cette fonction ne nécessite pas un objet 'regul', mais simplement un vecteur temps. Elle peut donc être utilisée pour étudier les possibilités existantes* ***avant*** *d'effectuer la régularisation, alors que* ***hist.regul()*** *est un outil d'analyse* post hoc*, donc* ***après*** *régularisation.*

Arguments:

- **x:** un vecteur indiquant le temps auquel les données irrégulières ont été échantillonnées, dans une unité arbitraire, mais au format décimal.

- **xmin:** la valeur de temps minimal à tester pour la régularisation

- **frequency:** la fréquence à tester pour la régularisation

- **deltat:** le pas de temps à tester pour la régularisation. Le pas est l'inverse de la fréquence. Ces deux arguments sont donc redondants. Un seul de ces deux arguments doit être précisé. Si les deux sont donnés, c'est la fréquence qui est utilisée en priorité.

- **tol:** la tolérance pour déterminer si une valeur mesurée correspond à une valeur régularisée. Si **tol = 0**, les temps dans la série initiale et dans la série régularisée doivent être strictement identiques pour que les valeurs correspondantes de **y** ne soient pas interpolées. **tol** doit être une fraction entière de **deltat** (**deltat**, **deltat/2**, **deltat/3**, etc…), et ne peut lui être supérieure. Dans le cas contraire, la valeur de **tol** sera automatiquement ajustée à la valeur permise la plus proche. Par défaut, **tol = deltat**, valeur maximale qu'il vaut mieux laisser comme cela pour avoir une vue complète des possibilités.

- **tol.type:** le type d'ajustement pour la tolérance, soit "**left**", "**both**" (par défaut), "**right**" ou "**none**". Si **tol.type = "left"**, les valeurs de **x** qui correspondent aux valeurs régularisées sont cherchées dans une fenêtre $]\mathbf{x_{régul}} - \mathbf{tol}, \mathbf{x_{régul}}]$. Si **tol.type = "both"**, ces valeurs sont recherchées dans la fenêtre $]\mathbf{x_{régul}} - \mathbf{tol}, \mathbf{x_{régul}} + \mathbf{tol}[$. Si **tol.type = "right"**, les fenêtres $[\mathbf{x_{régul}}, \mathbf{x_{régul}} + \mathbf{tol}[$ sont scrutées. Au cas où plusieurs valeurs seraient présentes dans la fenêtre, celle qui est la plus proche dans le temps de la valeur régularisée $\mathbf{x_{régul}}$ est conservée. Enfin, si **tol.type = "none"**, seules les valeurs qui coïncident exactement sont recherchées (pas très utile dans cette fonction!).

- **nclass:** paramètre de l'histogramme. Valeur initiale du nombre de classes à réaliser. Ce nombre est recalculé au mieux par le programme, mais il arrive qu'il ne puisse obtenir une séparation en classes adéquate avec le nombre fournit. Essayer alors une valeur supérieure. Par défaut, **nclass = 50**.

- **col:** les couleurs à utiliser pour remplir respectivement la première barre (coïncidence exacte), les suivantes (coïncidence dans la fenêtre de tolérance), et la dernière barre (valeurs interpolées). Par défaut, **col = c(4, 5, 2)**.

- **xlab:** intitulé de l'axe x. Par défaut: **"Time distance"**.

- **ylab:** intitulé de l'axe y. Par défaut: **"Frequency"**.

- **main:** titre principal du graphique. Par défaut: **"Number of matching observations"**.

- **plotit:** indique si le graphe doit être tracé ou non, en plus du renvoi des valeurs chiffrées. Par défaut, oui (**TRUE**).

- **…:** paramètres optionnels complémentaires pour les graphiques.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), regul.screen(), hist.regul()

Exemple:

Voir partie I.

regul.screen(x, weight = NULL, xmin, frequency, deltat, tol, tol.type = "both")

Description:

> *Effectue un « screening » croisé des deux paramètres* **xmin** *et* **deltat** *(ou* **frequency***) pour différentes valeurs fournies. Détermine la valeur optimale de* **n***, calcule* **tol***, et détermine le nombre de valeurs qui coïncident exactement et dans la fenêtre de tolérance entre la série initiale et la série régularisée (valeur éventuellement pondérée par* **weight***). Cette fonction sert à identifier la ou les combinaisons de paramètres* **xmin**, **deltat**, **n** *qui optimalisent le nombre de points qui coïncident au seuil de tolérance* **tol** *près. Utile pour calculer le meilleur pas de temps* **deltat** *et valeur d'origine* **xmin** *lorsqu'on est en présence d'une série très irrégulière au départ.*

Arguments:

- **x:** un vecteur contenant les valeurs de temps des séries irrégulières échantillonnées.

- **weight:** un vecteur de même longueur que **x**, et contenant la pondération à donner aux points. Une valeur de 0 indique d'ignorer le point correspondant. Une valeur de 1 donne une pondération normale au point, et une valeur supérieure donne plus d'importance à ce point. Toutes les valeurs négatives ou manquantes sont ramenées à 0. Si **weight = NULL** (par défaut), une pondération 1 est utilisée pour tous les points.

- **xmin:** l'ensemble des valeurs de temps minimal à tester pour la régularisation

- **frequency:** l'ensemble des valeurs de fréquence à tester pour la régularisation

- **deltat:** l'ensemble des valeurs de pas à tester pour la régularisation. Le pas est l'inverse de la fréquence. Ces deux arguments sont donc redondants. Un seul de ces deux arguments doit être précisé. Si les deux sont donnés, c'est la fréquence qui est utilisée en priorité.

- **tol:** la tolérance pour déterminer si une valeur mesurée correspond à une valeur régularisée. Si **tol = 0**, les temps dans la série initiale et dans la série régularisée doivent être strictement identiques pour que les valeurs correspondantes de **y** ne soient pas interpolées. **tol** doit être une fraction entière de **deltat** (**deltat**, **deltat/2**, **deltat/3**, etc…), et ne peut lui être supérieure. Dans le cas contraire, la valeur de **tol** sera automatiquement ajustée à la valeur permise la plus proche. Par défaut, **tol = deltat/5**.

- **tol.type:** le type d'ajustement pour la tolérance, soit "**left**", "**both**" (par défaut), "**right**" ou "**none**". Si **tol.type = "left"**, les valeurs de **x** qui correspondent aux valeurs régularisées sont cherchées dans une fenêtre $]\mathbf{x_{régul}} - \mathbf{tol}, \mathbf{x_{régul}}]$. Si **tol.type = "both"**, ces valeurs sont recherchées dans la fenêtre $]\mathbf{x_{régul}} - \mathbf{tol}, \mathbf{x_{régul}} + \mathbf{tol}[$. Si **tol.type = "right"**, les fenêtres $[\mathbf{x_{régul}}, \mathbf{x_{régul}} + \mathbf{tol}[$ sont scrutées. Au cas où plusieurs valeurs seraient présentes dans la fenêtre, celle qui est la plus proche dans le temps de la valeur régularisée $\mathbf{x_{régul}}$ est conservée. Enfin, si **tol.type = "none"**, seules les valeurs qui coïncident exactement sont recherchées.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), regul.adj(), hist.regul()

Exemple:

Voir partie I.

specs.regul(x)

Description:

> *Récupère et/ou imprime les spécifications pour la régularisation (i.e., les différents paramètres utilisés lors de la régularisation à partir d'un objet 'regul'.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), print.regul(), summary.regul(), plot.regul(), lines.regul(), identify.regul(), hist.regul(), extract.regul()

Exemple:

```
> data(releve)     # Seulement dans R
> rel.reg <- regul(releve$Day, releve$Melosul, xmin=0, n=78,
+ deltat=17, tol=2.125, methods="linear")
> specs(rel.reg)
              xmin                 n         frequency
                 0                78 0.0588235294117647
            deltat             units        dateformat
                17              days             m/d/Y
               tol          tol.type           methods
             2.125              both            linear
              rule                 f          periodic
                 1                 0             FALSE
            window             split
  17.1948051948052               100
```

On peut aussi récupérer des spécifications de paramètres à partir d'un objet *'regul'* existant et les appliquer à une nouvelle régularisation en précisant:

```
> new.reg <- regul(releve$Day, releve$Navi,
+ specs=specs(rel.reg), methods="area")
```

Cela évite de devoir rentrer à nouveau tous les paramètres à chaque régularisation. Les paramètres qui sont reprécisés explicitement, comme **methods** ici, sont utilisés prioritairement aux valeurs renvoyées par ***specs()***. A noter toutefois que, comme tous les arguments sont évalués lors du premier appel de la fonction ***regul()***, il n'est pas possible de récupérer des spécifications calculées. Par exemple, lors du premier appel de ***regul()***, **deltat = 10** et **tol = deltat/5**, soit **2**. Si l'on récupère les spécifications de l'objet créé, mais que l'on reprécise en même temps **deltat = 20** lors d'un second appel de ***regul()***, **tol** vaudra toujours **2**, et ne sera en aucun cas recalculé à **20/5 = 4**!

specs.tsd(x)

Description:

> *Récupère et/ou imprime les spécifications pour la décomposition (i.e., les différents paramètres utilisés lors du traitement d'une ou plusieurs séries par **tsd()**).*

Arguments:

- **x:** un objet *'tsd'* obtenu à partir de la fonction ***tsd()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

tsd(), print.tsd(), summary.tsd(), plot.tsd(), extract.tsd()

Exemple:

```
> data(releve)     # Seulement dans R
> rel.reg <- regul(releve$Day, releve$Melosul, xmin=0, n=78,
+ deltat=17, tol=2.125, methods="linear")
> rel.ts <- tseries(rel.reg)
> melo.dec <- tsd(rel.ts, method="average", order=2, times=10,
+ sides=2, ends="fill")
> specs(melo.dec)
  method     type      lag    axes1    axes2    axes3    axes4
 average additive        1        1        2        3        4
   axes5    order    times    sides     ends s.degree t.degree
       5        2       10        2     fill        0        2
  robust    trend
   FALSE    FALSE
```

On peut aussi récupérer des spécifications de paramètres à partir d'un objet *'tsd'* existant et les appliquer à une nouvelle décomposition en précisant:

```
> new.dec <- tsd(rel.ts, specs=specs(melo.dec), times=1)
```

Cela évite de devoir rentrer à nouveau tous les paramètres à chaque régularisation. Les paramètres qui sont reprécisés explicitement à l'appel de ***tsd()*** sont utilisés prioritairement aux valeurs renvoyées par ***specs()***. Dans le cas présent, on a redéfini l'argument **times**, tout en conservant tous les autres comme dans la première décomposition. A noter toutefois que, comme tous les arguments sont évalués lors du premier appel de la fonction ***tsd()***, il n'est pas possible de récupérer des spécifications calculées (voir ***specs.regul()*** pour un exemple).

stat.desc(x, basic = TRUE, desc = TRUE, norm = FALSE, p = 0.95)

Description:

> *Calcule des statistiques descriptives sur une ou plusieurs séries de données.*

Arguments:

- **x:** un data frame (contenant une ou plusieurs colonnes, et donc, une ou plusieurs variables à calculer).

- **basic:** si **TRUE** (par défaut), retourne des informations basiques sur les séries: le nombre de valeurs (nbr.val), le nombre de valeurs nulles (nbr.null), le nombre de valeurs manquantes (nbr.na), la valeur minimale (min), la valeur maximale (max), l'étendue des valeurs (range, soit max-min) et leur somme (sum).

- **desc:** si **TRUE** (par défaut), retourne des descripteurs classiques sur les séries: la médiane (median), la moyenne (mean), erreur standard de la moyenne (SE.mean), intervalle de confiance de la moyenne (CI.mean) au niveau indiqué par l'argument **p**, la variance (var), l'écart type (std.dev) et le coefficient de variation (coef.var) qui se définit comme le rapport de l'écart type à la moyenne.

- **norm:** si **TRUE** (**FALSE** par défaut), retourne des statistiques correspondant à la distribution normale: coefficient d'asymétrie g1 (skewness), critère de significativité (skew.2SE), soit g1/2.SEg1 (si ce paramètre est supérieur à 1, le coefficient d'asymétrie est significatif), coefficient d'aplatissement g2 (kurtosis), critère de significativité (kurt.2SE, même remarque que pour skew.2SE), et sous R, test de normalité de la distribution Shapiro-Wilk, statistique (normtest.W) et probabilité (normtest.p).

- **p:** la probabilité à utiliser pour le calcul de l'intervalle de confiance sur la moyenne (CI.mean). Par défaut: 0.95.

Correspondance dans PASSTEC 2000:

Menu ***Description -> Statistiques*** en mode ***Univarié***. Les items calculés sont légèrement différents. PASSTEC 2000 renvoie le maximum, le minimum, la moyenne, la variance, l'écart type, le nombre de zéros, ainsi que g1, g2 et un test de normalité. Les tests pour g1, g2 et la normalité sont des tests t de Student, et diffèrent donc des tests réalisés ici.

Voir aussi:

stat.pen(), stat.slide()

Exemple:

Voir partie I.

stat.pen(x, basic = FALSE, desc = FALSE)

Description:

> *Calcule des statistiques descriptives sur une ou plusieurs séries de données, dont les estimateurs de Pennington. Par défaut, la fonction retourne la moyenne (pen.mean), la variance (pen.var), l'écart type (pen.std.dev) et la variance de la moyenne (pen.mean.var) selon les estimateurs de Pennington.*

Arguments:

- **x:** un data frame (contenant une ou plusieurs colonnes, et donc, une ou plusieurs variables à calculer).

- **basic:** si **TRUE**, retourne également des informations basiques sur les séries: le nombre de valeurs (nbr.val), le nombre de valeurs nulles (nbr.null), le pourcentage de valeurs nulles (percnull), le nombre de valeurs manquantes (nbr.na). Par défaut, **basic = FALSE**.

- **desc:** si **TRUE**, retourne aussi des descripteurs classiques sur les séries: la médiane (median), la moyenne (mean), la variance (var), l'écart type (std.dev), les mêmes statistiques calculées pour les valeurs positives uniquement (pos.median, pos.mean, pos.var, pos.std.dev) et la moyenne géométrique (geo.mean), c'est à dire, l'exponentielle de la moyenne des valeurs exprimées en logarithme, à l'exclusion des valeurs nulles. Par défaut, **desc = FALSE**.

Correspondance dans PASSTEC 2000:

Menu ***Description -> Estimateurs de Pennington*** en mode ***Univarié***. Dans PASSTEC 2000, on ne peut calculer ces statistiques que pour une variable à la fois, alors que dans PASTECS, on peut sélectionner autant de variables que l'on veut, et effectuer tous les calculs en une seule étape. Les items calculés sont également légèrement différents.

Voir aussi:

pennington(), stat.desc(), stat.slide()

Exemple:

Voir partie I.

stat.slide(x, y, xcut = NULL, xmin = min(x), n = NULL, frequency = NULL, deltat = 1/frequency, basic = FALSE, desc = FALSE, norm = FALSE, pen = FALSE, p = 0.95)

Description:

> *Calcule des statistiques glissantes sur une série de données. Crée un objet de classe 'stat.slide' qui est associé aux méthodes **print()**, **plot()** et **lines()** pour afficher les résultats et en faire une représentation graphique.*

Arguments:

- **x:** un vecteur indiquant le temps correspondant des observations au format décimal.

- **y:** un vecteur contenant la série de données à traiter.

- **xcut:** un vecteur contenant la position des différentes périodes pour le calcul des statistiques glissantes sur des périodes irrégulières. Sa valeur est **NULL** par défaut. Dans ce cas, un vecteur de périodes régulières est créé à partir de **xmin**, **n** et **frequency** ou **deltat**.

- **xmin:** la date initiale à considérer pour la constitution des différentes périodes régulièrement espacées dans le temps. Par défaut, la valeur minimale de **x** est utilisée.

- **n:** le nombre de périodes régulières à calculer. Par défaut, **n = NULL**; le programme va estimer quelle est la meilleur valeur de **n** possible.

- **frequency:** la fréquence des périodes dans l'unité temporelle de **x** (par exemple, **frequency = 4** pour des saisons sous une échelle de temps "**years"**). Par défault, **frequency = NULL**; c'est alors **deltat** qui est utilsé.

- **deltat:** l'intervalle de temps entre les périodes. Précisez soit **frequency**, soit **deltat**. Si les deux sont spécifiés, **frequency** est utilisé en priorité.

- **basic:** si **TRUE** (**FALSE** par défaut), retourne des informations basiques sur les séries: le nombre de valeurs (nbr.val), le nombre de valeurs nulles (nbr.null), le nombre de valeurs manquantes (nbr.na), la valeur minimale (min), la valeur maximale (max), l'étendue des valeurs (range, soit max-min) et leur somme (sum).

- **desc:** si **TRUE** (**FALSE** par défaut), retourne des descripteurs classiques sur les séries: la médiane (median), la moyenne (mean), erreur standard de la moyenne (SE.mean), intervalle de confiance de la moyenne (CI.mean) au niveau indiqué par l'argument **p**, la variance (var), l'écart type (std.dev) et le coefficient de variation (coef.var) qui se définit comme le rapport de l'écart type à la moyenne.

- **norm:** si **TRUE** (**FALSE** par défaut), retourne des statistiques correspondant à la distribution normale: coefficient d'asymétrie g1 (skewness), critère de significativité (skew.2SE), soit g1/2.SEg1 (si ce paramètre est supérieur à 1, le coefficient d'asymétrie est significatif), coefficient d'aplatissement g2 (kurtosis), critère de significativité (kurt.2SE, même remarque que pour skew.2SE), test de normalité de la distribution Shapiro-Wilk, statistique (normtest.W) et probabilité (normtest.p).

- **pen:** si **TRUE** (**FALSE** par défaut), retourne des statistiques de Pennington, ainsi que d'autres statistiques associées: pos.median, pos.mean, pos.var, pos.std.dev, respectivement la médiane, la moyenne, l'écart type et la variance en ne considérant que les valeurs positives (non nulles), geo.mean, la moyenne

géométrique, c'est à dire, l'exponentielle de la moyenne des valeurs exprimées en logarithme, à l'exclusion des valeurs nulles. pen.mean, pen.var, pen.std.dev, pen.mean.var, respectivement la moyenne, la variance, l'écart type et la variance de la moyenne selon les estimateurs de Pennington.

- **p:** la probabilité à utiliser pour le calcul de l'intervalle de confiance sur la moyenne (CI.mean). Par défaut, **p = 0.95**.

Correspondance dans PASSTEC 2000:

Menu ***Description -> Statistiques glissantes*** en mode ***Univarié***. PASSTEC 2000 renvoie uniquement le maximum, le minimum, la moyenne, la variance, l'écart type et le nombre de zéros.

Voir aussi:

print.stat.slide(), plot.stat.slide(), lines.stat.slide(), stat.desc(), stat.pen()

Exemple:

Voir partie I.

summary.abund(x)

Description:

> *Méthode **summary()** des objets de classe 'abund'. Affiche les résultats d'un tri par ordre d'abondance.*

Arguments:

- **x:** un objet de classe *'abund'* issu d'une analyse lancée par la fonction ***abund()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

abund(), print.abund(), plot.abund(), lines.abund(), identify.abund(), extract.abund()

Exemple:

Voir partie I.

summary.escouf(x)

Description:

> *Méthode **summary()** des objets de classe 'escouf'. Affiche les résultats d'une analyse selon la méthode d'Escoufier.*

Arguments:

- **x:** un objet de classe '*escouf*' issu d'une analyse lancée par la fonction ***escouf()***.

Correspondance dans PASSTEC 2000:

Peut être assimilé à l'option '**Sorties statistiques**' dans la boite de dialogue **frmEscoufier** de PASSTEC 2000.

Voir aussi:

escouf(), print.escouf(), plot.escouf(), lines.escouf(), identify.escouf(), extract.escouf()

Exemple:

Voir partie I.

summary.regul(x)

Description:

> *Imprime un résumé détaillé de la régularisation obtenue par **regul()**. Ce résumé correspond aux informations fournies par **print.regul()** suivies du tableau des valeurs interpolées pour les différentes séries traitées.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), print.regul(), plot.regul(), lines.regul(), identify.regul(), hist.regul()

Exemple:

Voir partie I.

summary.tsd(x)

Description:

> *Imprime un résumé détaillé de la décomposition obtenue par **tsd()**. En plus des informations renvoyées par la méthode **print()**, la méthode **summary()** retourne aussi les composantes de la série s'il n'y en a qu'une, ou les 10 premières lignes de chaque composante de chaque série, s'il y a plusieurs séries.*

Arguments:

- **x:** un objet *'tsd'* obtenu à partir de la fonction ***tsd()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

tsd(), print.tsd(), specs.tsd(), plot.tsd(), extract.tsd()

Exemple:

Voir partie I.

summary.turnogram(x)

Description:

> *Imprime un résumé détaillé du calcul du tournogramme par **turnogram()**. En plus des informations renvoyées par la méthode **print()**, la méthode **summary()** retourne aussi le nombre d'observations et d'extremums (moyen, mininum et maximum dans le cas d'un tournogramme complet).*

Arguments:

- **x:** un objet *'turnogram'* obtenu à partir de la fonction ***turnogram()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

turnogram(), print.turnogram(), plot.turnogram(), identify.turnogram(), extract.turnogram(), pgleissberg()

Exemple:

Voir partie I.

summary.turnpoints(x)

Description:

> *Imprime un résumé détaillé du calcul des points de retournement obtenu par **turnpoints()**. En plus des informations renvoyées par la méthode **print()**, la méthode **summary()** retourne aussi la liste de ces points de retournement, leur position, s'il s'agit d'un pic ou d'un creux, ainsi que la probabilité et la quantité d'information qui leurs sont associées.*

Arguments:

- **x:** un objet *'turnpoints'* obtenu à partir de la fonction ***turnpoints()***.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

turnpoints(), print.turnpoints(), plot.turnpoints(), lines.turnpoints(), extract.turnpoints()

Exemple:

Voir partie I.

trend.test(tseries, R = 1, scramble=TRUE)

Description:

> *Teste si la série possède une tendance croissante ou décroissante à l'aide d'un test de corrélation non paramétrique de Spearman entre les observations et le temps.*

Arguments:

- **tseries:** une série régulière *'rts'* sous S+ ou *'ts'* sous R.
- **R:** le nombre de fois que le test est effectué. Si **R = 1** (par défaut), un test classique est réalisé. Si **R > 1**, le test est bootstrapé.
- **scramble:** utilisé lorsque R > 1. Si scramble=TRUE (par défaut), alors les observations et le temps sont mélangés, c'est-à-dire que les observations sont attribuées de manière aléatoire à des temps différents. Si scramble=FALSE, alors les paires (temps, observation) sont conservées et le bootstrap échantillonne les paires de manière aléatoire avec remise (ex-aequos permis). Notez que lorsque scramble=TRUE, une erreur peut être introduite en cas de forte correlation entre les observations, ce qui est souvent le cas des séries spatio-temporelles par définition! Dans ce dernier cas, la probabilité associée avec le test de bootstrap est estimée par le rapport du nombre de fois que la statistique bootstrapée est inférieure (pour une corrélation négative) ou supérieure (pour une corrélation positive) à la statistique initialement observée, divisé par R+1. Lorsque scramble=TRUE, la significativité du test est déterminée en observant la distribution des statistiques bootstrapées: si cette distribution inclus zéro, la corrélation n'est pas significative.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, cette fonction correspond à ***Tendance -> Test d'Existence*** en mode ***Univarié***.

Voir aussi:

local.trend()

Exemples:

Voir partie I.

tsd(x, specs = NULL, method = "loess", type, …)

Description:

> *Effectue la décomposition d'une ou plusieurs séries spatio-temporelles régulières. Toutes les méthodes implémentées dans PASTECS sont disponibles (argument **method**) à partir de cette fonction. Un objet 'tsd' est créé. Il possède les méthodes suivantes: **print()**, **summary()**, **specs()**, **plot()** et **extract()**.*

Arguments:

- **x:** un objet *'rts'* sous S+ ou *'ts'* sous R contenant une ou plusieurs séries régulières.

- **specs:** un objet *'specs.tsd'* obtenu en appliquant la méthode ***specs()*** à un objet *'tsd'*. Récupère toutes les spécifications du traitement qui a servi à générer cet objet et les appliquent au traitement en cours. Si un argument est fourni explicitement en plus de **specs**, celui-ci sera prioritaire par rapport à la valeur trouvée dans **specs**. Par défaut: **NULL**, ce qui signifie qu'aucune spécification ne doit être appliquée. La fonction utilise les arguments fournis, ou s'ils manquent, leurs valeurs par défaut.

- **method:** la méthode à utiliser pour réaliser la décomposition. Les valeurs possibles sont: **"diff"**, **"average"**, **"median"**, **"evf"**, **"reg"** **"loess"** (par défaut) ou **"census"** seulement sous R**)**. La fonction ***decxxx()*** équivalente est appliquée tour à tour à chaque série présente dans **x**.

- **type:** le type de modèle: soit **"additive"** pour un modèle additif (valeur par défaut), soit **"multiplicative"** pour un modèle multiplicatif. Notez que seul le modèle additif est permis pour la méthode LOESS, et seul le modèle multiplicatif est accepté pour la méthode CENSUS II.

- **…:** arguments à fournir à la fonction ***decxxx()*** où ***xxx*** est le nom de la méthode fournit à l'argument **method** ci-dessus. Voir les différentes fonctions ***decxxx()*** pour connaître les arguments à utiliser, ainsi que leurs significations.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, cette fonction effectue l'équivalent de plusieurs procédures: dans le menu ***Filtrage***: ***Census***, ***Elimination de tendance ou de cycle***, ***Filtrage par les vecteurs propres*** et dans le menu ***Tendance***: ***estimation statistique*** en mode ***Univarié.***

Voir aussi:

print.tsd(), summary.tsd(), specs.tsd(), plot.tsd(), extract.tsd(), tseries(), decdiff(), decaverage(), decmedian(), decevf(), decreg(), decloess(), deccensus()

Exemple:

Voir partie I.

tseries(x)

Description:

> *Convertit toutes les séries présentes dans un objet 'regul' ou 'tsd' en 'time series' régulière(s) et les placent dans un objet 'rts' sous S+, ou 'ts' sous R. Toutes les méthodes des objets 'rts' ou 'ts' sont alors disponibles pour traiter ces données.*

Arguments:

- **x:** un objet *'regul'* obtenu à partir de la fonction ***regul()*** ou *'tsd'* obtenu à l'aide de la fonction ***tsd()*** ou de toute autre fonction qui renvoient des objets similaires.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

regul(), extract.regul(), tsd(), extract.tsd(), is.tseries()

Exemple:

Voir partie I.

turnogram(series, intervals = c(1, length(series)/5), step = 1, complete = FALSE, two.tailed = TRUE, FUN = mean, plotit = TRUE, level = 0.05, lhorz=TRUE, lvert = FALSE, xlog = TRUE)

Description:

> *Calcule un tournogramme soit simple (**complete** = **FALSE**) soit complet (**complete** = **TRUE**), c'est à dire, soit pour les intervalles à partir de la 1ère observation seulement, soit à partir de tous les intervalles possibles d'une série temporelle régulière. L'indice de monotonie peut être calculé par rapport à un test unilatéral à gauche (**two.tailed** = **FALSE**) ou bilatéral (**two.tailed** = **TRUE**). Enfin, la fonction qui sert à aggréger les données dans les intervalles peut être choisie librement.*

Arguments:

- **series:** un objet *'rts'* sous S+ ou *'ts'* sous R contenant une seule série régulière.
- **intervals:** l'étendue des intervalles à calculer (valeur minimale et maximale). Par défaut, l'intervalle minimal est **1** (la série originelle), et l'intervalle maximum est **n/5**.
- **step:** l'incrément à utiliser pour calculer les différents intervalles. Par défaut, un incrément de **1** est utilisé. Si l'étendue des intervalles est grande, on peut réduire les temps de calcul en choisissant un incrément supérieur (2, 3, 4,…).
- **complete:** si **TRUE**, un tournogramme complet est calculé, avec les courbes moyennes, minimales et maximales. Si **FALSE** (par défaut), un tournogramme simple (commençant toujours à la première observation) est calculé.
- **two.tailed:** si **TRUE** (par défaut), l'indice de monotonie associé à un test bilatéral est calculé, sinon (**FALSE**), un test unilatéral à gauche est utilisé.
- **FUN:** une fonction à utiliser pour aggréger les données dans les intervalles. Il s'agit d'une fonction du type **FUN(x, na.rm,…)**. La fonction d'aggrégation la plus utilisée est la moyenne (**mean**), mais il est également possible de ne prendre que la première valeur sur l'intervalle (**first**), la dernière valeur (**last**), la valeur médiane (**median**) ou la somme des valeurs (**sum**). Cette dernière est utile pour les données cumulatives, tel que la pluviométrie. A noter que le tournagramme avec **FUN = mean** et **FUN = sum** est le même, mais que l'extraction finale de la série la plus informative est différente pour un intervalle supérieur à 1.
- **plotit:** si **TRUE** (par défaut), le graphe du tournogramme est également tracé.
- **level:** le seuil de significativité à indiquer sur le graphe. Par défaut, **level = 0.05** (p = 5%).
- **lhorz:** si une ligne (test unilatéral) ou deux lignes (test bilatéral) doivent être dessinées sur le graphe pour indiquer le niveau de significativité du test. Par défaut, **lhorz = TRUE**.
- **lvert:** si une ligne verticale doit-elle être indiquée sur le graphe du tournogramme, montrant l'intervalle correspondant au maximum d'information calculé automatiquement. Par défaut, cette ligne n'est pas tracée (**lvert = FALSE**).
- **xlog:** si **TRUE** (par défaut), l'échelle des abscisses sur le graphe du tournogramme, c'est-à-dire la succession des intervalles testés, est exprimée en logarithmes.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, il s'agit de ***Description***: ***Tournogramme*** en mode ***Univarié***. PASSTEC 2000 ne calcule que le tournogramme simple, avec aggrégation par la moyenne et le test bilatéral. PASTECS offre donc plus de possibilités. A noter que le calcul et le test sur les longueurs de phases ne sont pas encore implémentés dans PASTECS.

Voir aussi:

print.turnogram(), summary.turnogram(), plot.turnogram(), identify.turnogram(), extract.turnogram(), pgleissberg(), turnpoints()

Exemple:

Voir partie I.

turnpoints(x)

Description:

> *Calcule le nombre et la position des extremums (points de retournement) dans une série, ainsi que la quantité d'information qui leur est associée. Crée un objet 'turnpoints' qui possède les méthodes* ***print()****,* ***summary()****,* ***plot()****,* ***lines()*** *et* ***extract()****.*

Arguments:

- **x:** un vecteur ou une série temporelle régulière unique du type *'ts'* (sous R) ou *'rts'* (sous S+).

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, il s'agit de ***Description***: ***Points de Retournement*** en mode ***Univarié***.

Voir aussi:

print.turnpoints(), summary.turnpoints(), plot.turnpoints(), lines.turnpoints(), extract.turnpoints(), turnogram(), stat.slide()

Exemple:

Voir partie I.

vario(x, max.dist = length(x)/3, plotit = TRUE, vario.data = NULL)

Description:

> *Calcule et trace le semi-variogramme pour un vecteur ou une série spatio-temporelle régulière.*

Arguments:

- **x:** un vecteur ou une série spatio-temporelle univariée.
- **max.dist:** la distance maximale à calculer. Par défaut, il s'agit du tiers du nombre d'observations (c'est-à-dire, du nombre de lignes dans la matrice).
- **plotit:** si **TRUE** (par défaut), le graphe du semi-variogramme est tracé.
- **vario.data:** des données issues d'un appel précédent de la fonction, fournies afin d'en tracer le graphe. Par défaut, **NULL**, les données du semi-variogramme sont (re)calculées.

Correspondance dans PASSTEC 2000:

Dans PASSTEC 2000, cette fonction est accessible par le menu ***Description -> Variogramme*** en mode ***Univarié***. A noter que PASSTEC 2000 calcule le variogramme, qui est simplement le double du semi-variogramme calculé ici.

Voir aussi:

disto()

Exemple:

Voir partie I.

yearstodays(x, xmin = NULL)

Description:

> *Convertit les données temporelles contenues dans* **x** *de l'échelle* **"years"** *(une unité = un an) à l'échelle* **"days"** *(une unité = un jour). Vois aussi le chapitre régularisation pour plus de détails sur la représentation du temps dans S+ et R.*

Arguments:

- **x:** un vecteur représentant le temps à l'échelle **"years"**.
- **xmin:** un nombre indiquant quelle doit être la valeur minimale du vecteur temps à l'échelle **"days"** obtenue. Par défaut, vaut **NULL** auquel cas la valeur du jour le plus petit est calculée d'après la date minimale dans **x** à l'échelle **"years"**.

Correspondance dans PASSTEC 2000:

Aucune.

Voir aussi:

daystoyears(), strip.Time.Date(), copy.Time.Date().

Exemple:

L'instruction suivante change l'échelle temporelle de **"days"** en **"years"**, et puis inversément pour la colonne ***Day*** du jeu de données ***releve***:

```
> data(releve)     # Seulement sous R
> # Transforme de "days" en "years"
> rel.years <- daystoyears(releve$Day, datemin = "10/05/2000")
> # Retransforme en unité "days" en partant de 1
> rel.days <- yearstodays(rel.years, xmin = 1)
> rel.days

 [1]    1   51  108  163  176  206  248  315  339  356  389
[12]  449  480  493  501  508  522  554  568  597  613  624
[23]  639  676  697  723  751  786  814  842  863  877  891
[34]  906  922  940  954  971  983  999 1010 1027 1038 1054
[45] 1069 1081 1094 1110 1129 1143 1156 1173 1186 1207 1226
[56] 1235 1249 1271 1290 1314 1325
```

Annexe

Ceci est une traduction libre en français de la "GNU General Public License" originale sous laquelle la librairie PASTECS est distribuée. Il va sans dire que seule la version officielle en anglais est légalement valable.

License Publique Générale GNU

Version 2, juin 1991

Préambule

Les licences relatives à la plupart des logiciels sont destinées à supprimer votre liberté de les partager et de les modifier. Par contraste, la licence publique générale GNU General Public License veut garantir votre liberté de partager et de modifier les logiciels libres, pour qu'ils soient vraiment libres pour tous leurs utilisateurs. La présente licence publique générale s'applique à la plupart des logiciels de la Free Software Foundation, ainsi qu'à tout autre programme dont les auteurs s'engagent à l'utiliser. (Certains autres logiciels sont couverts par la Licence Publique Générale pour Bibliothèques GNU à la place). Vous pouvez aussi l'appliquer à vos programmes.

Quand nous parlons de logiciels libres, nous parlons de liberté, non de gratuité. Nos licences publiques générales veulent vous garantir:

- que vous avez toute liberté de distribuer des copies des logiciels libres (et de facturer ce service, si vous le souhaitez);
- que vous recevez les codes sources ou pouvez les obtenir si vous le souhaitez;
- que vous pouvez modifier les logiciels ou en utiliser des éléments dans de nouveaux programmes libres;
- et que vous savez que vous pouvez le faire.

Pour protéger vos droits, nous devons apporter des restrictions, qui vont interdire à quiconque de vous dénier ces droits, ou de vous demander de vous en désister. Ces restrictions se traduisent par certaines responsabilités pour ce qui vous concerne, si vous distribuez des copies de logiciels, ou si vous les modifiez.

Par exemple, si vous distribuez des copies d'un tel programme, gratuitement ou contre une rémunération, vous devez transférer aux destinataires tous les droits dont vous disposez. Vous devez vous garantir qu'eux-mêmes, par ailleurs, reçoivent ou peuvent recevoir le code source. Et vous devez leur montrer les présentes dispositions, de façon qu'ils connaissent leurs droits.

Nous protégeons vos droits en deux étapes :

1. Nous assurons le droit d'auteur (copyright) du logiciel, et

2. Nous vous proposons cette licence, qui vous donne l'autorisation légale de dupliquer, distribuer et/ou modifier le logiciel.

De même, pour la protection de chacun des auteurs, et pour notre propre protection, nous souhaitons nous assurer que tout le monde comprenne qu'il n'y a aucune garantie portant sur ce logiciel libre. Si le logiciel est modifié par quelqu'un d'autre puis transmis à des tiers, nous souhaitons que les destinataires sachent que ce qu'ils possèdent n'est pas l'original, de façon que tous problèmes introduits par d'autres ne se traduisent pas par une répercussion négative sur la réputation de l'auteur original.

Enfin, tout programme libre est en permanence menacé par des brevets de logiciels. Nous souhaitons éviter le danger que des sous-distributeurs d'un programme libre obtiennent à titre individuel des licences de brevets, avec comme conséquence qu'ils ont un droit de propriété sur le programme. Pour éviter cette situation, nous avons fait tout ce qui est nécessaire pour que tous brevets doivent faire l'objet d'une concession de licence qui en permette l'utilisation libre par quiconque, ou bien qu'il ne soit pas concédé du tout.

Nous présentons ci-dessous les clauses et dispositions concernant la duplication, la distribution et la modification.

Conditions d'exploitation portant sur la duplication, la distribution et la modification

0. Le présent contrat de licence s'applique à tout programme ou autre ouvrage contenant un avis, apposé par le détenteur du droit de propriété, disant qu'il peut être distribué au titre des dispositions de la présente Licence Publique Générale. Ci-après, le "programme" désigne l'un quelconque de ces programmes ou ouvrages, et un "ouvrage fondé sur le programme" désigne soit le programme, soit un ouvrage qui en dérive au titre de la loi sur le droit d'auteur; plus précisément, il s'agira d'un ouvrage contenant le programme ou une version de ce dernier, soit mot à mot, soit avec des modifications et/ou traduit en une autre langue (ci-après, le terme "modification" englobe, sans aucune limitation, les traductions qui en sont faites). Chaque titulaire de licence sera appelé "ayant droit".

Les activités autres que la duplication, la distribution et la modification ne sont pas couvertes par la présente licence; elles n'entrent pas dans le cadre de cette dernière. L'exécution du programme n'est soumise à aucune restriction, et les résultats du programme ne sont couverts que si son contenu constitue un ouvrage fondé sur le programme (indépendamment du fait qu'il a été réalisé par exécution du programme). La véracité de ce qui précède dépend de ce que fait le programme.

1. L’ayant droit peut dupliquer et distribuer des copies mot à mot du code source du programme tel qu'il les reçoit, et ce sur un support quelconque, du moment qu'il appose, d'une manière parfaitement visible et appropriée, sur chaque exemplaire, un avis approprié de droits d'auteur (copyright) et de renonciation à garantie; qu'il maintient intacts tous les avis qui se rapportent à la présente licence et à l'absence de toute garantie; et qu'il transmet à tout destinataire du programme un exemplaire de la présente licence en même temps que le programme.

L’ayant droit peut facturer l'acte physique de transfert d'un exemplaire, et il peut, à sa discrétion, proposer en échange d'une rémunération une protection en garantie.

2. L’ayant droit peut modifier son ou ses exemplaires du programme ou de toute portion de ce dernier, en formant ainsi un ouvrage fondé sur le programme, et dupliquer et distribuer ces modifications ou cet ouvrage selon les dispositions de la section 1 ci-dessus, du moment qu’il satisfait aussi à toutes ces conditions:

- A. L’ayant droit doit faire en sorte que les fichiers modifiés portent un avis, parfaitement visible, disant que l’ayant droit a modifié les fichiers, avec la date de tout changement.

- B. L’ayant droit doit faire en sorte que tout ouvrage qu'il distribue ou publie, et qui, en totalité ou en partie, contient le programme ou une partie quelconque de ce dernier ou en dérive, soit concédé en bloc, à titre gracieux, à tous tiers au titre des dispositions de la présente licence.

- C. Si le programme modifié lit normalement des instructions interactives lors de son exécution, l’ayant droit doit, quand il commence l'exécution du programme pour une telle utilisation interactive de la manière la plus usuelle, faire en sorte que ce programme imprime ou affiche une annonce, comprenant un avis approprié de droits d'auteur, et un avis selon lequel il n'y a aucune garantie (ou autrement, que l’ayant droit fournit une garantie), et que les utilisateurs peuvent redistribuer le programme au titre de ces dispositions, et disant à l'utilisateur comment visualiser une copie de cette licence (exception: si le programme par lui-même est interactif mais n'imprime normalement pas une telle annonce, l'ouvrage du concessionnaire se fondant sur le programme n'a pas besoin d'imprimer une annonce).

Les exigences ci-dessus s'appliquent à l'ouvrage modifié pris en bloc. Si des sections identifiables de cet ouvrage ne dérivent pas du programme et peuvent être considérées raisonnablement comme représentant des ouvrages indépendants et distincts par eux-mêmes, alors la présente licence, et ses dispositions, ne s'appliquent pas à ces sections quand l’ayant droit les distribue sous forme d'ouvrages distincts. Mais quand l’ayant droit distribue ces mêmes sections en tant qu'élément d'un tout qui représente un ouvrage se fondant sur le programme, la distribution de ce tout doit se faire conformément aux dispositions de la présente licence, dont les autorisations, portant sur d'autres ayant droits, s'étendent à la totalité dont il est question, et ainsi à chacune de ces parties, indépendamment de celui qu'il a écrite.

Ainsi, cette section n'a pas pour but de revendiquer des droits ou de contester vos droits sur un ouvrage entièrement écrit par l’ayant droit; bien plus, l'intention est d'exercer le droit de surveiller la distribution d'ouvrages dérivée ou collective se fondant sur le programme.

De plus, un simple assemblage d'un autre ouvrage ne se fondant pas sur le programme, avec le programme (ou avec un ouvrage se fondant sur le programme) sur un volume d'un support de stockage ou distribution, ne fait pas entrer l'autre ouvrage dans le cadre de la présente licence.

3. L’ayant droit peut dupliquer et distribuer le programme (ou un ouvrage se fondant sur ce dernier, au titre de la Section 2), en code objet ou sous une forme exécutable, au titre des dispositions des Sections 1 et 2 ci-dessus, du moment que l’ayant droit effectue aussi l'une des opérations suivantes :

- A. Lui joindre le code source complet correspondant, exploitable par une machine, code qui doit être distribué au titre des Sections 1 et 2 ci-dessus sur un support couramment utilisé pour l'échange de logiciels; ou bien

- B. Lui joindre une offre écrite, dont la validité se prolonge pendant au moins 3 ans, de transmettre à un tiers quelconque, pour un montant non supérieur au coût pour l’ayant droit, de réalisation physique de la distribution de la source, un exemplaire complet, exploitable par une machine, du code source correspondant, qui devra être distribué au titre des dispositions des Sections 1 et 2 ci-dessus sur un support couramment utilisé pour l'échange des logiciels; ou bien

- C. Lui joindre les informations que l’ayant droit a reçues, pour proposer une distribution du code source correspondant (cette variante n'est autorisée que pour la distribution non commerciale, et seulement si l’ayant droit a reçu le programme sous forme exécutable ou sous forme d'un code objet, avec une telle offre, conformément à l'alinéa B) ci-dessus).

Le code source d'un ouvrage représente la forme préférée de l'ouvrage pour y effectuer des modifications. Pour un ouvrage exécutable, le code source complet représente la totalité du code source pour tous les modules qu'il contient, plus tous fichiers de définitions d'interface associés, plus les informations en code machine pour commander la compilation et l'installation du programme exécutable. Cependant, à titre d'exceptions spéciales, le code source distribué n'a pas besoin de comprendre quoi que ce soit qui est normalement distribué (sous forme source ou sous forme binaire) avec les composants principaux (compilateur, noyau de système d'exploitation, etc.) du système d'exploitation sur lequel est exécuté le programme exécutable, à moins que le composant, par lui-même, soit joint au programme exécutable.

Si la distribution de l‘exécutable ou du code objet est réalisée de telle sorte qu'elle offre d'accéder à une copie à partir d'un lieu désigné, alors le fait d'offrir un accès équivalent à la duplication du code source à partir de ce même lieu s'entend comme distribution du code source, même si des tiers ne sont pas contraints de dupliquer la source en même temps que le code objet.

4. L’ayant droit ne peut dupliquer, modifier, concéder en sous-licence ou distribuer le programme, sauf si cela est expressément prévu par les dispositions de la présente licence. Toute tentative pour autrement dupliquer, modifier, concéder en sous-licence ou distribuer le programme est interdite, et met automatiquement fin aux droits de l’ayant droit au titre de la présente licence. Cependant, les parties qui ont reçu des copies, ou des droits, de la part de l’ayant droit au titre de la présente licence, ne verront pas expirer leur contrat de licence, tant que ces parties agissent d'une manière parfaitement conforme.

5. Il n'est pas exigé de l’ayant droit qu'il accepte la présente licence, car il ne l'a pas signée. Cependant, rien d'autre n'octroie à l’ayant droit l'autorisation de modifier ou de distribuer le programme ou ses ouvrages dérivés. Ces actions sont interdites par la loi si l’ayant droit n'accepte pas la présente licence. En conséquence, par le fait de modifier ou de distribuer le programme (ou un ouvrage quelconque se fondant sur le programme), l’ayant droit indique qu'il accepte la présente licence, et qu'il a la volonté de se conformer à toutes les closes et dispositions concernant la duplication, la distribution ou la modification du programme ou d'ouvrages se fondant sur ce dernier.

6. Chaque fois que l’ayant droit redistribue le programme (ou tout ouvrage se fondant sur le programme), le destinataire reçoit automatiquement une licence de l'émetteur initial de la licence, pour dupliquer, distribuer ou modifier le programme, sous réserve des présentes closes et dispositions. L’ayant droit ne peut imposer aucune restriction plus poussée sur l'exercice, par le destinataire, des droits octroyés au titre des présentes. L’ayant droit n'a pas pour responsabilité d'exiger que des tiers se conforment à la présente licence.

7. Si, en conséquence une décision de justice ou une allégation d'infraction au droit des brevets, ou pour toute autre raison (qui n'est pas limitée à des problèmes de propriétés industrielles), des conditions sont imposées à l’ayant droit (par autorité de justice, par convention ou autrement), qui entrent en contradiction avec les dispositions de la présente licence, elles n'exemptent pas l’ayant droit de respecter les dispositions de la présente licence. Si l’ayant droit ne peut procéder à la distribution de façon à satisfaire simultanément à ces obligations au titre de la présente licence et à toutes autres obligations pertinentes, alors, en conséquence de ce qui précède, l’ayant droit ne peut pas procéder du tout à la distribution du programme. Par exemple, si une licence de brevet ne permettait pas une redistribution du programme, sans redevances, par tous ceux qui reçoivent des copies directement ou indirectement par l'intermédiaire de l’ayant droit, alors le seul moyen par lequel l’ayant droit pourrait satisfaire tant à cette licence de brevet qu'à la présente licence, consisterait à s'abstenir complètement de distribuer le programme.

Si une partie quelconque de cette section est considérée comme nulle ou non exécutoire dans certaines circonstances particulières, le reste de cette section reste applicable, et la

section dans son ensemble est considérée comme s'appliquant dans les autres circonstances.

La présente section n'a pas pour objet de pousser l'ayant droit à enfreindre tous brevets ou autres revendications à droit de propriété, ou encore à contester la validité d'une ou plusieurs quelconques de ces revendications; la présente section a pour objet unique de protéger l'intégrité du système de distribution des logiciels libres, système qui est mis en oeuvre par les pratiques liées aux licences publiques. De nombreuses personnes ont apporté une forte contribution à la gamme étendue des logiciels distribués par ce système, en comptant sur l'application systématique de ce système; c'est à l'auteur/donateur de décider s'il a la volonté de distribuer le logiciel par un quelconque autre système, et un ayant droit ne peut imposer ce choix.

La présente section veut rendre parfaitement claire ce que l'on pense être une conséquence du reste de la présente licence.

8. Si la distribution et/ou l'utilisation du Programme est restreinte dans certains pays, sous l'effet de brevets ou d'interfaces présentant un droit d'auteur, le détenteur du droit d'auteur original, qui soumet le Programme aux dispositions de la présente licence, pourra ajouter une limitation expresse de distribution géographique excluant ces pays, de façon que la distribution ne soit autorisée que dans les pays ou parmi les pays qui ne sont pas ainsi exclus. Dans ce cas, la limitation fait partie intégrante de la présente licence, comme si elle était écrite dans le corps de la présente licence.

9. La Free Software Foundation peut, de temps à autre, publier des versions révisées et/ou nouvelles du General Public License. Ces nouvelles versions seront analogues, du point de vue de leur esprit, à la présente version, mais pourront en différer dans le détail, pour résoudre de nouveaux problèmes ou de nouvelles situations.

Chaque version reçoit un numéro de version qui lui est propre. Si le programme spécifie un numéro de version de la présente licence, qui s'applique à ce dernier et "à toute autre version ultérieure", l'ayant droit a le choix de respecter les closes et dispositions de cette version, ou une quelconque version ultérieure publiée par la Free Software Foundation. Si le programme ne spécifie pas de numéro de version de la présente licence, l'ayant droit pourra choisir une version quelconque publiée à tout moment par la Free Software Foundation.

10. Si l'ayant droit souhaite incorporer des parties du programme dans d'autres programmes libres dont les conditions de distribution sont différentes, il devrait écrire à l'auteur pour demander son autorisation. Pour un logiciel soumis à droit d'auteur par la Free Software Foundation, il devra écrire à la Free Software Foundation; nous faisons quelquefois des exceptions à cette règle. Notre décision va être guidée par le double objectif de protéger le statut libre de tous les dérivés de nos logiciels libres, et de favoriser le partage et la réutilisation des logiciels en général.

Absence de garantie

11. Comme la licence du programme est concédée à titre gratuit, il n'y a aucune garantie s'appliquant au programme, dans la mesure autorisée par la loi en vigueur. Sauf mention contraire écrite, les détenteurs du droit d'auteur et/ou les autres parties mettent le programme à disposition "en l'état", sans aucune garantie de quelque nature que ce soit, expresse ou implicite, y compris, mais sans limitation, les garanties implicites de commercialisation et de l'aptitude à un objet particulier. C'est l'ayant droit qui prend la totalité du risque quant à la qualité et aux performances du programme. Si le programme se révélait défectueux, c'est l'ayant droit qui prendrait à sa charge le coût de l'ensemble des opérations nécessaires d'entretien, réparation ou correction.

12. En aucun cas, sauf si la loi en vigueur l'exige ou si une convention écrite existe à ce sujet, aucun détenteur de droit d'auteur, ou aucune partie ayant le pouvoir de modifier

et/ou de redistribuer le programme conformément aux autorisations ci-dessus, n'est responsable vis-à-vis de l'ayant droit pour ce qui est des dommages, y compris tous dommages généraux, spéciaux, accidentels ou indirects, résultant de l'utilisation du programme ou de l'impossibilité d'utiliser le programme (y compris, mais sans limitation, la perte de données, ou le fait que des données sont rendues imprécises, ou encore les pertes éprouvées par l'ayant droit ou par des tiers, ou encore un manquement du programme à fonctionner avec tout autre programme), même si ce détenteur ou cette autre partie a été avisé de la possibilité de tels dommages.

Index

Les fonctions sont présentées en gras et italique suivies de parenthèses –***function()***– dans les entrées d'index. Les numéros de pages en italique renvoient vers les exemples d'utilisation de ces fonctions (au début de l'exemple… la fonction est parfois utilisée plus loin), tandis que les numéros de page en gras renvoient vers la définition détaillée de ces mêmes fonctions. Les méthodes sont présentées comme les fonctions, via leur descriptif complet. Ainsi, la méthode ***extract()*** de l'objet *'abund'* se retrouvera sous l'entrée d'index –***extract.abund()***–. Lorsqu'une méthode est référencée en tant que telle, seul son intitulé apparaît (par exemple, –***extract()***–), et il est précisé 'méthode' dans l'entrée d'index.

Les noms de classes d'objets sont présentés en italique –*classe*–, et il est précisé 'classe' dans les entrées d'index. Les arguments de fonction répertoriés (seulement quelques arguments clefs sont repris), sont présentés en gras –**argument**–, et il est précisé 'argument' dans l'entrée d'index. La même présentation est utilisée pour les attributs –**attribut**–, mais il est précisé 'attribut' dans l'entrée d'index. Enfin les mots clefs dans le texte sont présentés en caractères normaux.