Package ‘CoSMoS’

October 12, 2022

Type Package

Title Complete Stochastic Modelling Solution

Version 2.1.0

Date 2021-05-20

Description
Makes univariate, multivariate, or random fields simulations precise and simple. Just select the desired time series or random fields’ properties and it will do the rest. CoSMoS is based on the framework described in Papalexiou (2018, <doi:10.1016/j.advwatres.2018.02.013>), extended for random fields in Papalexiou and Serinaldi (2020, <doi:10.1029/2019WR026331>), and further advanced in Papalexiou et al. (2021, <doi:10.1029/2020WR029466>) to allow fine-scale space-time simulation of storms (or even cyclone-mimicking fields).

Depends R (>= 3.5.0), ggplot2, data.table

Imports utils, methods, stats, grDevices, nloptr, MBA, Matrix, mAr,
matrixcalc, mvtnorm, cowplot, directlabels, animation,
ggquiver, pracma, plot3D

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

Author Simon Michael Papalexiou [aut],
Francesco Serinaldi [aut],
Filip Strnad [aut],
Yannis Markonis [aut],
Kevin Shook [ctb, cre]

Maintainer Kevin Shook <`kevin.shook@usask.ca`>

License GPL-3

URL `https://github.com/TycheLab/CoSMoS`

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-29 23:20:08 UTC

R topics documented:

`CoSMoS-package` *CoSMoS: Complete Stochastic Modelling Solution*

Description

CoSMoS is an R package that makes time series generation with desired properties easy. Just choose the characteristics of the time series you want to generate, and it will do the rest.

Details

The generated time series preserve any probability distribution and any linear autocorrelation structure. Users can generate as many and as long time series from processes such as precipitation, wind, temperature, relative humidity etc. It is based on a framework that unified, extended, and improved a modelling strategy that generates time series by transforming "parent" Gaussian time series having specific characteristics (Papalexiou, 2018).

Funding

The package was partly funded by the Global institute for Water Security (GIWS; https://water.usask.ca/) and the Global Water Futures (GWF; https://gwf.usask.ca/) program.

Author(s)

Coded by: Filip Strnad `<strnadf@fzp.czu.cz>` and Francesco Serinaldi `<francesco.serinaldi@ncl.ac.uk>`

Conceptual design by: Simon Michael Papalexiou `<sm.papalexiou@usask.ca>`

Tested and documented by: Yannis Markonis `<markonis@fzp.czu.cz>`

Maintained by: Kevin Shook `<kevin.shook@usask.ca>`

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources 115, 234-252, doi: 10.1016/j.advwatres.2018.02.013

Papalexiou, S.M., Markonis, Y., Lombardo, F., AghaKouchak, A., Foufoula-Georgiou, E. (2018). Precise Temporal Disaggregation Preserving Marginals and Correlations (DiPMaC) for Stationary and Nonstationary Processes. Water Resources Research, 54(10), 7435-7458, doi: 10.1029/2018WR022726

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi: 10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

`acs` *AutoCorrelation Structure*

Description

Provides a parametric function that describes the values of the linear autocorrelation up to desired lags. For more details on the parametric autocorrelation structures see section 3.2 in Papalexiou (2018).

Usage

```
acs(id, ...)
```

Arguments

`id`	autocorrelation structure id
`...`	other arguments (t as lag and acs parameters)

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi: 10.1016/j.advwatres.2018.02.013

Examples

```
library(CoSMoS)

## specify lag
t <- 0:10

## get the ACS
f <- acs('fgn', t = t, H = .75)
b <- acs('burrXII', t = t, scale = 1, shape1 = .6, shape2 = .4)
w <- acs('weibull', t = t, scale = 2, shape = 0.8)
p <- acs('paretoII', t = t, scale = 3, shape = 0.3)

## visualize the ACS
dta <- data.table(t, f, b, w, p)

m.dta <- melt(dta, id.vars = 't')

ggplot(m.dta,
       aes(x = t,
           y = value,
           group = variable,
           colour = variable)) +
  geom_point(size = 2.5) +
```

```
  geom_line(lwd = 1) +
  scale_color_manual(values = c('steelblue4', 'red4', 'green4', 'darkorange'),
                     labels = c('FGN', 'Burr XII', 'Weibull', 'Pareto II'),
                     name = '') +
  labs(x = bquote(lag ~ tau),
       y = 'Acf') +
  scale_x_continuous(breaks = t) +
  theme_classic()
```

`actpnts` *AutoCorrelation Transformed Points*

Description

Transforms a Gaussian process in order to match a target marginal lowers its autocorrelation values. The actpnts evaluates the corresponding autocorrelations for the given target marginal for a set of Gaussian correlations, i.e., it returns (ρ_x, ρ_z) points where ρ_x and ρ_z represent, respectively, the autocorrelations of the target and Gaussian process.

Usage

```
actpnts(margdist, margarg, p0 = 0, distbounds = c(-Inf, Inf))
```

Arguments

`margdist`	target marginal distribution
`margarg`	list of marginal distribution arguments
`p0`	probability zero
`distbounds`	distribution bounds (default set to c(-Inf, Inf))

Examples

```
library(CoSMoS)

## here we target to a process that has the Pareto type II
## marginal distribution with scale parameter 1 and shape parameter 0.3
## (note that all parameters have to be named)
dist <- 'paretoII'
distarg <- list(scale = 1, shape = .3)

x <- actpnts(margdist = dist, margarg = distarg, p0 = 0)
x

## you can see the points by using
ggplot(x,
       aes(x = rhox,
```

```
             y = rhoz)) +
  geom_point(colour = 'royalblue4', size = 2.5) +
  geom_abline(lty = 5) +
  labs(x = bquote(Autocorrelation ~ rho[x]),
       y = bquote(Gaussian ~ rho[z])) +
  scale_x_continuous(limits = c(0, 1)) +
  scale_y_continuous(limits = c(0, 1)) +
  theme_classic()
```

`advectionF` *Advection fields*

Description

Provides parametric functions that describe different types of advection fields.

Usage

```
advectionF(id, ...)
```

Arguments

`id`	advection type id (`uniform`, `rotation`, `spiral`, `spiralCE`, `radial`, and `hyperbolic`)
`...`	other arguments (vector of coordinates and parameters of advection field functions)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)

## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

## get the advection field
af <- advectionF('spiral',
                 spacepoints = coord,
                 x0 = floor(m / 2),
                 y0 = floor(m / 2),
```

```
                a = 3,
                b = 2,
                rotation = 1)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFhyperbolic` *Hyperbolic advection field*

Description

Provides an advection field with hyperbolic trajectories.

Usage

```
advectionFhyperbolic(spacepoints, x0, y0, a, b)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of hyperbola
`y0`	y coordinate of the center of hyperbola
`a`	parameter controlling the x component of rotational velocity
`b`	parameter controlling the y component of rotational velocity

Note

- if $a > 0$, $b > 0$: toward bottom-left and top-right corner
- if $a < 0$, $b < 0$: toward top-left and bottom-right corner

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFhyperbolic(spacepoints = coord,
                           x0 = floor(m / 2),
                           y0 = floor(m / 2),
                           a = 3,
                           b = 2)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFradial` *Radial advection field*

Description

Provides an advection field corresponding to radial motion from or towards a specified reference point.

Usage

```
advectionFradial(spacepoints, x0, y0, a, b)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of radial motion
`y0`	y coordinate of the center of radial motion
`a`	parameter controlling the x component of radial velocity
`b`	parameter controlling the y component of radial velocity

Note

- if a > 0, b > 0: divergence from (x0, y0) (source point effect)
- if a < 0, b < 0: convergence to (x0, y0) (sink effect)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)

## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFradial(spacepoints = coord,
                       x0 = floor(m / 2),
                       y0 = floor(m / 2),
                       a = 3,
                       b = 2)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFrotation` *Rotational advection field*

Description

Provides an advection field corresponding to rotation around a specified center.

Usage

```
advectionFrotation(spacepoints, x0, y0, a, b)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of rotation
`y0`	y coordinate of the center of rotation
`a`	parameter controlling the x component of rotational velocity
`b`	parameter controlling the y component of rotational velocity

Note

- if a > 0, b > 0: clockwise rotation around (x0, y0)
- if a < 0, b < 0: counter-clockwise rotation around (x0, y0)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFrotation(spacepoints = coord,
                        x0 = floor(m / 2),
                        y0 = floor(m / 2),
                        a = 3,
                        b = 2)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFspiral` *Spiraling advection field*

Description

Provides an advection field corresponding to a spiral motion to/from a specified reference point (sink).

Usage

```
advectionFspiral(spacepoints, x0, y0, a, b, rotation = 1)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of reference point (sink)
`y0`	y coordinate of reference point (sink)
`a`	parameter controlling the x component of rotational velocity
`b`	parameter controlling the y component of rotational velocity
`rotation`	parameter controlling the rotational direction. The following combinations hold: • if a > 0, b > 0, and direction = 1: spiraling CLOCKWISE TO (x0, y0) • if a < 0, b < 0, and direction = 1: spiraling COUNTER-CLOCKWISE FROM (x0, y0) • if a > 0, b > 0, and direction = 2: spiraling COUNTER-CLOCKWISE TO (x0, y0) • if a < 0, b < 0, and direction = 2: spiraling CLOCKWISE FROM (x0, y0)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFspiral(spacepoints = coord,
                        x0 = floor(m / 2),
                        y0 = floor(m / 2),
                        a = 3,
                        b = 2,
                        rotation = 1)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFspiralCE` *Spiraling advection field satisfying continuity equation*

Description

Provides an advection field corresponding to a spiral motion to/from a specified reference point (sink) satisfying continuity equation (from John Burkardt's website).

Usage

```
advectionFspiralCE(spacepoints, a, C)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`a`	parameter controlling the intensity of rotational velocity (a > 0 clokwise; a < 0 conter-clockwise)
`C`	parameter ranging in (0, 2*pi)

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFspiralCE(spacepoints = coord,
                        a = 5,
                        C = 1)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`advectionFuniform` *Uniform advection field*

Description

Provides an advection field with constant orthogonal (u and v) components at each grid point. This mimics rigid translation in a given direction according to the components u and v of the velocity vector.

Usage

```
advectionFuniform(spacepoints, u, v)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`u`	velocity component along the x axis
`v`	velocity component along the y axis

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(ggquiver)
library(ggplot2)
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

af <- advectionFuniform(spacepoints = coord,
                       u = 2,
                       v = 6)

## visualize advection field
dta <- data.frame(lon = coord[ ,1], lat = coord[ ,2], u = af[ ,1], v = af[ ,2])
ggplot(dta, aes(x = lon, y = lat, u = u, v = v)) +
geom_quiver() +
theme_light()
```

`analyzeTS` *The Functions analyzeTS, reportTS, and simulateTS*

Description

Provide a complete set of tools to make time series analysis a piece of cake - `analyzeTS` automatically performs seasonal analysis, fits distributions and correlation structures, `reportTS` provides visualizations of the fitted distributions and correlation structures, and a table with the values of the fitted parameters and basic descriptive statistics, `simulateTS` automatically takes the results of `analyzeTS` and generates synthetic ones.

Usage

```
analyzeTS(
  TS,
  season = "month",
  dist = "ggamma",
  acsID = "weibull",
  norm = "N1",
  n.points = 30,
  lag.max = 30,
  constrain = FALSE,
  opts = NULL
)

reportTS(aTS, method = "dist")

simulateTS(aTS, from = NULL, to = NULL)
```

Arguments

`TS`	time series in format - date, value
`season`	name of the season (e.g. month, week)
`dist`	name of the distribution to be fitted
`acsID`	ID of the autocorrelation structure to be fitted
`norm`	norm used for distribution fitting - id ('N1', 'N2', 'N3', 'N4')
`n.points`	number of points to be subsetted from ecdf
`lag.max`	max lag for the empirical autocorrelation structure
`constrain`	logical - constrain shape2 parametes for finite tails
`opts`	minimization options
`aTS`	analyzed timeseries
`method`	report method - `dist` for distribution fits, `acs` for ACS fits and `stat` for basic statistical report
`from`	starting date/time of the simulation
`to`	end date/time of the simulation

Details

In practice, we usually want to simulate a natural process using some sampled time series. To generate a synthetic time series with similar characteristics to the observed values, we have to determine marginal distribution, autocorrelation structure and probability zero for each individual month. This can is done by fitting distributions and autocorrelation structures with `analyzeTS`. Result can be checked with `reportTS`. Syynthetic time series with the same statistical properties can be produced with `simulateTS`.

Recomended distributions for variables:

- *precipitation*: ggamma (Generalized Gamma), burr### (Burr type)
- *streamflow*: ggamma (Generalized Gamma), burr### (Burr type)
- *relative humidity*: beta
- *temperature*: norm (Normal distribution)

Examples

```
library(CoSMoS)

## Load data included in the package
## (to find out more about the data use ?precip)
data('precip')

## Fit seasonal ACSs and distributions to the data
a <- analyzeTS(precip)

reportTS(a, 'dist') ## show seasonal distribution fit
reportTS(a, 'acs') ## show seasonal ACS fit
reportTS(a, 'stat') ## display basic descriptive statisctics

######################################
## 'duplicate' analyzed time series ##
sim <- simulateTS(a)

## plot the result
precip[, id := 'observed']
sim[, id := 'simulated']

dta <- rbind(precip, sim)

ggplot(dta) +
  geom_line(aes(x = date, y = value)) +
  facet_wrap(~id, ncol = 1) +
  theme_classic()

################################################
## or simulate timeseries of different length ##
sim <- simulateTS(a,
                  from = as.POSIXct('1978-12-01 00:00:00'),
                  to = as.POSIXct('2008-12-01 00:00:00'))
```

```
## and plot the result
precip[, id := 'observed']
sim[, id := 'simulated']

dta <- rbind(precip, sim)

ggplot(dta) +
  geom_line(aes(x = date, y = value)) +
  facet_wrap(~id, ncol = 1) +
  theme_classic()
```

`anisotropyT` *Anisotropy transformation*

Description

Provides parametric functions that describe different types of planar deformation fields, including affine (rotation and stretching), and swirl-like deformation. For more details see Papalexiou et al.(2021) and references therein.

Usage

```
anisotropyT(id, ...)
```

Arguments

`id`	anisotropy type id (`affine`, `swirl`, and `wave`)
`...`	additional arguments (vector of coordinates and parameters of the anisotropy transformations)

References

Papalexiou, S. M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond, Water Resources Research, doi: 10.1029/2020WR029466

Examples

```
library(CoSMoS)

## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)
```

```
## get the anisotropy field
at1 <- anisotropyT('affine',
                   spacepoints = coord,
                   phi1 = 0.5,
                   phi2 = 2,
                   phi12 = 0,
                   theta = -pi/3)
at2 <- anisotropyT('swirl',
                   spacepoints = coord,
                   x0 = floor(m / 2),
                   y0 = floor(m / 2),
                   b = 10,
                   alpha = 1.5 * pi)
at3 <- anisotropyT('wave',
                   spacepoints = coord,
                   phi1 = 0.5,
                   phi2 = 2,
                   beta = 3,
                   theta = 0)

## visualize anisotropy field
aux = data.frame(lon = at2[ ,1], lat = at2[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

`anisotropyTaffine` *Affine anisotropy transformation*

Description

Affine anisotropy transformation.

Usage

```
anisotropyTaffine(spacepoints, phi1, phi2, phi12, theta)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`phi1`	stretching parameter along the x axis
`phi2`	stretching parameter along the y axis
`phi12`	shear effect
`theta`	rotation angle

References

Allard, D., Senoussi, R., Porcu, E. (2016). Anisotropy Models for Spatial Data. Mathematical Geosciences, 48(3), 305-328, doi: 10.1007/s110040159594x

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

at <- anisotropyTaffine(spacepoints = coord,
                       phi1 = 0.5,
                       phi2 = 2,
                       phi12 = 0,
                       theta = -pi/3)

## visualize transformed coordinate system
aux = data.frame(lon = at[ ,1], lat = at[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

`anisotropyTswirl` *Swirl anisotropy transformation*

Description

Swirl anisotropy transformation.

Usage

```
anisotropyTswirl(spacepoints, x0, y0, b, alpha)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`x0`	x coordinate of the center of the swirl deformation
`y0`	y coordinate of the center of the swirl deformation
`b`	scaling parameter controlling the swirl deformation
`alpha`	rotation angle

References

Ligas, M., Banas, M., Szafarczyk, A. (2019). A method for local approximation of a planar deformation field. Reports on Geodesy and Geoinformatics, 108(1), 1-8, doi: 10.2478/rgg20190007

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

at <- anisotropyTswirl(spacepoints = coord,
                       x0 = floor(m / 2),
                       y0 = floor(m / 2),
                       b = 10,
                       alpha = 1.5 * pi)

## visualize transformed coordinate system
aux = data.frame(lon = at[ ,1], lat = at[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

`anisotropyTwave` *Wave anisotropy transformation*

Description

Wave anisotropy transformation.

Usage

```
anisotropyTwave(spacepoints, phi1, phi2, beta, theta)
```

Arguments

`spacepoints`	vector of coordinates (2 x d), where d is the number of locations/grid points
`phi1`	stretching parameter along the x axis
`phi2`	stretching parameter along the y axis
`beta`	amplitude of sinusoidal wave
`theta`	rotation angle

References

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
## specify coordinates
m = 25
aux <- seq(0, m - 1, length = m)
coord <- expand.grid(aux, aux)

at <- anisotropyTwave(spacepoints = coord,
                      phi1 = 0.5,
                      phi2 = 2,
                      beta = 3,
                      theta = 0)

## visualize transformed coordinate system
aux = data.frame(lon = at[ ,1], lat = at[ ,2], id1 = rep(1:m, each = m), id2 = rep(1:m, m))
ggplot(aux, aes(x = lon, y = lat)) +
geom_path(aes(group = id1)) +
geom_path(aes(group = id2)) +
geom_point(col = 2) +
theme_light()
```

`BurrIII` *Burr Type III distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the Burr Type III distribution.

Usage

```
dburrIII(x, scale, shape1, shape2, log = FALSE)

pburrIII(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qburrIII(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rburrIII(n, scale, shape1, shape2)

mburrIII(r, scale, shape1, shape2)
```

Arguments

`x, q`	vector of quantiles.
`scale, shape1, shape2`	scale and shape parameters; the shape arguments cannot be a vectors (must have length one).
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \le x]$ otherwise, $P[X > x]$.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`r`	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(1, 15)),
       aes(x)) +
  stat_function(fun = dburrIII,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

`BurrXII` *Burr Type XII distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the Burr Type XII distribution.

Usage

```
dburrXII(x, scale, shape1, shape2, log = FALSE)

pburrXII(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qburrXII(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rburrXII(n, scale, shape1, shape2)

mburrXII(r, scale, shape1, shape2)
```

Arguments

`x, q`	vector of quantiles.
`scale, shape1, shape2`	scale and shape parameters; the shape arguments cannot be a vector (must have length one).
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \le x]$ otherwise, $P[X > x]$.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`r`	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 10)),
       aes(x)) +
  stat_function(fun = dburrXII,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

---

checkRF *Numerical and visual check of generated random fields*

Description

Compares generated random fields sample statistics with the theoretically expected values (similar to `checkTS`). It also returns graphical output for visual check.

Usage

```
checkRF(RF, lags = 30, nfields = 49, method = "stat")
```

Arguments

`RF`	output of `generateRF`
`lags`	number of lags of empirical STCF to be considered in the graphical output (default set to 30)
`nfields`	number of fields to be used in the numerical and graphical output (default set to 49). As the plots are arranged in a matrix with nrows as close as possible to ncol, we suggest using values such as 3x3, 3x4, 7x8, etc.
`method`	report method - `'stat'` for basic statistical report, `'statplot'` for graphical check of lagged SCS, target STCS, and marginal distribution, `'field'` for plotting a matrix of the first `nfields`, and `'movie'` to save the first `nfields` as a GIF file named "movieRF.gif" in the current working directory

Examples

```
## The example below refers to the fitting and simulation of 10 random fields
## of size 10x10 with AR(1) temporal correlation. As the fitting algorithm has
## O((mxm)^3) complexity for a mxm field, this setting allows for quick fitting
## and simulation (short CPU time). However, for a more effective visualization
## and reliable performance assessment, we suggest to generate a larger number
## of fields (e.g. 100 or more) of size about 30X30. This setting needs more
## CPU time but enables more effective comparison of theoretical and
## empirical statistics. Sizes larger than about 50x50 can be unpractical
##  on standard machines.

fit <- fitVAR(
  spacepoints = 10,
  p = 1,
  margdist ='burrXII',
  margarg = list(scale = 3, shape1 = .9, shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull", tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20, shape = 0.7),
                tcfarg = list(scale = 1.1, shape = 0.8))
)

sim <- generateRF(n = 12,
                    STmodel = fit)
checkRF(RF = sim,
          lags = 10,
          nfields = 12)
```

---

checkTS *Check generated timeseries*

Description

Compares generated time series sample statistics with the theoretically expected values.

Usage

```
checkTS(TS, distbounds = c(-Inf, Inf))
```

Arguments

`TS`	generated timeseries
`distbounds`	distribution bounds (default set to c(-Inf, Inf))

Examples

```
library(CoSMoS)

## check your generated timeseries
x <- generateTS(margdist = 'burrXII',
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .25),
                acsvalue = acs(id = 'weibull',
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .5, TSn = 5)

checkTS(x)
```

`disch` *Daily streamflow data data*

Description

Station details

- Name: Nassawango Creek near Snow Hill, Worcester County, Maryland, Hydrologic Unit 02080111
- Network Id: , USGS 01485500
- Latitude/Longitude: 38°13’44.1", 75°28’17.2"
- Elevation: 11.49 ft above North American Vertical Datum of 1988.
- Measurement unit: cubic feet per second

Usage

```
disch
```

Format

A data.table with 23315 rows and 2 variables:

date POSIXct format date/time

value daily avarage values

Details

more details can be found here.

Source

The United States Geological Survey (USGS) National Water Information System (NWIS)

`fitactf` *Fit the AutoCorrelation Transformation Function*

Description

Fits the ACTF (Autocorrelation Transformation Function) to the estimated points (ρ_x, ρ_z) using `nls`.

Usage

```
fitactf(actpnts, discrete = FALSE)
```

Arguments

`actpnts`	estimated ACT points
`discrete`	logical - is the marginal distribution discrete?

Examples

```

library(CoSMoS)

## choose the marginal distribution as Pareto type II
## with corresponding parameters
dist <- 'paretoII'
distarg <- list(scale = 1, shape = .3)

## estimate rho 'x' and 'z' points using ACTI
p <- actpnts(margdist = dist, margarg = distarg, p0 = 0)

## fit ACTF
fit <- fitactf(p)

## plot the result
plot(fit)
```

`fitDist` *Distribution fitting*

Description

Uses Nelder-Mead simplex algorithm to minimize fitting norms.

Usage

```
fitDist(
  data,
  dist,
  n.points,
  norm,
  constrain,
 opts = list(algorithm = "NLOPT_LN_NELDERMEAD", xtol_rel = 1e-08, maxeval = 10000)
)
```

Arguments

`data`	value to be fitted
`dist`	name of the distribution to be fitted
`n.points`	number of points to be subsetted from ecdf
`norm`	norm used for distribution fitting - id ('N1', 'N2', 'N3', 'N4')
`constrain`	logical - constrain shape2 parametes for finite tails
`opts`	minimization options

Examples

```
x <- fitDist(rnorm(1000), 'norm', 30, 'N1', FALSE)
x
```

`fitVAR` *VAR model parameters to simulate correlated parent Gaussian random vectors and fields*

Description

Compute VAR model parameters to simulate parent Gaussian random vectors with specified spatiotemporal correlation structure using the method described by Biller and Nelson (2003).

Usage

```
fitVAR(
  spacepoints,
  p,
  margdist,
  margarg,
  p0,
  distbounds = c(-Inf, Inf),
  stcsid,
  stcsarg,
  scalefactor = 1,
  anisotropyid = "affine",
  anisotropyarg = list(phi1 = 1, phi2 = 1, phi12 = 0, theta = 0),
  advectionid = "uniform",
  advectionarg = list(u = 0, v = 0)
)
```

Arguments

`spacepoints`	it can be a numeric integer, which is interpreted as the side length m of the square field (m x m), or a matrix (d x 2) of coordinates (e.g. longitude and latitude) of d spatial locations (e.g. d gauge stations)
`p`	order of VAR(p) model
`margdist`	target marginal distribution of the field
`margarg`	list of marginal distribution arguments. Please consult the documentation of the selected marginal distribution indicated in the argument `margdist` for the list of required parameters
`p0`	probability zero
`distbounds`	distribution bounds (default set to `c(-Inf, Inf)`)
`stcsid`	spatiotemporal correlation structure ID
`stcsarg`	list of spatiotemporal correlation structure arguments. Please consult the documentation of the selected spatiotemporal correlation structure indicated in the argument `stcsid` for the list of required parameters
`scalefactor`	factor specifying the distance between the centers of two pixels (default set to 1)
`anisotropyid`	spatial anisotropy ID (`affine` by default, `swirl` or `wave`)
`anisotropyarg`	list of arguments characterizing the spatial anisotropy according to the syntax of the function `anisotropyT`. Isotropic fields by default
`advectionid`	advection field ID (`uniform` by default, `rotation`, `spiral`, `spiralCE`, `radial`, or `hyperbolic`)
`advectionarg`	list of arguments characterizing the advection field according to the syntax of the function `advectionF`. No advection by default

Details

The fitting algorithm has $O(m*m)^3$ complexity for a $(m*m)$ field or equivalently $O(d^3)$ complexity for a d-dimensional vector. Very large values of $(m*m)$ (or d) and high order AR correlation structures can be unpractical on standard machines.

Here, we give indicative CPU times for some settings, referring to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32GB RAM.
: CPU time:
d = 100 or m = 10, p = 1: ~ 0.4s
d = 900 or m = 30, p = 1: ~ 6.0s
d = 900 or m = 30, p = 5: ~ 47.0s
d = 2500 or m = 50, p = 1: ~100.0s

Note

While all the advection types can be applied to isotropic random fields, anisotropic random fields require more care. We suggest combining `affine` anysotropy with `uniform` advection, and `swirl` anisotropy with `rotation` or `spiral` advection with the same rotation center.

References

Biller, B., Nelson, B.L. (2003). Modeling and generating multivariate time-series input processes using a vector autoregressive technique. ACM Trans. Model. Comput. Simul. 13(3), 211-237, doi: 10.1145/937332.937333

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi: 10.1016/j.advwatres.2018.02.013

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi: 10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
## for multivariate simulation
coord <- cbind(runif(4)*30, runif(4)*30)

fit <- fitVAR(
  spacepoints = coord,
  p = 1,
  margdist ='burrXII',
  margarg = list(scale = 3,
                 shape1 = .9,
                 shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
```

```
  stcsarg = list(scfid = "weibull",
                 tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20,
                               shape = 0.7),
                 tcfarg = list(scale = 1.1,
                               shape = 0.8))
)

dim(fit$alpha)
dim(fit$res.cov)

fit$m
fit$margarg
fit$margdist

## for random fields simulation
fit <- fitVAR(
  spacepoints = 10,
  p = 1,
  margdist ='burrXII',
  margarg = list(scale = 3, shape1 = .9, shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull", tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20, shape = 0.7),
                 tcfarg = list(scale = 1.1, shape = 0.8))
)

dim(fit$alpha)
dim(fit$res.cov)

fit$m
fit$margarg
fit$margdist
```

`generateMTS` *Simulation of multiple time series with given marginals and spatiotemporal properties*

Description

Generates multiple time series with given marginals and spatiotemporal properties, just provide (1) the output of `fitVAR` function, and (2) the number of time steps to simulate.

Usage

```
generateMTS(n, STmodel)
```

Arguments

`n`	number of fields (time steps) to simulate
`STmodel`	list of arguments resulting from `fitVAR` function

Details

Referring to the documentation of `fitVAR` for details on computational complexity of the fitting algorithm, here we report indicative simulation CPU times for some settings, assuming that the model parameters are already evaluated. CPU times refer to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32GB RAM.
CPU time:
d = 900, p = 1, n = 1000: ~17s
d = 900, p = 1, n = 10000: ~75s
d = 900, p = 5, n = 100: ~280s
d = 900, p = 5, n = 1000: ~302s
d = 2500, p = 1, n = 1000 : ~160s
d = 2500, p = 1, n = 10000 : ~570s
where d denotes the number of spatial locations

Examples

```
## Simulation of a 4-dimensional vector with VAR(1) correlation structure
coord <- cbind(runif(4)*30, runif(4)*30)

fit <- fitVAR(
  spacepoints = coord,
  p = 1,
  margdist ='burrXII',
  margarg = list(scale = 3,
                 shape1 = .9,
                 shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull",
                 tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20,
                               shape = 0.7),
                 tcfarg = list(scale = 1.1,
                               shape = 0.8))
)

sim <- generateMTS(n = 100,
                     STmodel = fit)
```

`generateMTSFast` *Faster simulation of multiple time series with approximately separable spatiotemporal correlation structure*

Description

For more details see section 6 in Serinaldi and Kilsby (2018), and section 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
generateMTSFast(
  n,
  spacepoints,
  margdist,
  margarg,
  p0,
  distbounds = c(-Inf, Inf),
  stcsid,
  stcsarg,
  scalefactor = 1,
  anisotropyid = "affine",
  anisotropyarg = list(phi1 = 1, phi2 = 1, phi12 = 0, theta = 0)
)
```

Arguments

`n`	number of fields (time steps) to simulate
`spacepoints`	matrix `(d x 2)` of coordinates (e.g. longitude and latitude) of `d` spatial locations (e.g. `d` gauge stations)
`margdist`	target marginal distribution
`margarg`	list of marginal distribution arguments. Please consult the documentation of the selected marginal distribution indicated in the argument `margdist` for the list of required parameters
`p0`	probability zero
`distbounds`	distribution bounds (default set to `c(-Inf, Inf)`)
`stcsid`	spatiotemporal correlation structure ID
`stcsarg`	list of spatiotemporal correlation structure arguments. Please consult the documentation of the selected spatiotemporal correlation structure indicated in the argument `stcsid` for the list of required parameters
`scalefactor`	factor specifying the distance between the centers of two pixels (default set to 1)
`anisotropyid`	spatial anisotropy ID (`affine` by default, `swirl` or `wave`)
`anisotropyarg`	list of arguments characterizing the spatial anisotropy according to the syntax of the function `anisotropyT`. Isotropic fields by default

Details

generateMTSFast provides a faster approach to multivariate simulation compared to generateMTS by exploiting circulant embedding fast Fourier transformation. However, this approach is feasible only for approximately separable target spatiotemporal correlation functions. generateMTSFast comprises fitting and simulation in a single function. Here, we give indicative CPU times for some settings, referring to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32GB RAM.
CPU time:
d = 2500, n = 1000: ~58s
d = 2500, n = 10000: ~160s
d = 10000, n = 1000: ~2955s (~50min)

References

Serinaldi, F., Kilsby, C.G. (2018). Unsurprising Surprises: The Frequency of Record-breaking and Overthreshold Hydrological Extremes Under Spatial and Temporal Dependence. Water Resources Research, 54(9), 6460-6487, doi: 10.1029/2018WR023055

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi: 10.1029/2019WR026331

Examples

```
coord <- cbind(runif(4)*30, runif(4)*30)

sim <- generateMTSFast(
    n = 50,
    spacepoints = coord,
    p0 = 0.7,
    margdist ='paretoII',
    margarg = list(scale = 1,
                   shape = .3),
    stcsarg = list(scfid = "weibull",
                   tcfid = "weibull",
                   scfarg = list(scale = 20,
                                 shape = 0.7),
                   tcfarg = list(scale = 1.1,
                                 shape = 0.8))
)
```

`generateRF` *Simulation of random field with given marginals and spatiotemporal properties*

Description

Generates random field with given marginals and spatiotemporal properties, just provide (1) the output of `fitVAR` function, and (2) the number of time steps to simulate.

Usage

```
generateRF(n, STmodel)
```

Arguments

`n`	number of fields (time steps) to simulate
`STmodel`	list of arguments resulting from `fitVAR` function

Details

Referring to the documentation of `fitVAR` for details on computational complexity of the fitting algorithm, here we report indicative simulation CPU times for some settings, assuming that the model parameters are already evaluated. CPU times refer to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32GB RAM.
CPU time:
m = 30, p = 1, n = 1000: ~17s
m = 30, p = 1, n = 10000: ~75s
m = 30, p = 5, n = 100: ~280s
m = 30, p = 5, n = 1000: ~302s
m = 50, p = 1, n = 1000 : ~160s
m = 50, p = 1, n = 10000 : ~570s where m denotes the side length of a square field (mxm)

Examples

```
## The example below refers to the simulation of few random fields of
## size 10x10 with AR(1) temporal correlation for the sake of illustration.
## For a more effective visualization and reliable performance assessment,
## we suggest to generate a larger number of fields (e.g. 100 or more)
## of size about 30X30.
## See section 'Details' for additional information on running times
## with different settings.

fit <- fitVAR(
  spacepoints = 10,
  p = 1,
  margdist ='burrXII',
  margarg = list(scale = 3, shape1 = .9, shape2 = .2),
  p0 = 0.8,
  stcsid = "clayton",
  stcsarg = list(scfid = "weibull", tcfid = "weibull",
                 copulaarg = 2,
                 scfarg = list(scale = 20, shape = 0.7),
                tcfarg = list(scale = 1.1, shape = 0.8))
)
```

```
sim <- generateRF(n = 12,
                  STmodel = fit)
checkRF(sim,
        lags = 10,
        nfields = 12)
```

`generateRFFast` *Faster simulation of random fields with approximately separable spatiotemporal correlation structure*

Description

For more details see section 6 in Serinaldi and Kilsby (2018), and section 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
generateRFFast(
  n,
  spacepoints,
  margdist,
  margarg,
  p0,
  distbounds = c(-Inf, Inf),
  stcsid,
  stcsarg,
  scalefactor = 1,
  anisotropyid = "affine",
  anisotropyarg = list(phi1 = 1, phi2 = 1, phi12 = 0, theta = 0)
)
```

Arguments

`n`	number of fields (time steps) to simulate
`spacepoints`	side length m of the square field (`m` x `m`)
`margdist`	target marginal distribution of the field
`margarg`	list of marginal distribution arguments. Please consult the documentation of the selected marginal distribution indicated in the argument `margdist` for the list of required parameters
`p0`	probability zero
`distbounds`	distribution bounds (default set to `c(-Inf, Inf)`)
`stcsid`	spatiotemporal correlation structure ID
`stcsarg`	list of spatiotemporal correlation structure arguments. Please consult the documentation of the selected spatiotemporal correlation structure indicated in the argument `stcsid` for the list of required parameters

`scalefactor`	factor specifying the distance between the centers of two pixels (default set to 1)
`anisotropyid`	spatial anisotropy ID (`affine` by default, `swirl` or `wave`)
`anisotropyarg`	list of arguments characterizing the spatial anisotropy according to the syntax of the function `anisotropyT`. Isotropic fields by default

Details

`generateRFFast` provides a faster approach to RF simulation compared to `generateRF` by exploiting circulant embedding fast Fourier transformation. However, this approach is feasible only for approximately separable target spatiotemporal correlation functions. `generateRFFast` comprises fitting and simulation in a single function. Here, we give indicative CPU times for some settings, referring to a Windows 10 Pro x64 laptop with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 4-core, 8 logical processors, and 32GB RAM.
CPU time:
m = 50, n = 1000: ~58s
m = 50, n = 10000: ~160s
m = 100, n = 1000: ~2955s (~50min)

References

Serinaldi, F., Kilsby, C.G. (2018). Unsurprising Surprises: The Frequency of Record-breaking and Overthreshold Hydrological Extremes Under Spatial and Temporal Dependence. Water Resources Research, 54(9), 6460-6487, doi: 10.1029/2018WR023055

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi: 10.1029/2019WR026331

Examples

```
sim <- generateRFFast(
    n = 50,
    spacepoints = 3,
    p0 = 0.7,
    margdist ='paretoII',
    margarg = list(scale = 1,
                   shape = .3),
    stcsarg = list(scfid = "weibull",
                   tcfid = "weibull",
                   scfarg = list(scale = 20,
                                 shape = 0.7),
                   tcfarg = list(scale = 1.1,
                                 shape = 0.8))
)

checkRF(sim,
          lags = 10,
          nfields = 49)
```

`generateTS` *Generate timeseries*

Description

Generates timeseries with given properties, just provide (1) the target marginal distribution and its parameters, (2) the target autocorrelation structure or individual autocorrelation values up to a desired lag, and (3) the probablility zero if you wish to simulate an intermittent process.

Usage

```
generateTS(
  n,
  margdist,
  margarg,
  p = NULL,
  p0 = 0,
  TSn = 1,
  distbounds = c(-Inf, Inf),
  acsvalue = NULL
)
```

Arguments

`n`	number of values
`margdist`	target marginal distribution
`margarg`	list of marginal distribution arguments
`p`	integer - model order (if NULL - limits maximum model order according to auto-correlation structure values)
`p0`	probability zero
`TSn`	number of timeseries to be generated
`distbounds`	distribution bounds (default set to c(-Inf, Inf))
`acsvalue`	target auto-correlation structure (from lag 0)

Details

A step-by-step guide:

- First define the target marginal (`margdist`), that is, the probability distribution of the generated data. For example set `margdist = 'ggamma'` if you wish to generate data following the Generalized Gamma distribution, `margidst = 'burrXII'` for Burr type XII distribution etc. For a full list of the distributions we support see the help vignette. In general, the package supports all build-in distribution functions of R and of other packages.

- Define the parameters' values (`margarg`) of the distribution you selected. For example the Generalized Gamma has one scale and two shape parameters so set the desired value, e.g., `margarg = list(scale = 2, shape1 = 0.9, shape2 = 0.8)`. Note distributions might have different number of parameters and different type of parameters (location, scale, shape). See the help vignette for details on the parameters of each distribution we support.
- If you wish your time series to be intermittent (e.g., precipitation), then define the probability zero. For example, set p0 = 0.9, if you wish your generated data to have 90% of zero values (dry days).
- Define your linear autocorrelations.
  - You can supply specific lag autocorrelations starting from lag 0 and up to a desired lag, e.g., `acs = c(1, 0.9, 0.8, 0.7)`; this will generate a process with lag1, 2 and 3 autocorrelations equal with 0.9, 0.8 and 0.7.
  - Alternatively, you can use a parametric autocorrelation structure (see section 3.2 in Papalexiou (2018). We support the following autocorrelation structures (acs) weibull, paretoII, fgn and burrXII. See also `acs` examples.
- Define the order to the autoregressive model p. For example if you aim to preserve the first 10 lag autocorrelations then just set p = 10. Otherwise set it p = NULL and the model will decide the value of p in order to preserve the whole autocorrelation structure.
- Lastly just define the time series length, e.g., `n = 1000` and number of time series you wish to generate, e.g., `TSn = 10`.

Play around with the following given examples which will make the whole process a piece of cake.

References

Papalexiou, S.M. (2018). Unified theory for stochastic modelling of hydroclimatic processes: Preserving marginal distributions, correlation structures, and intermittency. Advances in Water Resources, 115, 234-252, doi: 10.1016/j.advwatres.2018.02.013

Examples

```
library(CoSMoS)

## Case1:
## You wish to generate 3 time series of size 1000 each
## that follow the Generalized Gamma distribution with parameters
## scale = 1, shape1 = 0.8, shape2 = 0.8
## and autocorrelation structure the ParetoII
## with parameters scale = 1 and shape = .75
x <- generateTS(margdist = 'ggamma',
                margarg = list(scale = 1,
                               shape1 = .8,
                               shape2 = .8),
                acsvalue = acs(id = 'paretoII',
                               t = 0:30,
                               scale = 1,
                               shape = .75),
                n = 1000,
```

```
                p = 30,
                TSn = 3)

## see the results
plot(x)



## Case2:
## You wish to generate time series the same distribution
## and autocorrelations as is Case1 but intermittent
## with probability zero equal to 90%
y <- generateTS(margdist = 'ggamma',
                margarg = list(scale = 1,
                               shape1 = .8,
                               shape2 = .8),
                acsvalue = acs(id = 'paretoII',
                               t = 0:30,
                               scale = 1,
                               shape = .75),
                p0 = .9,
                n = 1000,
                p = 30,
                TSn = 3)

## see the results
plot(y)

## Case3:
## You wish to generate a time series of size 1000
## that follows the Beta distribution
## (e.g., relative humidity ranging from 0 to 1)
## with parameters shape1 = 0.8, shape2 = 0.8, is defined from 0 to 1
## and autocorrelation structure the ParetoII
## with parameters scale = 1 and shape = .75
z <- generateTS(margdist = 'beta',
                margarg = list(shape1 = .6,
                               shape2 = .8),
                distbounds = c(0, 1),
                acsvalue = acs(id = 'paretoII',
                               t = 0:30,
                               scale = 1,
                               shape = .75),
                n = 1000,
                p = 20)

## see the results
plot(z)

## Case4:
## Same in previous case but now you provide specific
## autocorrelation values for the first three lags,
## ie.., lag 1 to 3 equal to 0.9, 0.8 and 0.7
```

```
z <- generateTS(margdist = 'beta',
                margarg = list(shape1 = .6,
                               shape2 = .8),
                distbounds = c(0, 1),
                acsvalue = c(1, .9, .8, .7),
                n = 1000,
                p = TRUE)

## see the results
plot(z)
```

`GEV` *Generalized extreme value distribution*

Description

Provides density, distribution function, quantile function, and random value generation, for the generalized extreme value distribution.

Usage

```
dgev(x, loc, scale, shape, log = FALSE)

pgev(q, loc, scale, shape, lower.tail = TRUE, log.p = FALSE)

qgev(p, loc, scale, shape, lower.tail = TRUE, log.p = FALSE)

rgev(n, loc, scale, shape)

mgev(r, loc, scale, shape)
```

Arguments

`x, q`	vector of quantiles.
`loc, scale, shape`	location, scale and shape parameters.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`r`	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dgev,
                args = list(loc = 1,
                            scale = .5,
                            shape = .15),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

`GGamma` *Generalized gamma distribution*

Description

Provides density, distribution function, quantile function, random value generation, and raw moments of order *r* for the generalized gamma distribution.

Usage

```
dggamma(x, scale, shape1, shape2, log = FALSE)

pggamma(q, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qggamma(p, scale, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rggamma(n, scale, shape1, shape2)

mggamma(r, scale, shape1, shape2)
```

Arguments

`x, q`	vector of quantiles.
`scale, shape1, shape2`	scale and shape parameters; the shape arguments cannot be a vectors (must have length one).
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \le x]$ otherwise, $P[X > x]$.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`r`	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dggamma,
                args = list(scale = 5,
                            shape1 = .25,
                            shape2 = .75),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

`moments` *Numerical estimation of moments*

Description

Uses numerical integration to caclulate the theoretical raw or central moments of the specified distribution.

Usage

```
moments(
  dist,
  distarg,
  p0 = 0,
  raw = T,
  central = T,
  coef = T,
  distbounds = c(-Inf, Inf),
  order = 1:4
)
```

Arguments

`dist`	distribution
`distarg`	list of distribution arguments
`p0`	probability zero
`raw`	logical - calculate raw moments?
`central`	logical - calculate central moments?
`coef`	logical - calculate coefficients (coefficient of variation, skewness and kurtosis)?
`distbounds`	distribution bounds (default set to c(-Inf, Inf))
`order`	vector of integers - raw moment orders

Examples

```
library(CoSMoS)

## Normal Distribution
moments('norm', list(mean = 2, sd = 1))

## Pareto type II
scale <- 1
shape <- .2

moments(dist = 'paretoII',
        distarg = list(shape = shape,
                       scale = scale))
```

`ParetoII` *Pareto type II distribution*

Description

Provides density, distribution function, quantile function, random value generation and raw moments of order *r* for the Pareto type II distribution.

Usage

```
dparetoII(x, scale, shape, log = FALSE)

pparetoII(q, scale, shape, lower.tail = TRUE, log.p = FALSE)

qparetoII(p, scale, shape, lower.tail = TRUE, log.p = FALSE)

rparetoII(n, scale, shape)

mparetoII(r, scale, shape)
```

Arguments

`x, q`	vector of quantiles.
`scale, shape`	scale and shape parameters; the shape argument cannot be a vector (must have length one).
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
`p`	vector of probabilities.
`n`	number of observations. If `length(n)` > 1, the length is taken to be the number required.
`r`	raw moment order

Examples

```
## plot the density

ggplot(data.frame(x = c(0, 20)),
       aes(x)) +
  stat_function(fun = dparetoII,
                args = list(scale = 1,
                            shape = .3),
                colour = 'royalblue4') +
  labs(x = '',
       y = 'Density') +
  theme_classic()
```

`plot.acti` *AutoCorrelation Transformation Function visualisation*

Description

Visualizes the autocorrelation tranformation integral (there are two possible methods for plotting - base graphics and ggplot2 package).

Usage

```
## S3 method for class 'acti'
plot(x, ...)
```

Arguments

`x`	`fitactf` result object
`...`	other arguments

Examples

```
library(CoSMoS)

## choose the marginal distribution as Pareto type II with corresponding parameters
dist <- 'paretoII'
distarg <- list(scale = 1, shape = .3)

## estimate rho 'x' and 'z' points using ACTI
p <- actpnts(margdist = dist, margarg = distarg, p0 = 0)

## fit ACTF
fit <- fitactf(p)

## plot the results
```

```
plot(fit)
plot(fit, main = 'Pareto type II distribution \nautocorrelation tranformation')
```

`plot.checkTS` *Plot method for check results*

Description

Plot method for check results.

Usage

```
## S3 method for class 'checkTS'
plot(x, ...)
```

Arguments

`x`	check result
`...`	other args

Examples

```
library(CoSMoS)

## check your generated timeseries
x <- generateTS(margdist = 'burrXII',
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .15),
                acsvalue = acs(id = 'weibull',
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .25, TSn = 100)

chck <- checkTS(x)

plot(chck)
```

`plot.cosmosts` *Plot generated Timeseries*

Description

Visualizes Timeseries generated by the package CoSMoS.

Usage

```
## S3 method for class 'cosmosts'
plot(x, ...)
```

Arguments

`x`	`fitactf` result object
`...`	other arguments

Examples

```

library(CoSMoS)

## generate TS
ts <- generateTS(margdist = 'ggamma',
                 margarg = list(scale = 1,
                                shape1 = .8,
                                shape2 = .8),
                 acsvalue = acs(id = 'paretoII',
                                t = 0:30,
                               scale = 1,
                                shape = .75),
                 n = 1000,
                 p = 30,
                 TSn = 2)

## plot the TS
plot(ts)
```

`precip` *Hourly station precipitation data*

Description

Station details

- Name: Philadelphia International Airport
- Network ID: COOP:366889
- Latitude/Longitude: 39.87327°, -75.22678°
- Elevation: 3m

Usage

```
precip
```

Format

A data.table with 79633 rows and 2 variables:

date POSIXct format date/time

value precipitation totals

Details

more details can be found here.

Source

The National Oceanic and Atmospheric Administration (NOAA)

`quickTSPlot` *Quick visualization of basic timeseries properties*

Description

Return timeseries diagram, empirical density function, and empirical autocorrelation function.

Usage

```
quickTSPlot(TS, ci = 0.95)
```

Arguments

`TS`	timeseries to plot
`ci`	confidence interval around the zero autocorrelation value (default set to 0.95, i.e. 95% CI)

Examples

```
no <- 1000
ggamma_sim <- rggamma(n = no, scale = 1, shape1 = 1, shape2 = .5)
quickTSPlot(ggamma_sim)
```

`regenerateTS` *Bulk Timeseries generation*

Description

Resamples given timeseries.

Usage

```
regenerateTS(ts, TSn = 1)
```

Arguments

`ts`	generated timeseries using ARp
`TSn`	number of timeseries to be (re)generated

Details

You have used the `generateTS` function and you wish to generate more time series. Instead of rerunning `generateTS` you can use `regenerateTS`, which generates timeseries using the parameters previously calculated by the `generateTS` function, and thus it is faster.

Examples

```
library(CoSMoS)

## define marginal distribution and arguments with target
## autocorrelation structure
x <- generateTS(margdist = 'burrXII',
                margarg = list(scale = 1,
                               shape1 = .75,
                               shape2 = .25),
                acsvalue = acs(id = 'weibull',
                               t = 0:30,
                               scale = 10,
                               shape = .75),
                n = 1000, p = 30, p0 = .5, TSn = 3)

## generate new values with same parameters
r <- regenerateTS(x)

plot(r)
```

`sample.moments` *Estimation of sample moments*

Description

Estimation of sample moments.

Usage

```
sample.moments(x, na.rm = FALSE, raw = T, central = T, coef = T, order = 1:4)
```

Arguments

`x`	a numeric vector of values
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds
`raw`	logical - calculate raw moments?
`central`	logical - calculate central moments?
`coef`	logical - calculate coefficients (coefficient of variation, skewness and kurtosis)?
`order`	vector of integers - raw moment orders

Examples

```

library(CoSMoS)

x <- rnorm(1000)
sample.moments(x)

y <- rparetoII(1000, 10, .1)
sample.moments(y)
```

`stcfclayton` *Clayton SpatioTemporal Correlation Structure*

Description

Provides spatiotemporal correlation structure function based on Clayton copula. For more details on the parametric spatiotemporal correlation structures see section 2.3 and 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
stcfclayton(t, s, scfid, tcfid, copulaarg, scfarg, tcfarg)
```

Arguments

`t`	time lag
`s`	spatial lag (distance)
`scfid`	ID of the spatial (marginal) correlation structure (e.g. weibull)
`tcfid`	ID of the temporal (marginal) correlation structure (e.g. weibull)
`copulaarg`	parameter of the Clayton copula linking the marginal correlation structures
`scfarg`	parameters of spatial (marginal) correlation structure
`tcfarg`	parameters of temporal (marginal) correlation structure

References

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi: 10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d - 1),
                  0:(d - 1))

## get the STCS
wc <- stcfclayton(t = st[, 1],
                  s = st[, 2],
                  scfid = 'weibull',
                  tcfid = 'weibull',
                  copulaarg = 2,
                  scfarg = list(scale = 20,
                                shape = 0.7),
                  tcfarg = list(scale = 1.1,
                                shape = 0.8))

## visualize the STCS
wc.m <- matrix(wc,
               nrow = d)

persp3D(z = wc.m, x = 1: nrow(wc.m), y = 1:ncol(wc.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))
```

`stcfgneiting14` *Gneiting-14 SpatioTemporal Correlation Structure*

Description

Provides spatiotemporal correlation structure function proposed by Gneiting (2002; Eq.14 at p. 593).

Usage

```
stcfgneiting14(t, s, a, c, alpha, beta, gamma, tau)
```

Arguments

`t`	time lag
`s`	spatial lag (distance)
`a`	nonnegative scaling parameter of time
`c`	nonnegative scaling parameter of space
`alpha`	smoothness parameter of time. Valid range: $(0, 1]$
`beta`	space-time interaction parameter. Valid range: $[0, 1]$
`gamma`	smoothness parameter of space. Valid range: $(0, 1]$
`tau`	space-time interaction parameter. Valid range: ≥ 1 (for 2-dimensional fields)

References

Gneiting, T. (2002). Nonseparable, Stationary Covariance Functions for Space-Time Data, Journal of the American Statistical Association, 97:458, 590-600, doi: 10.1198/016214502760047113

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d - 1),
                  0:(d - 1))

## get the STCS
g14 <- stcfgneiting14(t = st[, 1],
                      s = st[, 2],
                      a = 1/50,
                      c = 1/10,
                      alpha = 1,
                      beta = 1,
                      gamma = 0.5,
                      tau = 1)
```

```
## visualize the STCS

g14.m <- matrix(g14,
                nrow = d)

persp3D(z = g14.m, x = 1: nrow(g14.m), y = 1:ncol(g14.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))
```

`stcfgneiting16` *Gneiting-16 SpatioTemporal Correlation Structure*

Description

Provides spatiotemporal correlation structure function proposed by Gneiting (2002; Eq.16 at p. 594).

Usage

```
stcfgneiting16(t, s, a, c, alpha, beta, nu, tau)
```

Arguments

`t`	time lag
`s`	spatial lag (distance)
`a`	nonnegative scaling parameter of time
`c`	nonnegative scaling parameter of space
`alpha`	smoothness parameter of time. Valid range: $(0, 1]$
`beta`	space-time interaction parameter. Valid range: $[0, 1]$
`nu`	smoothness parameter of space. Valid range: > 0
`tau`	space-time interaction parameter. Valid range: ≥ 1 (for 2-dimensional fields)

References

Gneiting, T. (2002). Nonseparable, Stationary Covariance Functions for Space-Time Data, Journal of the American Statistical Association, 97:458, 590-600, doi: 10.1198/016214502760047113

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d - 1),
                  0:(d - 1))

## get the STCS
g16 <- stcfgneiting16(t = st[, 1],
                      s = st[, 2],
                      a = 1/50,
                      c = 1/10,
                      alpha = 1,
                      beta = 1,
                      nu = 0.5, tau = 1)

## visualize the STCS

g16.m <- matrix(g16,
                nrow = d)

persp3D(z = g16.m, x = 1: nrow(g16.m), y = 1:ncol(g16.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))
```

`stcs` *SpatioTemporal Correlation Structure*

Description

Provides a parametric function that describes the values of the linear spatiotemporal autocorrelation up to desired lags. For more details on the parametric spatiotemporal correlation structures see section 2.3 and 2.4 in Papalexiou and Serinaldi (2020).

Usage

```
stcs(id, ...)
```

Arguments

`id`	spatiotemporal correlation structure ID
`...`	additional arguments (t as time lag, s as spatial lag (distance), and stcs parameters)

References

Papalexiou, S.M., Serinaldi, F. (2020). Random Fields Simplified: Preserving Marginal Distributions, Correlations, and Intermittency, With Applications From Rainfall to Humidity. Water Resources Research, 56(2), e2019WR026331, doi: 10.1029/2019WR026331

Papalexiou, S.M., Serinaldi, F., Porcu, E. (2021). Advancing Space-Time Simulation of Random Fields: From Storms to Cyclones and Beyond. Water Resources Research, 57, e2020WR029466, doi: 10.1029/2020WR029466

Examples

```
library(plot3D)

## specify grid of spatial and temporal lags
d <- 31
st <- expand.grid(0:(d-1),
                  0:(d-1))

## get the STCS
wc <- stcs("clayton",
           t = st[, 1],
           s = st[, 2],
           scfid = 'weibull',
           tcfid = 'weibull',
           copulaarg = 2,
           scfarg = list(scale = 20,
                         shape = 0.7),
           tcfarg = list(scale = 1.1,
                         shape = 0.8))

g14 <- stcs("gneiting14",
            t = st[, 1],
            s = st[, 2],
            a = 1/50,
            c = 1/10,
            alpha = 1,
            beta = 1,
            gamma = 0.5,
            tau = 1)

g16 <- stcs("gneiting16",
            t = st[, 1],
            s = st[, 2],
            a = 1/50,
            c = 1/10,
            alpha = 1,
            beta = 1,
            nu = 0.5,
            tau = 1)

## note: for nu = 0.5 stcfgneiting16 is equivalent to
```

```
## stcfgneiting14 with gamma = 0.5

## visualize the STCS

wc.m <- matrix(wc,
               nrow = d)

persp3D(z = wc.m, x = 1: nrow(wc.m), y = 1:ncol(wc.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))

g14.m <- matrix(g14,
                nrow = d)

persp3D(z = g14.m, x = 1: nrow(wc.m), y = 1:ncol(wc.m),
        expand = 1, main = "", scale = TRUE, facets = TRUE,
        xlab="Time lag", ylab = "Distance", zlab = "STCF",
        colkey = list(side = 4, length = 0.5), phi = 20, theta = 120,
        resfac = 5,  col= gg2.col(100))
```

Index