How to produce precipitation-weighted annual average isoscapes in IsoriX?

The IsoriX core team

2017-07-07

Welcome to **IsoriX**, in this vignette we present the steps required for you to build an annual average isoscape weighted by the amount of monthly precipitation.

Before starting

Please read the vignette ***Workflow***, if you haven't done so. You can access it by simply typing:

```
vignette("Workflow", package = "IsoriX")
```

Note that the current vignette, like the one introducing the workflow, takes some time to run. It has thus not been compiled by CRAN but by us. Again, due to constraints on how big this document could be, we had to reduce a lot the resolution of the figures.

Before starting, don't forget to load our package:

```
library(IsoriX)
```

```
##
##  IsoriX version 0.6 is loaded!
##
##  Many functions and objects have changed names since the version 0.4.
##  This is to make IsoriX more intuitive for you to use.
##  We will do our best to limit changes in names in the future!!
##
##  Type:
##     * ?IsoriX for a short description.
##     * browseVignettes(package = 'IsoriX') for tutorials.
##     * news(package = 'IsoriX') for news.
```

Step 1 - Select the isoscape data

Start by selecting the precipitation data needed for you to build an isoscape. In this example, we will consider all the data available in `GNIPDataDE`. The difference with what we did in the vignette ***Workflow*** is that here the function `queryGNIP` is called with the argument `split.by = "month"`, which lead to data aggregated across years (as before), but not across months.

```
GNIPDataDE12 <- queryGNIP(data = GNIPDataDE, split.by = "month")
```

The dataset we created contains up to twelve different rows per location (i.e. one per month if records are available for all twelve months) instead of the single one:

```
knitr::kable(head(GNIPDataDE12, 15L))
```

stationID	isoscape.value	var.isoscape.value	n.isoscape.value	lat	long	elev	month
ARKONA	-69.13636	253.07997	11	54.67	13.43	42	1
ARKONA	-73.05455	238.47221	11	54.67	13.43	42	2

stationID	isoscape.value	var.isoscape.value	n.isoscape.value	lat	long	elev	month
ARKONA	-65.60455	312.38313	11	54.67	13.43	42	3
ARKONA	-63.04636	214.81135	11	54.67	13.43	42	4
ARKONA	-51.00000	83.21076	11	54.67	13.43	42	5
ARKONA	-46.87583	118.54494	12	54.67	13.43	42	6
ARKONA	-48.43083	48.32270	12	54.67	13.43	42	7
ARKONA	-54.38182	78.66940	11	54.67	13.43	42	8
ARKONA	-57.02000	211.12920	11	54.67	13.43	42	9
ARKONA	-59.22545	216.15013	11	54.67	13.43	42	10
ARKONA	-71.56273	170.03014	11	54.67	13.43	42	11
ARKONA	-74.99455	128.47743	11	54.67	13.43	42	12
ARTERN	-79.14375	505.39862	16	51.37	11.29	164	1
ARTERN	-77.76875	366.71829	16	51.37	11.29	164	2
ARTERN	-71.66875	309.94096	16	51.37	11.29	164	3

Step 2 - Fit the geostatistical models

We will now fit not one pair of models as during the ***Workflow*** but twelve pairs of models. We indeed want to fit one mean model and one residual dispersion model for each of the twelve months of a year. Each of the twelve pairs of models are technically fitted independently, but to save you the manual labor of calling twelve times the function `isofit`, we have created the function `isomultifit` that does that for you. This latter function also combines all fitted models in one object of class `multiisofit` which other functions will recognize. As `isofit`, `isomultifit` can fit several model structures, but we will restrict the demonstration to a single example.

```
GermanyFit12 <- isomultifit(iso.data = GNIPDataDE12, split.by = "month")
```

We check that all models are there:

```
names(GermanyFit12$multi.fits)
```

```
##  [1] "month_1"  "month_2"  "month_3"  "month_4"  "month_5"  "month_6"
##  [7] "month_7"  "month_8"  "month_9"  "month_10" "month_11" "month_12"
```

You could then look at the output of a given model (here, January) by simply typing `GermanyFit12$multi.fits$month_1`.

Step 3 - Prepare the elevation raster

As for the ***Workflow***, we prepare the elevation raster from the tif file we downloaded (see ***Workflow*** for details):

```
library(raster)
```

```
## Warning: package 'sp' was built under R version 3.4.1
```

```
elevationraster <- raster("../vignette_workflow/gmted2010_30mn.tif")
elev <- relevate(elevation.raster = elevationraster, isofit = GermanyFit12)
```

```
## class       : RasterLayer
## dimensions  : 950, 1148, 1090600  (nrow, ncol, ncell)
## resolution  : 0.008333333, 0.008333333  (x, y)
## extent      : 5.816527, 15.38319, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : /private/var/folders/r4/nmf9vqyj7pz968sk2vrtmbvm0000gn/T/RtmpVAr2ez/raster/r_tmp_2017-
## names       : gmted2010_30mn
```

```
## values      : -179, 3230  (min, max)
```

Step 4 - Prepare the precipitation rasters

We now need the rasters containing the average precipitation amount for each month of the year and each location from the elevation raster. We start by downloading such file (mind that the file is ca. 1Gb and takes a while to download):

```
getprecip()
```

```
## the file wc2.0_30s_prec.zip is already present in /Users/alex/Dropbox/Boulot/Mes_projets_de_recherch
```

```
## [1] the file seems OK (md5sums do match)
```

Note that if the zip file is already in the working directory (as it is the case here), it won't be downloaded again.

We then resize the RasterBrick obtained to the size of the elevation raster:

```
precipitations <- prepcipitate(elevation.raster = elev)
```

Step 5 - Build the isoscape

To build the precipitation-weighted annual average isoscapes stemming from `GermanyFit12` we need to use the function `isomultiscape`. This function is a wrapper to `isoscape` handling several models at once. This is also the function to which we need to provide the prepared precipitation data:

```
isoscapes <- isomultiscape(elevation.raster = elev,
                           isofit = GermanyFit12,
                           weighting = precipitations)
```

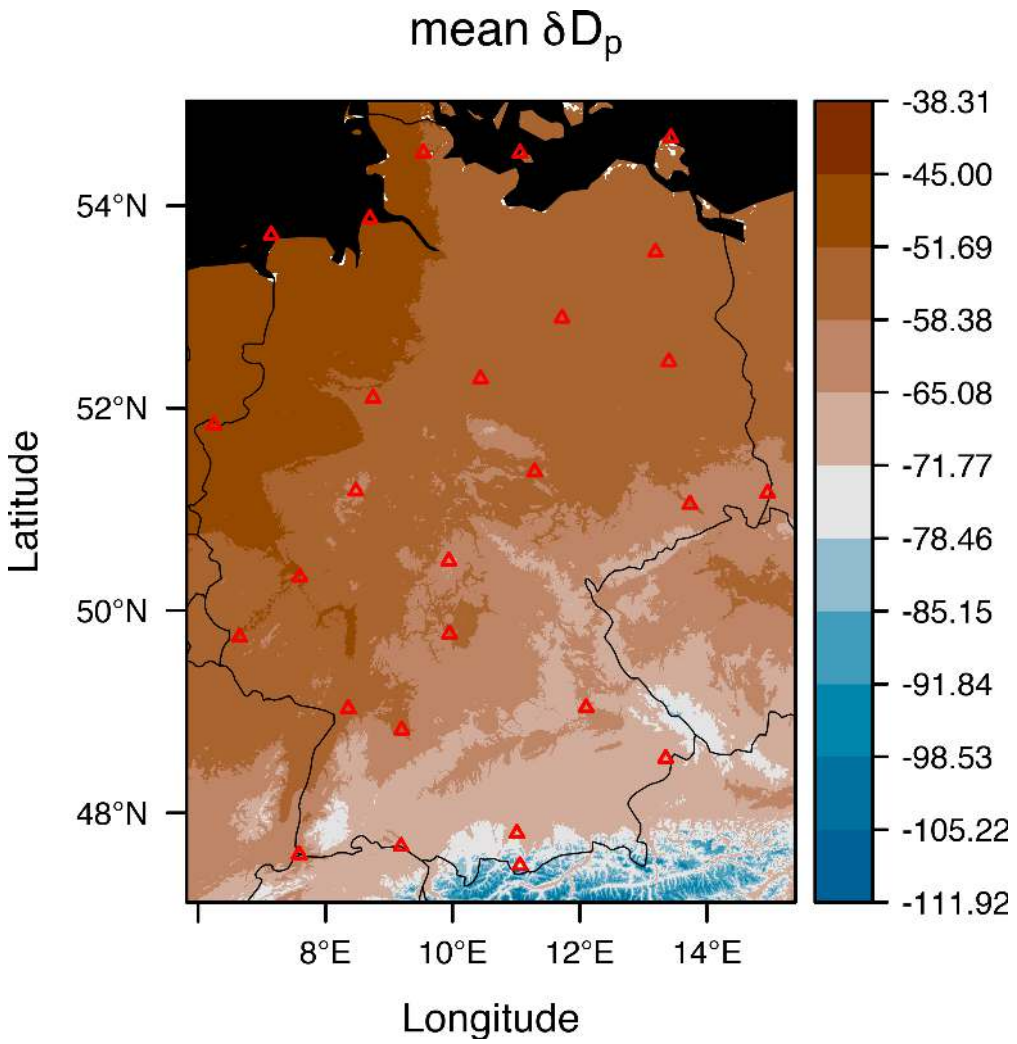
```
## Warning in spaMM::predict.HLfit(object = isofit$disp.fit, newdata = xs, :
## Prior weights are not taken in account in residVar computation.
```

```
## Warning in .calc_logdisp_cov_new(object = object, dvdloglamMat =
## dvdloglamMat, : phi dispVar component not yet available for phi model !=
## ~1.
```

Step 6 - Plotting the isoscapes

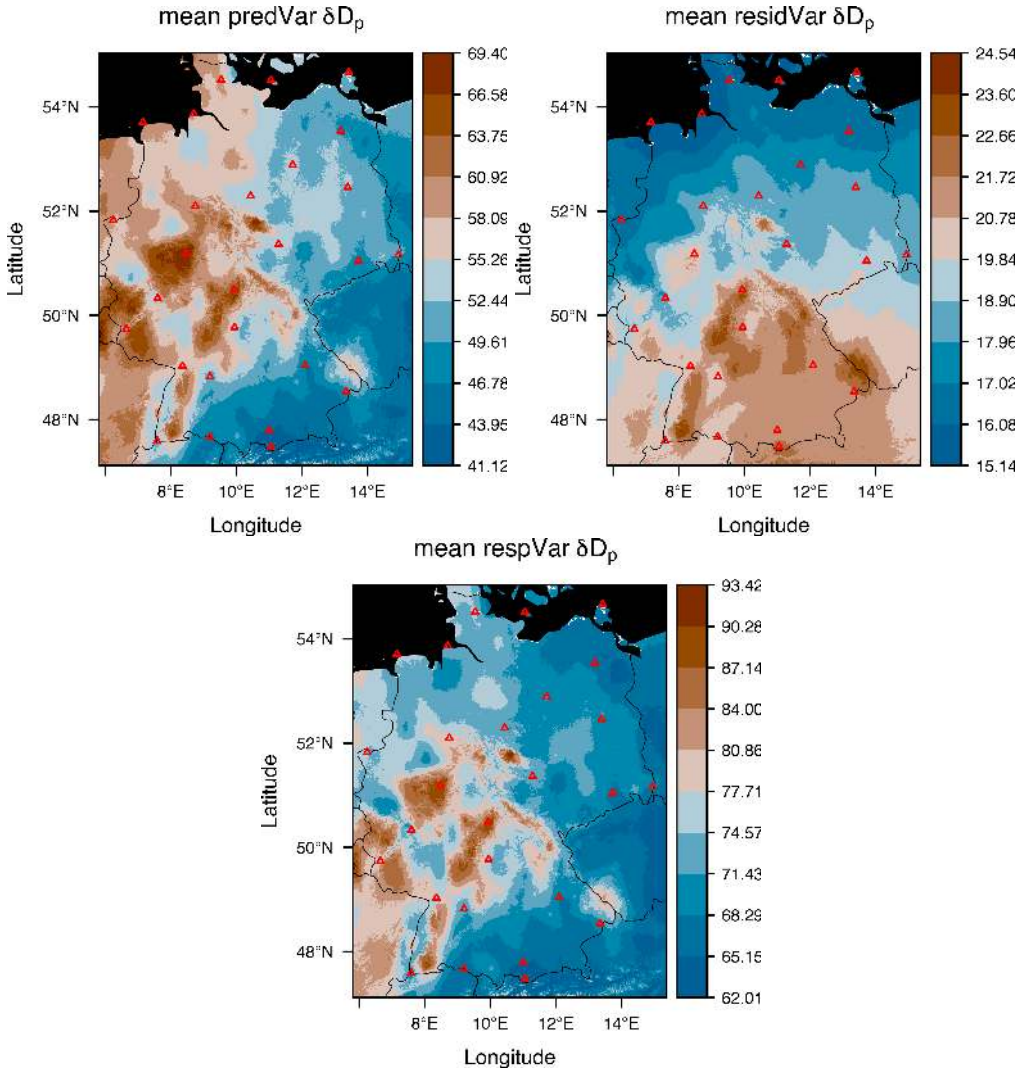
We can finally plot the precipitation-weighted annual average isoscape as we did in the ***Workflow***:

```
plot(x = isoscapes)
```



As for any isoscape fitted with **IsoriX** we can also plot the isoscape for the prediction variance, the one for the residual variance, and the one for the response variance:

```
plot(x = isoscapes, which = "mean.predVar")
plot(x = isoscapes, which = "mean.residVar")
plot(x = isoscapes, which = "mean.respVar")
```



Does the isoscapes differ from the one not accounting for precitation?

Above two differences were introduced compared to the simple approach we followed during the ***Workflow***. First, models were fitted by month; second, they were weighted by precipitation amounts before the aggregation. We can simply study the influence of such additional steps by comparing the isoscapes produced by different workflows.

We will here compare the isoscape for point predictions produced between the two different workflow. To do so, we need to produce a simple isoscape for Germany using the simple workflow introduced in the vignette ***Workflow***:

```
GNIPDataDEagg <- queryGNIP(data = GNIPDataDE)
GermanyFit    <- isofit(iso.data = GNIPDataDEagg)
isoscape      <- isoscape(elevation.raster = elev, isofit = GermanyFit)
```

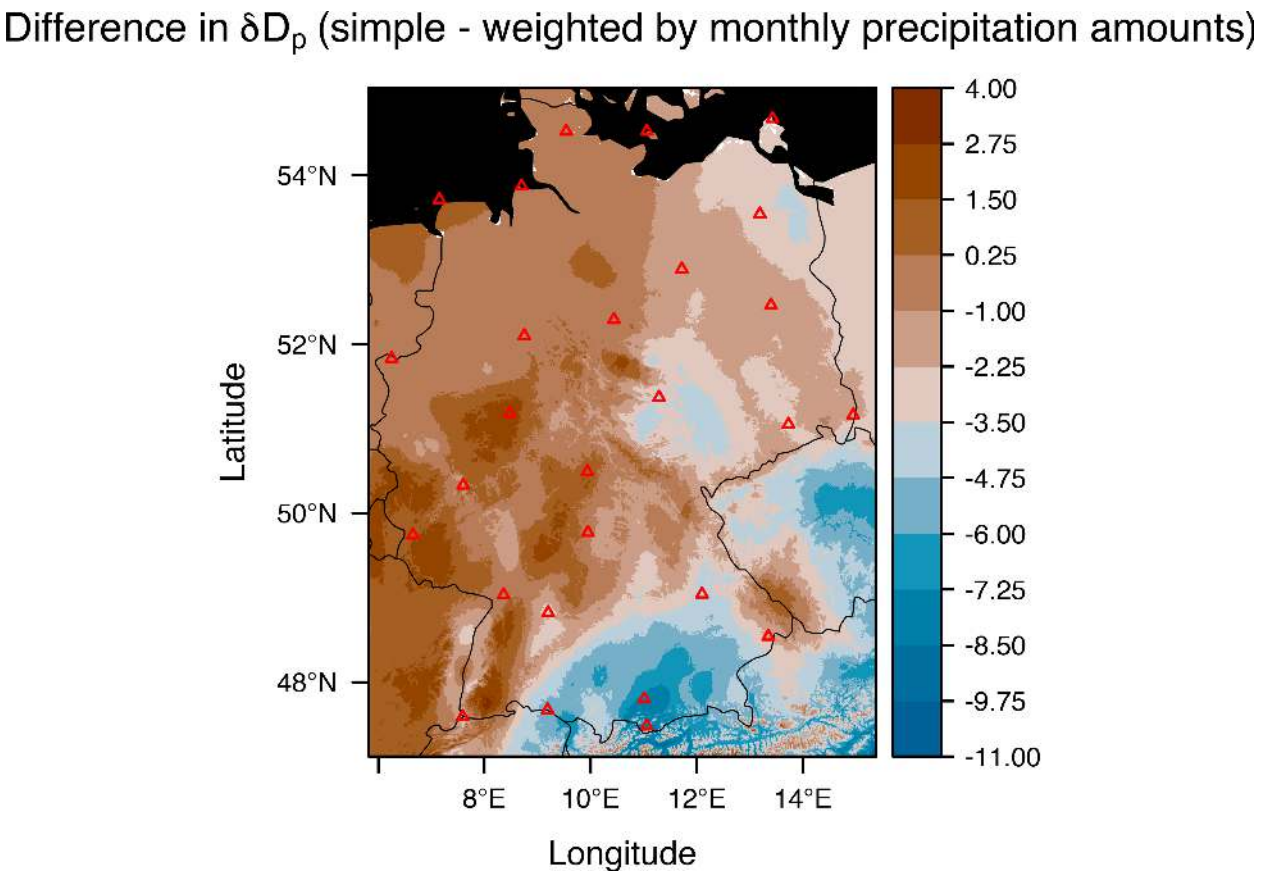
We now compute the difference between the isoscapes produced by the two different workflows (mind that the two isoscapes must have same resolution and extent to do that directly, which is the case here):

```
isoscape.diff <- isoscape  ## We create a new object of class isoscape
isoscape.diff$isoscape <- isoscape$isoscape - isoscapes$isoscape  ## We replace the isoscape by
```

```
                                                    ## the difference in isoscapes
```

You could plot the point predictions as before, but we choose to add one small step to adjust the title:

```
plotdiff <- plot(x = isoscape.diff,
                 palette = list(step = 1.25, range = c(-11, 4), n.labels = Inf),
                 plot = FALSE)
plotdiff$main <- "Difference in" ~ delta * D[p] ~ "(simple - weighted by monthly precipitation amounts)"
plotdiff
```



The End

That is all for now! Here are the information of the R session we used:

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.5
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] raster_2.5-8 sp_1.2-5     IsoriX_0.6   knitr_1.16
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.11        magrittr_1.5        rasterVis_0.41
##  [4] MASS_7.3-47         viridisLite_0.2.0   lattice_0.20-35
##  [7] stringr_1.2.0       highr_0.6           tools_3.4.0
## [10] parallel_3.4.0      rgdal_1.2-8         grid_3.4.0
## [13] spaMM_2.1.6         nlme_3.1-131        latticeExtra_0.6-28
## [16] htmltools_0.3.6     yaml_2.1.14         rprojroot_1.2
## [19] digest_0.6.12       Matrix_1.2-10       RColorBrewer_1.1-2
## [22] nloptr_1.0.4        codetools_0.2-15    evaluate_0.10.1
## [25] rmarkdown_1.6       proxy_0.4-17        stringi_1.1.5
## [28] compiler_3.4.0      backports_1.1.0     hexbin_1.27.1
## [31] Cairo_1.5-9         zoo_1.8-0
```