IsoriX: an R package for building isoscapes and inferring the geographic origin of organisms using mixed models

Courtiol A, Rousset F, Kramer-Schadt S

14 September 2016

Table of contents

1. Generalities about IsoriX
2. The IsoriX workflow in 6 steps
3. Analysing the Dechamm dataset
4. Future of IsoriX

Table of contents

1. Generalities about IsoriX
2. The IsoriX workflow in 6 steps
3. Analysing the Dechamm dataset
4. Future of IsoriX

Why IsoriX?

Benefits:

- reproducible workflow
- simple
- highly customizable
- compatible with GIS
- cutting edge statistical methods

Limits:

- R knowledge required
- constrained model structure
- not multivariate (1 isotope only)
- not Bayesian (prior information not considered)

How to start IsoriX?

```r
install.packages("IsoriX")
```

```r
library(IsoriX)

##
##  ##########################################
##  #                                        #
##  #    IsoriX (version 0.4-1) is loaded!   #
##  #                                        #
##  #  Type ?IsoriX for a short description  #
##  #                                        #
##  #  Type browseVignettes(package='IsoriX')  #
##  #    for details about how this package  #
##  #        works and how to use it         #
##  #                                        #
##  #  Type news(package='IsoriX') for news  #
##  #                                        #
##  ##########################################
```

What is IsoriX?

An interface between packages!
(without help files: only 1513 lines of codes)

- **spaMM**, numDeriv (for statistical computation)
- **raster**, sp, proxy (for GIS)
- **rasterVis**, latticeExtra, grid (for plotting)
- **RandomFields** (for simulations)

```r
## Package dependencies
tools::package_dependencies("IsoriX")
```

```
##  [1] "stats"        "utils"        "graphics"     "grDevices"
##  [5] "spaMM"        "numDeriv"     "raster"       "sp"
##  [9] "proxy"        "rasterVis"    "latticeExtra" "grid"
## [13] "RandomFields"
```

Content of IsoriX: datasets

```r
data(package="IsoriX")$results[, c("Item", "Title")]
```

```
##       Item                    Title
##  [1,] "GNIP_World"            "World-wide weather station data"
##  [2,] "GNIP_Europe"           "European weather station data"
##  [3,] "GNIP_Model"            "The fitted isoscape model for GNIP_Europe"
##  [4,] "elevationrastersmall" "The raster of world elevation"
##  [5,] "calibrationdata"       "Calibration dataset"
##  [6,] "assignmentdata"        "Assignment dataset"
##  [7,] "isoscapepalette"       "Colour palettes for plotting"
##  [8,] "isorixpalette"         "Colour palettes for plotting"
##  [9,] "oceanmask"             "Mask of world oceans"
## [10,] "worldcountries"        "Borders of world countries"
```

Content of IsoriX: functions

```r
ls("package:IsoriX")
```

```
##  [1] "Isofit"           "Isorix"           "Isosim"
##  [4] "GetElev"          "Calibfit"         "Isoscape"
##  [7] "RElevate"         "QueryGNIP"        "plot.isofit"
## [10] "plot.isorix"      "print.isofit"     "print.isorix"
## [13] "plot.calibfit"    "plot.isoscape"    "print.calibfit"
## [16] "print.isoscape"   "summary.isofit"   "summary.isorix"
## [19] "summary.calibfit" "summary.isoscape"
```

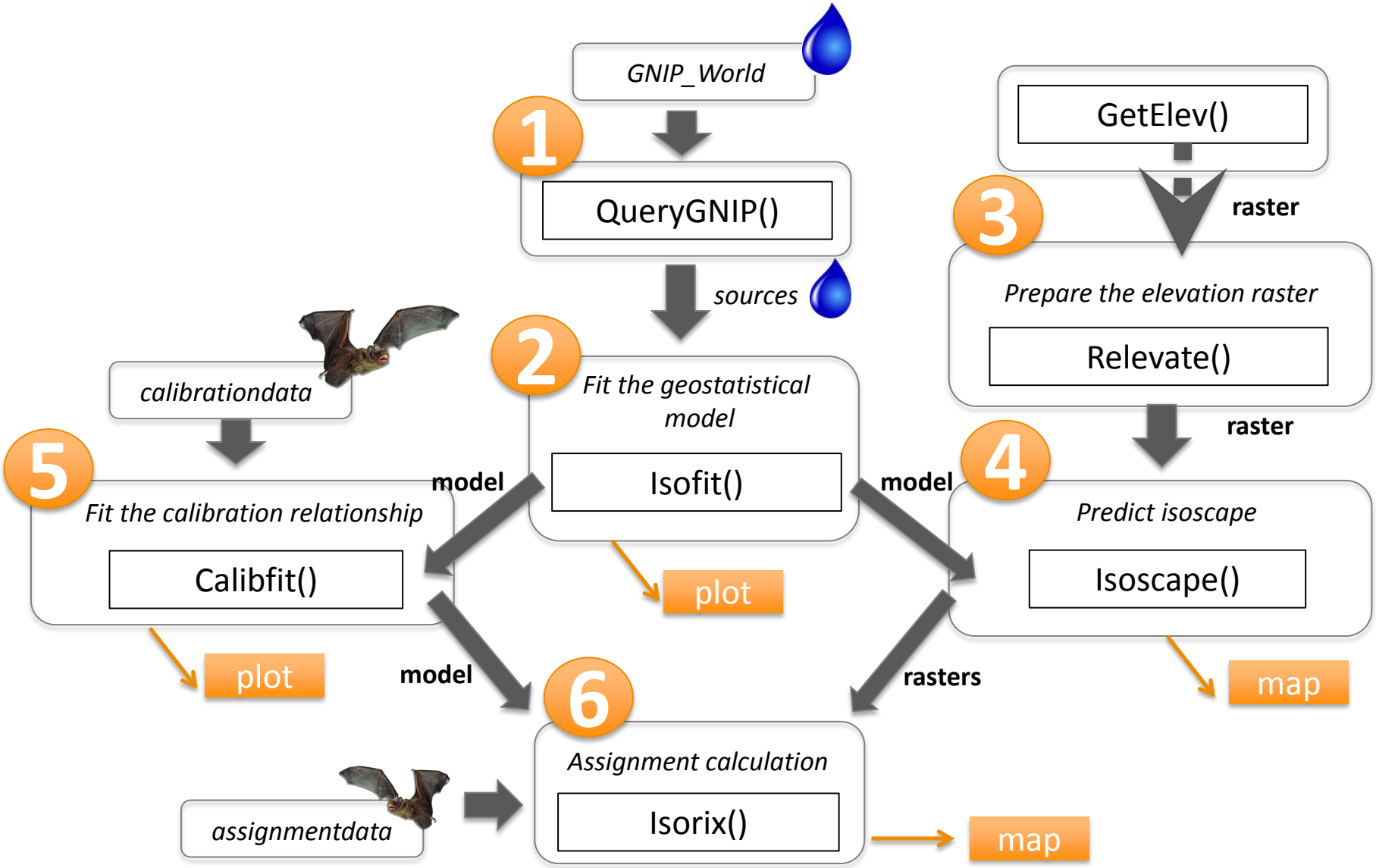
Table of contents

1. Generalities about IsoriX
2. The IsoriX workflow in 6 steps
3. Analysing the Dechamm dataset
4. Future of IsoriX

Workflow

1. Prepare the source data with `QueryGNIP()`
2. Fit the Isoscape model with `Isofit()`
3. Prepare the elevation raster with `GetElev()` & `RElevate()`
4. Predict the isoscape with `Isoscape()`
5. Fit the calibration function with `Calibfit()`
6. Perform the assignment with `Isorix()`

The assignment workflow in IsoriX 0.4-1



Step 1. Prepare the source data

Load GNIP_World:

```
## load the raw data
data(GNIP_World)

## display first 6 rows
head(GNIP_World)
```

lat	long	elev	isoscape.value	year	month	stationID
78.25	11.91667	7	-72.8	1990	1	52950
78.25	11.91667	7	-67.6	1990	2	52950
78.25	11.91667	7	-107.0	1990	3	52950
78.25	11.91667	7	-77.8	1990	4	52950
78.25	11.91667	7	-77.1	1990	5	52950
78.25	11.91667	7	-67.3	1990	6	52950

```
## size of the dataset
dim(GNIP_World)

## [1] 54513     7

## number of unique weather stations
nlevels(GNIP_World$stationID)

## [1] 858
```

Step 1. Prepare the source data

`QueryGNIP()` filters and aggregates the data:

```
QueryGNIP(
        data,
        month.min = 1,
        month.max = 12,
        year.min,
        year.max,
        long.min ,
        long.max,
        lat.min,
        lat.max,
        prop.random = 0,
        random.level = "station"
        )
```

Step 1. Prepare the source data

`QueryGNIP()` filters and aggregates the data:

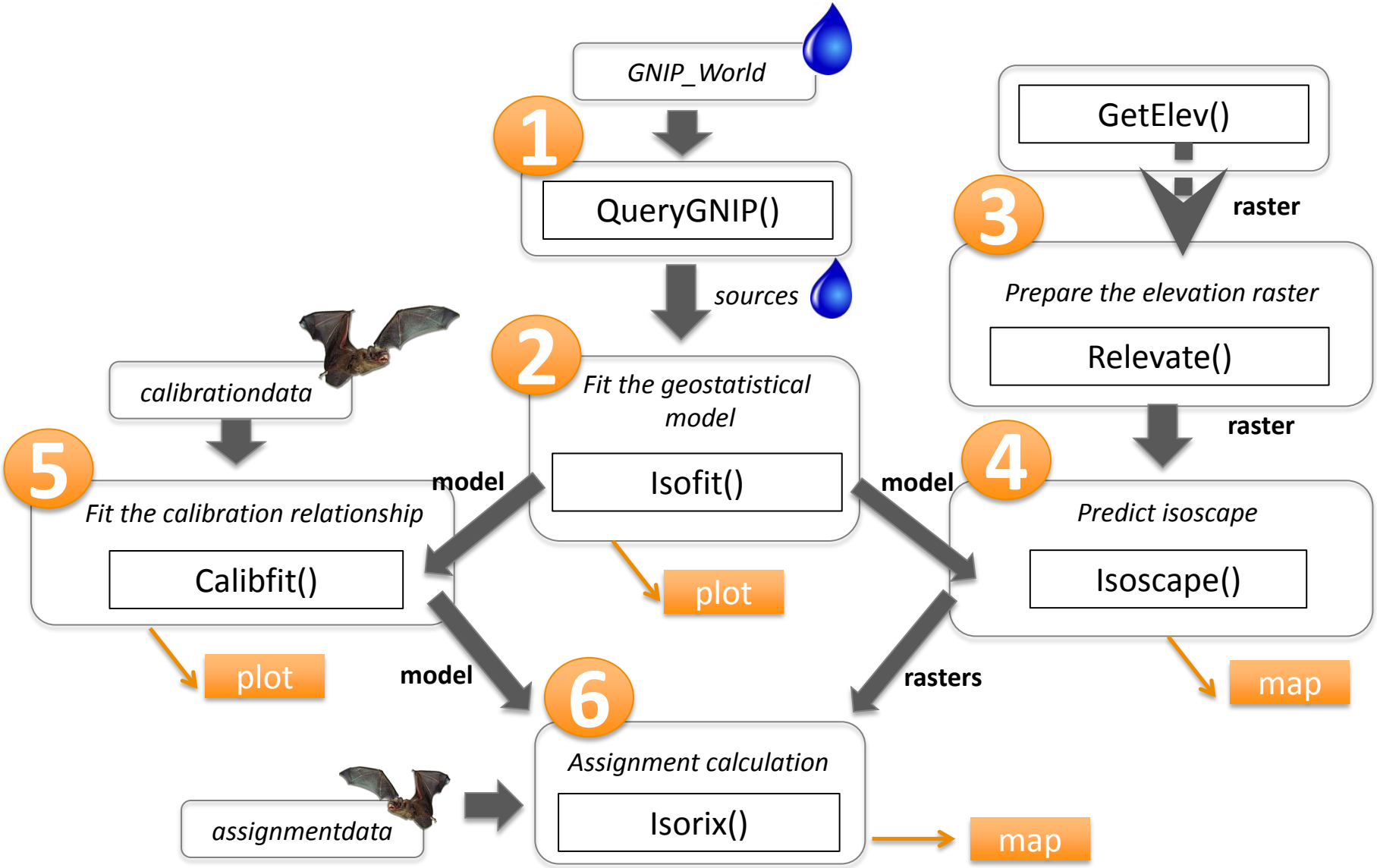
```
## using QueryGNIP
sources_Europe <- QueryGNIP(data=GNIP_World, long.min = -30, long.max = 60,
        lat.min = 30, lat.max = 70)

## Warning in FUN(X[[i]], ...):  some elevation values are not unique but should
be, so the first element was taken among:  65 190
```

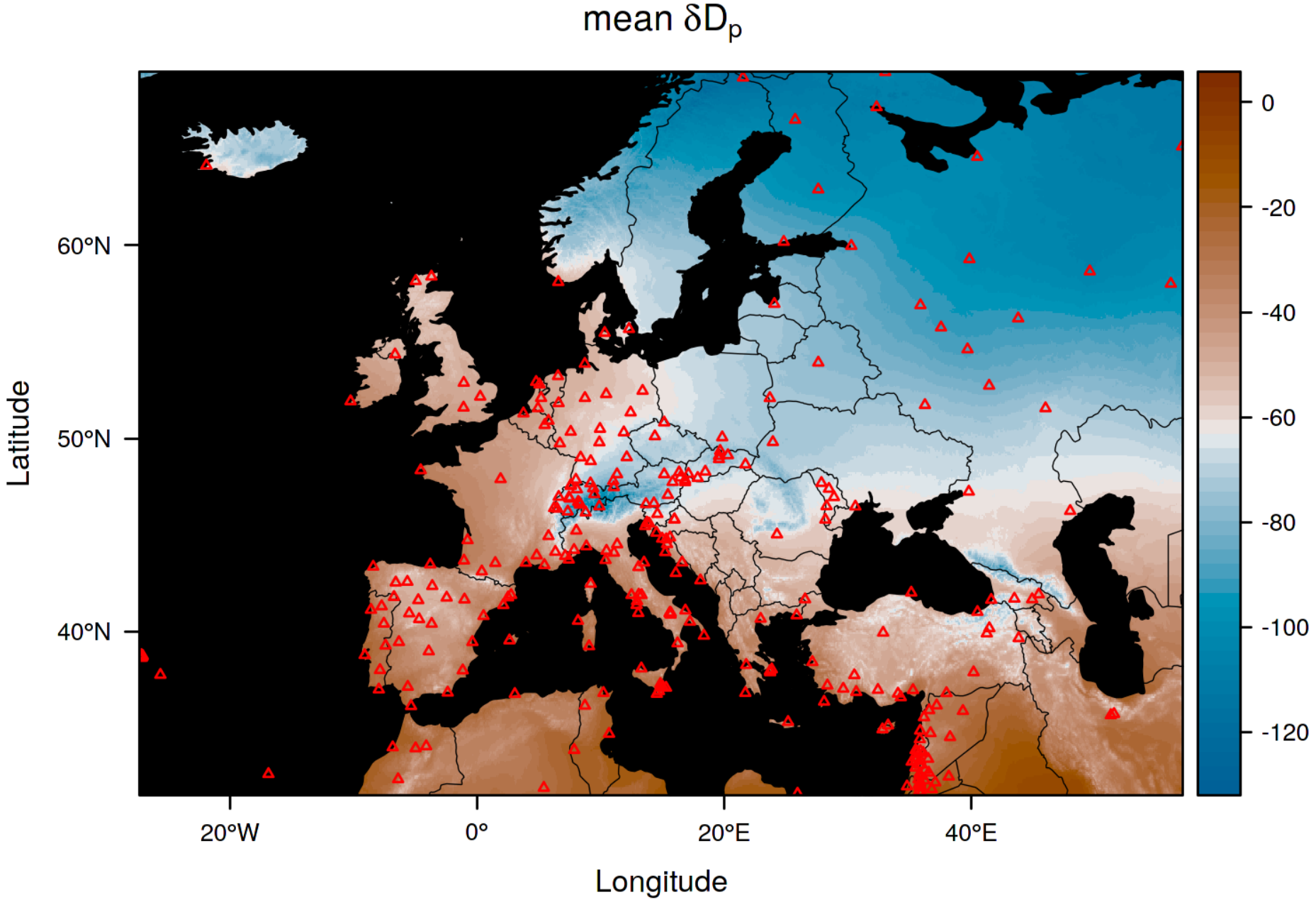
```
head(sources_Europe, n=4)
```

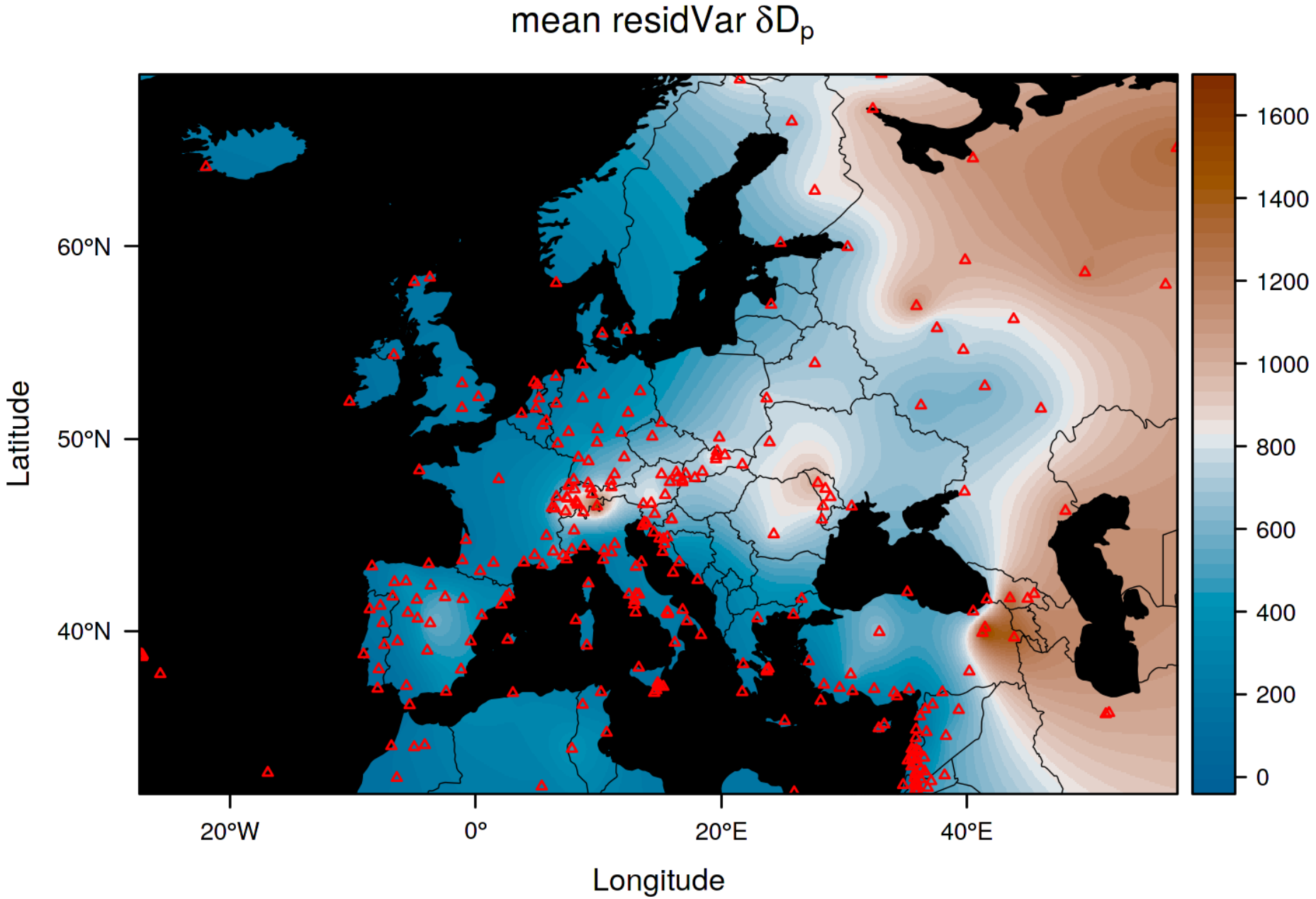
stationID	isoscape.value	var.isoscape.value	n.isoscape.value	lat	long	el
18963	-4.189474	472.4396	19	30.08000	31.28000	3
19005	-25.282353	207.0240	17	30.27000	35.58000	130
19022	0.480000	721.8720	5	30.33755	36.14619	87
19245	-9.829412	302.6347	17	31.08000	33.83000	3

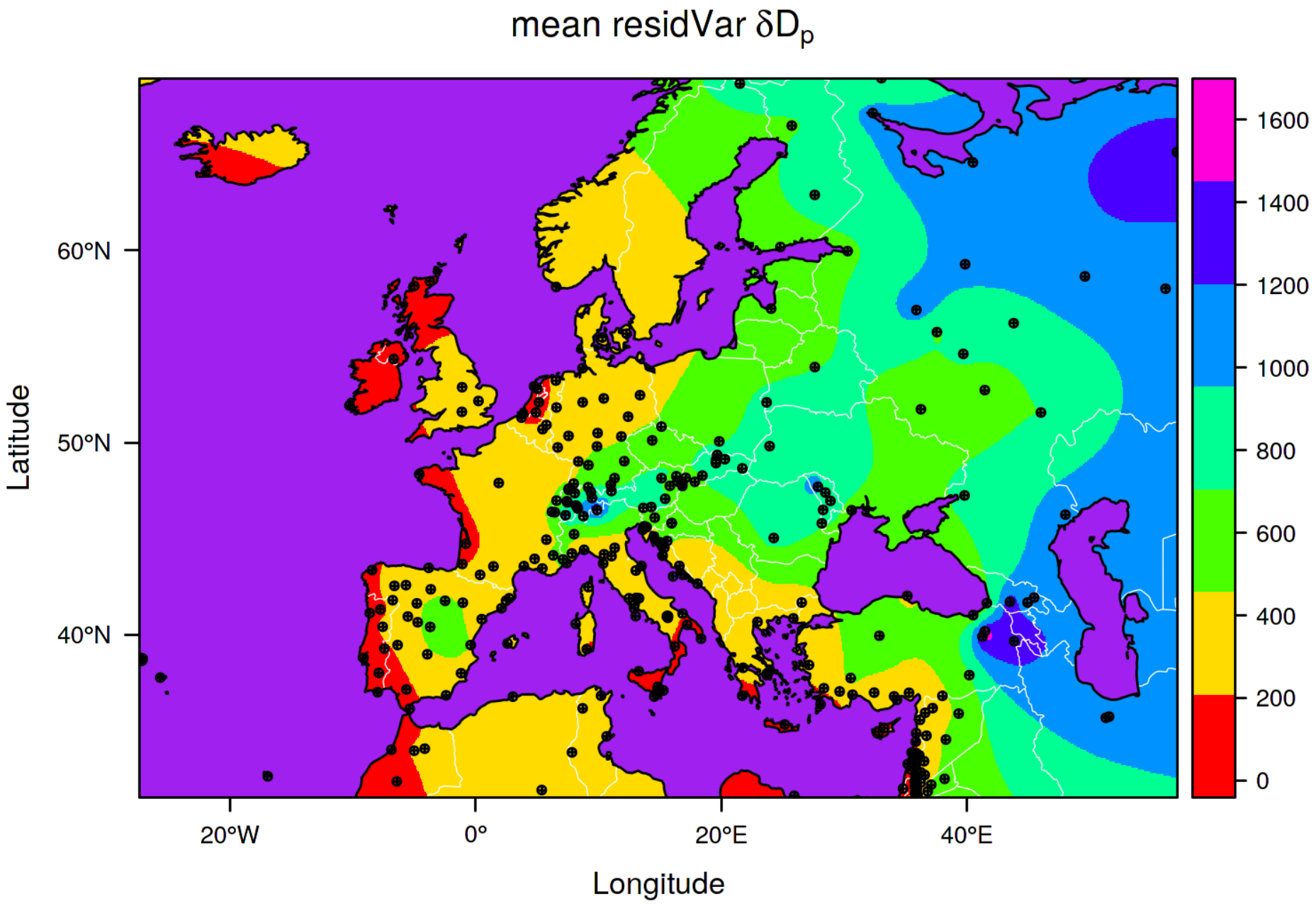
```
## we check variables
range(sources_Europe$isoscape.value)

## [1] -122.4757    0.4800

range(sources_Europe$elev)

## [1] -224 2210
```

The assignment workflow in IsoriX 0.4-1




mean δD$_p$
60°N
50°N
40°N
Latitude
20°W
0°
20°E
40°E
Longitude
0
-20
-40
-60
-80
-100
-120


mean residVar δD$_p$
60°N
50°N
40°N
Latitude
20°W
0°
20°E
40°E
Longitude
1600
1400
1200
1000
800
600
400
200
0


mean residVar δD$_p$
60°N
50°N
40°N
Latitude
20°W
0°
20°E
40°E
Longitude
1600
1400
1200
1000
800
600
400
200
0

Step 2. Fit the isoscape

The statistical model behind the isoscape:

- Mean model (LMM):

$$\texttt{isoscape.value}_{gi} = \texttt{Fix}_g + \texttt{Random}_g + \texttt{Error}_{gi}$$

Step 2. Fit the isoscape

The statistical model behind the isoscape:

- Mean model (LMM):

$$\texttt{isoscape.value}_{gi} = \texttt{Fix}_g + \texttt{Random}_g + \texttt{Error}_{gi}$$

- where

$$\texttt{Fix}_g = \beta_0 + \beta_{\texttt{lat.abs}} \times \texttt{lat.abs}_g + \beta_{\texttt{elev}} \times \texttt{elev}_g$$
$$\texttt{Random}_g = \texttt{RandomSpatial}_g + \texttt{RandomUncorr}_g$$

and

$$\texttt{Correlation}(\texttt{RandomSpatial}_a, \texttt{RandomSpatial}_b) = \texttt{Matern}(d_{ab}, \rho, \nu)$$

Step 2. Fit the isoscape

The statistical model behind the isoscape:

- Mean model (LMM):

$$\texttt{isoscape.value}_{gi} = \texttt{Fix}_g + \texttt{Random}_g + \texttt{Error}_{gi}$$

- where

$$\begin{aligned} \texttt{Fix}_g &= \beta_0 + \beta_{\texttt{lat.abs}} \times \texttt{lat.abs}_g + \beta_{\texttt{elev}} \times \texttt{elev}_g \\ \texttt{Random}_g &= \texttt{RandomSpatial}_g + \texttt{RandomUncorr}_g \end{aligned}$$

  and

$$\texttt{Correlation}(\texttt{RandomSpatial}_a, \texttt{RandomSpatial}_b) = \texttt{Matern}(d_{ab}, \rho, \nu)$$

- Residual dispersion variance model (disp model; Γ GLMM):

$$\mathrm{E}\left(\texttt{Error}_{gi}^2\right) = \exp\left(\texttt{Fix}_g^{\mathrm{D}} + \texttt{RandomSpatial}_g^{\mathrm{D}} + \texttt{RandomUncorr}_g^{\mathrm{D}}\right)$$

Step 2. Fit the isoscape

`Isofit()` fits the following geostatistical isoscape model:

```
Isofit(
        iso.data,
        mean.model.fix=list(elev=FALSE, lat.abs=FALSE, lat.2=FALSE,
                long=FALSE, long.2=FALSE),
        disp.model.fix=list(elev=FALSE, lat.abs=FALSE, lat.2=FALSE,
                long=FALSE, long.2=FALSE),
        mean.model.rand=list(uncorr=FALSE, spatial=TRUE),
        disp.model.rand=list(uncorr=FALSE, spatial=TRUE),
        uncorr.terms=list(mean.model="lambda", disp.model="lambda"),
        spaMM.method=list(mean.model="fitme", disp.model="fitme"),
        dist.method="Earth",
        control.mean=list(),
        control.disp=list(),
        verbose=interactive()
        )
```

Step 2. Fit the isoscape

`Isofit()` fits the following geostatistical isoscape model:

```
iso.model <- Isofit(iso.data = sources_Europe,
        mean.model.fix = list(elev = TRUE, lat.abs = TRUE))
```

```
## [1] "fitting the following residual dispersion model using spaMM  :"
## [1] "var.isoscape.value ~ 1 + Matern(1|long + lat)"
## [1] "it may take a while..."
## [1] "fitting the following mean model using spaMM  :"
## [1] "isoscape.value ~ 1 + elev + lat.abs + Matern(1|long + lat)"
## [1] "it may take a while..."

## 'fitme' not optimized for models with structured dispersion.

## [1] "Done! Models were fitted in 203 sec."
```

```
## explore your output
names(iso.model)

## [1] "mean.fit" "disp.fit" "info.fit"
```

Step 2. Fit the isoscape

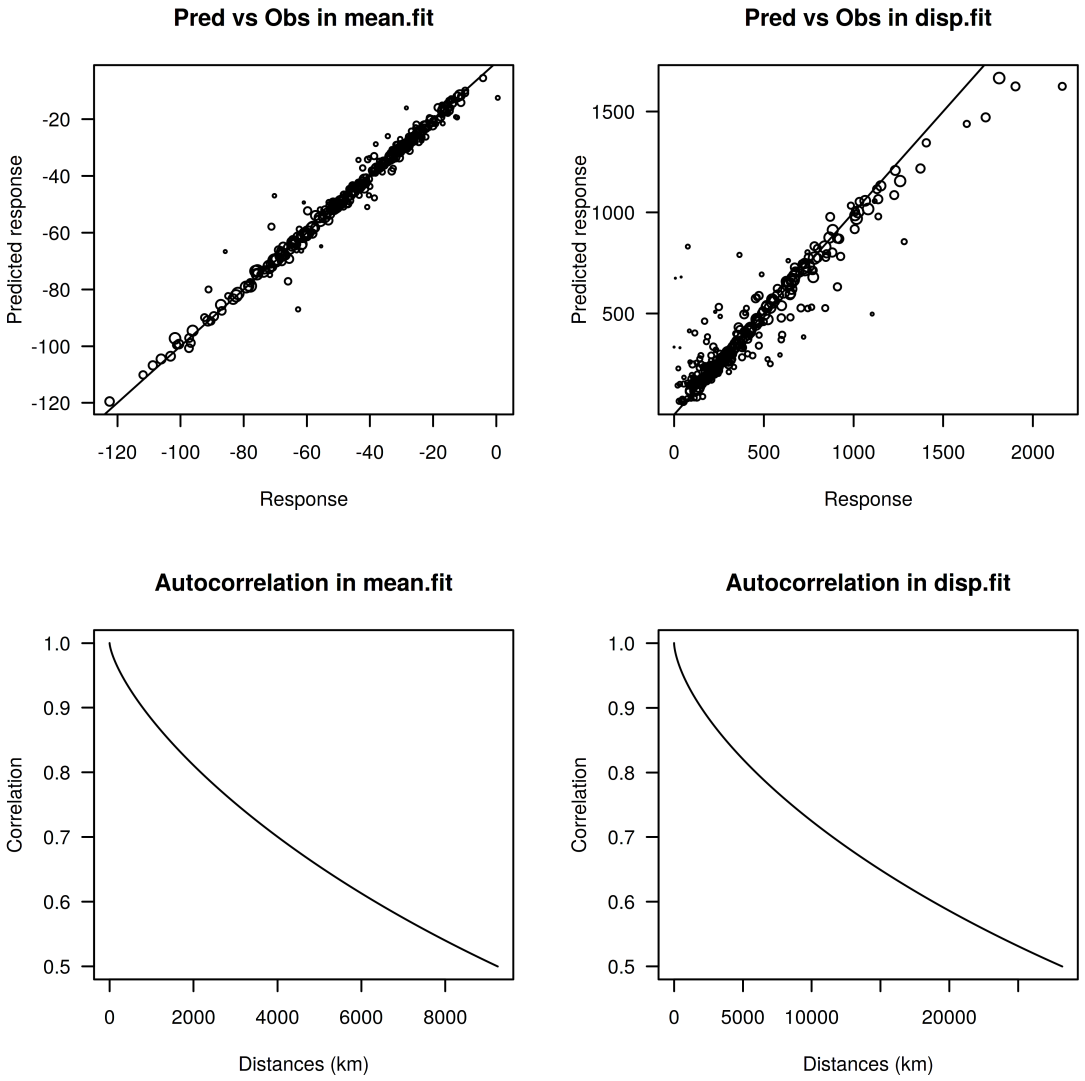
```
iso.model$mean.fit

## formula: isoscape.value ~ 1 + elev + lat.abs + Matern(1 | long + lat)
## REML: Estimation of lambda, phi, nu and rho by REML.
##       Estimation of fixed effects by ML.
## Estimation of nu and rho by 'outer' REML, maximizing p_bv.
## Family: gaussian ( link = identity )
##  ------------ Fixed effects (beta) ------------
##             Estimate  Cond. SE t-value
## (Intercept) 65.44187 3.263e+01   2.006
## elev        -0.01064 7.856e-04 -13.545
## lat.abs     -2.33792 4.707e-01  -4.967
##  --------------- Random effects ---------------
## Family: gaussian ( link = identity )
## Distance: Earth
## Correlation parameters:
##           nu          rho
## 3.502815e-01 5.076208e-05
## Coefficients for log[ lambda = var(u) ]:
##       Group        Term Estimate Cond.SE
##  long + lat (Intercept)    6.762  0.1011
## Estimate of lambda ( long + lat ):  864.2
## # of obs: 306; # of groups: long + lat, 306
##  ------------- Residual variance  -------------
## Prior weights: 19 17 5 17 78 ...
## phi was fixed by an offset term:  "phi" ~ 0 + offset(pred.disp)
##  ------------- Likelihood values  -------------
##                        logLik
## p_v(h) (marginal L): -1027.42
##   p_beta,v(h) (ReL): -1029.34
```

Step 2. Fit the isoscape

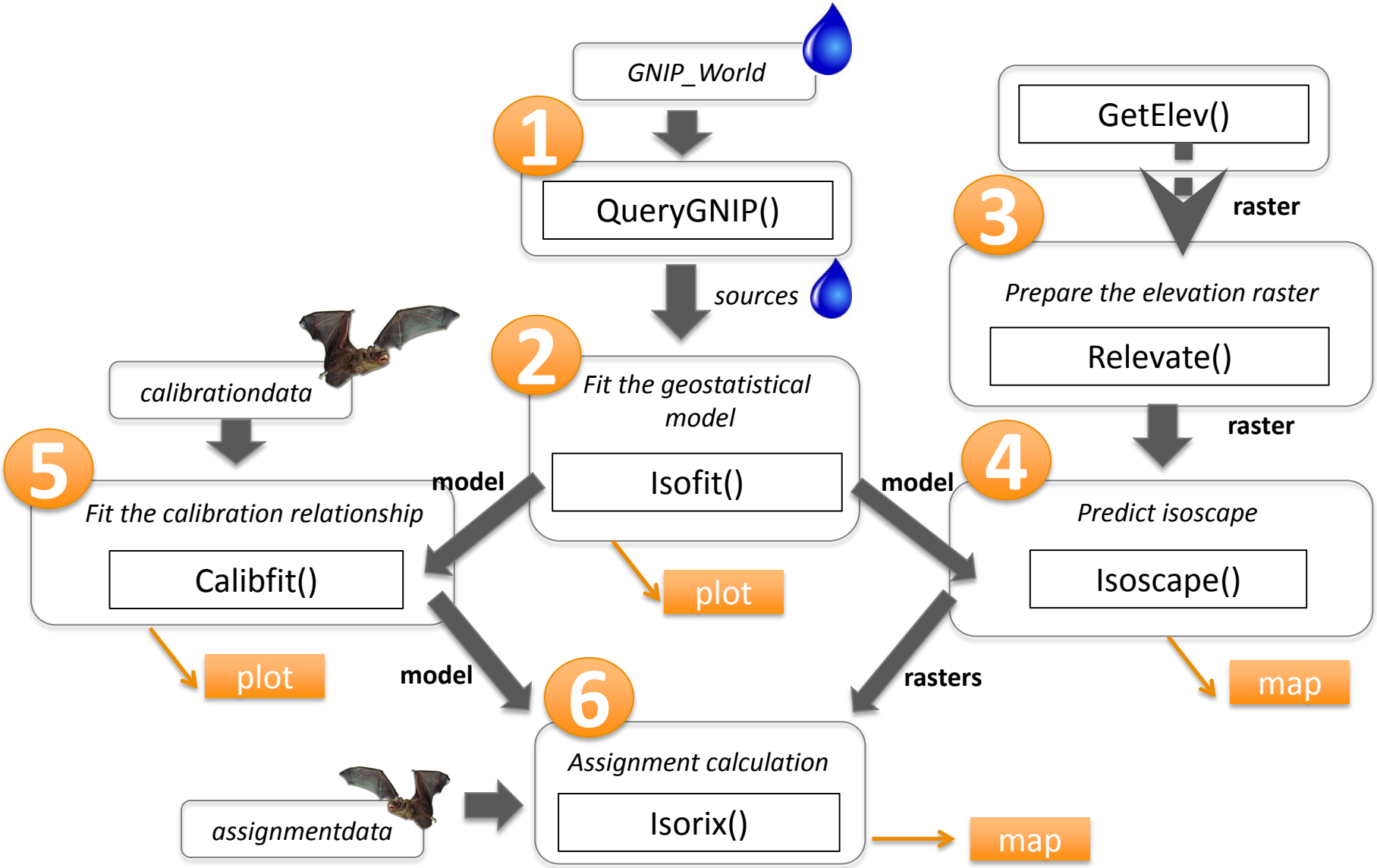
```
iso.model$disp.fit

## formula: var.isoscape.value ~ 1 + Matern(1 | long + lat)
## Estimation of lambda, nu and rho by REML approximation (p_bv).
## Estimation of fixed effects by ML approximation (p_v).
## Estimation of lambda, nu and rho by 'outer' REML, maximizing p_bv.
## Family: Gamma ( link = log )
##  ------------ Fixed effects (beta) ------------
##             Estimate Cond. SE t-value
## (Intercept)    5.863    2.023   2.898
##  --------------- Random effects ---------------
## Family: gaussian ( link = identity )
## Distance: Earth
## Correlation parameters:
##           nu          rho
## 3.188215e-01 1.478021e-05
## Outer estimate of lambda ( long + lat ):  4.755
## # of obs: 306; # of groups: long + lat, 306
##  -- Residual variation ( var = phi * mu^2 )  --
## Prior weights: 18 16 4 16 77 ...
## phi was fixed to 2
##  ------------- Likelihood values  -------------
##                         logLik
## p_v(h) (marginal L): -1971.680
##   p_beta,v(h) (ReL): -1970.056
```

Step 2. Fit the isoscape

```
plot(iso.model)
```



The assignment workflow in IsoriX 0.4-1



Step 3. Prepare the elevation raster

First, we need to obtain the elevation raster:

```
## option 1 (only for playing!)
data(elevationrastersmall)
elevationrastersmall

## class       : RasterLayer
## dimensions  : 209, 432, 90288  (nrow, ncol, ncell)
## resolution  : 0.8333333, 0.8333333  (x, y)
## extent      : -180.0001, 179.9999, -90.16681, 83.99986  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : elevationrastersmall.asc
## values      : -72.4432, 5498.742  (min, max)
```

Step 3. Prepare the elevation raster

```
## option 2
mypath <- paste(getwd(),"/",sep="")  ## temporary fix of a bug!
GetElev(path=mypath)
```

```
## [1] "this function attempts to download a high resolution elevation raster from
## [1] "the raster elevation_raster_gmted2010_30mn.tif is stored in the folder /hom
```

```
library(raster)
elevationrasterbig <- raster("elevation_raster_gmted2010_30mn.tif")
```

```
### IF IT DOES NOT WORK:

## 1. check that the file is in the working directory:
file.exists("elevation_raster_gmted2010_30mn.tif")

## 2. if it is, but still did not work re-download the tif using:
download.file(
  url="http://62.141.164.7/download/elevation_raster_gmted2010_30mn.tif",
  destfile = "elevation_raster_gmted2010_30mn.tif", mode="wb")

elevationrasterbig <- raster("elevation_raster_gmted2010_30mn.tif")
```

Step 3. Prepare the elevation raster

```
# elevationrastersmall
elevationrasterbig

## class       : RasterLayer
## dimensions  : 20880, 43200, 902016000  (nrow, ncol, ncell)
## resolution  : 0.008333333, 0.008333333  (x, y)
## extent      : -180.0001, 179.9999, -90.00014, 83.99986  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : /home/alex/Dropbox/Boulot/Mes_talks/IsoriX_2016/elevation_raster_gm
## names       : elevation_raster_gmted2010_30mn
## values      : -430, 8625  (min, max)
```

Step 3. Prepare the elevation raster

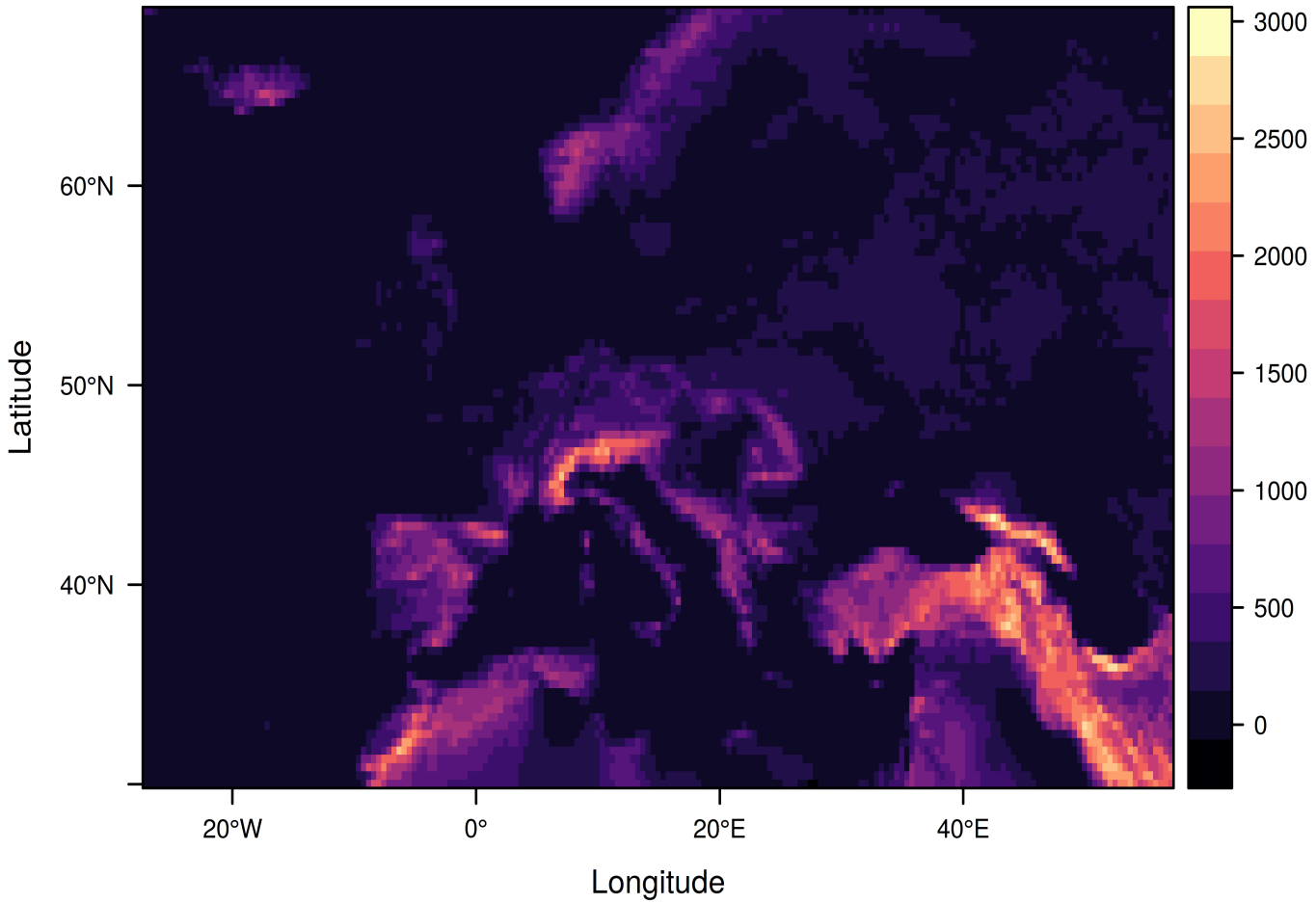
Second, we need to crop and aggregate the obtained elevation raster:

```
elev <- RElevate(elevation.raster = elevationrasterbig,
                 isofit = iso.model, aggregation.factor = 50)
```

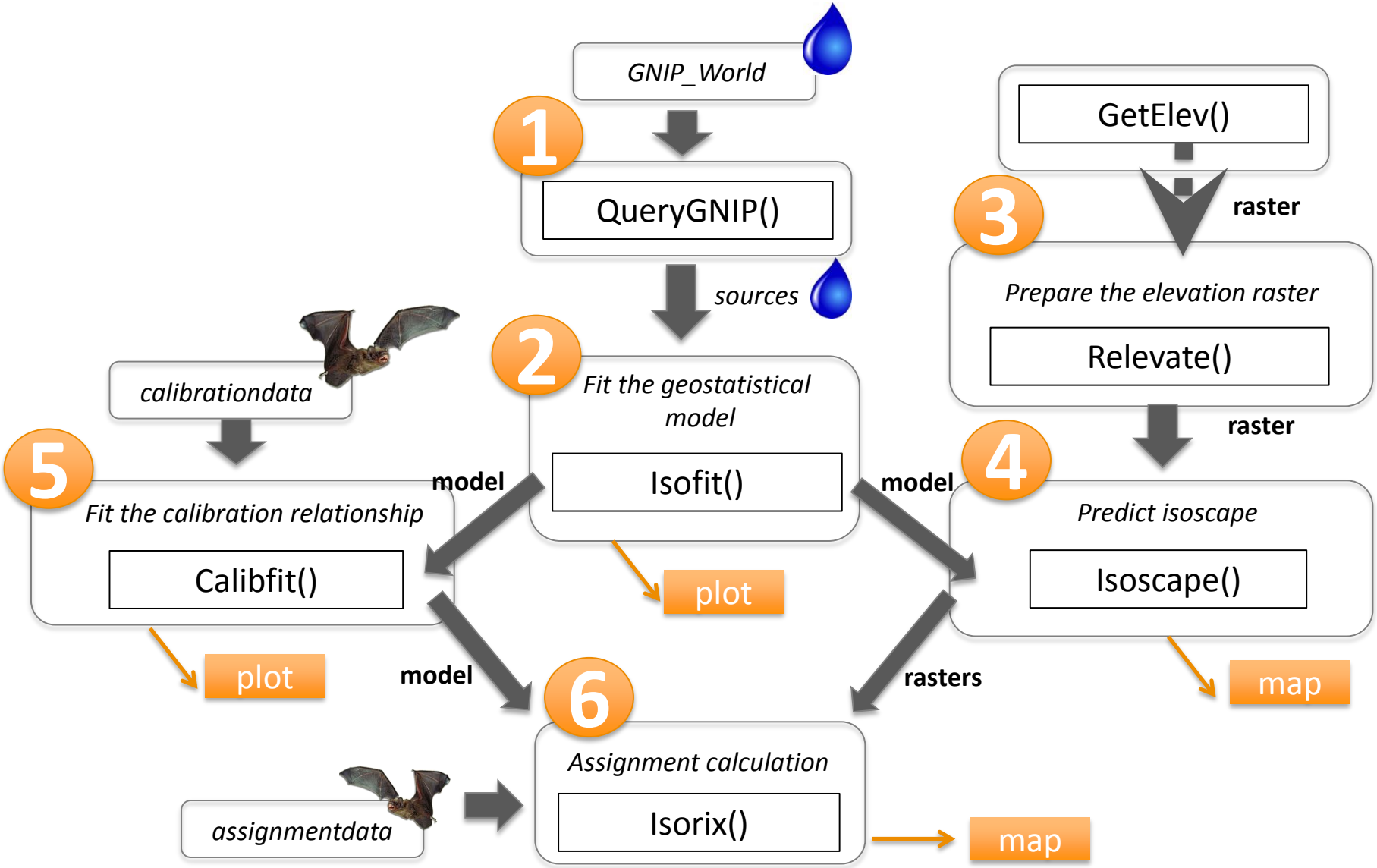
```
## [1] "cropping..."
## [1] "done!"
##    user  system elapsed
##   7.672   1.604   9.278
## class       : RasterLayer
## dimensions  : 94, 203, 19082  (nrow, ncol, ncell)
## resolution  : 0.4166667, 0.4166667  (x, y)
## extent      : -27.34181, 57.24153, 29.79986, 68.96653  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : /tmp/RtmpZM1GlS/raster/r_tmp_2016-09-15_080613_7042_50640.grd
## names       : elevation_raster_gmted2010_30mn
## values      : -66.01125, 2856.432  (min, max)
```

Step 3. Prepare the elevation raster

```
library(rasterVis)
levelplot(elev, margin=FALSE)  ## this is not an IsoriX plot!
```



The assignment workflow in IsoriX 0.4-1



Step 4. Predict the isoscape

`Isoscape()` predicts the isoscape:

```
Isoscape(
  elevation.raster,
  isofit,
  verbose = interactive()
  )
```

```
myisoscape <- Isoscape(elevation.raster = elev, isofit = iso.model)
```

```
## [1] "building the isoscape... (may take a while)"
## [1] "predictions for all 19082 locations have been computed in 28 sec."
```

Step 4. Predict the isoscape

```
names(myisoscape)

## [1] "Isoscape"  "sp.points"

myisoscape$Isoscape

## class       : RasterStack
## dimensions  : 94, 203, 19082, 8  (nrow, ncol, ncell, nlayers)
## resolution  : 0.4166667, 0.4166667  (x, y)
## extent      : -27.34181, 57.24153, 29.79986, 68.96653  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## names       :          mean,  mean.predVar, mean.residVar,  mean.respVar,
## min values  : -120.84491945,    2.86862910,   75.61020877,   79.37903961,   75.6
## max values  :     0.3532501,   170.9956985,  1547.3991373,  1568.6595688,  1547.
```

Step 4. Predict the isoscape

```
names(myisoscape$Isoscape)

## [1] "mean"          "mean.predVar"  "mean.residVar" "mean.respVar"
## [5] "disp"          "disp.predVar"  "disp.residVar" "disp.respVar"
```

- `mean` = prediction of the mean isotopic value
- `mean.predVar` = quantify the error on `mean`
- `mean.residVar` = prediction of the variance between new observation
- `mean.respVar` = `mean.predVar` + `mean.residVar`
- `disp` = `mean.residVar`
- `disp.predVar` = quantify the error on `disp`
- `disp.residVar` = 2
- `disp.respVar` = `disp.predVar` + `disp.residVar`

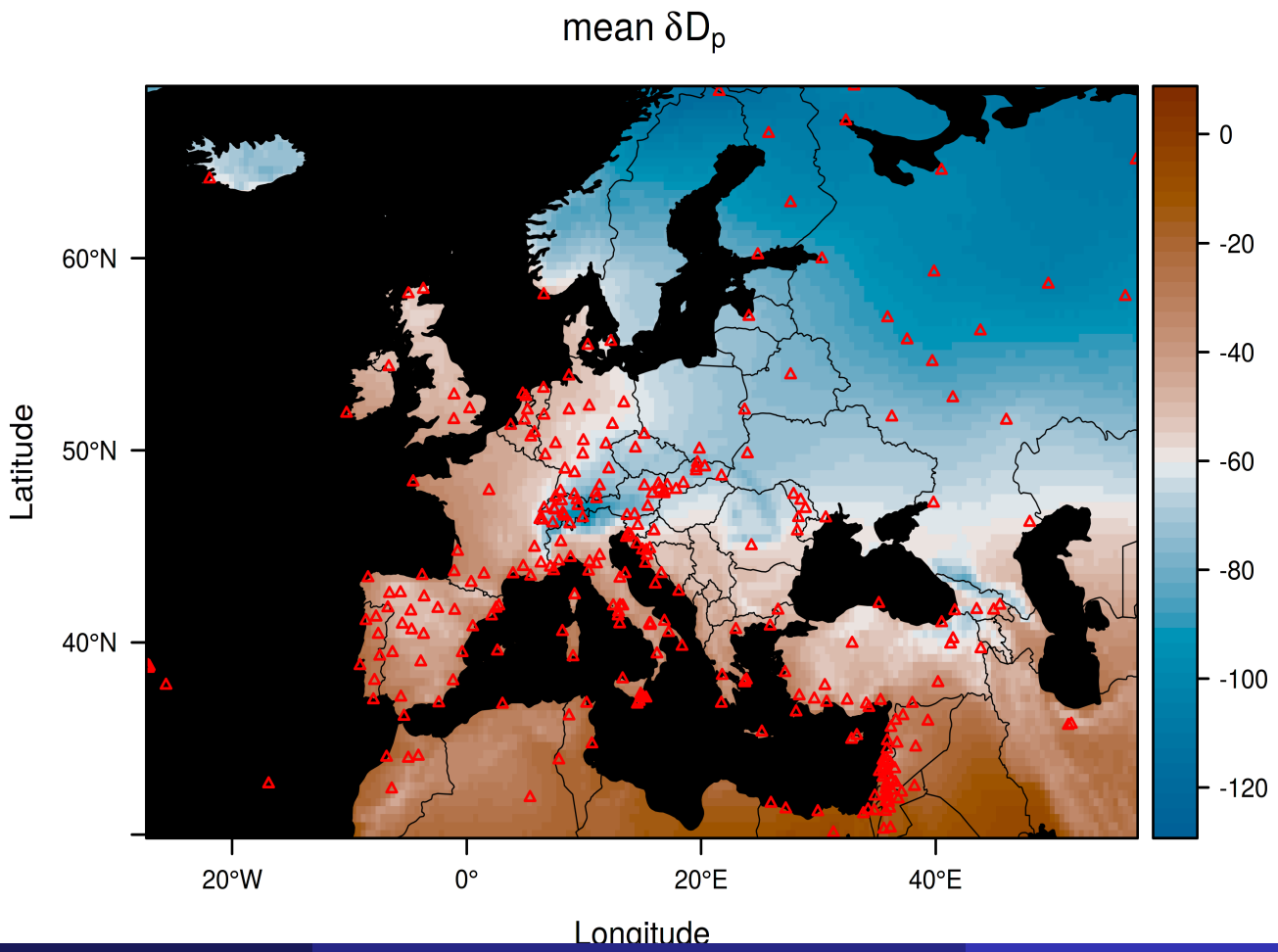
Step 4. Predict the isoscape

```r
plot.isoscape(
  x,
  which = "mean",
  sources = list(draw = TRUE, cex = 0.5, pch = 2, lwd = 1, col = "red"),
  borders = list(borders = NULL, lwd = 0.5, col = "black"),
  mask = list(mask = NULL, col = "black"),
  palette = terrain.colors(20),
  plot = TRUE
)
```

```r
## we load the decorators
data(worldcountries)
data(oceanmask)
data(isoscapepalette)
```

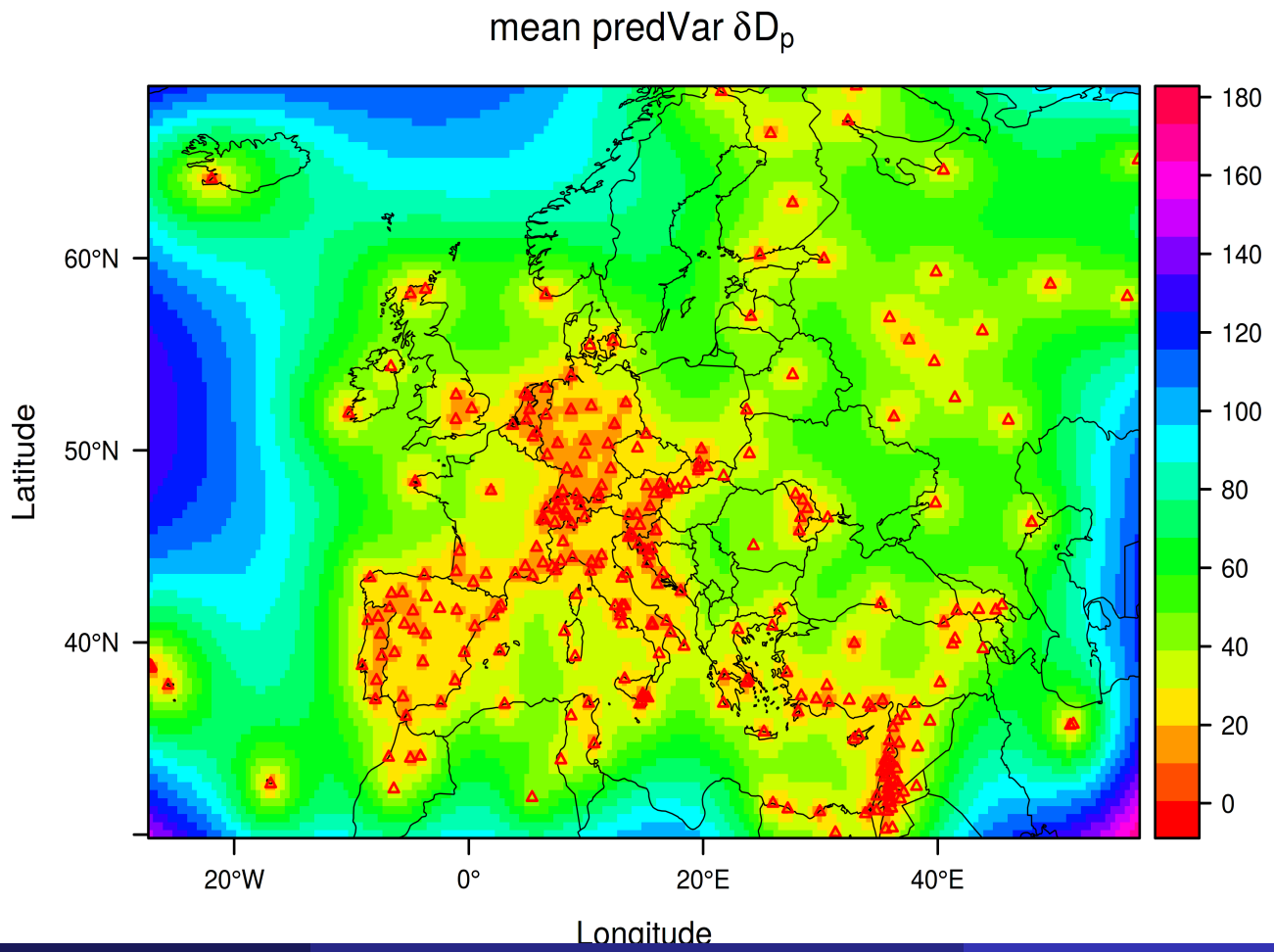
Step 4. Predict the isoscape

```
plot(myisoscape,
  which="mean",
  borders = list(borders = worldcountries),
  mask = list(mask = oceanmask), palette = isoscapepalette)
```



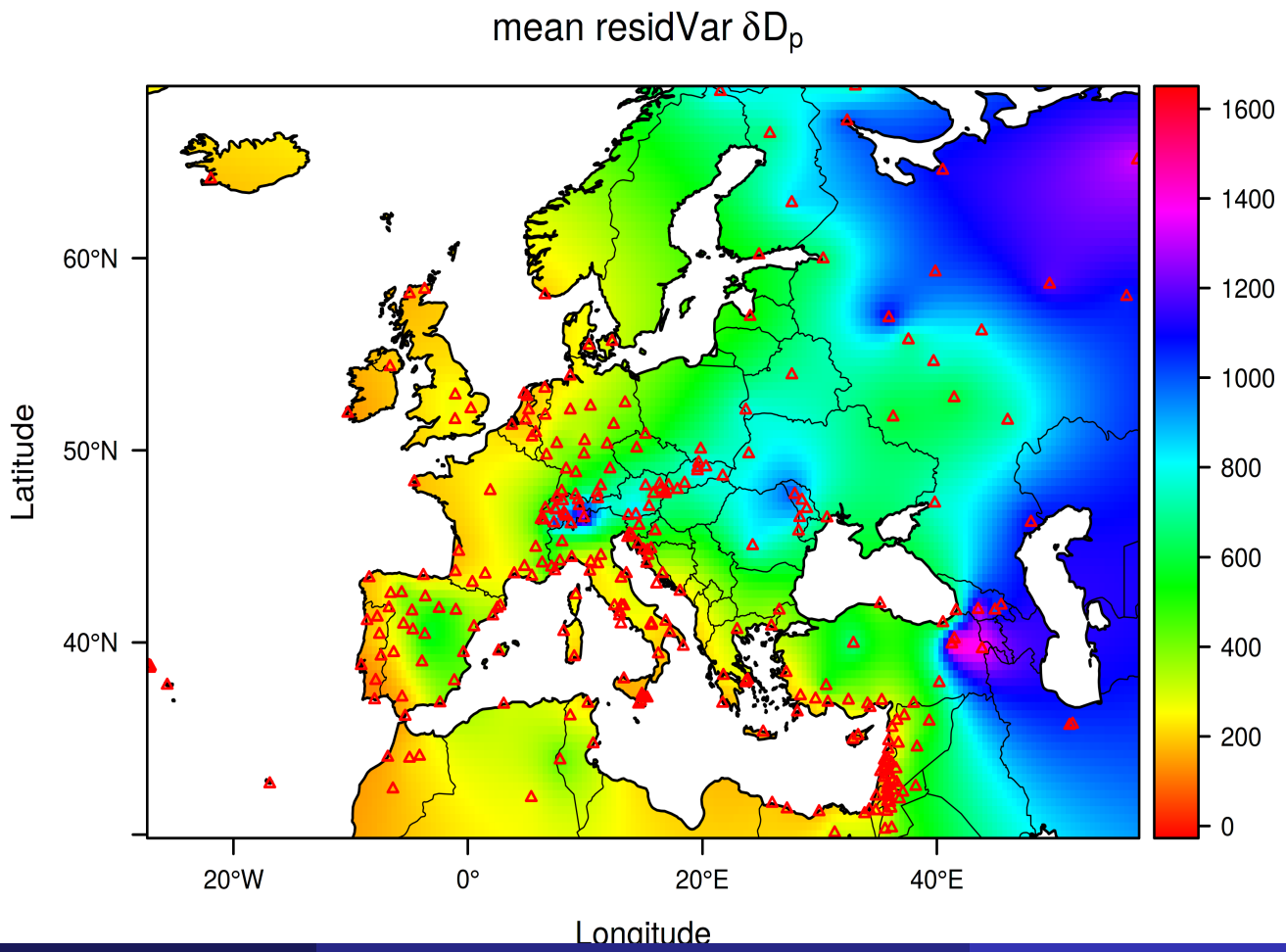
Step 4. Predict the isoscape

```
plot(myisoscape,
  which="mean.predVar",
  borders = list(borders = worldcountries),
  mask = list(mask = NULL), palette = rainbow(20))
```

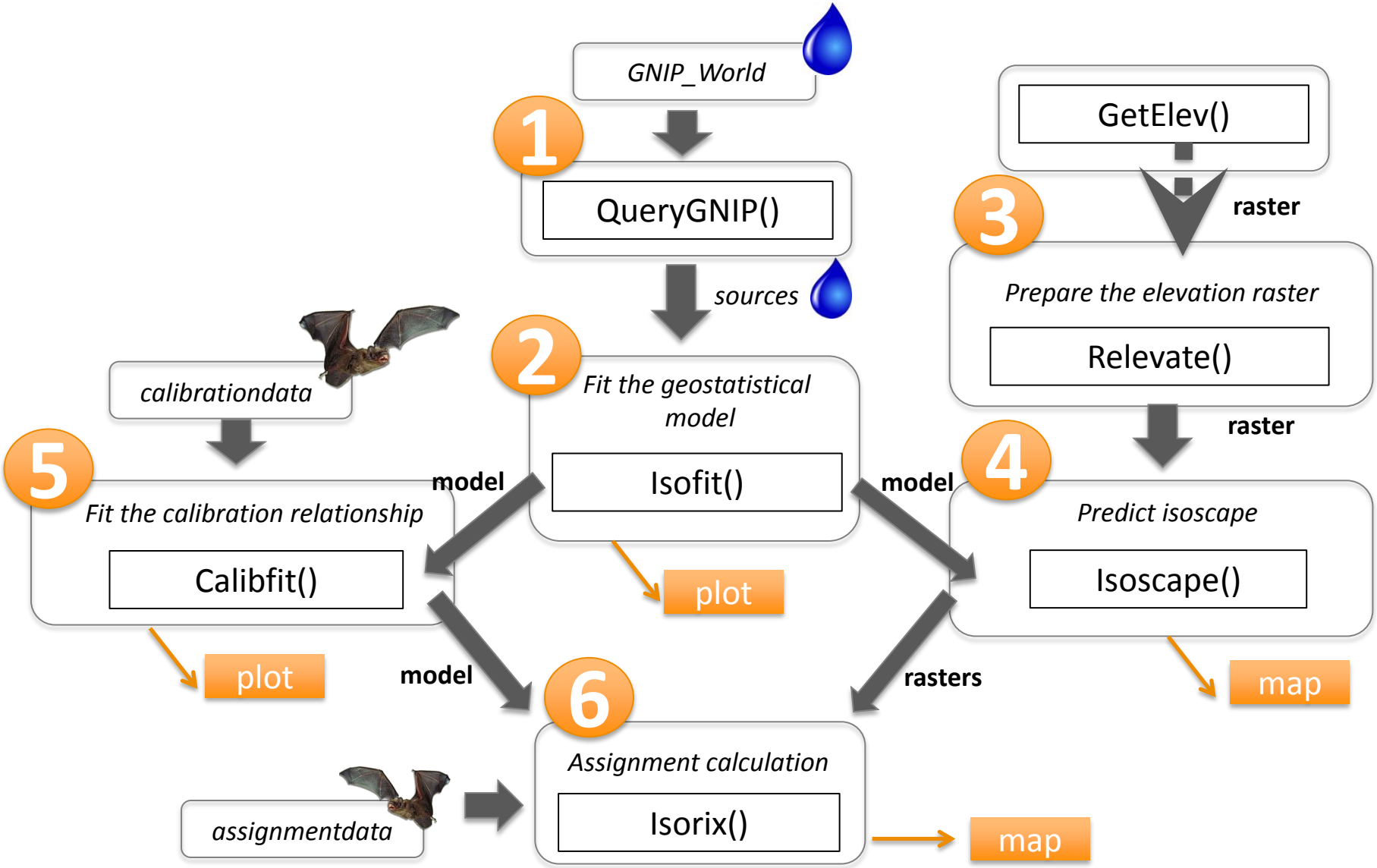


Step 4. Predict the isoscape

```
plot(myisoscape,
  which="mean.residVar",
  borders = list(borders = worldcountries),
  mask = list(mask = oceanmask, col="white"), palette = rainbow(1000))
```



The assignment workflow in IsoriX 0.4-1



Step 5. Fit the calibration function

First, we load the calibration data:

```
data(calibrationdata)
head(calibrationdata, n=15)
```

siteID	long	lat	elev	animalID	species	tissue.value
site_1	-5.590503	36.52158	667	Ese_2337	Eptesicus_serotinus	-36.4
site_1	-5.590503	36.52158	667	Ese_2338	Eptesicus_serotinus	-43.9
site_2	-5.977033	37.51861	31	Eis_2614	Eptesicus_isabellinus	-43.8
site_2	-5.977033	37.51861	31	Eis_2615	Eptesicus_isabellinus	-59.8
site_2	-5.977033	37.51861	31	Eis_2616	Eptesicus_isabellinus	-49.0
site_2	-5.977033	37.51861	31	Eis_2617	Eptesicus_isabellinus	-51.1
site_2	-5.977033	37.51861	31	Eis_2618	Eptesicus_isabellinus	-51.4
site_2	-5.977033	37.51861	31	Eis_2619	Eptesicus_isabellinus	-50.6
site_2	-5.977033	37.51861	31	Eis_2620	Eptesicus_isabellinus	-51.4
site_2	-5.977033	37.51861	31	Eis_2621	Eptesicus_isabellinus	-45.5
site_2	-5.977033	37.51861	31	Eis_2622	Eptesicus_isabellinus	-51.6
site_2	-5.977033	37.51861	31	Eis_2623	Eptesicus_isabellinus	-42.2
site_2	-5.977033	37.51861	31	Eis_2624	Eptesicus_isabellinus	-41.7
site_2	-5.977033	37.51861	31	Eis_2625	Eptesicus_isabellinus	-51.1
site_2	-5.977033	37.51861	31	Eis_2626	Eptesicus_isabellinus	-53.5

Step 5. Fit the calibration function

Then, we fit the calibration function with `Calibfit`:

```
Calibfit(
  calib.data,
  isofit,
  verbose = interactive(),
  control.optim = list()
  )
```

```
mycalib <- Calibfit(calib.data = calibrationdata, isofit = iso.model)
```

```
## [1] "predicting the isoscape value in each calibration site..."
## [1] "fitting the calibration function..."
## [1] "computing the covariance matrix of fixed effects..."
```

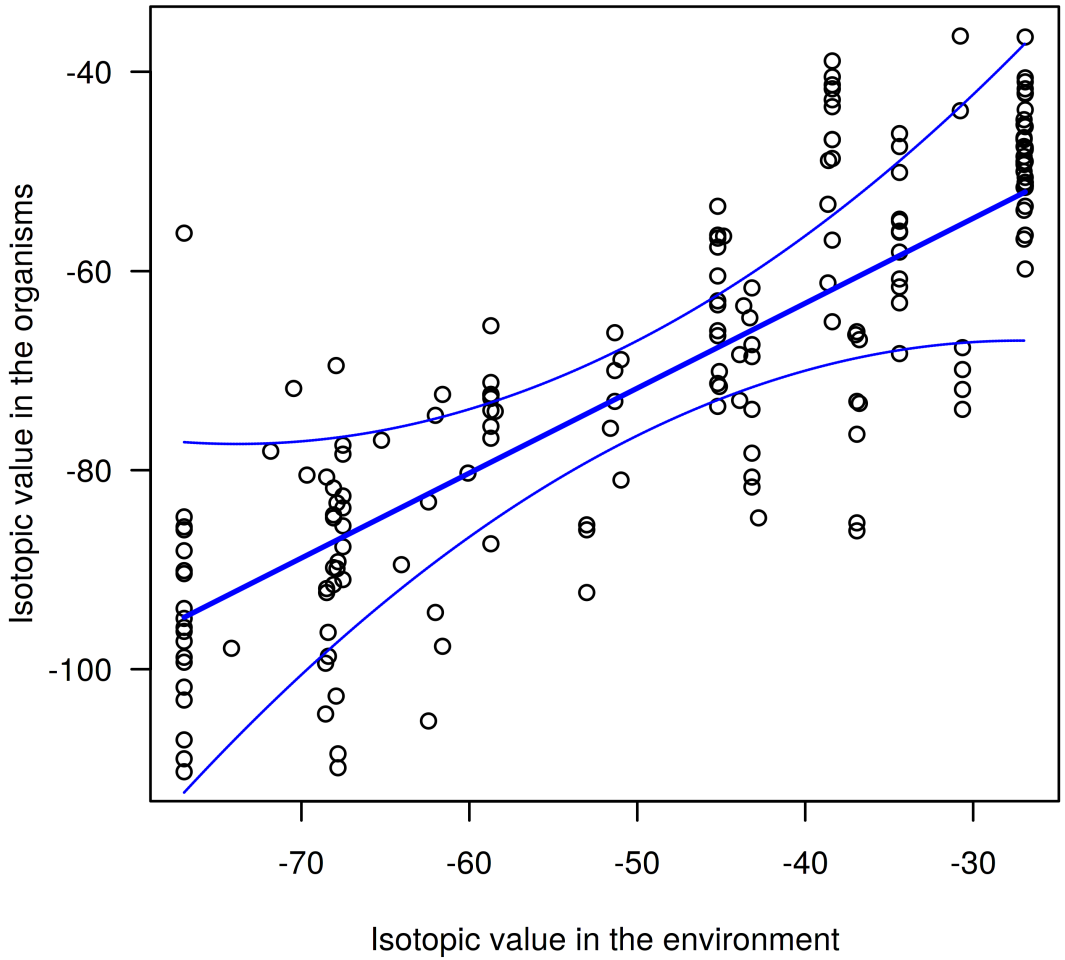
(The statistical model is quite complex...)

Step 5. Fit the calibration function

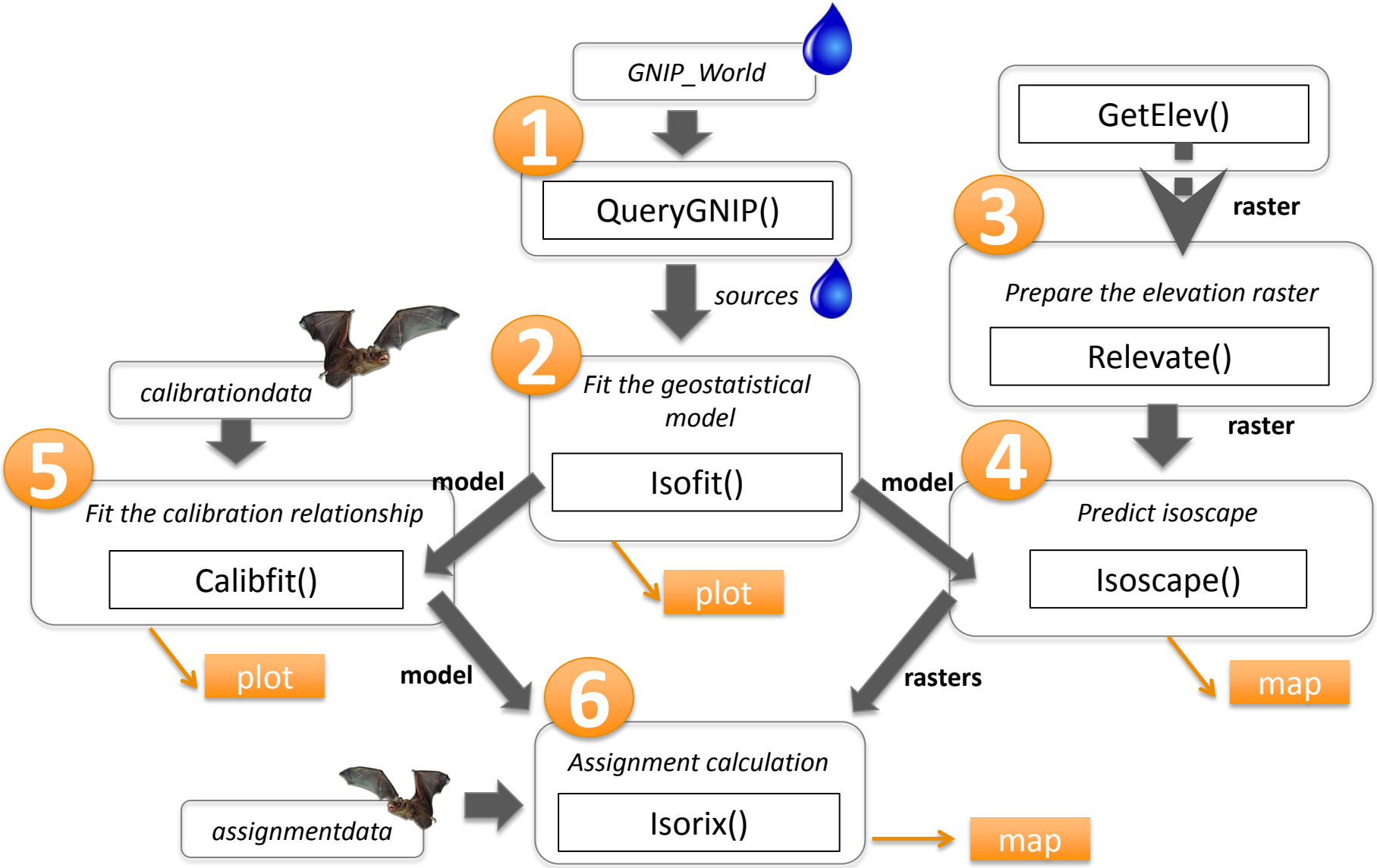
```
## don't mind the displayed formula (it is wrong, but only on the display)
mycalib

##
## Fixed effect estimates of the calibration fit
## tissue.value = intercept + slope * predicted isoscape.value + SpatialRandom + Er
##
##            intercept (+/- SE) = -29.10 +/- 5.12
##            slope     (+/- SE) =   0.85 +/- 0.10
##
## [for more information, use summary()]
```

Step 5. Fit the calibration function

```
plot(mycalib)
```



The assignment workflow in IsoriX 0.4-1



Step 6. Perform the assignment

The dataset:

```
data(assignmentdata)
M_bechsteinii <- subset(assignmentdata, species=="Myotis_bechsteinii")
```

```
M_bechsteinii
```

animalID	species	tissue.value
Mbe_1	Myotis_bechsteinii	-68.8
Mbe_2	Myotis_bechsteinii	-64.3
Mbe_3	Myotis_bechsteinii	-72.9
Mbe_4	Myotis_bechsteinii	-76.1
Mbe_5	Myotis_bechsteinii	-69.9
Mbe_6	Myotis_bechsteinii	-60.9
Mbe_7	Myotis_bechsteinii	-63.4
Mbe_8	Myotis_bechsteinii	-68.0
Mbe_9	Myotis_bechsteinii	-72.9
Mbe_10	Myotis_bechsteinii	-60.4

Step 6. Perform the assignment

The assignment test:

- let z be the isotopic measurement in a bat
- let α be the intercept of the calibration function
- let β be the slope of the calibration function
- let δ_o be the isotopic measurement at the bat original location
- let ε be the error in the calibration fit
- let δ_i be the isotopic measurement at the location i
- let d be the test statistic of the assignment

$$z = \alpha + \beta\delta_o + \varepsilon$$

$$\hat{\delta}_o = \frac{z - \hat{\alpha}}{\hat{\beta}}$$

$$d = \hat{\delta}_i - \hat{\delta}_o$$

If $i = o$ (null hypothesis), then $\text{mean}(d) = 0$ and $\text{var}(d) =$

$$\text{mean.predVar}_i + \frac{\text{mycalib.residVar} + \text{mycalib.fixedVar}}{\hat{\beta}^2} + \frac{2 \times \text{cov}(mean.predVar, mean.fixedVar)}{\hat{\beta}}$$

Step 6. Perform the assignment

`Isorix` performs the assignment test:

```
Isorix(
  assign.data,
  isoscape,
  calibfit,
  mask = NULL,
  verbose = interactive()
  )
```

```
myassignment <- Isorix(assign.data = M_bechsteinii,
                       isoscape = myisoscape,
                       calibfit = mycalib,
                       mask=oceanmask)
```

Step 6. Perform the assignment

```
names(myassignment)

## [1] "indiv"     "group"     "sp.points"

names(myassignment$indiv)

## [1] "stat"     "stat.var" "pv"

myassignment$indiv$stat

## class       : RasterBrick
## dimensions  : 94, 203, 19082, 10  (nrow, ncol, ncell, nlayers)
## resolution  : 0.4166667, 0.4166667  (x, y)
## extent      : -27.34181, 57.24153, 29.79986, 68.96653  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :     Mbe_1,     Mbe_2,     Mbe_3,     Mbe_4,     Mbe_5,     Mbe_6,
## min values  : -42.27230, -36.99986, -47.07608, -50.82537, -43.56112, -33.01624,
## max values  :  74.32477,  79.59721,  69.52099,  65.77170,  73.03595,  83.58083,
```

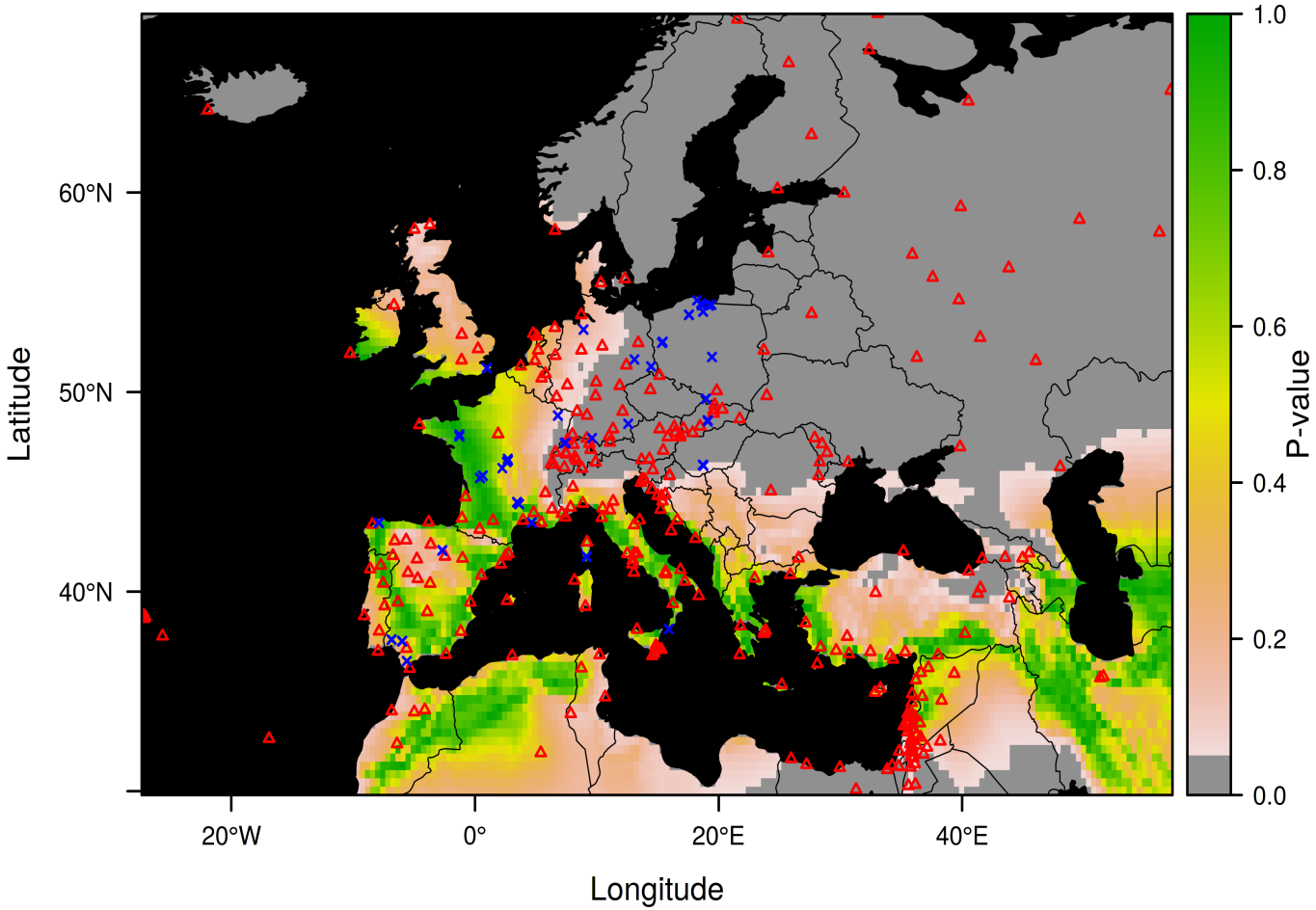
Step 6. Perform the assignment

```
plot.isorix(
  x,
  who = "group",
  what = "pv",
  cutoff = list(draw = TRUE, level = 0.05, col = "#909090"),
  sources = list(draw = TRUE, cex = 0.5, pch = 2, lwd = 1, col = "red"),
  calib = list(draw = TRUE, cex = 0.5, pch = 4, lwd = 1, col = "blue"),
  borders = list(borders = NULL, lwd = 0.5, col = "black"),
  mask = list(mask = NULL, col = "black"),
  palette = rev(terrain.colors(100)),
  plot = TRUE
  )
```

```
data(isorixpalette)
```

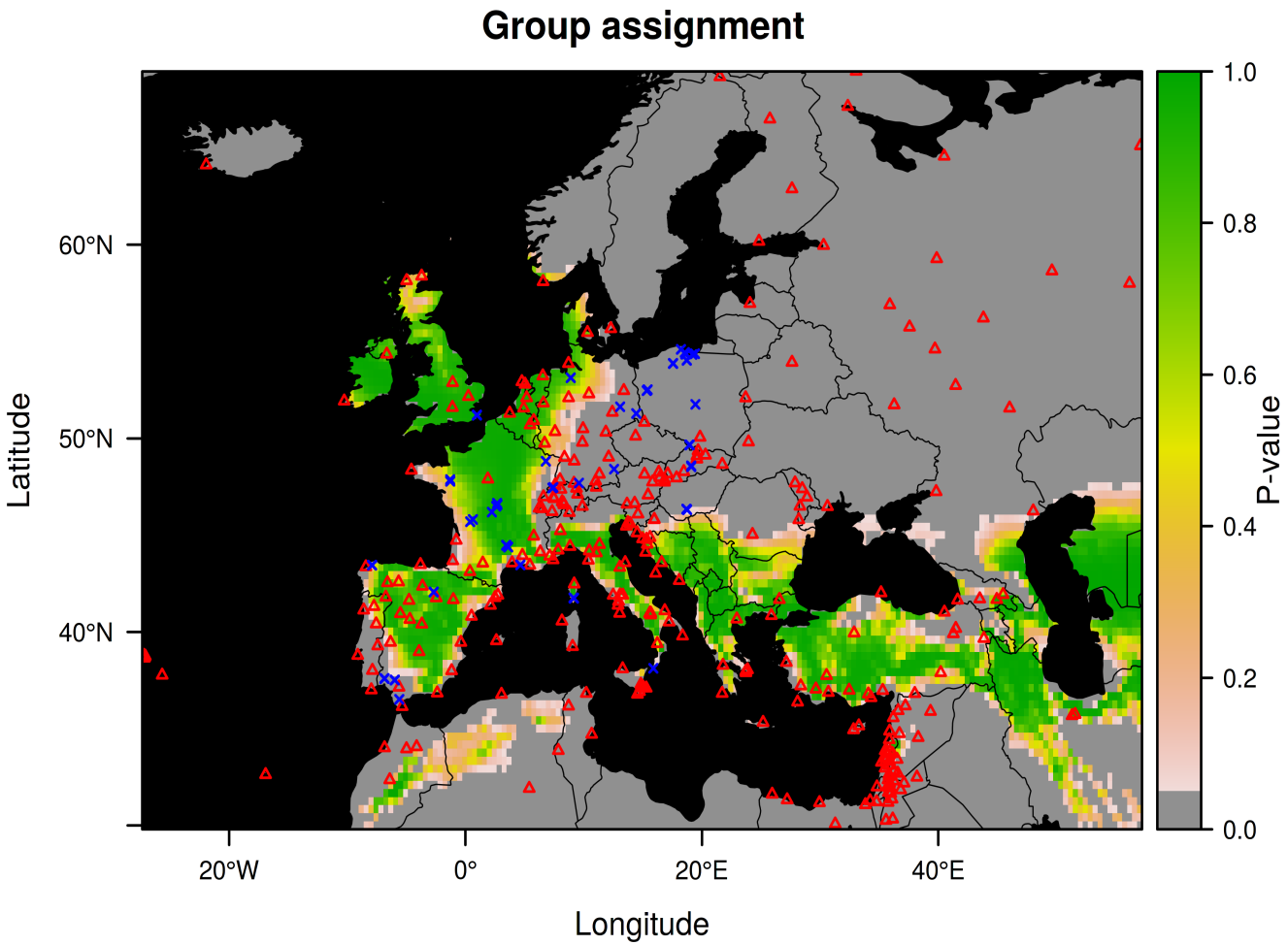
Step 6. Perform the assignment

```
plot(x=myassignment, who="Mbe_10",
  borders=list(borders=worldcountries), mask=list(mask=oceanmask),
  palette=isorixpalette)
```

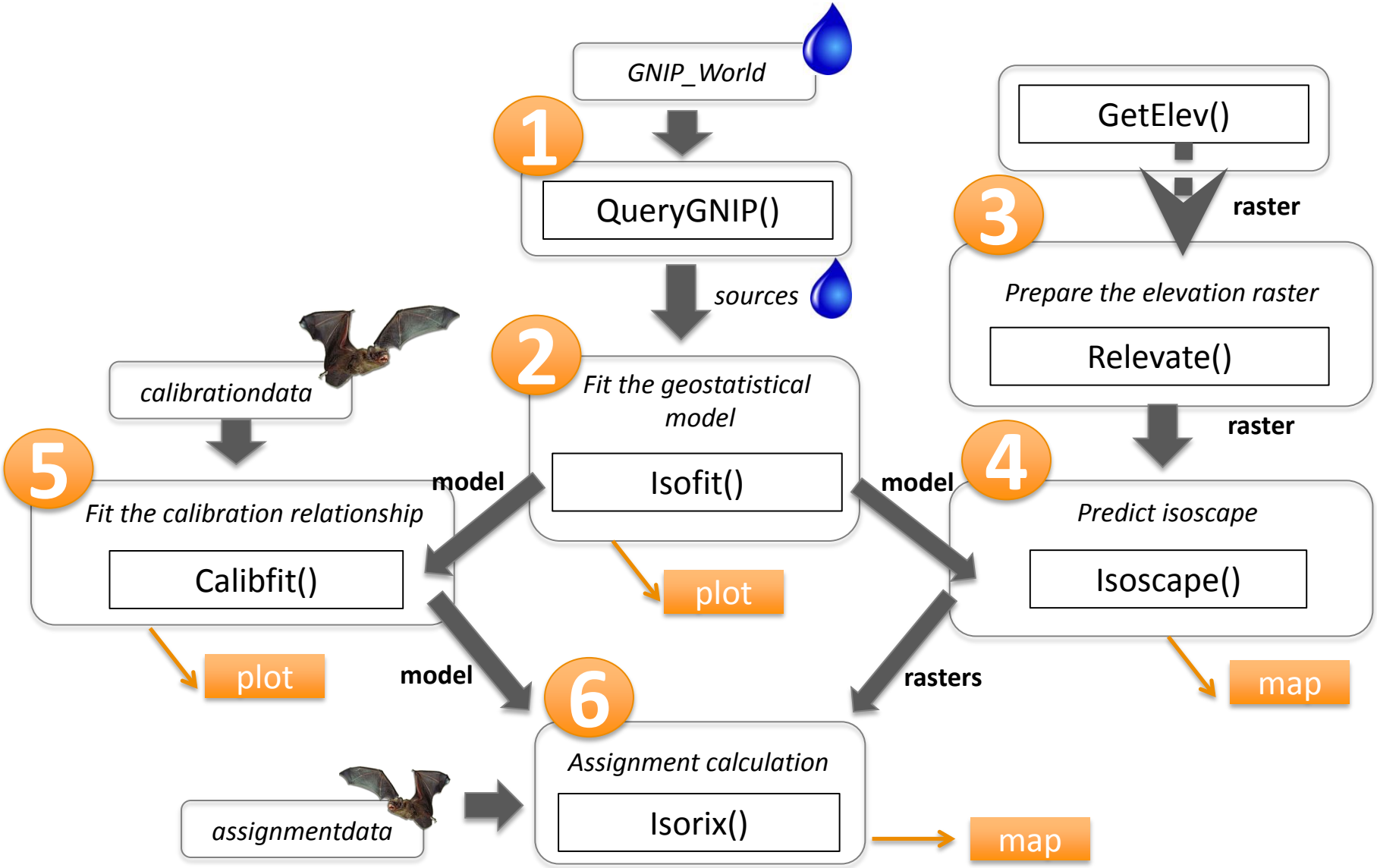


Step 6. Perform the assignment

```
assplot <- plot(x=myassignment, who="group",
  borders=list(borders=worldcountries), mask=list(mask=oceanmask),
  palette=isorixpalette)
```

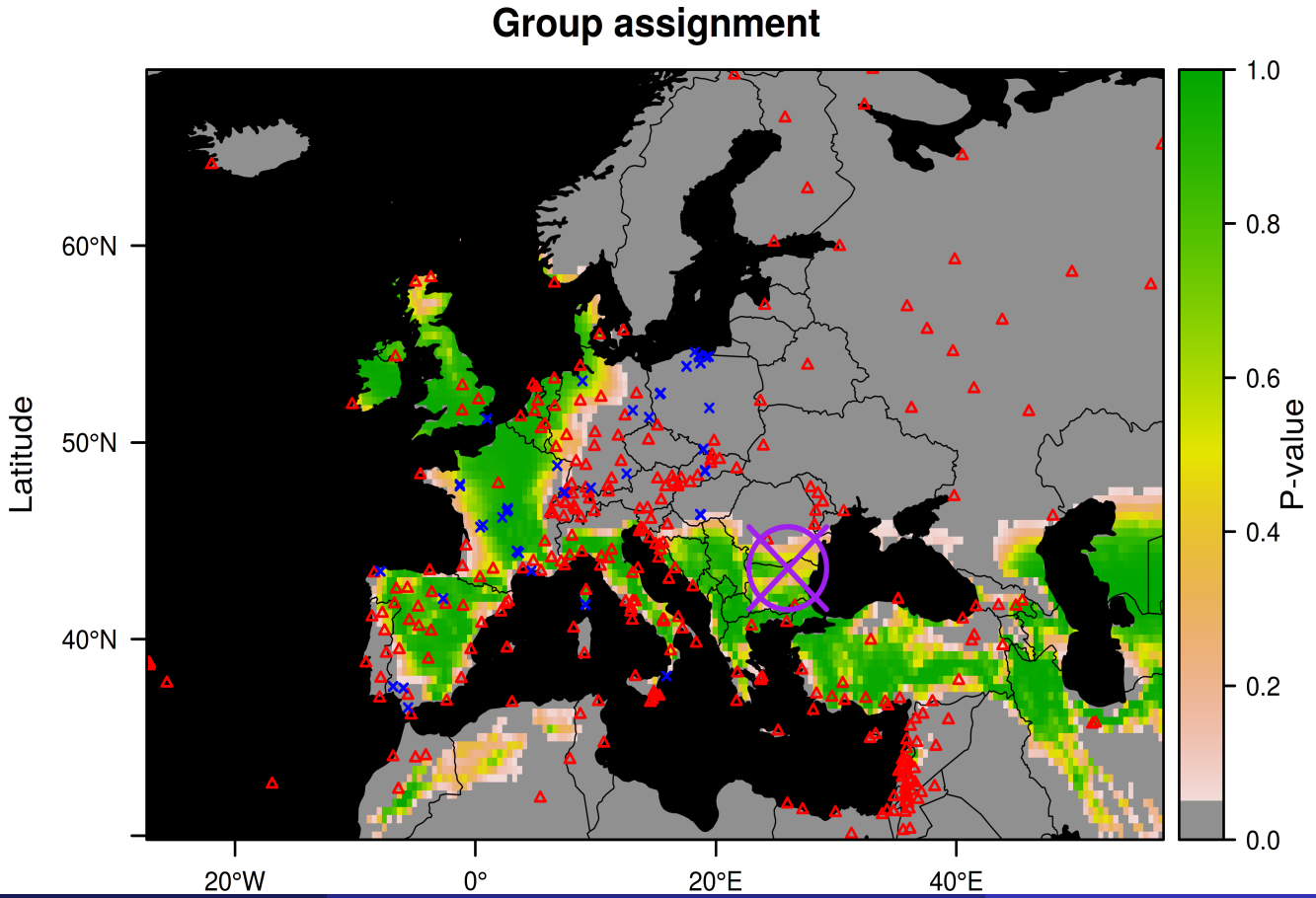


The assignment workflow in IsoriX 0.4-1



How good is the assignment?

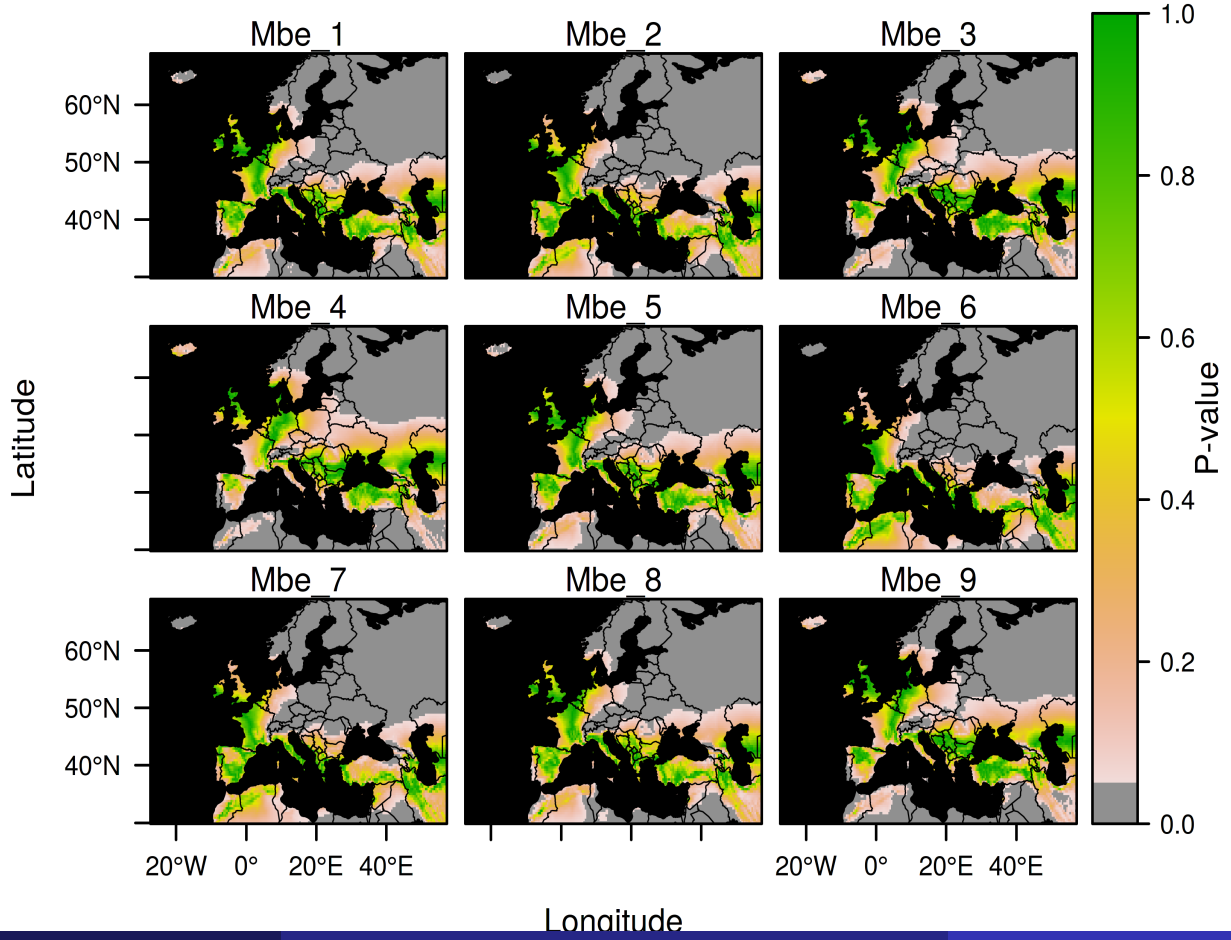
```
## the real origin of these bats (as known by us)
origin <- data.frame(long=25.982122, lat=43.611536)
library(lattice)
assplot + xyplot(origin$lat~origin$long, pch=13, col="purple", cex=5, lwd=2,
         panel=panel.points)
```



Why can assignment fail?

1. Because individuals may come from different places:

```
plot(x=myassignment, who=1:9,
  borders=list(borders=worldcountries), mask=list(mask=oceanmask),
  sources=list(draw=FALSE), calib= list(draw=FALSE), palette=isorixpalette)
```



Why can assignment fail?

2. Because calibration may be poor:

```
test <- data.frame(env=extract(myisoscape$Isoscape$mean, origin),
           tissue=M_bechsteinii$tissue.value)
```

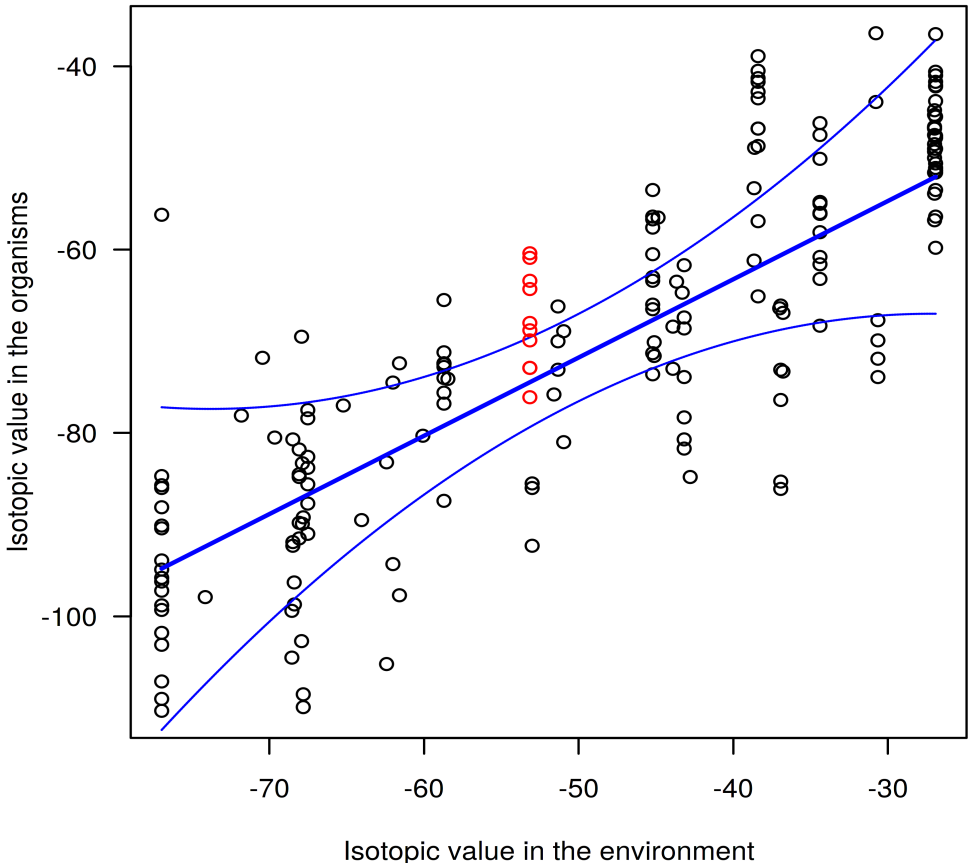
```
test
```

env	tissue
-53.15699	-68.8
-53.15699	-64.3
-53.15699	-72.9
-53.15699	-76.1
-53.15699	-69.9
-53.15699	-60.9
-53.15699	-63.4
-53.15699	-68.0
-53.15699	-72.9
-53.15699	-60.4

Why can assignment fail?

2. Because calibration may be poor:

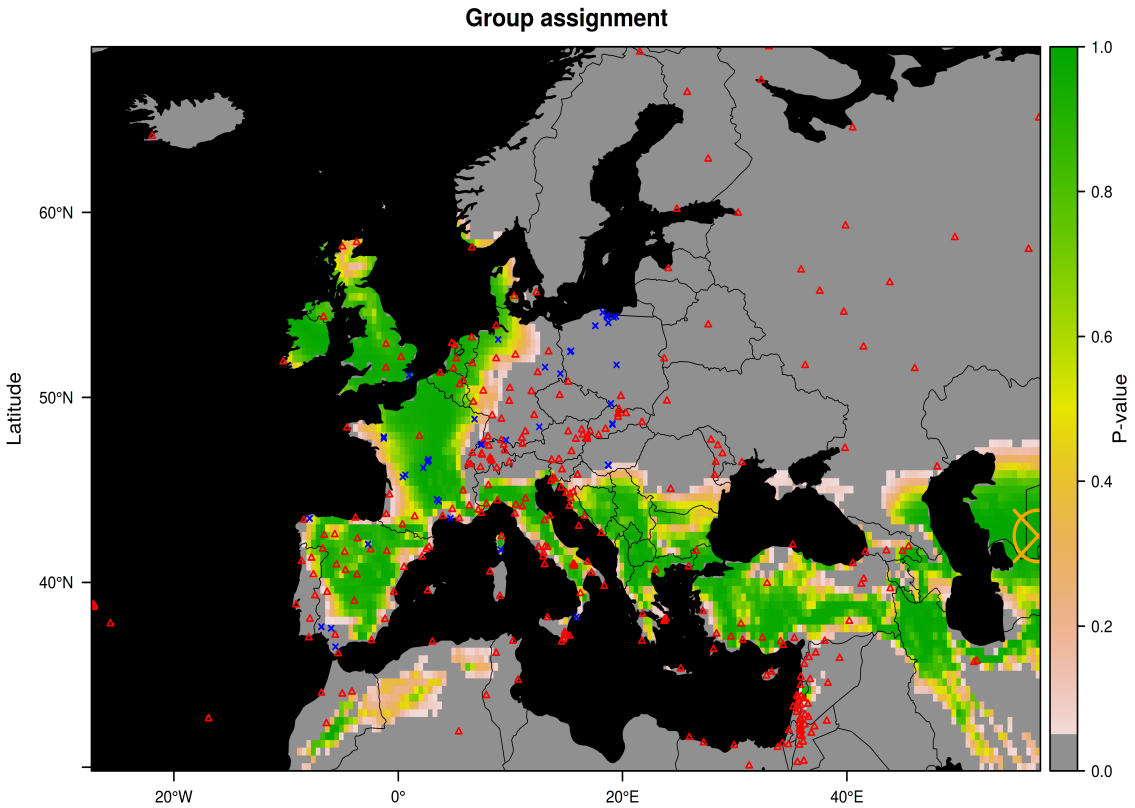
```
plot(mycalib)
points(test$tissue~test$env, col="red")
```



Why can assignment fail?

3. Because the isoscape may be poor:

```
## the location with the largest p-value
coord <- coordinates(myassignment$group$pv)
max.location <- coord[which.max(values(myassignment$group$pv)), ]
maximum <- data.frame(long=max.location[1], lat=max.location[2])
assplot + xyplot(maximum$lat~maximum$long, pch=13, col="orange", cex=5, lwd=2,
         panel=panel.points)
```



Why can assignment fail?

3. Because the isoscape may be poor:

```
predVarplot <- plot(myisoscape, which="mean.predVar",
                    borders = list(borders = worldcountries),
                    mask = list(mask = oceanmask),
                    palette = isoscapepalette, plot=FALSE)
predVarplot + xyplot(maximum$lat~maximum$long, pch=13, col="orange", cex=5, lwd=2,
         panel=panel.points)
```

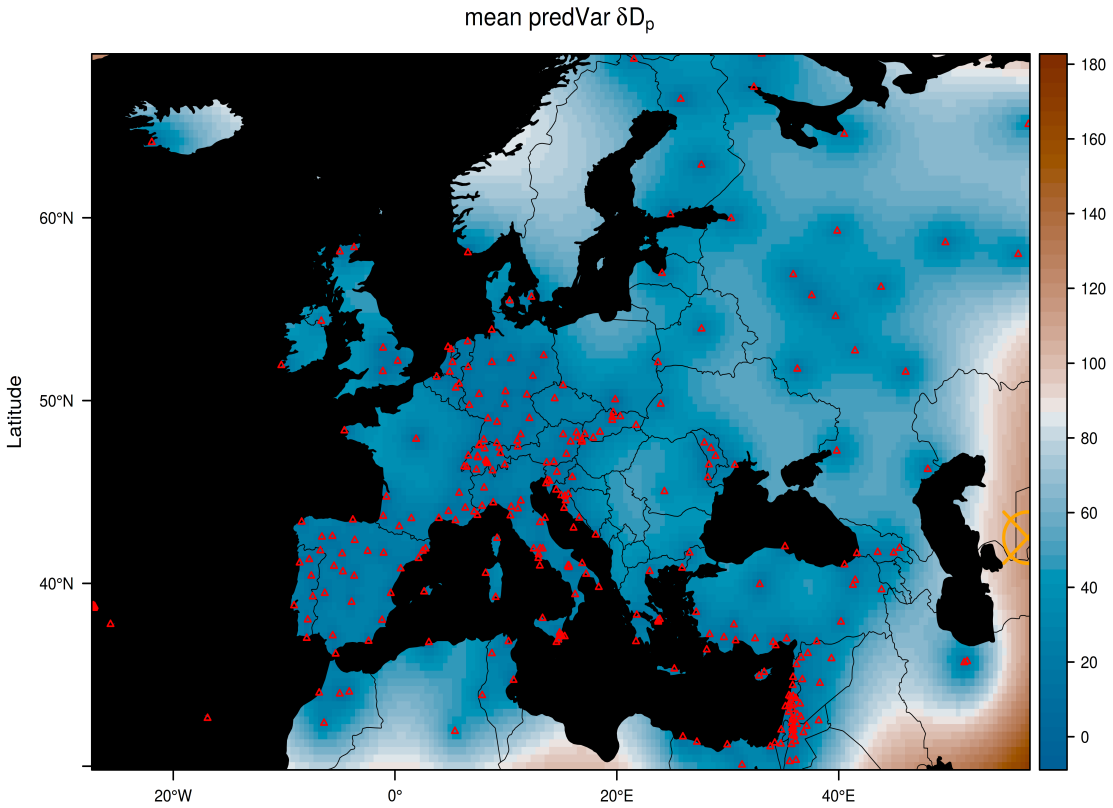


Table of contents

1. Generalities about IsoriX
2. The IsoriX workflow in 6 steps
3. Analysing the Dechamm dataset
4. Future of IsoriX

The new dataset

```r
dechamm <- read.csv("Dechamm.csv")

dechamm
```

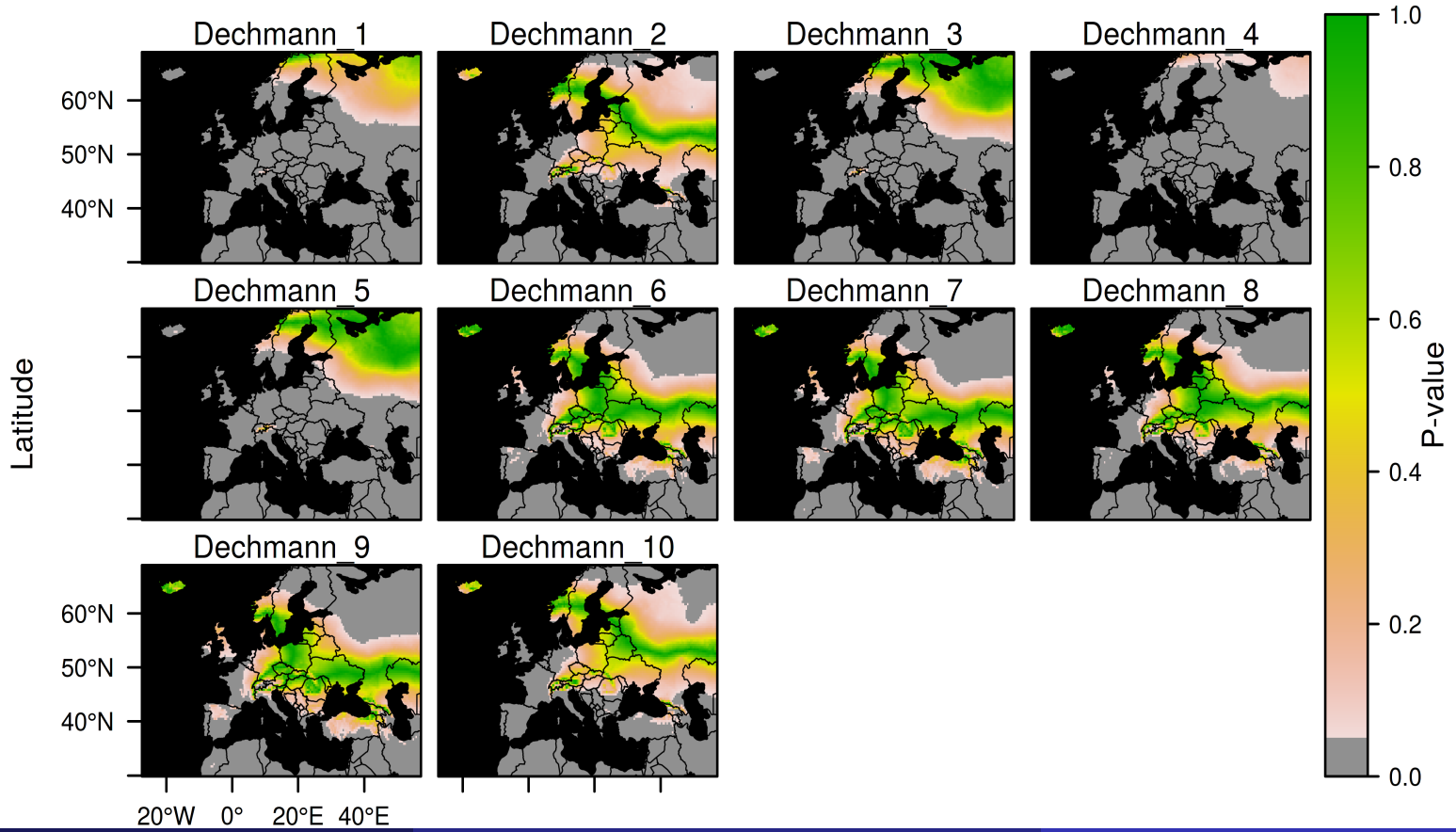
animalID	raw.tissue.value	tissue.value
Dechmann_1	-145.215	-128.846
Dechmann_2	-124.245	-99.677
Dechmann_3	-138.814	-119.942
Dechmann_4	-155.815	-143.591
Dechmann_5	-137.720	-118.421
Dechmann_6	-117.128	-89.777
Dechmann_7	-115.159	-87.038
Dechmann_8	-118.696	-91.958
Dechmann_9	-114.932	-86.722
Dechmann_10	-122.601	-97.390

New assignment

```
dechamm_assignment <- Isorix(assign.data = dechamm,
                             isoscape = myisoscape,
                             calibfit = mycalib,
                             mask=oceanmask)
```

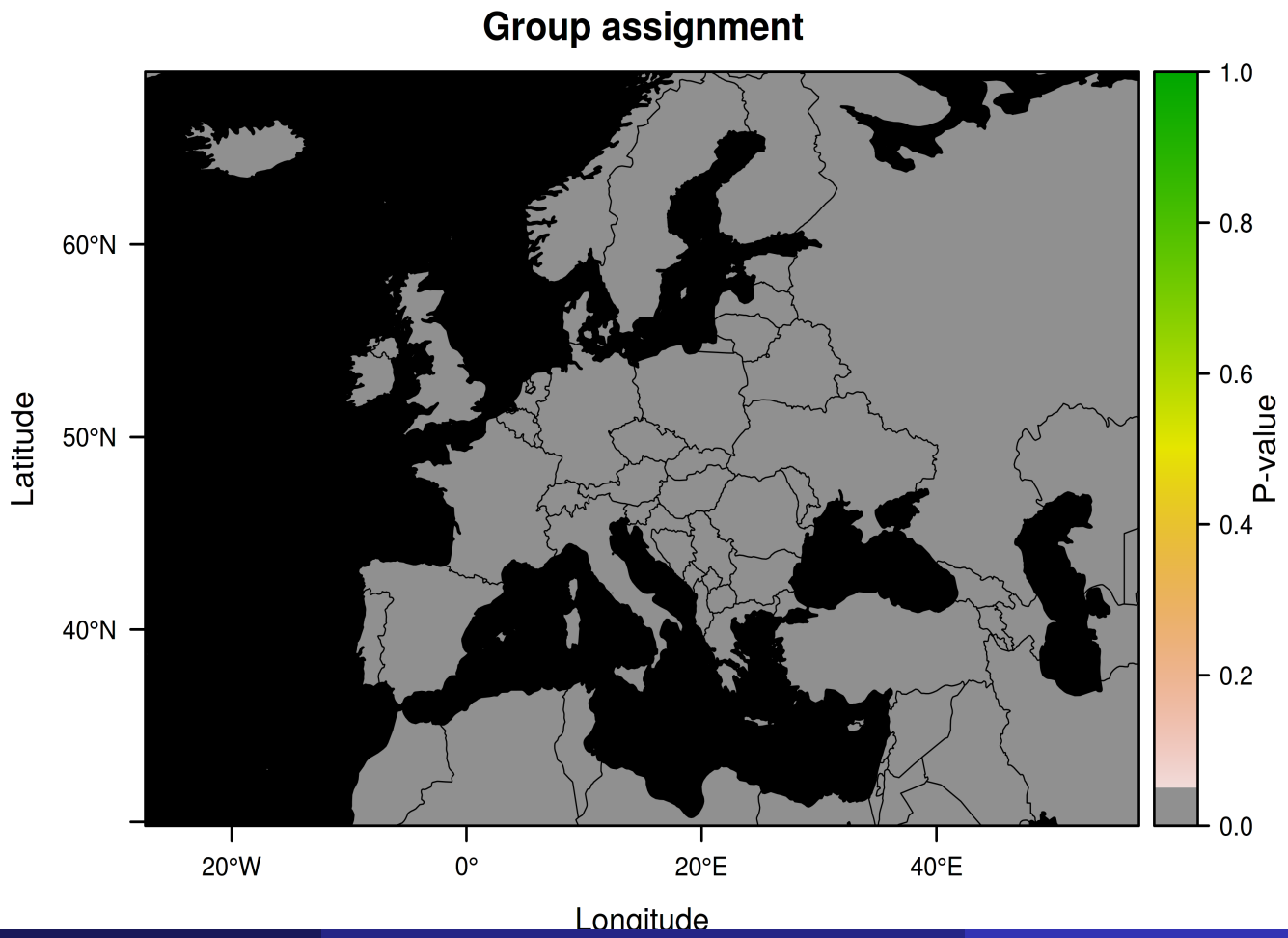
New assignment

```
plot(x=dechamm_assignment, who=1:10,
  borders=list(borders=worldcountries),
  mask=list(mask=oceanmask),
  sources=list(draw=FALSE),
  calib= list(draw=FALSE), palette=isorixpalette)
```



New assignment

```
plot(x=dechamm_assignment, who="group",
  borders=list(borders=worldcountries), mask=list(mask=oceanmask),
  sources=list(draw=FALSE), calib= list(draw=FALSE),
  palette=isorixpalette)
```

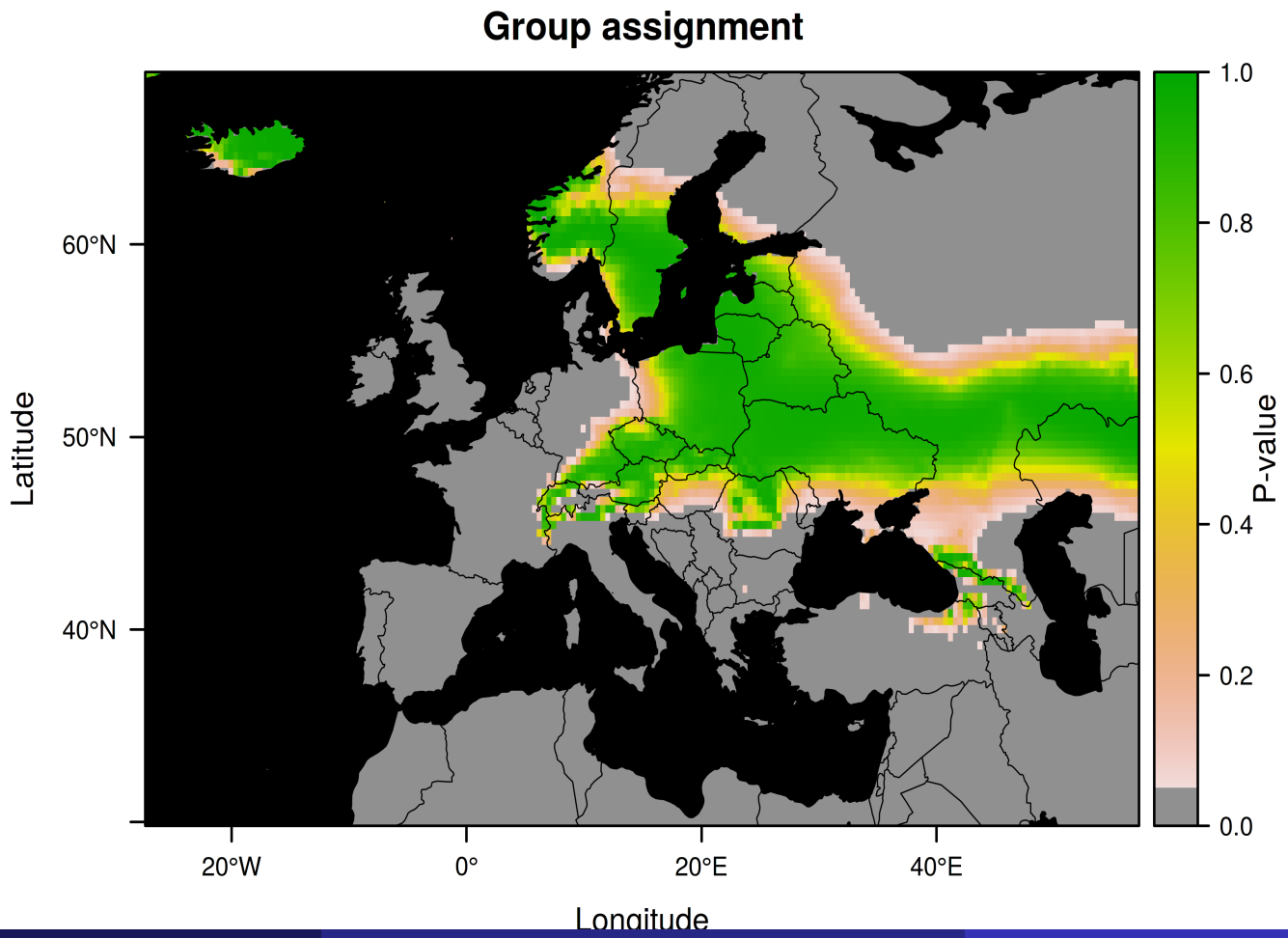


New assignment 2

```
dechamm2 <- dechamm[-c(1, 3, 4, 5), ]
dechamm3 <- dechamm[ c(1, 3, 4, 5), ]

dechamm_assignment2 <- Isorix(assign.data = dechamm2,
                              isoscape = myisoscape,
                              calibfit = mycalib,
                              mask=oceanmask)

dechamm_assignment3 <- Isorix(assign.data = dechamm3,
                              isoscape = myisoscape,
                              calibfit = mycalib,
                              mask=oceanmask)
```

New assignment 2

```
plot(x=dechamm_assignment2, who="group",
  borders=list(borders=worldcountries), mask=list(mask=oceanmask),
  sources=list(draw=FALSE), calib= list(draw=FALSE),
  palette=isorixpalette)
```



New assignment 3

```
plot(x=dechamm_assignment3, who="group",
  borders=list(borders=worldcountries), mask=list(mask=oceanmask),
  sources=list(draw=FALSE), calib= list(draw=FALSE),
  palette=isorixpalette)
```

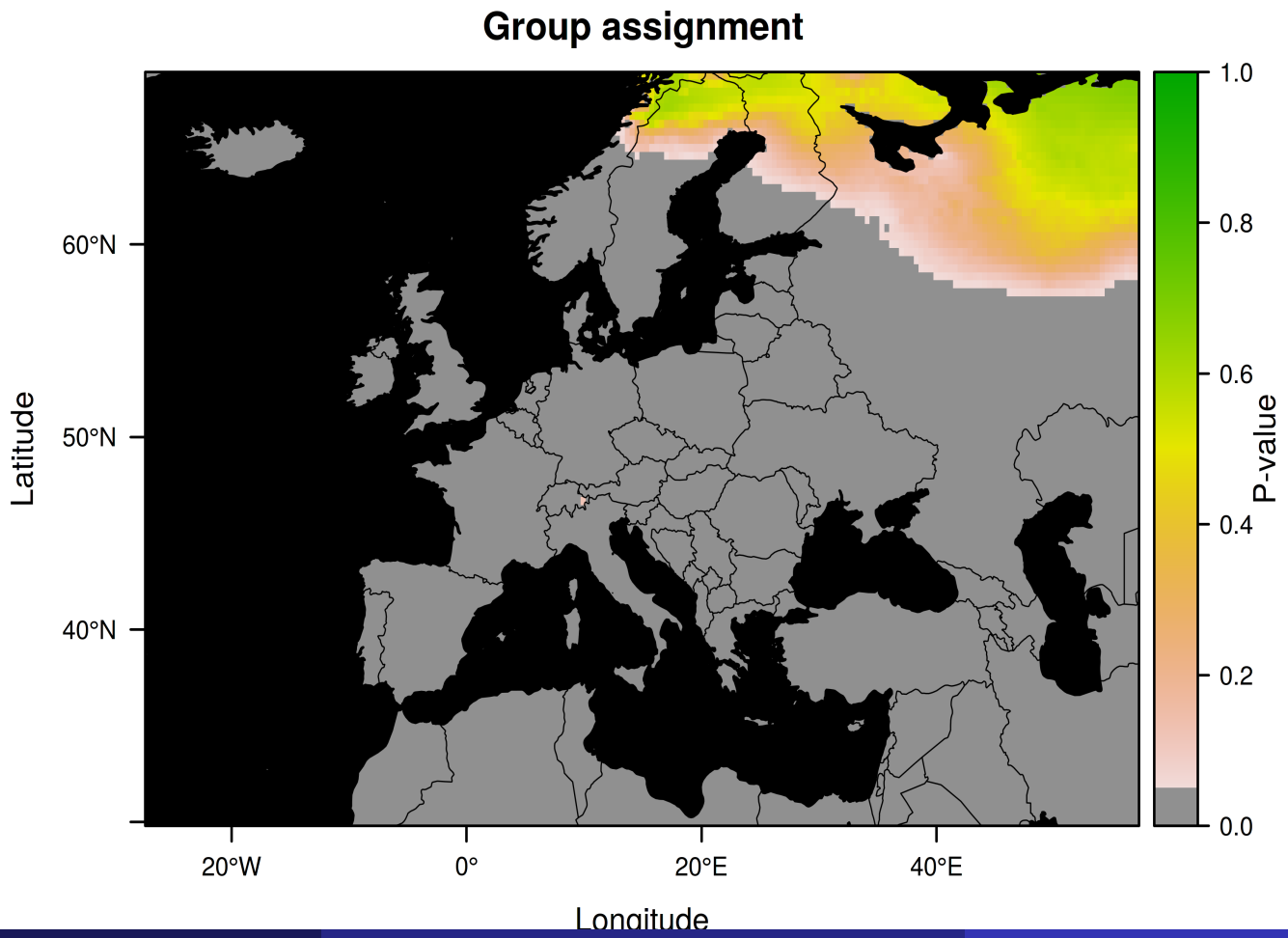


Table of contents

1. Generalities about IsoriX
2. The IsoriX workflow in 6 steps
3. Analysing the Dechamm dataset
4. Future of IsoriX

Life of IsoriX

Current versions:

- IsoriX 0.4-1: stable version (1st submission to CRAN, 11/09/2016)
- IsoriX 0.4-2: development version

Soon:

- more tutorials and better help files
- supporting publications
- optional calibration
- 3D plots
- map projections
- finer statistical methods (DHGLM)

To follow the development of IsoriX

https://github.com/courtiol/IsoriX_project