# IsoriX

## an R package for isoscape computation and inference of spatial origins using mixed models

The IsoriX core team

September 2018

# Table of contents

# Table of contents

# Table of contents

Benefits:

- fully reproducible methods & results
- simple enough (but not too simple) and highly customizable
- compatible with Geographic Information System within and without **R**
- cutting edge statistical methods
- works on Windows, MacOS, Linux, Unix; localy or remotely; with or without Internet
- free & open source (anyone can use and improve IsoriX!)

# Why IsoriX?

Benefits:

- fully reproducible methods & results
- simple enough (but not too simple) and highly customizable
- compatible with Geographic Information System within and without **R**
- cutting edge statistical methods
- works on Windows, MacOS, Linux, Unix; localy or remotely; with or without Internet
- free & open source (anyone can use and improve IsoriX!)

Limits:

- **R** knowledge required
- not multivariate (1 isotope only)
- not Bayesian (prior information not considered)

# Table of contents

An **R** package:

```
install.packages("IsoriX")  ## to install
```

```
library(IsoriX)  ## to load
```

```
##
##  IsoriX version 0.8.1 is now loaded!
##
##  The names of all objects (including functions) are finally getting stable,
##  but they have changed a lot since the version 0.7.1 (sorry!).
##  We kept revising them to make IsoriX more intuitive for you to use.
##  We will do our best to limit changes from version 0.8 onward!!
##
##  Type:
##
##    * ?IsoriX
##      for a very short description
##
##    * help(package = 'IsoriX', help_type = 'html')
##      for a list of the package objects and their help files
##
##    * browseURL('https://bookdown.org/content/782/')
##      for online tutorials & documentation
##
##    * citation('IsoriX')
##      for information on how to cite IsoriX (i.e.  the papers you should read)
##
##    * news(package = 'IsoriX')
##      for info on what has changed between the different versions of IsoriX
```

An interface between **R** packages (without help files: only 3614 lines of codes):

```
tools::package_dependencies("IsoriX") ## note: list dependencies, not suggested packages
```

```
## $IsoriX
##  [1] "graphics"     "grDevices"    "grid"         "lattice"      "latticeExtra" "numDeriv"     "raster"
##  [8] "rasterVis"    "sp"           "spaMM"        "stats"        "tools"        "utils"        "viridisLite"
```

- numDeriv, **spaMM**, stats (for statistical computation)
- **raster**, sp (for GIS)
- graphics, grDevices, grid, lattice, latticeExtra, **rasterVis**, viridisLite (for plotting)
- tools, utils (for small geeky details)

# What is IsoriX?

A collection of functions:

```
setdiff(ls("package:IsoriX"), data(package = "IsoriX")$results[, "Item"])  ## note: does not show methods and unexported functions
```

```
##  [1] "area"              "calibfit"          "Calibfit"          "create_aliens"     "CRS"               "downloadfile"
##  [7] "extent"            "extent<-"          "extract"           "get_ranPars"       "getelev"           "GetElev"
## [13] "getOption_IsoriX" "getprecip"         "gpar"              "grid.text"         "isofind"           "isofit"
## [19] "Isofit"           "isomultifit"       "isomultiscape"     "Isorix"            "isoscape"          "Isoscape"
## [25] "isosim"           "Isosim"            "layer"             "levelplot"         "options_IsoriX"    "panel.points"
## [31] "prepcipitate"     "prepdata"          "prepelev"          "prepiso"           "prepraster"        "prepsources"
## [37] "projection<-"     "queryGNIP"         "QueryGNIP"         "raster"            "RdBuTheme"         "relevate"
## [43] "RElevate"         "shift"             "sp.points"         "sp.polygons"       "values"            "xyplot"
```

# What is IsoriX?

A collection of functions:

```
setdiff(ls("package:IsoriX"), data(package = "IsoriX")$results[, "Item"])  ## note: does not show methods and unexported functions
```

```
##  [1] "area"              "calibfit"          "Calibfit"          "create_aliens"     "CRS"               "downloadfile"
##  [7] "extent"            "extent<-"          "extract"           "get_ranPars"       "getelev"           "GetElev"
## [13] "getOption_IsoriX" "getprecip"         "gpar"              "grid.text"         "isofind"           "isofit"
## [19] "Isofit"           "isomultifit"       "isomultiscape"     "Isorix"            "isoscape"          "Isoscape"
## [25] "isosim"           "Isosim"            "layer"             "levelplot"         "options_IsoriX"    "panel.points"
## [31] "prepcipitate"     "prepdata"          "prepelev"          "prepiso"           "prepraster"        "prepsources"
## [37] "projection<-"     "queryGNIP"         "QueryGNIP"         "raster"            "RdBuTheme"         "relevate"
## [43] "RElevate"         "shift"             "sp.points"         "sp.polygons"       "values"            "xyplot"
```

The main ones being:

1. `prepsources()`
2. `isofit()`
3. `prepraster()`
4. `isoscape()`
5. `calibfit()`
6. `isofind()`

All these functions (as well as the methods behind `plot()`) are documented!
So you can type e.g. `?prepsources()` to get help.

# What is IsoriX?

A collection of data to try things out or to make plots nicer:

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

```
##       Item              Title
##  [1,] "AssignDataAlien" "Simulated assignment dataset"
##  [2,] "AssignDataBat"   "Assignment dataset for bat species"
##  [3,] "AssignDataBat2"  "Assignment dataset for bat species"
##  [4,] "CalibDataAlien"  "Simulated calibration dataset"
##  [5,] "CalibDataBat"    "Calibration dataset for bat species"
##  [6,] "CalibDataBat2"   "Calibration dataset for bat species"
##  [7,] "CountryBorders"  "Borders of world CountryBorders"
##  [8,] "ElevRasterDE"    "The raster of elevation for Germany"
##  [9,] "GNIPDataALLagg"  "Hydrogen delta values in precipitation water (aggregated per location)"
## [10,] "GNIPDataDE"      "Hydrogen delta values in precipitation water, Germany"
## [11,] "GNIPDataEUagg"   "Hydrogen delta values in precipitation water (aggregated per location)"
## [12,] "OceanMask"       "Mask of world oceans"
## [13,] "PrecipBrickDE"   "The precipitation monthly amounts for Germany"
## [14,] "isopalette1"     "Colour palettes for plotting"
## [15,] "isopalette2"     "Colour palettes for plotting"
```

# What is IsoriX?

A collection of data to try things out or to make plots nicer:

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

```
##       Item              Title
##  [1,] "AssignDataAlien" "Simulated assignment dataset"
##  [2,] "AssignDataBat"   "Assignment dataset for bat species"
##  [3,] "AssignDataBat2"  "Assignment dataset for bat species"
##  [4,] "CalibDataAlien"  "Simulated calibration dataset"
##  [5,] "CalibDataBat"    "Calibration dataset for bat species"
##  [6,] "CalibDataBat2"   "Calibration dataset for bat species"
##  [7,] "CountryBorders"  "Borders of world CountryBorders"
##  [8,] "ElevRasterDE"    "The raster of elevation for Germany"
##  [9,] "GNIPDataALLagg"  "Hydrogen delta values in precipitation water (aggregated per location)"
## [10,] "GNIPDataDE"      "Hydrogen delta values in precipitation water, Germany"
## [11,] "GNIPDataEUagg"   "Hydrogen delta values in precipitation water (aggregated per location)"
## [12,] "OceanMask"       "Mask of world oceans"
## [13,] "PrecipBrickDE"   "The precipitation monthly amounts for Germany"
## [14,] "isopalette1"     "Colour palettes for plotting"
## [15,] "isopalette2"     "Colour palettes for plotting"
```

And of course you can use your own data, that is the whole point!

# What is IsoriX?

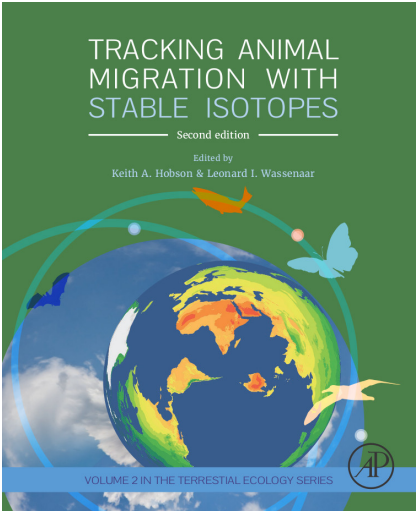
A project that aims at becoming a collaborative platform:

www.github.com/courtiol/IsoriX

# Table of contents

# Where can you learn about IsoriX?

- In our book chapter




Note: only in the 2nd edition of the book.

- In the package documentation

```
help(package = 'IsoriX')
```

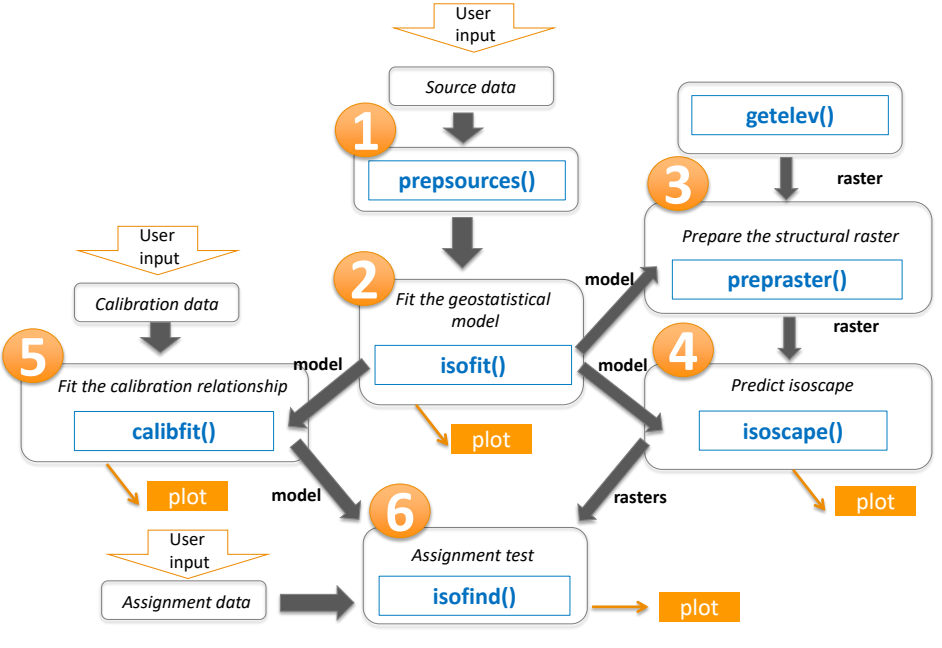
- In our (growing) bookdown

`https://bookdown.org/content/782/`

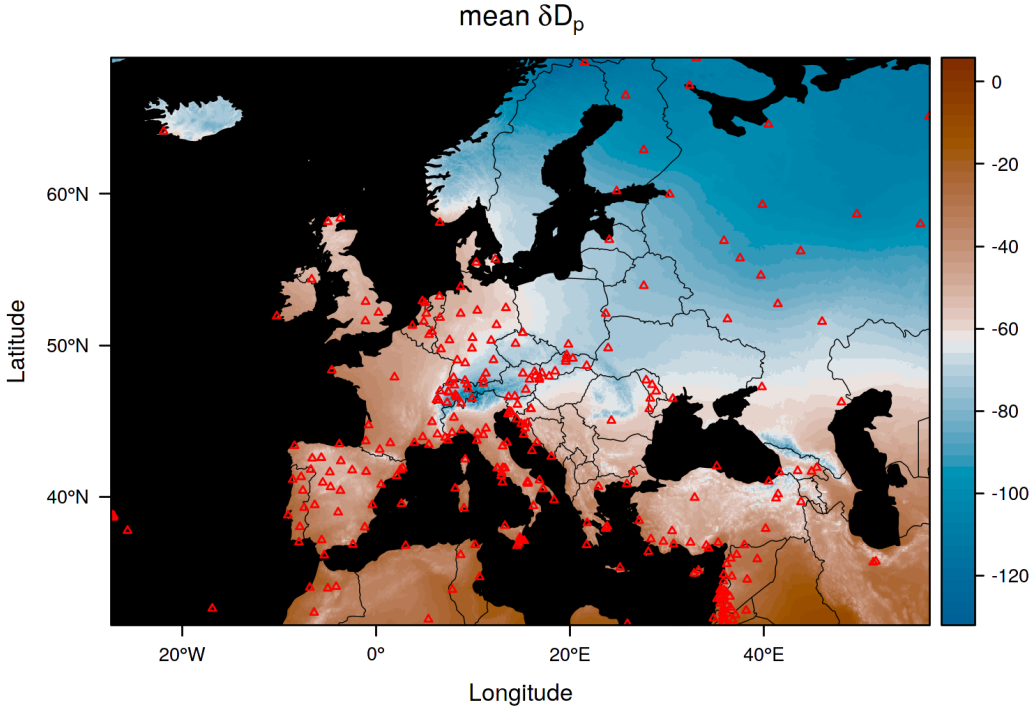
# Table of contents

# Standard workflow in 6 main functions

1. Prepare the source data with `prepsources()`
2. Fit the Isoscape model with `isofit()`
3. Prepare the structural raster with `prepraster()`
4. Predict the isoscape with `isoscape()`
5. Fit the calibration function with `calibfit()`
6. Perform the assignment with `isofind()`


User input
Source data
1
prepsources()
getelev()
raster
3
Prepare the structural raster
prepraster()
raster
User input
Calibration data
2
Fit the geostatistical model
isofit()
model
model
5
Fit the calibration relationship
calibfit()
model
model
4
Predict isoscape
isoscape()
plot
plot
plot
model
6
rasters
User input
Assignment data
Assignment test
isofind()
plot

# Table of contents


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

# Step 1. Prepare the source data

1a. Retrieve (e.g. from GNIP)/Collect data and shape them as the toy dataset `GNIPDataDE`:

```
head(GNIPDataDE)
```

```
##   source_ID   lat long elev year month source_value
## 1 SCHLESWIG 54.52 9.54   43 1997     6        -59.0
## 2 SCHLESWIG 54.52 9.54   43 1997     7        -56.0
## 3 SCHLESWIG 54.52 9.54   43 1997     8        -60.8
## 4 SCHLESWIG 54.52 9.54   43 1997     9        -51.0
## 5 SCHLESWIG 54.52 9.54   43 1997    10        -58.7
## 6 SCHLESWIG 54.52 9.54   43 1997    11        -74.6
```

```
tail(GNIPDataDE)
```

```
##                    source_ID   lat  long elev year month source_value
## 8586 GARMISCH-PARTENKIRCHEN 47.48 11.06  719 2013     7       -46.42
## 8587 GARMISCH-PARTENKIRCHEN 47.48 11.06  719 2013     8       -48.07
## 8588 GARMISCH-PARTENKIRCHEN 47.48 11.06  719 2013     9       -63.75
## 8589 GARMISCH-PARTENKIRCHEN 47.48 11.06  719 2013    10      -100.23
## 8590 GARMISCH-PARTENKIRCHEN 47.48 11.06  719 2013    11      -109.23
## 8591 GARMISCH-PARTENKIRCHEN 47.48 11.06  719 2013    12      -105.82
```

To avoid troubles:

- use a `data.frame` (and not a `tibble` or a matrix)
- `source_ID` better be a factor

# Step 1. Prepare the source data

1b. Use `prepsources()` to aggregate (and filter) the data:

```
prepsources(
  data,
  month = 1:12, year,
  long_min, long_max, lat_min, lat_max,
  split_by = NULL,
  prop_random = 0, random_level = "source",
  col_source_value = "source_value",
  col_source_ID = "source_ID", col_lat = "lat", col_long = "long",
  col_elev = "elev", col_month = "month", col_year = "year"
  )
```

# Step 1. Prepare the source data

1b. Use `prepsources()` to aggregate (and filter) the data:

```
prepsources(
  data,
  month = 1:12, year,
  long_min, long_max, lat_min, lat_max,
  split_by = NULL,
  prop_random = 0, random_level = "source",
  col_source_value = "source_value",
  col_source_ID = "source_ID", col_lat = "lat", col_long = "long",
  col_elev = "elev", col_month = "month", col_year = "year"
  )
```

Example:

```
GNIPDataDEagg <- prepsources(data = GNIPDataDE)
```

```
head(GNIPDataDEagg)
```

```
##       source_ID mean_source_value var_source_value n_source_value   lat  long elev
## 1        ARKONA         -60.99231         247.8767            134 54.67 13.43   42
## 2        ARTERN         -61.00653         510.5720            199 51.37 11.29  164
## 3 BAD SALZUFLEN         -53.80770         310.1046            408 52.10  8.75  135
## 4        BERLIN         -57.53477         395.6484            419 52.46 13.40   48
## 5  BRAUNSCHWEIG         -52.73350         339.4752            420 52.29 10.44   81
## 6      CUXHAVEN         -49.25271         221.6594            420 53.87  8.70    5
```

# Step 1. Prepare the source data

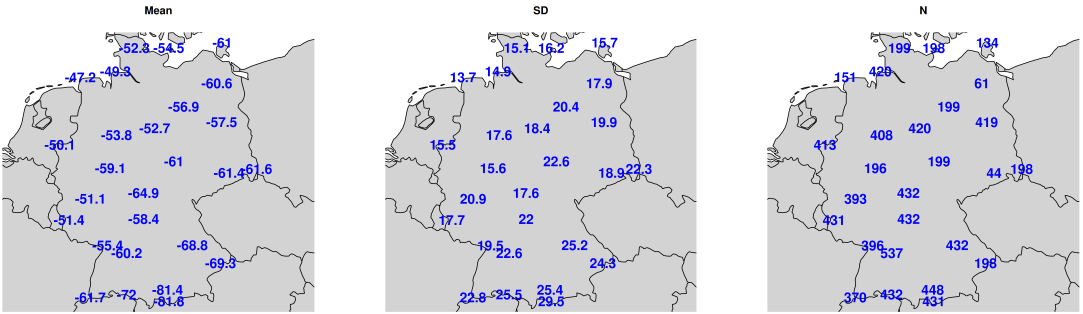
## 1c. Check your data (it is always a good idea to plot your data!)

```
library(sp) ## for plotting polygons

plot(CountryBorders, xlim = range(GNIPDataDEagg$long), ylim = range(GNIPDataDEagg$lat), col = "lightgrey", main = "Mean")
text(x = GNIPDataDEagg$long, y = GNIPDataDEagg$lat, labels = round(GNIPDataDEagg$mean_source_value, digits = 1), col = "blue", cex = 1.5, font = 2)

plot(CountryBorders, xlim = range(GNIPDataDEagg$long), ylim = range(GNIPDataDEagg$lat), col = "lightgrey", main = "SD")
text(x = GNIPDataDEagg$long, y = GNIPDataDEagg$lat, labels = round(sqrt(GNIPDataDEagg$var_source_value), digits = 1), col = "blue", cex = 1.5, font = 2)

plot(CountryBorders, xlim = range(GNIPDataDEagg$long), ylim = range(GNIPDataDEagg$lat), col = "lightgrey", main = "N")
text(x = GNIPDataDEagg$long, y = GNIPDataDEagg$lat, labels = GNIPDataDEagg$n_source_value, col = "blue", cex = 1.5, font = 2)
```



# Step 1. Prepare the source data

The most important questions you must answer:

- Which years?
- Which months?
- Which geographic area?
- Weighted by precipitation or not? If yes 2 possibilities, before or after fitting:
  - before: do the aggregation yourself (for now)
    NB: don't forget to rise the weights to the square when averaging variances!
  - after: using `split_by` and later on `isomultifit` and `isomultiscape` (but not ready yet for calibration)

# Step 1. Prepare the source data

The most important questions you must answer:

- Which years?
- Which months?
- Which geographic area?
- Weighted by precipitation or not? If yes 2 possibilities, before or after fitting:
  - before: do the aggregation yourself (for now)
    NB: don't forget to rise the weights to the square when averaging variances!
  - after: using `split_by` and later on `isomultifit` and `isomultiscape` (but not ready yet for calibration)

Issue:

- More data = more signal or more noise?

# Step 1. Prepare the source data

The most important questions you must answer:

- Which years?
- Which months?
- Which geographic area?
- Weighted by precipitation or not? If yes 2 possibilities, before or after fitting:
  - before: do the aggregation yourself (for now)
    NB: don't forget to rise the weights to the square when averaging variances!
  - after: using `split_by` and later on `isomultifit` and `isomultiscape` (but not ready yet for calibration)

Issue:

- More data = more signal or more noise?

Examples:

- The isotopic signature in an environment at time $t$ is not solely influenced by the precipitation at time $t$.
- Too small area leads to extrapolation which is unreliable but too large leads to impose a single model over several regions that may differ in how the environment relates to isotopes.

# Step 1. Prepare the source data

The most important questions you must answer:

- Which years?
- Which months?
- Which geographic area?
- Weighted by precipitation or not? If yes 2 possibilities, before or after fitting:
  - before: do the aggregation yourself (for now)
    NB: don't forget to rise the weights to the square when averaging variances!
  - after: using `split_by` and later on `isomultifit` and `isomultiscape` (but not ready yet for calibration)

Issue:

- More data = more signal or more noise?

Examples:

- The isotopic signature in an environment at time $t$ is not solely influenced by the precipitation at time $t$.
- Too small area leads to extrapolation which is unreliable but too large leads to impose a single model over several regions that may differ in how the environment relates to isotopes.

Note:

- In IsoriX it is easy to change a given line of code and re-run your entire workflow to study the impact of your choices.

# Step 1. Prepare the source data

Uncertainty check list:

- ✓ variation across space
- ✓ variation across years and months
- × variation within months (includes technical variation)

# Step 1. Prepare the source data

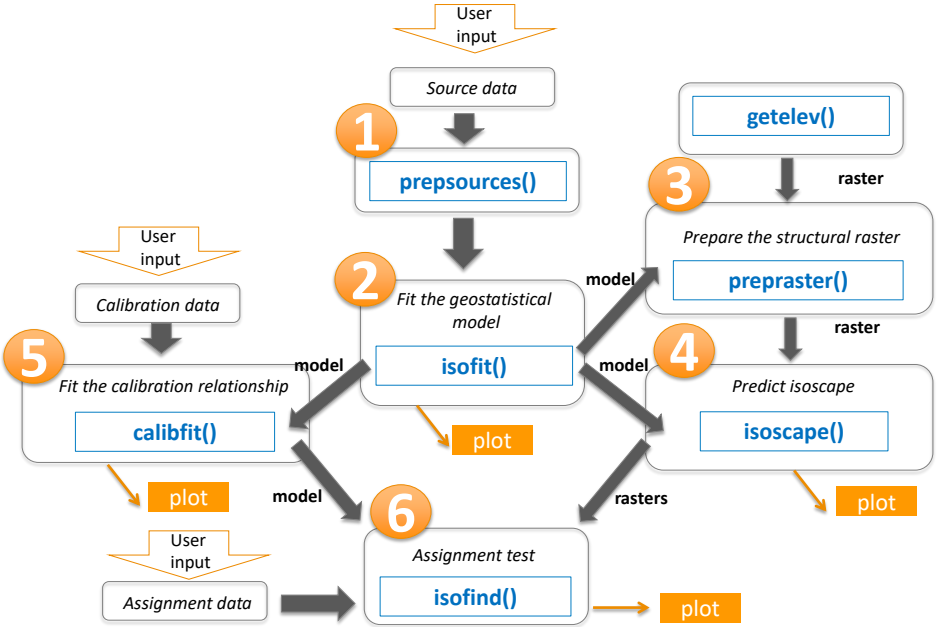
Uncertainty check list:

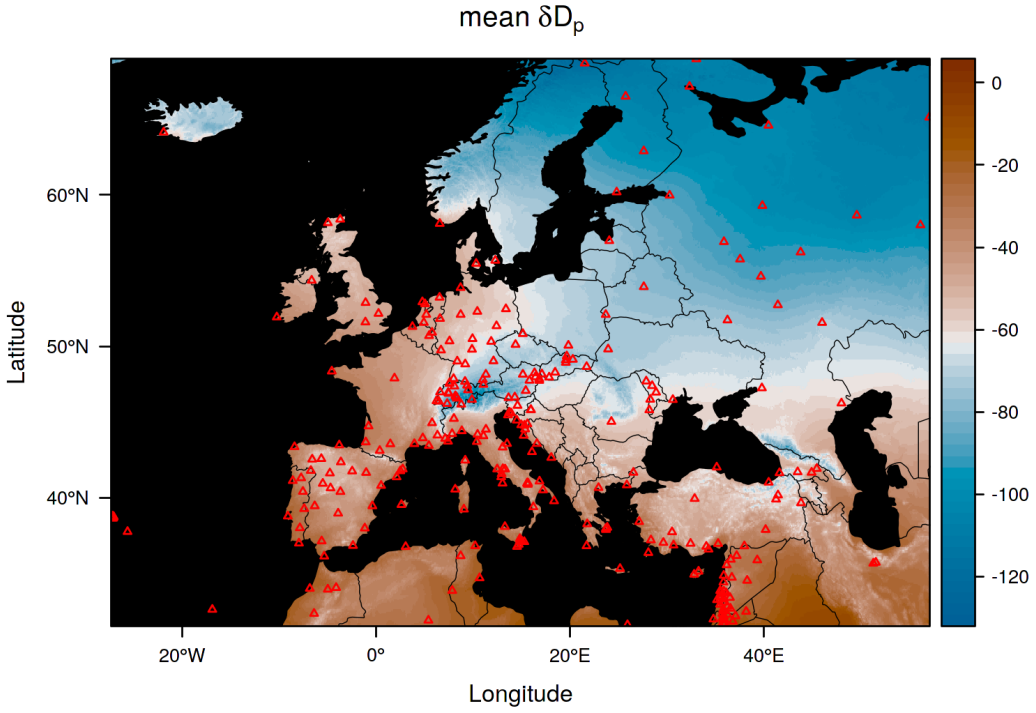
- ✓ variation across space
- ✓ variation across years and months
- × variation within months (includes technical variation)

Remember:

- Always try to be explicit about which sources of uncertainty are acounted for and which are not!
- No work can consider them all, so it is worth recognising and discussing the limitations of any approach.
- Being open source, in IsoriX, with a little effort you can figure out which assumptions are being made.

# Table of contents


User input
Source data
1
prepsources()
getelev()
raster
3
Prepare the structural raster
prepraster()
raster
User input
Calibration data
2
Fit the geostatistical model
isofit()
model
model
5
Fit the calibration relationship
calibfit()
model
plot
4
Predict isoscape
isoscape()
plot
plot
model
6
rasters
User input
Assignment data
Assignment test
isofind()
plot


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

# Step 2. Fit the isoscape

The statistical model(s) behind the isoscape:

- Mean model (LMM):

$$\texttt{Source}_{gi} = \texttt{Fix}_{g} + \texttt{Random}_{g} + \texttt{Error}_{gi}$$

# Step 2. Fit the isoscape

The statistical model(s) behind the isoscape:

- Mean model (LMM):

$$\texttt{Source}_{gi} = \texttt{Fix}_g + \texttt{Random}_g + \texttt{Error}_{gi}$$

- where

$$\texttt{Fix}_g = \beta_0 + \beta_{\texttt{lat}} \times |\texttt{lat}|_g + \beta_{\texttt{elev}} \times \texttt{elev}_g$$

$$\texttt{Random}_g = \texttt{RandomSpatial}_g + \texttt{RandomUncorr}_g$$

and

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

# Step 2. Fit the isoscape

The statistical model(s) behind the isoscape:

- Mean model (LMM):

$$\texttt{Source}_{gi} = \texttt{Fix}_g + \texttt{Random}_g + \texttt{Error}_{gi}$$

- where

$$\texttt{Fix}_g = \beta_0 + \beta_{\texttt{lat}} \times |\texttt{lat}|_g + \beta_{\texttt{elev}} \times \texttt{elev}_g$$

$$\texttt{Random}_g = \texttt{RandomSpatial}_g + \texttt{RandomUncorr}_g$$

and

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

- Residual dispersion model (disp model; $\Gamma$ 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

2a. Use `isofit()` to fit the geostatistical model(s):

```
isosfit(
  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 = TRUE, spatial = TRUE),
  disp_model_rand = list(uncorr = TRUE, 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

2a. Use `isofit()` to fit the geostatistical model(s):

```
isosfit(
  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 = TRUE, spatial = TRUE),
  disp_model_rand = list(uncorr = TRUE, 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()
  )
```

Example:

```
GermanFit <- isofit(data = GNIPDataDEagg,
                    mean_model_fix = list(elev = TRUE, lat_abs = TRUE))
```

2b. Check the fitted geostatistical model(s):

```
names(GermanFit)
```

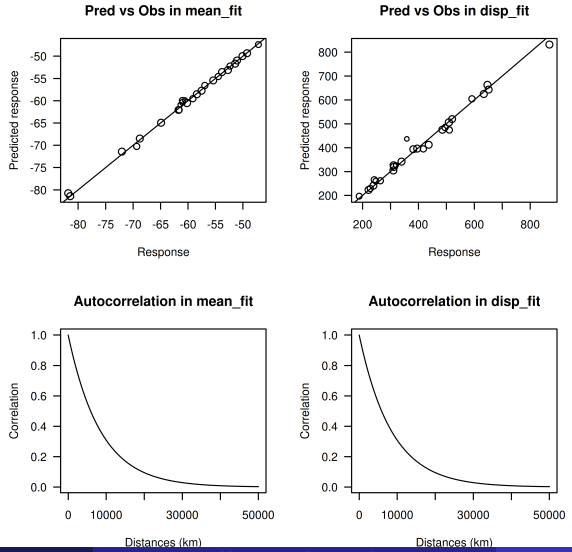
```
## [1] "mean_fit" "disp_fit" "info_fit"
```

```
GermanFit$mean_fit
```

```
## formula: mean_source_value ~ 1 + elev + lat_abs + (1 | source_ID) + Matern(1 |
##     long + lat)
## REML: Estimation of lambda, phi and corrPars by REML.
##       Estimation of fixed effects by ML.
## Family: gaussian ( link = identity )
##  ------------ Fixed effects (beta) ------------
##              Estimate  Cond. SE t-value
## (Intercept) -145.78873 65.386537  -2.230
## elev          -0.01163  0.002449  -4.750
## lat_abs        1.68318  1.210013   1.391
##  ---------------- Random effects ----------------
## Family: gaussian ( link = identity )
## Distance(s): Earth
##                    --- Correlation parameters:
##        2.nu       2.rho
## 0.5000000000 0.0001176491
##            --- Variance parameters ('lambda'):
## lambda = var(u) for u ~ Gaussian;
##    source_ID  :  6.993e-10
##    long + lat  :  482
##             --- Coefficients for log(lambda):
##       Group        Term Estimate   Cond.SE
##   source_ID (Intercept)  -21.081 2721.6553
##  long + lat (Intercept)    6.178    0.3232
## # of obs: 27; # of groups: source_ID, 27; long + lat, 27
##  ------------ Residual variance  ------------
## Prior weights: 134 199 408 419 420 ...
## phi was fixed by an offset term:  "phi" ~ 0 + offset(pred_disp)
##  ------------ Likelihood values  ------------
##                          logLik
## p_v(h) (marginal L): -71.72788
```

2b. Check the fitted geostatistical model(s):

```
plot(GermanFit)
```



2c. You can compare the predictive power of different fitted model(s):

```
GermanFit2 <- isofit(data = GNIPDataDEagg,
                     mean_model_fix = list(elev = TRUE, lat_abs = FALSE))
```

```
AIC(GermanFit$mean_fit)
```

```
##        marginal AIC:     conditional AIC:      dispersion AIC:        effective df:
##           153.455765           105.958823           147.485986             4.858697
```

```
AIC(GermanFit2$mean_fit)
```

```
##        marginal AIC:     conditional AIC:      dispersion AIC:        effective df:
##           153.453002           106.076824           151.619921             4.733679
```

2c. You can compare the predictive power of different fitted model(s):

```
GermanFit2 <- isofit(data = GNIPDataDEagg,
                     mean_model_fix = list(elev = TRUE, lat_abs = FALSE))
```

```
AIC(GermanFit$mean_fit)
```

```
##        marginal AIC:     conditional AIC:      dispersion AIC:        effective df:
##           153.455765           105.958823           147.485986             4.858697
```

```
AIC(GermanFit2$mean_fit)
```

```
##        marginal AIC:     conditional AIC:      dispersion AIC:        effective df:
##           153.453002           106.076824           151.619921             4.733679
```

Note:

- Use the conditional AIC (cAIC, which is different from AICc!) as it takes into account of the realization of the random effects.

2c. You can compare the predictive power of different fitted model(s):

```
GermanFit2 <- isofit(data = GNIPDataDEagg,
                     mean_model_fix = list(elev = TRUE, lat_abs = FALSE))
```

```
AIC(GermanFit$mean_fit)
```

```
##        marginal AIC:     conditional AIC:      dispersion AIC:        effective df:
##           153.455765           105.958823           147.485986                4.858697
```

```
AIC(GermanFit2$mean_fit)
```

```
##        marginal AIC:     conditional AIC:      dispersion AIC:        effective df:
##           153.453002           106.076824           151.619921                4.733679
```

Note:

- Use the conditional AIC (cAIC, which is different from AICc!) as it takes into account of the realization of the random effects.
- If you compare models with different structures for the residual dispersion model, you must correct the cAIC of the mean model by adding to it twice the number of parameters of the residual dispersion model. Here this latter model has 5 parameters (1 intercept, 2 variances of random effects, 2 correlation parameters for the Matern). [Better always correct the cAIC value, even if residual dispersion models are the same to get unbiased estimates of the cAIC]

# Step 2. Fit the isoscape

The most important questions you must answer:

- Did the models fit properly your data?
- Which model structure?

# Step 2. Fit the isoscape

The most important questions you must answer:

- Did the models fit properly your data?
- Which model structure?

Issue:

- More parameters = more signal or more noise?

# Step 2. Fit the isoscape

The most important questions you must answer:

- Did the models fit properly your data?
- Which model structure?

Issue:

- More parameters = more signal or more noise?

Note:

- The optimal model depends on your data!
- As before, in IsoriX it is easy to change a given line of code and re-run your entire workflow to study the impact of your choices.

# Step 2. Fit the isoscape

Uncertainty check list:

- mean fit
  - ✓ uncertainty in fixed-effect estimates
  - ✓ uncertainty in random-effect estimates
  - × uncertainty in correlation parameters
- ✓/× residual variation

# Step 2. Fit the isoscape

Uncertainty check list:

- mean fit
  - ✓ uncertainty in fixed-effect estimates
  - ✓ uncertainty in random-effect estimates
  - × uncertainty in correlation parameters
  - ✓/× residual variation

- residual dispersion fit
  - ✓ mean residual variation at a given location
  - × uncertainty in this residual variation

# Step 2. Fit the isoscape

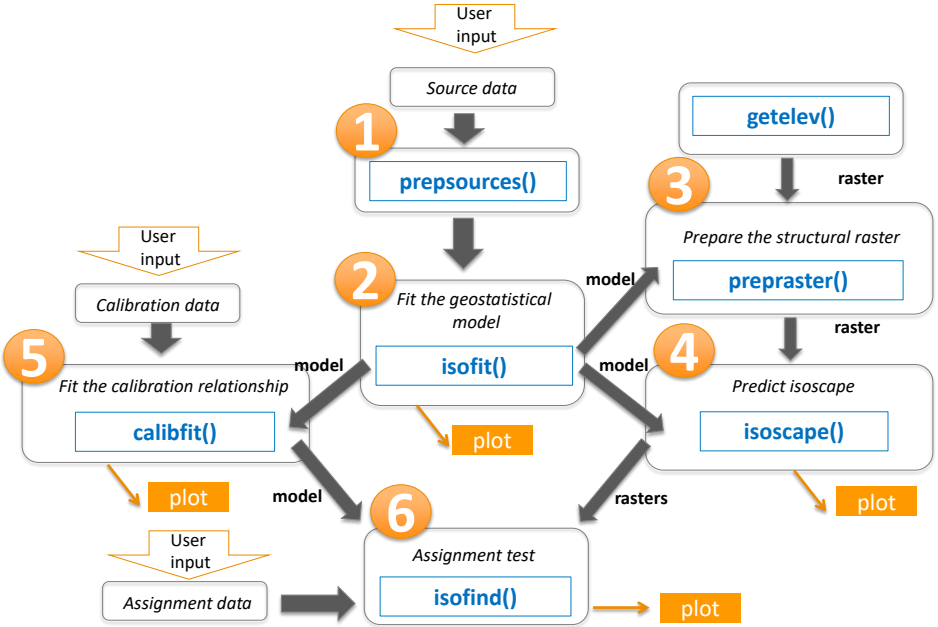
Uncertainty check list:

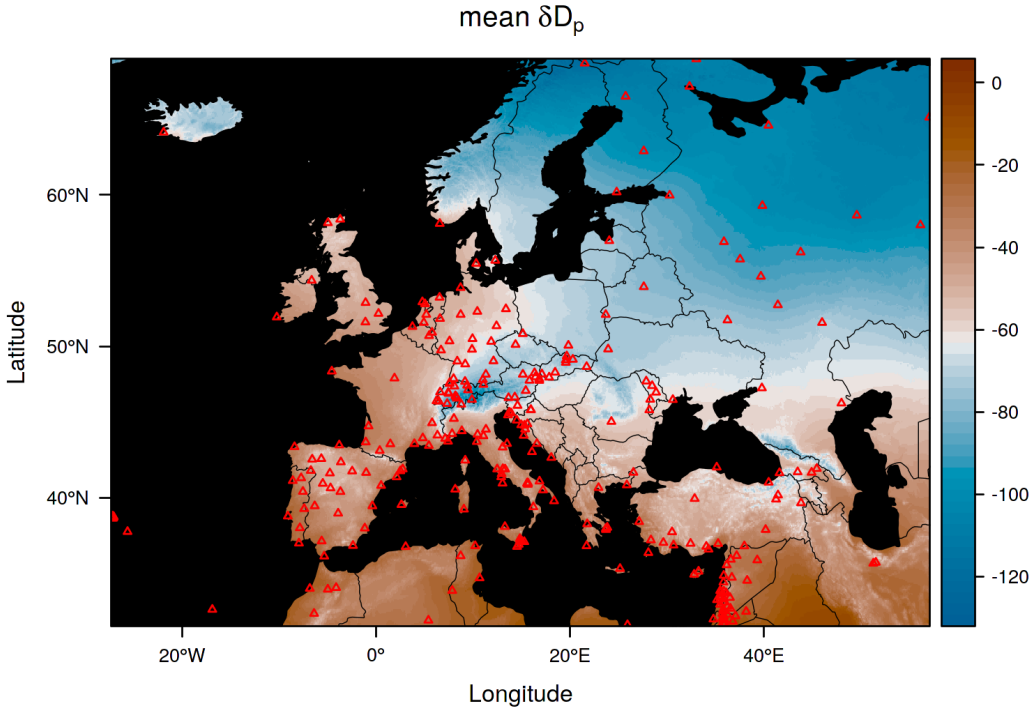
- mean fit
  - ✓ uncertainty in fixed-effect estimates
  - ✓ uncertainty in random-effect estimates
  - × uncertainty in correlation parameters
  - ✓/× residual variation
- residual dispersion fit
  - ✓ mean residual variation at a given location
  - × uncertainty in this residual variation

Note:

- Most of these uncertainties are neglected from alternative approaches!
- Some of these uncertainties are huge!!

# Table of contents


User input
Source data
1
prepsources()
getelev()
raster
3
Prepare the structural raster
prepraster()
raster
User input
Calibration data
2
Fit the geostatistical model
isofit()
model
model
5
Fit the calibration relationship
calibfit()
model
4
Predict isoscape
isoscape()
plot
plot
plot
model
6
rasters
User input
Assignment data
Assignment test
isofind()
plot


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

# Step 3. Prepare the structural raster

3a. Download your raw structural raster (e.g. an elevation raster):

The structural raster is ultimately used to provide the locations and covariates for predictions!

That is easiest if you want to use our own one (from Global Multi-resolution Terrain Elevation Data 2010):

```
getelev(path = "~/Desktop/")
```

```
ElevWorld <- raster("~/Desktop/gmted2010_30mn.tif")  ## turn the tif into raster
```

```
ElevWorld
```

```
## 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/Desktop/gmted2010_30mn.tif
## names       : gmted2010_30mn
## values      : -430, 8625  (min, max)
```

# Step 3. Prepare the structural raster

3b. Use `prepraster()` to aggregate and resize the structural raster:

```
prepraster(
  raster,
  isofit = NULL,
  margin_pct = 5,
  aggregation_factor = 0L,
  aggregation_fn = mean,
  manual_crop = NULL,
  verbose = interactive()
  )
```

3b. Use `prepraster()` to aggregate and resize the structural raster:

```
prepraster(
  raster,
  isofit = NULL,
  margin_pct = 5,
  aggregation_factor = 0L,
  aggregation_fn = mean,
  manual_crop = NULL,
  verbose = interactive()
  )
```

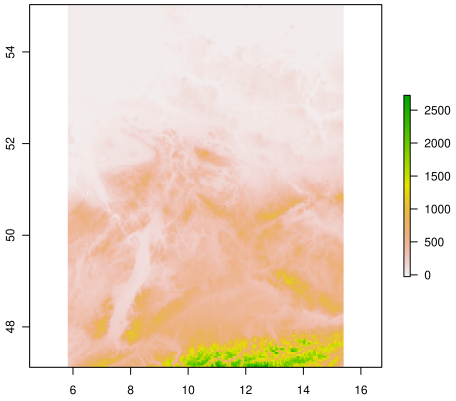
Example:

```
ElevRasterDE <- prepraster(ElevWorld, isofit = GermanFit, aggregation_factor = 5)
```

```
## class       : RasterLayer
## dimensions  : 190, 230, 43700  (nrow, ncol, ncell)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : layer
## values      : -27.72, 2725.08  (min, max)
```

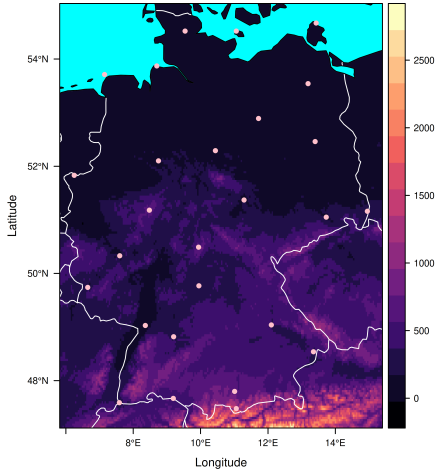
3c. Check the obtained the structural raster:

```
plot(ElevRasterDE)
```



3c. Check the obtained the structural raster:

```
levelplot(ElevRasterDE, margin = FALSE) +
  layer(sp.polygons(CountryBorders, col = "white")) +
  layer(sp.polygons(OceanMask, fill = "cyan")) +
  xyplot(GNIPDataDEagg$lat ~ GNIPDataDEagg$long, col = "pink")
```



# Step 3. Prepare the structural raster

The most important questions you must answer:

- Over which geographic area you want to draw an isoscape?
- At which resolution?
- Using which aggegation function?

# Step 3. Prepare the structural raster

The most important questions you must answer:

- Over which geographic area you want to draw an isoscape?
- At which resolution?
- Using which aggegation function?

Issue:

- Higher resolution = more precise, but requires more computer memory and computation time
- For the aggregation function, the mean is not always the best (it depends on the question!)

# Step 3. Prepare the structural raster

The most important questions you must answer:

- Over which geographic area you want to draw an isoscape?
- At which resolution?
- Using which aggegation function?

Issue:

- Higher resolution = more precise, but requires more computer memory and computation time
- For the aggregation function, the mean is not always the best (it depends on the question!)

Recommendations:

- Start coarse
- Then make try to get quite fine if you can!

# Step 3. Prepare the structural raster

Uncertainty check list:

- × uncertainty behind the raw raster
  (as isoscapes, structural rasters are the results of some smoothing procedure which contains uncertainty)
- ✓ variation between the cells of the processed raster
- × variation within the cells of the processed raster

# Step 3. Prepare the structural raster

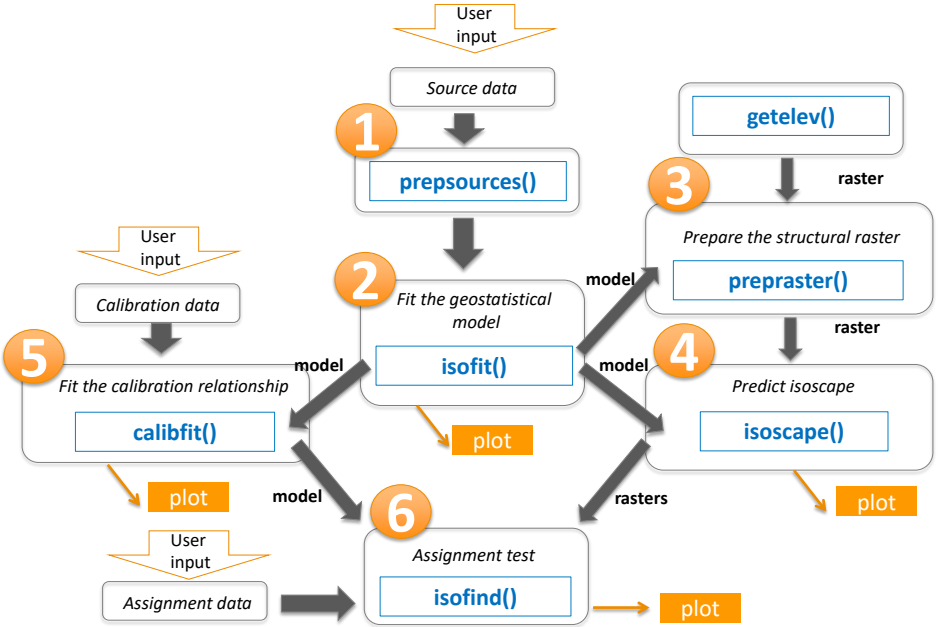
Uncertainty check list:

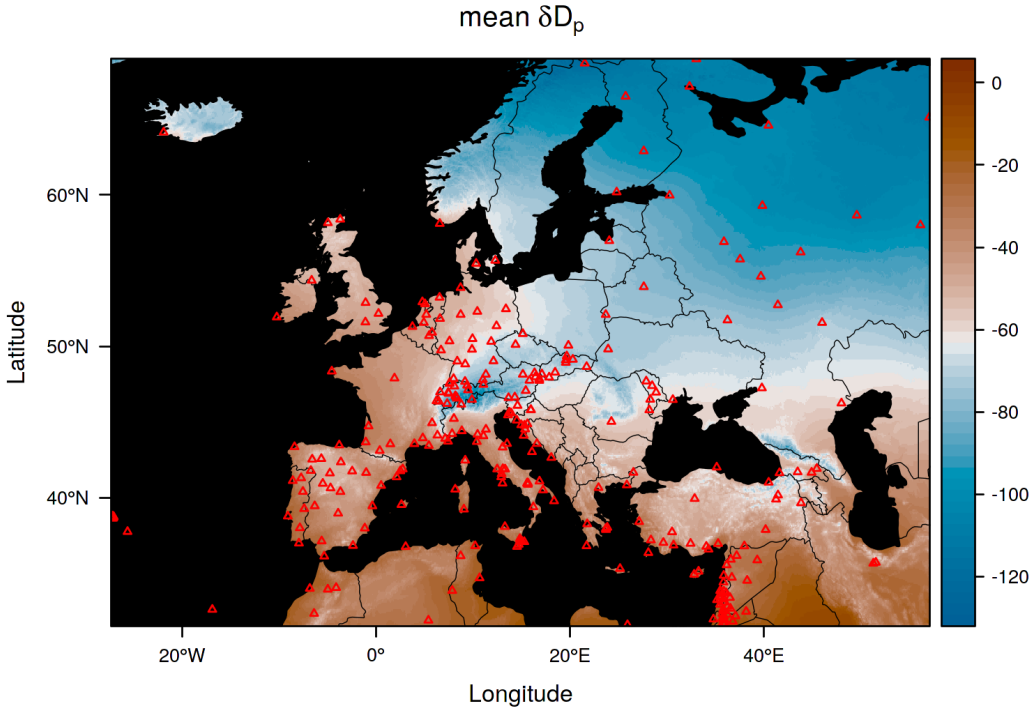
- × uncertainty behind the raw raster
  (as isoscapes, structural rasters are the results of some smoothing procedure which contains uncertainty)
- ✓ variation between the cells of the processed raster
- × variation within the cells of the processed raster

Recommendations:

- Get to know how the raster you use has been created:
  - Which uncertainty? (not just the average one)
  - Which smoothing or aggregation function?
- Increasing the resolution reduces the variance not considered!

# Table of contents


User input
Source data
1
prepsources()
getelev()
raster
3
Prepare the structural raster
prepraster()
raster
User input
Calibration data
2
Fit the geostatistical model
isofit()
model
model
5
Fit the calibration relationship
calibfit()
model
plot
4
Predict isoscape
isoscape()
plot
plot
model
6
rasters
User input
Assignment data
Assignment test
isofind()
plot


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

# Step 4. Predict the isoscape

4a. Use `isoscape()` predicts the isoscape:

```
isoscape(
  raster,
  isofit,
  verbose = interactive()
  )
```

# Step 4. Predict the isoscape

4a. Use `isoscape()` predicts the isoscape:

```
isoscape(
  raster,
  isofit,
  verbose = interactive()
  )
```

Example:

```
GermanScape <- isoscape(raster = ElevRasterDE, isofit = GermanFit)
```

```
## Warning in .predict_body(object = object, newdata = newdata, re.form = re.form, :  Prior weights
are not taken in account in residVar computation.
## Warning in .calc_logdisp_cov(res, dvdloglamMat = dvdloglamMat, dvdlogphiMat = dvdlogphiMat, :  phi
dispVar component not yet available for phi model != ~1.
```

```
names(GermanScape)
```

```
## [1] "isoscapes" "sp_points"
```

# Step 4. Predict the isoscape

There are 8 rasters behind the isoscape:

```
GermanScape$isoscapes
```

```
## class       : RasterBrick
## dimensions  : 190, 230, 43700, 8  (nrow, ncol, ncell, nlayers)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :          mean,  mean_predVar, mean_residVar,  mean_respVar,          disp,  disp_predVar, disp_residVar,  disp_respVar
## min values  : -103.21314894,  128.66277282,  195.32101271,  333.66430742,  195.32101271,    0.01424225,    2.00000000,  798.51111596
## max values  : -4.672852e+01,  1.680549e+02,  7.748267e+02,  9.300578e+02,  7.748267e+02,  8.217779e-02,  2.000000e+00,  3.734482e+04
```

# Step 4. Predict the isoscape

There are 8 rasters behind the isoscape:

```
GermanScape$isoscapes
```

```
## class       : RasterBrick
## dimensions  : 190, 230, 43700, 8  (nrow, ncol, ncell, nlayers)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :          mean,  mean_predVar, mean_residVar,  mean_respVar,          disp,  disp_predVar, disp_residVar,  disp_respVar
## min values  : -103.21314894,  128.66277282,  195.32101271,  333.66430742,  195.32101271,    0.01424225,    2.00000000,  798.51111596
## max values  : -4.672852e+01,  1.680549e+02,  7.748267e+02,  9.300578e+02,  7.748267e+02,  8.217779e-02,  2.000000e+00,  3.734482e+04
```

- `mean` = prediction of the mean isotopic value
- `mean_predVar` = quantify the uncertainty associated to `mean`
- `mean_residVar` = prediction of the variance between new observations

# Step 4. Predict the isoscape

There are 8 rasters behind the isoscape:

```
GermanScape$isoscapes

## class       : RasterBrick
## dimensions  : 190, 230, 43700, 8  (nrow, ncol, ncell, nlayers)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :          mean,  mean_predVar, mean_residVar,  mean_respVar,          disp,  disp_predVar, disp_residVar,  disp_respVar
## min values  : -103.21314894,  128.66277282,  195.32101271,  333.66430742,  195.32101271,    0.01424225,    2.00000000,  798.51111596
## max values  : -4.672852e+01,  1.680549e+02,  7.748267e+02,  9.300578e+02,  7.748267e+02,  8.217779e-02,  2.000000e+00,  3.734482e+04
```

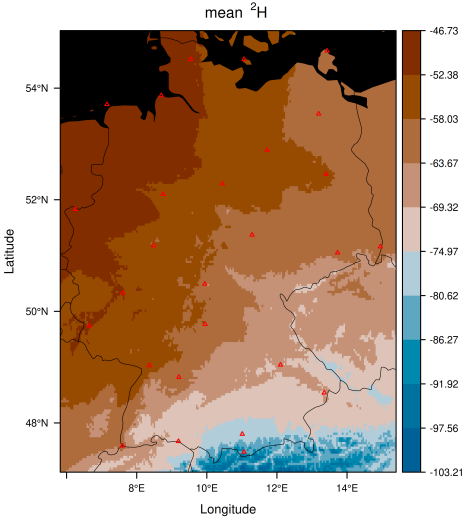
- `mean` = prediction of the mean isotopic value
- `mean_predVar` = quantify the uncertainty associated to `mean`
- `mean_residVar` = prediction of the variance between new observations
- `mean_respVar` = `mean_predVar` + `mean_residVar`
- `disp` = `mean_residVar`
- `disp_predVar` = quantify the uncertainty associated to `disp`
- `disp_residVar` = 2
- `disp_respVar` = exp(`disp_predVar` + `disp_residVar`)

4b. Plot the rasters with `plot()`:

```
plot.ISOSCAPE(
  x,
  which   = "mean",
  y_title = list(which = TRUE, title = bquote(delta^2 * H)),
  sources = list(draw = TRUE, cex = 0.5, pch = 2, lwd = 1, col = "red"),
  borders = list(borders = NA, lwd = 0.5, col = "black"),
  mask    = list(mask = NA, lwd = 0, col = "black", fill = "black"),
  palette = list(step = NA, range = c(NA, NA), n_labels = 11, digits = 2, fn = NA),
  plot    = TRUE,
  sphere  = list(build = FALSE, keep_image = FALSE),
  ...
  )
```

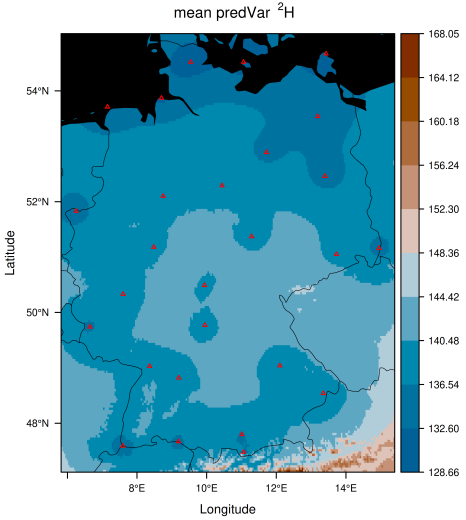
4b. Plot the rasters with `plot()`:

```
plot(GermanScape)
```



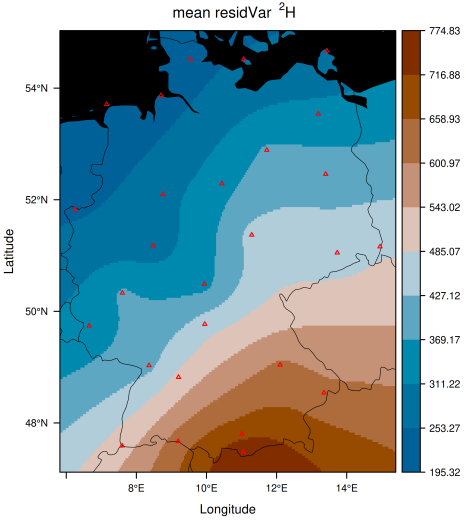
4b. Plot the rasters with `plot()`:

```
plot(GermanScape, which = "mean_predVar")
```



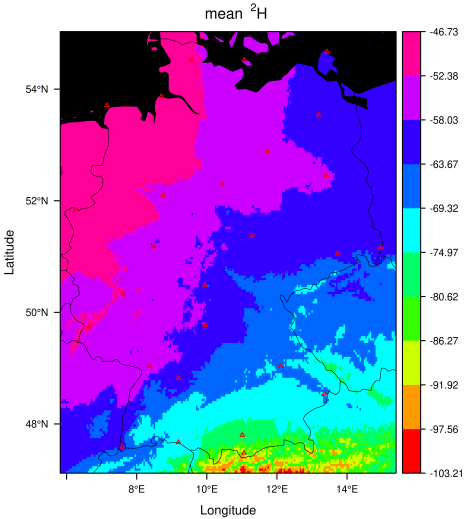
4b. Plot the rasters with `plot()`:

```
plot(GermanScape, which = "mean_residVar")
```



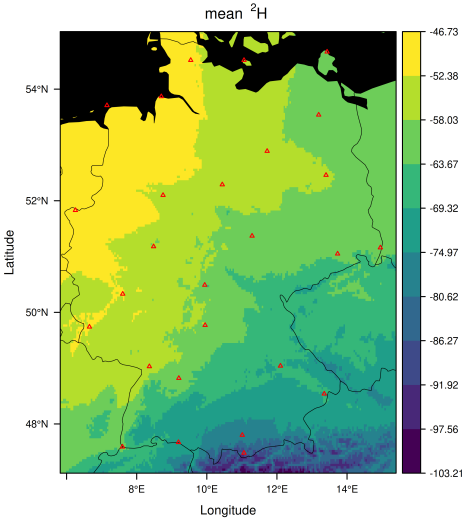
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     palette = list(fn = rainbow))
```



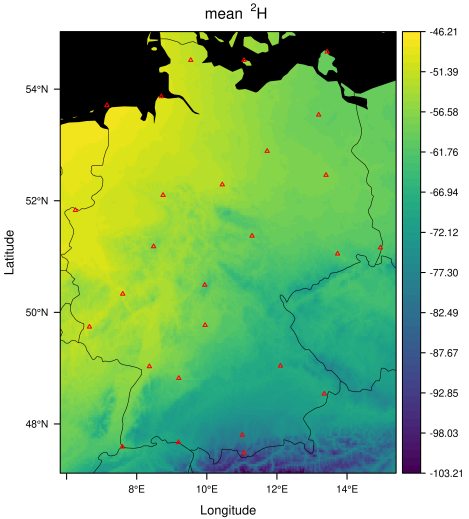
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     palette = list(fn = NULL))
```



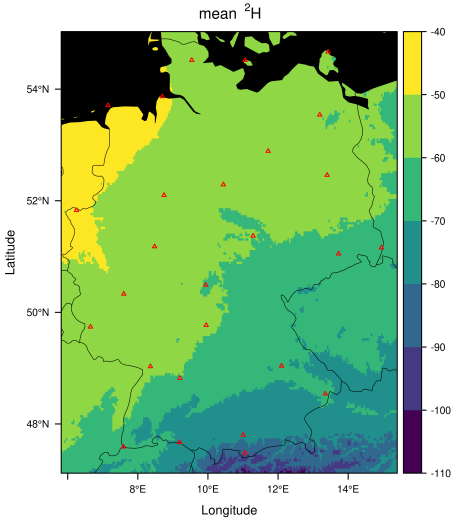
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     palette = list(fn = NULL, step = 1))
```



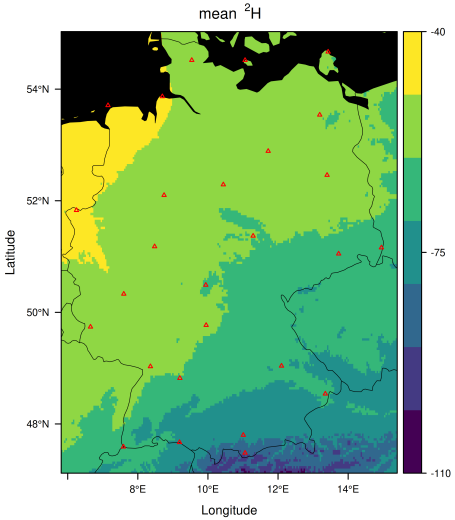
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     palette = list(fn = NULL, step = 10, range = c(-110, -40)))
```



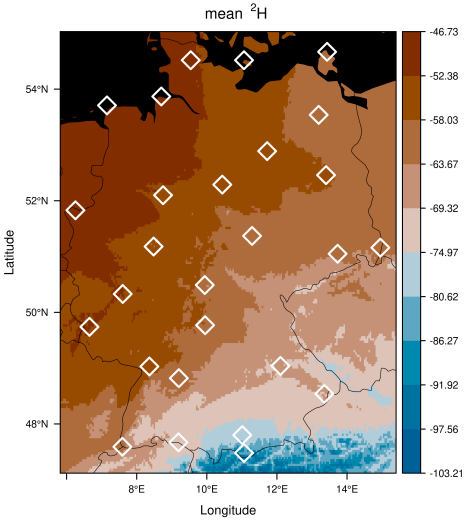
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     palette = list(fn = NULL, step = 10, range = c(-110, -40), n_labels = 3))
```



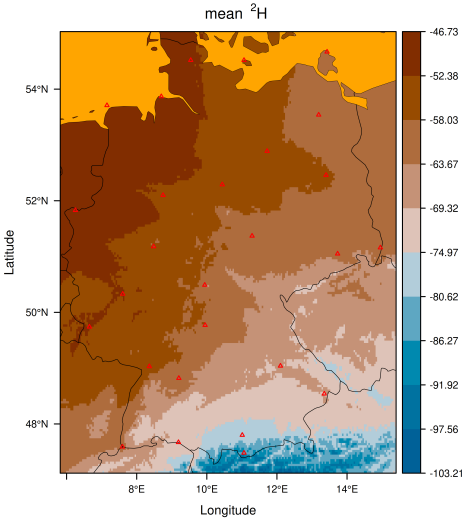
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     sources = list(draw = TRUE, cex = 2, pch = 5, lwd = 2, col = "white"))
```



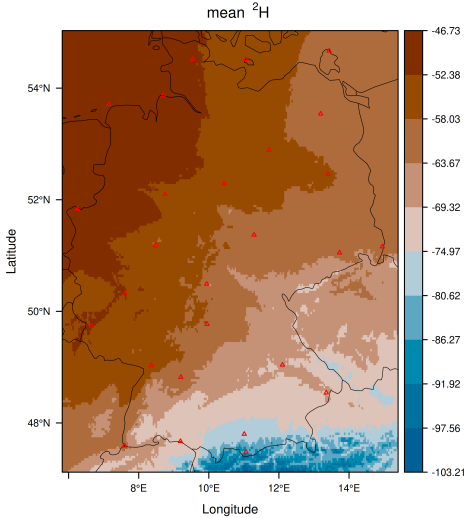
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     mask  = list(fill = "orange"))
```



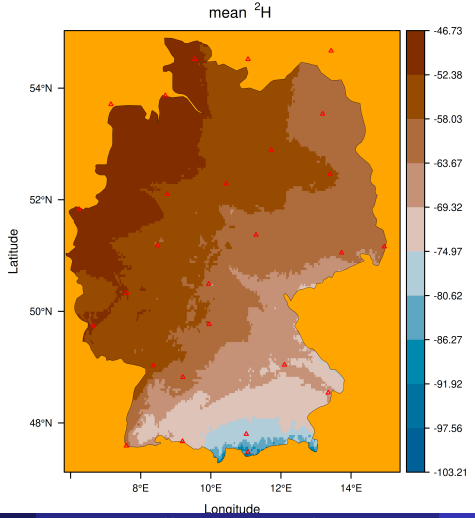
4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     mask  = list(mask = NULL))
```



4b. Plot the rasters with `plot()`:

```
plot(GermanScape,
     mask  = list(mask = rbind(OceanMask, CountryBorders[names(CountryBorders) != "Germany"]),
                  fill = "orange"))
```



# Step 4. Predict the isoscape

If the plotting possibilities of IsoriX are not good enough for you, you can always export the raster for other GIS software (e.g. QGIS).

Example:

```
library(raster)

writeRaster(GermanScape$isoscapes$mean,
            filename = "GermanScape.asc",
            format = "ascii",
            overwrite = TRUE,
            NAflag = -9999)

writeRaster(GermanScape$isoscapes$mean,
            filename = "GermanScape.tif",
            format = "GTiff",
            overwrite = TRUE,
            NAflag = -9999)
```

Note: you can also export other spatial objects included or created with IsoriX.

# Step 4. Predict the isoscape

Uncertainty check list → same as for `isofit` (cf Step 2).

# Step 4. Predict the isoscape

Uncertainty check list → same as for `isofit` (cf Step 2).

The most important questions you must answer:

- How many colours?
- Which colours?

# Step 4. Predict the isoscape

Uncertainty check list → same as for `isofit` (cf Step 2).

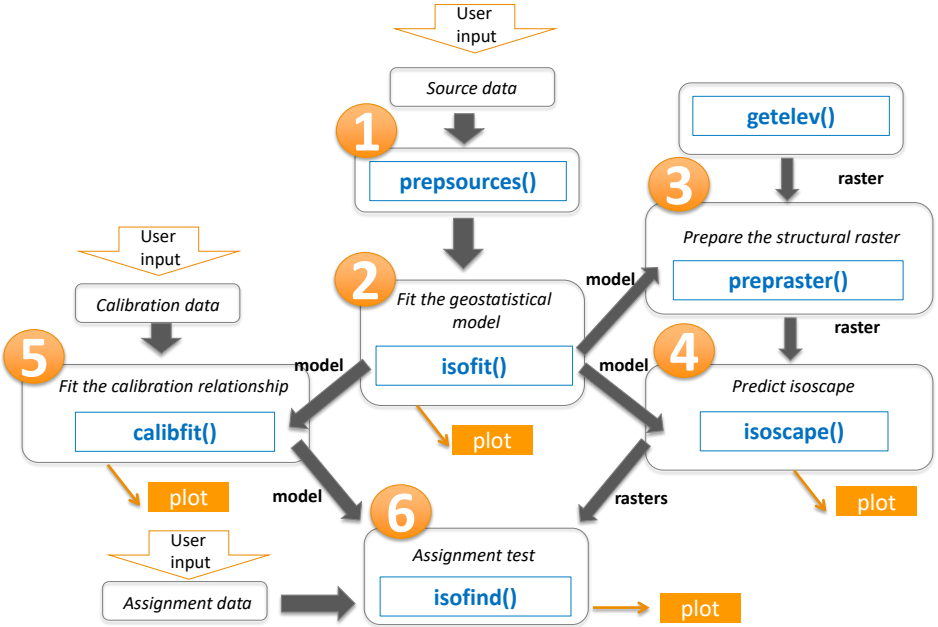
The most important questions you must answer:

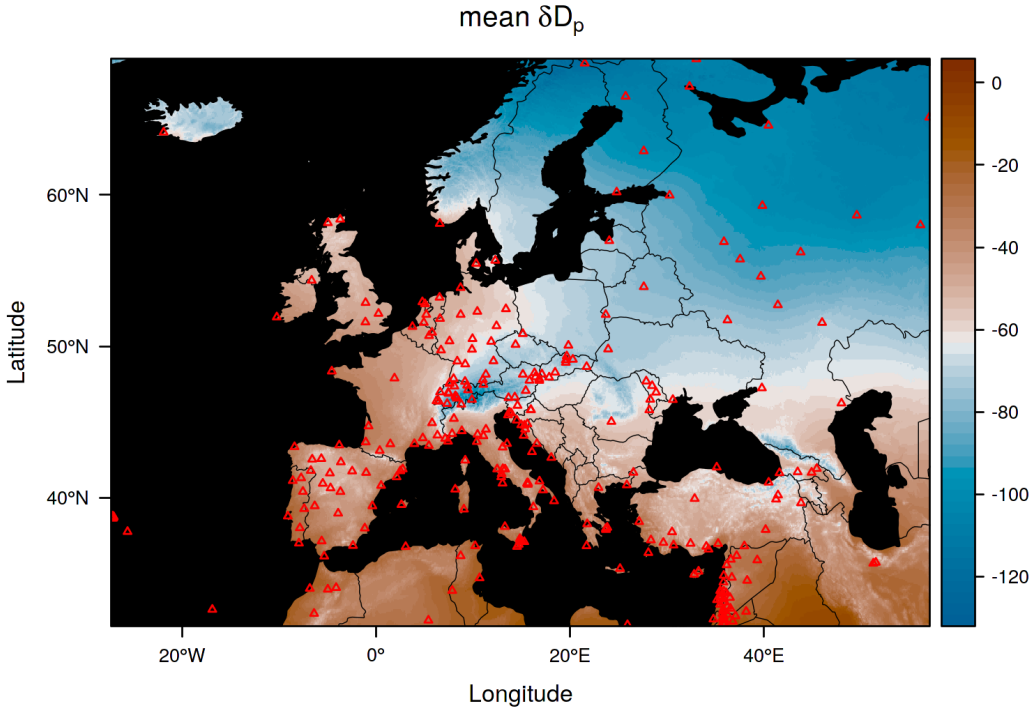
- How many colours?
- Which colours?

Recommendation:

- Perceptually uniform palettes are best
- Think of B&W print, think of colour-blind people

# Table of contents


User input
Source data
1
prepsources()
getelev()
raster
3
Prepare the structural raster
prepraster()
raster
User input
Calibration data
2
Fit the geostatistical model
isofit()
model
model
5
Fit the calibration relationship
calibfit()
model
4
Predict isoscape
isoscape()
plot
plot
plot
model
6
rasters
User input
Assignment data
Assignment test
isofind()
plot


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

# Step 5. Fit the calibration function

5a. Prepare your calibration data:

```
head(CalibDataBat)
```

```
##   site_ID     long      lat elev sample_ID species sample_value
## 1 site_41 18.87101 46.19747   90         1       4        -71.8
## 2 site_41 18.87101 46.19747   90         2       4       -103.4
## 3 site_41 18.87101 46.19747   90         3       4        -68.9
## 4 site_41 18.87101 46.19747   90         4       4       -103.3
## 5 site_67 19.10489 48.53405  522         5       4        -92.3
## 6 site_67 19.10489 48.53405  522         6       4       -102.0
```

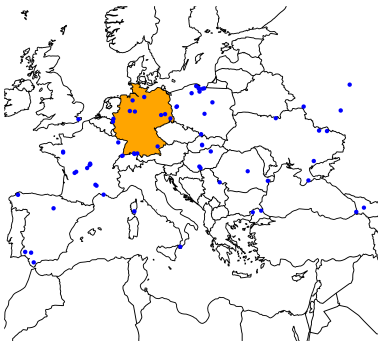
```
str(CalibDataBat)
```

```
## 'data.frame': 335 obs. of  7 variables:
##  $ site_ID     : Factor w/ 71 levels "site_10","site_11",..: 33 33 33 33 60 60 60 11 11 11 ...
##  $ long        : num  18.9 18.9 18.9 18.9 19.1 ...
##  $ lat         : num  46.2 46.2 46.2 46.2 48.5 ...
##  $ elev        : int  90 90 90 90 522 522 522 20 20 20 ...
##  $ sample_ID   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ species     : Factor w/ 6 levels "1","2","3","4",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ sample_value: num  -71.8 -103.4 -68.9 -103.3 -92.3 ...
```

Note: you need several replicates per location!

5b. Check your calibration data:

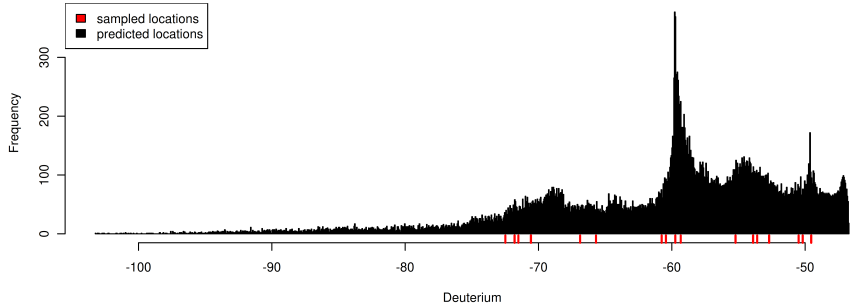
```
plot(CountryBorders, xlim = range(CalibDataBat$long), ylim = range(CalibDataBat$lat))
plot(CountryBorders["Germany"], col = "orange", add = TRUE)
plot(SpatialPoints(coords = CalibDataBat[, c("long", "lat")]), col = "blue", pch = 20, add = TRUE)
```



Note: this is NOT good! (but we will do with that for this tutorial)

5b. Check your calibration data:

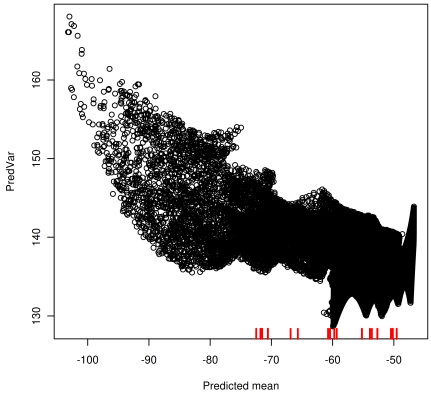
```
IsoPredicted <- values(GermanScape$isoscapes$mean)
IsoAtBats <- extract(GermanScape$isoscapes$mean, SpatialPoints(coords = CalibDataBat[, c("long", "lat")]))
hist(IsoPredicted, nclass = 1000, main = "", xlab = "Deuterium")
rug(IsoAtBats, col = "red", lwd = 2)
legend("topleft", fill = c("red", "black"), legend = c("sampled locations", "predicted locations"))
```




Note: this is quite good but very few points are considered (only the German ones).

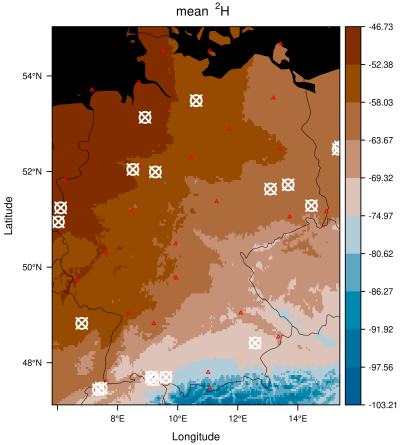
5b. Check your calibration data:

```
IsoPredictedPredVar <- values(GermanScape$isoscapes$mean_predVar)
plot(x = IsoPredicted, y = IsoPredictedPredVar, xlab = "Predicted mean", ylab = "PredVar")
rug(IsoAtBats, col = "red", lwd = 2)
```



5b. Check your calibration data:

```
plot(GermanScape, plot = FALSE) +
  xyplot(CalibDataBat$lat ~ CalibDataBat$long,
         pch = 13, col = "white", cex = 2, lwd = 2, panel = panel.points)
```



# Step 5. Fit the calibration function

5c. Use `calibfit()` to fit the calibration function:

```
calibfit(
  data,
  isofit,
  verbose = interactive(),
  control_optim = list()
  )
```

# Step 5. Fit the calibration function

5c. Use `calibfit()` to fit the calibration function:

```
calibfit(
  data,
  isofit,
  verbose = interactive(),
  control_optim = list()
  )
```

Example:

```
CalibBats <- calibfit(data = CalibDataBat, isofit = GermanFit)
```

# Step 5. Fit the calibration function

5d. Check the fitted the calibration model:

```
CalibBats
```

```
##
## Fixed effect estimates of the calibration fit
## sample_value = intercept + slope * mean_source_value + corrMatrix(1|site_ID) + slope^2 * (1|site_ID) + Error
##
##            intercept (+/- SE) = -34.29 +/- 12.91
##            slope     (+/- SE) = 0.85 +/- 0.14
##
## [for more information, use summary()]
```

5d. Check the fitted the calibration model:

```
CalibBats
```

```
##
## Fixed effect estimates of the calibration fit
## sample_value = intercept + slope * mean_source_value + corrMatrix(1|site_ID) + slope^2 * (1|site_ID) + Error
##
##            intercept (+/- SE) = -34.29 +/- 12.91
##            slope     (+/- SE) = 0.85 +/- 0.14
##
## [for more information, use summary()]
```

The model is quite complex...

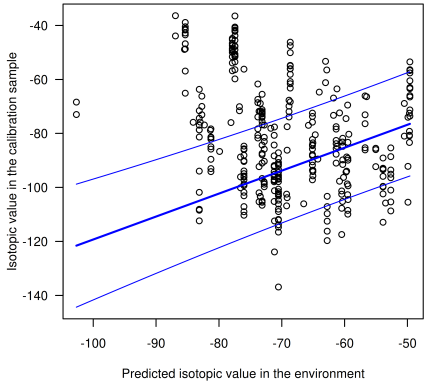
We would like to fit the model $\text{calib}_{gi} = \beta_0^C + \beta_\delta^C \delta_g + \epsilon_{gi}^C$.

However we often don't have $\delta_g$, but only $\hat{\delta}_g$ and you should not fit $\text{calib}_{gi} = \beta_0^C + \beta_\delta^C \hat{\delta}_g + \varepsilon_{gi}^C$, because there is error on $y$ (usual) but also on $x$ (violating exogeneity assumption).

Which leads us to $\text{calib}_{gi} = \beta_0^C + \beta_\delta^C \hat{\delta}_g + \beta_\delta^C \left(\delta_g - \hat{\delta}_g\right) + \epsilon_{gi}^C$, where difference $\delta_g - \hat{\delta}_g$ represents the prediction error of the mean source value at the sampling location $g$ by the mean fit. We deduce the prediction error from the prediction covariance matrix (hence the square above).

5d. Check the fitted the calibration model:

```
plot(CalibBats)
```




Note: this looks bad, but in fact it is increadibly good! Despite huge extrapolation, the fit is very close to a calibration fitted on an european isoscape (see `https://bookdown.org/content/782/calibration.html`). The reason why it works so well is that the uncertainty in $x$ is accounted for.

# Step 5. Fit the calibration function

The most important questions you must answer:

- How to get the best possible calibration data?
- Do they sample the full range of predicted source isoscape values?
- Are the samples of similar type than the ones you want to assign?
  - same species?
  - same reproductive state?
  - same age?
  - same diet?
  - ...

# Step 5. Fit the calibration function

The most important questions you must answer:

- How to get the best possible calibration data?
- Do they sample the full range of predicted source isoscape values?
- Are the samples of similar type than the ones you want to assign?
  - same species?
  - same reproductive state?
  - same age?
  - same diet?
  - ...

Recommendation:

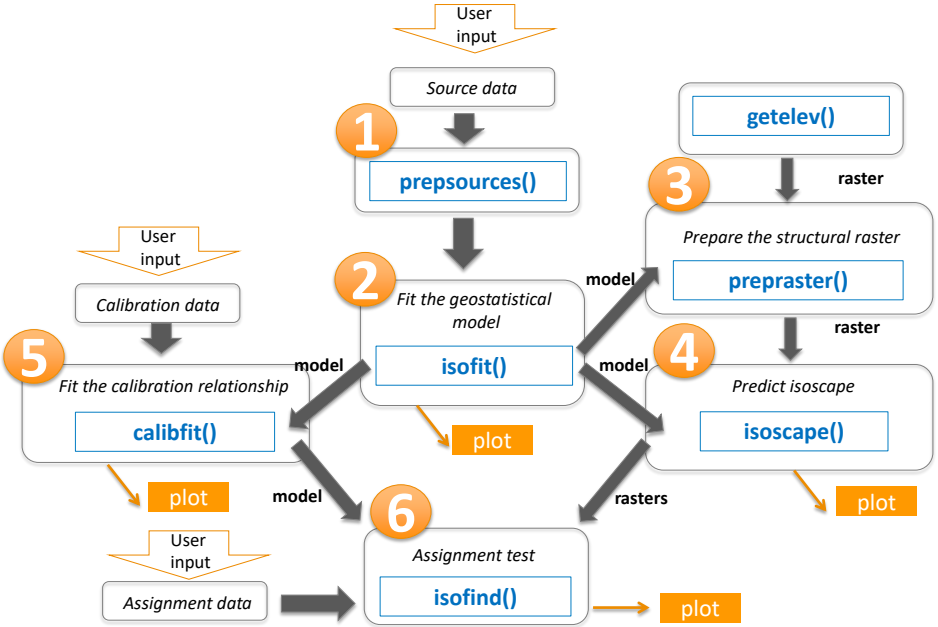
- do your very best, this is a critical step.

# Step 5. Fit the calibration function

Uncertainty check list:

- ✓ uncertainty in the isoscape
- ✓ uncertainty in the intercept and slope of the calibration function
- ✓ residual uncertainty

# Table of contents


User input
Source data
1
prepsources()
getelev()
raster
3
Prepare the structural raster
prepraster()
raster
User input
Calibration data
2
Fit the geostatistical model
isofit()
model
model
5
Fit the calibration relationship
calibfit()
model
4
Predict isoscape
isoscape()
plot
plot
plot
model
6
rasters
User input
Assignment data
Assignment test
isofind()
plot

# Step 6. Perform the assignment

6a. Prepare your assignment data:

```
AssignDataBat
```

```
##    sample_ID      lat     long sample_value
## 1     Nnoc_1 51.72323 13.67630    -86.58500
## 2     Nnoc_2 51.72323 13.67630    -93.96700
## 3     Nnoc_3 51.72323 13.67630    -88.71600
## 4     Nnoc_4 51.69520 14.64945    -84.85700
## 5     Nnoc_5 51.70366 14.63915   -104.21900
## 6     Nnoc_7 51.69520 14.64945    -85.58300
## 7     Nnoc_8 51.70536 14.63289    -89.99500
## 8     Nnoc_9 51.70536 14.63289    -92.62700
## 9    Nnoc_10 51.89348 13.76488    -85.55300
## 10   Nnoc_11 51.84883 13.64525    -87.05000
## 11   Nnoc_12 51.60535 11.92700    -89.34916
## 12   Nnoc_13 51.96695 12.08462   -103.65232
## 13   Nnoc_15 51.96695 12.08462   -121.18926
## 14   Nnoc_16 51.45457 11.50689    -98.49742
```

Our question: among the bats found dead at the bottom of wind turbines, could some be migratory bats?

# Step 6. Perform the assignment

6b. Use `isofind` performs the assignment test:

```
isofind(
  data,
  isoscape,
  calibfit = NULL,
  mask = NA,
  verbose = interactive()
  )
```

# Step 6. Perform the assignment

6b. Use `isofind` performs the assignment test:

```
isofind(
  data,
  isoscape,
  calibfit = NULL,
  mask = NA,
  verbose = interactive()
  )
```

Example:

```
AssignedBats <- isofind(data = AssignDataBat, isoscape = GermanScape, calibfit = CalibBats)

names(AssignedBats)

## [1] "sample"    "group"     "sp_points"
```

# Step 6. Perform the assignment

```
AssignedBats$sample
```

```
## $stat
## class       : RasterBrick
## dimensions  : 190, 230, 43700, 14  (nrow, ncol, ncell, nlayers)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :     Nnoc_1,     Nnoc_2,     Nnoc_3,     Nnoc_4,     Nnoc_5,     Nnoc_7,     Nnoc_8,     Nnoc_9,    Nnoc_10,    Nnoc_11,    Nnoc_12,    Nnoc_13,    Nnoc
## min values  :  -14.35264,  -23.03845,  -16.86002,  -12.31944,  -35.10116,  -13.17366,  -18.36491,  -21.46178,  -13.13836,  -14.89977,  -17.60500,  -34.43440,  -55.06
## max values  : 41.6854323, 32.9996196, 39.1780540, 43.7186326, 20.9369059, 42.8644061, 37.6731563, 34.5762911, 42.8997047, 41.1383038, 38.4330696, 21.6036709,  0.9693
##
##
## $stat_var
## class       : RasterBrick
## dimensions  : 190, 230, 43700, 14  (nrow, ncol, ncell, nlayers)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :   Nnoc_1,   Nnoc_2,   Nnoc_3,   Nnoc_4,   Nnoc_5,   Nnoc_7,   Nnoc_8,   Nnoc_9,  Nnoc_10,  Nnoc_11,  Nnoc_12,  Nnoc_13,  Nnoc_15,  Nnoc_16
## min values  :       0,       0,       0,       0,       0,       0,       0,       0,       0,       0,       0,       0,       0,       0
## max values  : 424.6313, 428.1552, 425.2054, 424.4298, 440.2086, 424.4857, 425.7227, 427.1949, 424.4825, 424.7259, 425.4453, 439.3250, 478.4556, 432.4551
##
##
## $pv
## class       : RasterBrick
## dimensions  : 190, 230, 43700, 14  (nrow, ncol, ncell, nlayers)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       :    Nnoc_1,    Nnoc_2,    Nnoc_3,    Nnoc_4,    Nnoc_5,    Nnoc_7,    Nnoc_8,    Nnoc_9,   Nnoc_10,   Nnoc_11,   Nnoc_12,   Nnoc_13,   Nnoc_15,   Nnoc_1
## min values  :        0,         0,         0,         0,         0,         0,         0,         0,         0,         0,         0,         0,         0,
## max values  : 0.9999895, 0.9999513, 0.9999984, 0.9999981, 0.9999486, 0.9999966, 0.9999896, 0.9999962, 0.9999882, 0.9999769, 0.9999977, 0.9999878, 0.9997887, 0.999980
```

## Step 6. Perform the assignment

Some important facts about the assignment test:

- The test is performed at the level of each sample and each cell defined by the structural raster.
- The null hypothesis of the test is that the sample comes from the cell being examined.
- A large p-value (i.e. non significant) indicates support for the null hypothesis.
- A small p-value (i.e. significant) indicates the rejection of the null hypothesis.

## Step 6. Perform the assignment

Some important facts about the assignment test:

- The test is performed at the level of each sample and each cell defined by the structural raster.
- The null hypothesis of the test is that the sample comes from the cell being examined.
- A large p-value (i.e. non significant) indicates support for the null hypothesis.
- A small p-value (i.e. significant) indicates the rejection of the null hypothesis.

Note:

- A very high p-value does not imply certainty of the location of origin, only that the isotopic signature between the location of origin and the location under evaluation are similar.
- A lack of rejection may stem from a good match, a large uncertainty, or both.
- The statistics behind the test are quite complex; see Appendix of book chapter.

# Step 6. Perform the assignment

You can extract the outcome of the assignment test at a given location!

Example 1: may the first bat (`Nnoc_1`) come from where it has been found?

```
extract(AssignedBats$sample$stat[[1]], cbind(AssignDataBat$long[1], AssignDataBat$lat[1]))
```

```
##
## -1.796566
```

```
extract(AssignedBats$sample$pv[[1]], cbind(AssignDataBat$long[1], AssignDataBat$lat[1]))
```

```
##
## 0.928049
```

Answer: it is possible (we cannot reject the proposition).

# Step 6. Perform the assignment

You can extract the outcome of the assignment test at a given location!

Example 1: may the first bat (Nnoc_1) come from where it has been found?

```
extract(AssignedBats$sample$stat[[1]], cbind(AssignDataBat$long[1], AssignDataBat$lat[1]))
```

```
##
## -1.796566
```

```
extract(AssignedBats$sample$pv[[1]], cbind(AssignDataBat$long[1], AssignDataBat$lat[1]))
```

```
##
## 0.928049
```

Answer: it is possible (we cannot reject the proposition).

Example 2: may the bat 13 (Nnoc_15) come from where it has been found?

```
extract(AssignedBats$sample$stat[[13]], cbind(AssignDataBat$long[13], AssignDataBat$lat[13]))
```

```
##
## -44.3473
```

```
extract(AssignedBats$sample$pv[[13]], cbind(AssignDataBat$long[13], AssignDataBat$lat[13]))
```

```
##
## 0.03669013
```

Answer: it is very unlikely (we can reject the proposition).

# Step 6. Perform the assignment

```
AssignedBats$group
```

```
## $pv
## class       : RasterLayer
## dimensions  : 190, 230, 43700  (nrow, ncol, ncell)
## resolution  : 0.04166667, 0.04166667  (x, y)
## extent      : 5.816527, 15.39986, 47.11653, 55.03319  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : layer
## values      : 0, 0.9955917  (min, max)
```

# Step 6. Perform the assignment

```
names(AssignedBats$sp_points)
```

```
## [1] "sources" "calibs"  "assigns"
```

```
AssignedBats$sp_points$sources
```

```
## class       : SpatialPointsDataFrame
## features    : 27
## extent      : 6.25, 14.95, 47.48, 54.67  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## variables   : 1
## names       : values
## min values  :  -9999
## max values  :  -9999
```