

IsoriX

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

The IsoriX core team

September 2024

Table of contents

1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

Table of contents

1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

Table of contents

1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

Why IsoriX?

Benefits:

- fully reproducible methods & results

Why IsoriX?

Benefits:

- fully reproducible methods & results
- as simple as necessary (but not beyond that)

Why IsoriX?

Benefits:

- fully reproducible methods & results
- as simple as necessary (but not beyond that)
- each step of the analysis can be thoroughly studied

Why IsoriX?

Benefits:

- fully reproducible methods & results
- as simple as necessary (but not beyond that)
- each step of the analysis can be thoroughly studied
- cutting edge statistical methods accounting for many sources of uncertainty

Why IsoriX?

Benefits:

- fully reproducible methods & results
- as simple as necessary (but not beyond that)
- each step of the analysis can be thoroughly studied
- cutting edge statistical methods accounting for many sources of uncertainty
- compatible with Geographic Information System within and without **R**

Why IsoriX?

Benefits:

- fully reproducible methods & results
- as simple as necessary (but not beyond that)
- each step of the analysis can be thoroughly studied
- cutting edge statistical methods accounting for many sources of uncertainty
- compatible with Geographic Information System within and without **R**
- works on Windows, MacOS, Linux, Unix; locally or remotely; with or without Internet

Why IsoriX?

Benefits:

- fully reproducible methods & results
- as simple as necessary (but not beyond that)
- each step of the analysis can be thoroughly studied
- cutting edge statistical methods accounting for many sources of uncertainty
- compatible with Geographic Information System within and without **R**
- works on Windows, MacOS, Linux, Unix; locally or remotely; with or without Internet
- free & open source (anyone can use and improve IsoriX)

Why IsoriX?

Benefits:

- fully reproducible methods & results
- as simple as necessary (but not beyond that)
- each step of the analysis can be thoroughly studied
- cutting edge statistical methods accounting for many sources of uncertainty
- compatible with Geographic Information System within and without **R**
- works on Windows, MacOS, Linux, Unix; locally or remotely; with or without Internet
- free & open source (anyone can use and improve IsoriX)

Limits:

- **R** knowledge required
- a little stable isotope knowledge also required
- not multivariate (1 isotope only, but isoscapes or assignment maps can be combined)
- not Bayesian (prior information not considered)

Table of contents

1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

What is IsoriX?

An **R** package:

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

library(IsoriX) ## to load
## Registered S3 methods overwritten by 'registry':
##   method           from
##   print.registry_field proxy
##   print.registry_entry proxy
##
## IsoriX version 0.9.3 is now loaded
##
## Type:
##   * '?IsoriX' for a very short description
##   * 'browseURL('https://bookdown.org/content/782/)' for a longer (online) documentation
##   * 'help(package = 'IsoriX', help_type = 'html')' for a list of the package objects and help files
##   * 'citation('IsoriX')' for how to cite IsoriX (i.e. the papers you should read)
##   * 'news(package = 'IsoriX')' for info on changed between versions of IsoriX
##
## Please join the mailing list 'https://groups.google.com/g/IsoriX'
## for help, news and discussions about IsoriX
##
## Attaching package: 'IsoriX'
## The following objects are masked from 'package:graphics':
##
##   plot, points
## The following object is masked from 'package:base':
##
##   plot
```

What is IsoriX?

An interface between **R** packages (without help files: only ca. 1400 lines of codes):

```
tools::package_dependencies("IsoriX") ## note: list dependencies, not suggested packages
## $IsoriX
## [1] "graphics"      "grDevices"      "grid"          "lattice"        "latticeExtra"   "methods"       "numDeriv"
## [8] "rasterVis"     "spaMM"         "stats"         "terra"         "tools"         "utils"         "viridisLite"
```

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

What is IsoriX?

A collection of functions:

```
setdiff(ls("package:IsoriX"), ## list all exported objects (ie. functions + data, but not methods)
       c("CountryBorders", "OceanMask", "ElevRasterDE", "PrecipBrickDE", ## remove lazy loaded data
         data(package = "IsoriX")$results[, "Item"]))) ## remove non loaded data

## [1] "calibfit"          "cellSize"           "create_aliens"      "crs"                "crs<-"             "downloadfile"
## [7] "ext"               "ext<-"              "extract"            "get_ranPars"        "getelev"            "getOption_IsoriX"
## [13] "getprecip"         "gpar"               "grid.text"          "isofind"            "isofit"              "isomultifit"
## [19] "isomultandscape"  "isoscape"           "layer"              "levelplot"          "lpolygon"           "options_IsoriX"
## [25] "plot"               "points"              "polys"              "precipitate"        "prepraster"         "prepsources"
## [31] "rast"               "RdBuTheme"          "readRDS"            "saveRDS"            "saveRDS_IsoriX"    "shift"
## [37] "values"             "vect"               "xyplot"
```

What is IsoriX?

A collection of functions:

```
setdiff(ls("package:IsoriX"), ## list all exported objects (ie. functions + data, but not methods)
       c("CountryBorders", "OceanMask", "ElevRasterDE", "PrecipBrickDE", ## remove lazy loaded data
         data(package = "IsoriX")$results[, "Item"]))) ## remove non loaded data

## [1] "calibfit"          "cellSize"           "create_aliens"      "crs"                "crs<-"            "downloadfile"
## [7] "ext"               "ext<-"             "extract"            "get_ranPars"        "getelev"           "getOption_IsoriX"
## [13] "getprecip"         "gpar"              "grid.text"         "isofind"            "isofit"             "isomultifit"
## [19] "isomultandscape"  "isoscape"          "layer"              "levelplot"          "lpolygon"          "options_IsoriX"
## [25] "plot"              "points"            "polys"              "precipitate"        "prepraster"        "prepsources"
## [31] "rast"              "RdBuTheme"         "readRDS"            "saveRDS"            "saveRDS_IsoriX"   "shift"
## [37] "values"            "vect"              "xyplot"
```

The main ones being:

- ① prepsources()
- ② isofit()
- ③ getelev()
- ④ prepraster()
- ⑤ isoscape()
- ⑥ calibfit()
- ⑦ 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 and to make plots nicer:

```
data(package = "IsoriX")$results[, c("Item", "Title")]
##      Item          Title
## [1,] "AssignDataAlien"    "Simulated assignment dataset"
## [2,] "AssignDataBat"      "Assignment datasets for bat species"
## [3,] "AssignDataBat2"     "Assignment datasets for bat species"
## [4,] "AssignDataBat2Rev"   "Assignment datasets for bat species"
## [5,] "AssignDataBatRev"    "Assignment datasets for bat species"
## [6,] "CalibDataAlien"     "Simulated calibration dataset"
## [7,] "CalibDataBat"       "Calibration datasets for bat species"
## [8,] "CalibDataBat2"      "Calibration datasets for bat species"
## [9,] "CalibDataBat2Rev"    "Calibration datasets for bat species"
## [10,] "CalibDataBatRev"    "Calibration datasets for bat species"
## [11,] "GNIPDataALLagg"    "Hydrogen delta values in precipitation water (aggregated per location)"
## [12,] "GNIPDataDE"        "Hydrogen delta values in precipitation water, Germany"
## [13,] "GNIPDataEUagg"     "Hydrogen delta values in precipitation water (aggregated per location)"
## [14,] "isopalette1"        "Colour palettes for plotting"
## [15,] "isopalette2"        "Colour palettes for plotting"
```

+ CountryBorders, OceanMask, ElevRasterDE, PrecipBrickDE which are spatial objects and thus pre-loaded.

What is IsoriX?

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

```
data(package = "IsoriX")$results[, c("Item", "Title")]
##      Item          Title
## [1,] "AssignDataAlien"    "Simulated assignment dataset"
## [2,] "AssignDataBat"      "Assignment datasets for bat species"
## [3,] "AssignDataBat2"     "Assignment datasets for bat species"
## [4,] "AssignDataBat2Rev"   "Assignment datasets for bat species"
## [5,] "AssignDataBatRev"    "Assignment datasets for bat species"
## [6,] "CalibDataAlien"     "Simulated calibration dataset"
## [7,] "CalibDataBat"       "Calibration datasets for bat species"
## [8,] "CalibDataBat2"      "Calibration datasets for bat species"
## [9,] "CalibDataBat2Rev"    "Calibration datasets for bat species"
## [10,] "CalibDataBatRev"    "Calibration datasets for bat species"
## [11,] "GNIPDataALLagg"    "Hydrogen delta values in precipitation water (aggregated per location)"
## [12,] "GNIPDataDE"        "Hydrogen delta values in precipitation water, Germany"
## [13,] "GNIPDataEUagg"     "Hydrogen delta values in precipitation water (aggregated per location)"
## [14,] "isopalette1"        "Colour palettes for plotting"
## [15,] "isopalette2"        "Colour palettes for plotting"
```

+ CountryBorders, OceanMask, ElevRasterDE, PrecipBrickDE which are spatial objects and thus pre-loaded.

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](https://github.com/courtiol/IsoriX)

Table of contents

1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

Where can you learn about IsoriX?

- There is a bookdown

<https://bookdown.org/content/782/>

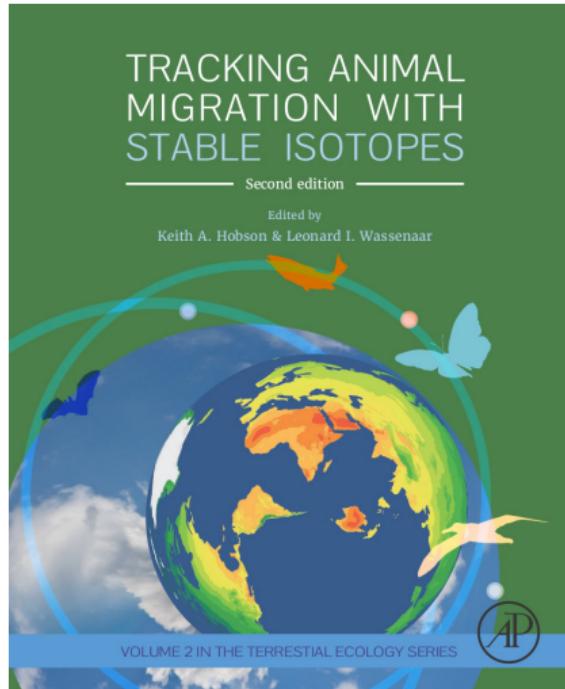
- There is the package documentation

```
help(package = "IsoriX")
```

- There is a mailing list

<https://groups.google.com/g/IsoriX>

- There is a book chapter
(in the 2nd edition of the book)



Is IsoriX better than alternatives?

- All methods differ in some details
(e.g. IsoriX does smoothing, not interpolation)
- All methods differ in their set of assumptions
(some uncertainties are considered, some are discarded)

Is IsoriX better than alternatives?

- All methods differ in some details
(e.g. IsoriX does smoothing, not interpolation)
- All methods differ in their set of assumptions
(some uncertainties are considered, some are discarded)

Conclusion:

- A single method is unlikely to be best across all situations
- Run several methods on your data

Is IsoriX better than alternatives?

- All methods differ in some details
(e.g. IsoriX does smoothing, not interpolation)
- All methods differ in their set of assumptions
(some uncertainties are considered, some are discarded)

Conclusion:

- A single method is unlikely to be best across all situations
- Run several methods on your data

What to do?

- if all methods give you the same answer, you have a robust result → great!
- if they don't, try to figure out which assumption(s) introduce the difference and figure out in which respect your data contradict the identified assumption(s)
→ you will better understand the biology/geology/... in your system → great!

Table of contents

1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

The standard workflow for an assignment involves 6 main steps and thus 6 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()`

A complete assignment can be done in 8 lines of code  `▶ proof`, but it is important to understand well and check the details of each step to get the best out of your data. So we will add a few extra steps.

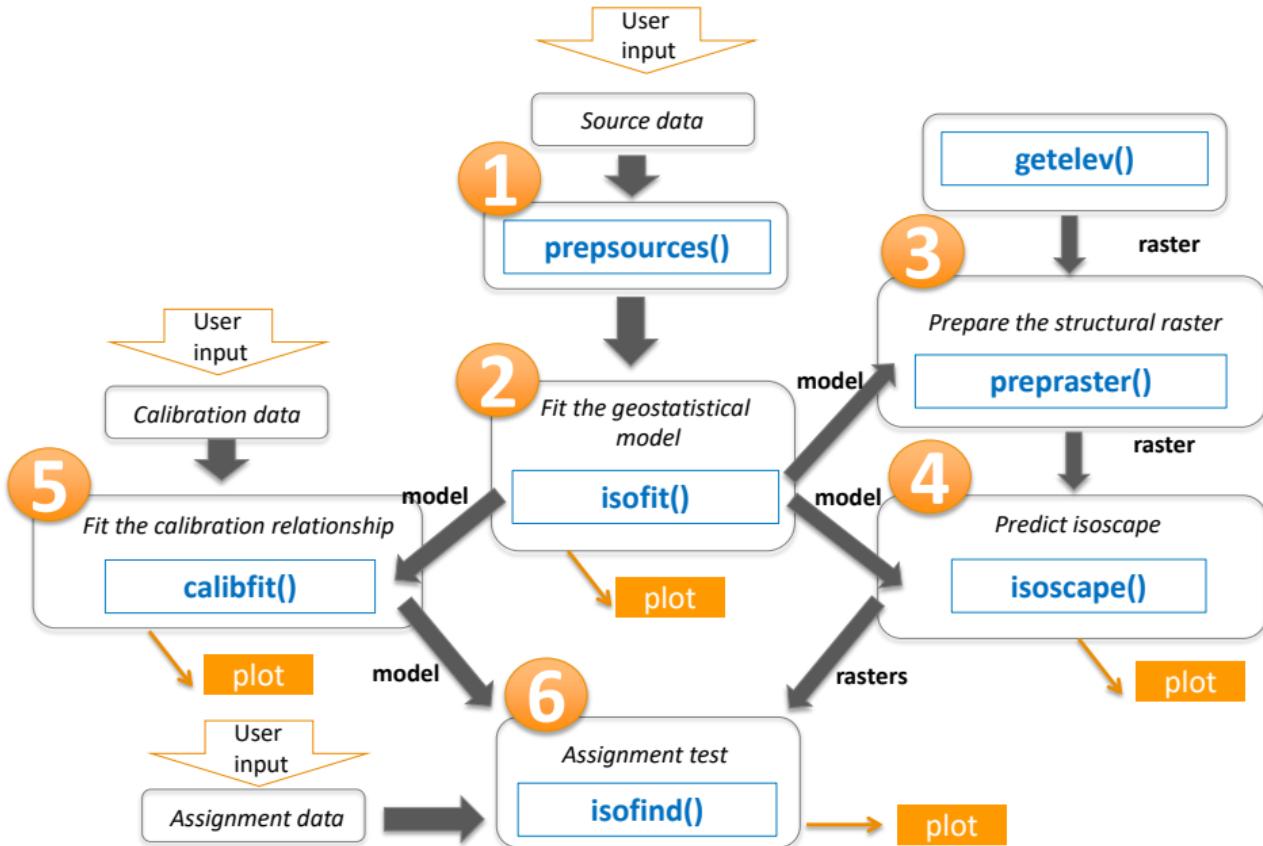


Table of contents

1 Generalities about IsoriX

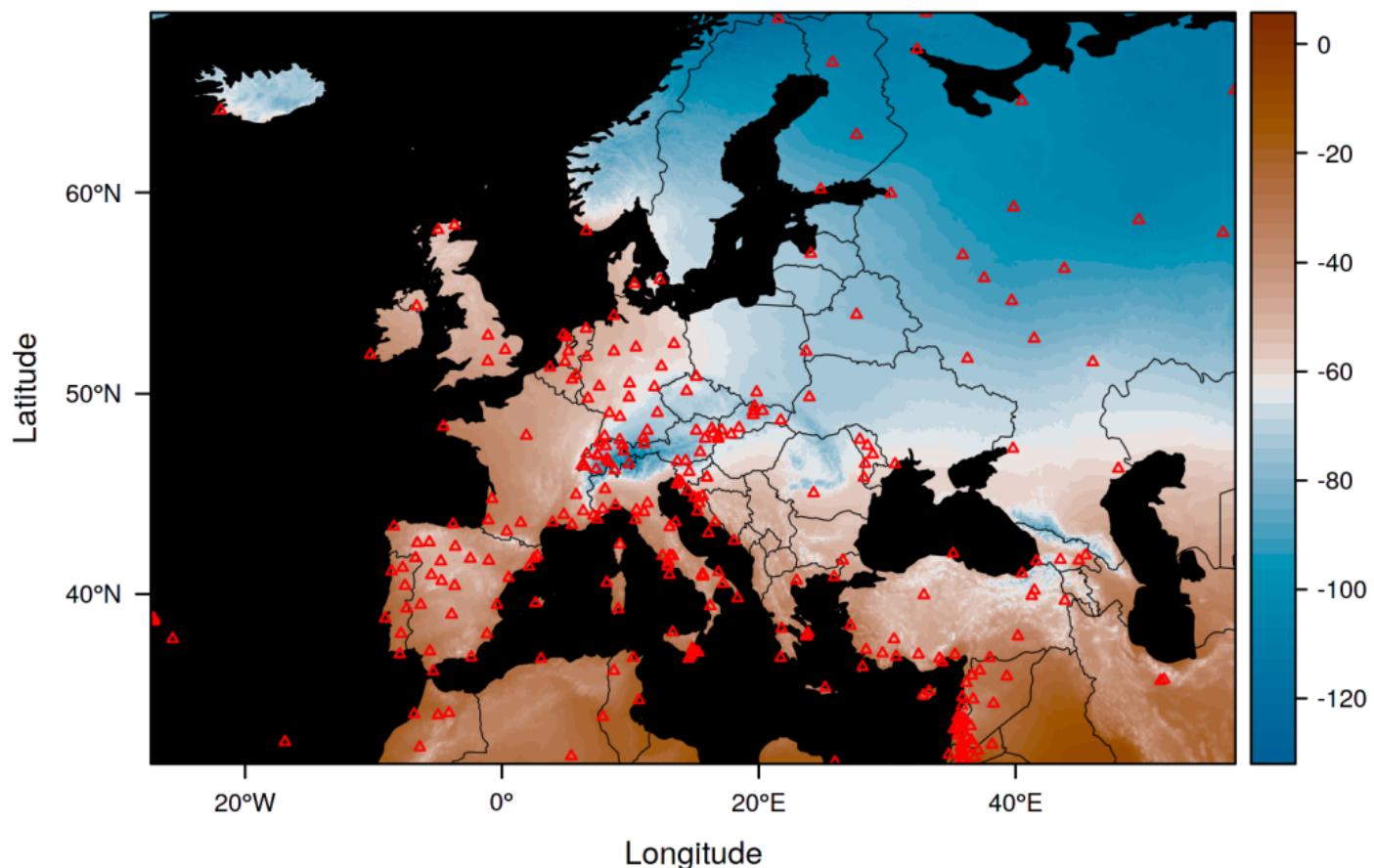
- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

mean δD_p



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
```

Step 1. Prepare the source data

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

```
prepsources(  
  data,  
  month = 1:12, year,  
  long_min = -180, long_max = 180, lat_min = -90, lat_max = 90,  
  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 filter and aggregate your source data:

```
prepsources(  
  data,  
  month = 1:12, year,  
  long_min = -180, long_max = 180, lat_min = -90, lat_max = 90,  
  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

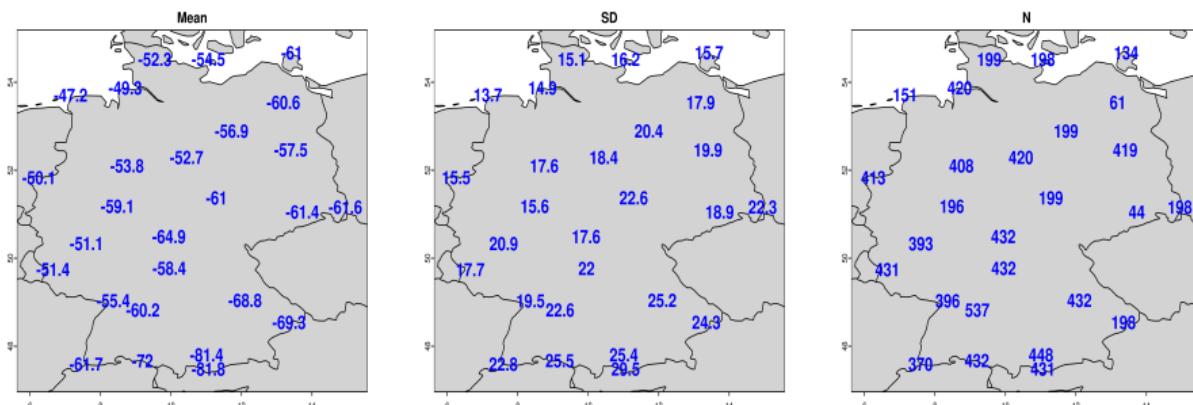
1c. Check your data (it is always a good idea to plot your data)

```
range_expanded <- function(x, stretch = 0.05) range(x) + diff(range(x))*c(-1, 1)*stretch

with(GNIPDataDEagg, plot(CountryBorders, xlim = range_expanded(long), ylim = range_expanded(lat), col = "lightgrey", main = "Mean"))
text(lat ~ long, data = GNIPDataDEagg, labels = round(mean_source_value, digits = 1), col = "blue", cex = 1.5, font = 2)

with(GNIPDataDEagg, plot(CountryBorders, xlim = range_expanded(long), ylim = range_expanded(lat), col = "lightgrey", main = "SD"))
text(lat ~ long, data = GNIPDataDEagg, labels = round(sqrt(var_source_value), digits = 1), col = "blue", cex = 1.5, font = 2)

with(GNIPDataDEagg, plot(CountryBorders, xlim = range_expanded(long), ylim = range_expanded(lat), col = "lightgrey", main = "N"))
text(lat ~ long, data = GNIPDataDEagg, labels = 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? See details in bookdown, section 6.3.3.

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? See details in bookdown, section 6.3.3.

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? See details in bookdown, section 6.3.3.

Issue:

- More data = more signal or more noise?

Trade-off:

- 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? See details in bookdown, section 6.3.3.

Issue:

- More data = more signal or more noise?

Trade-off:

- 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 accounted 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

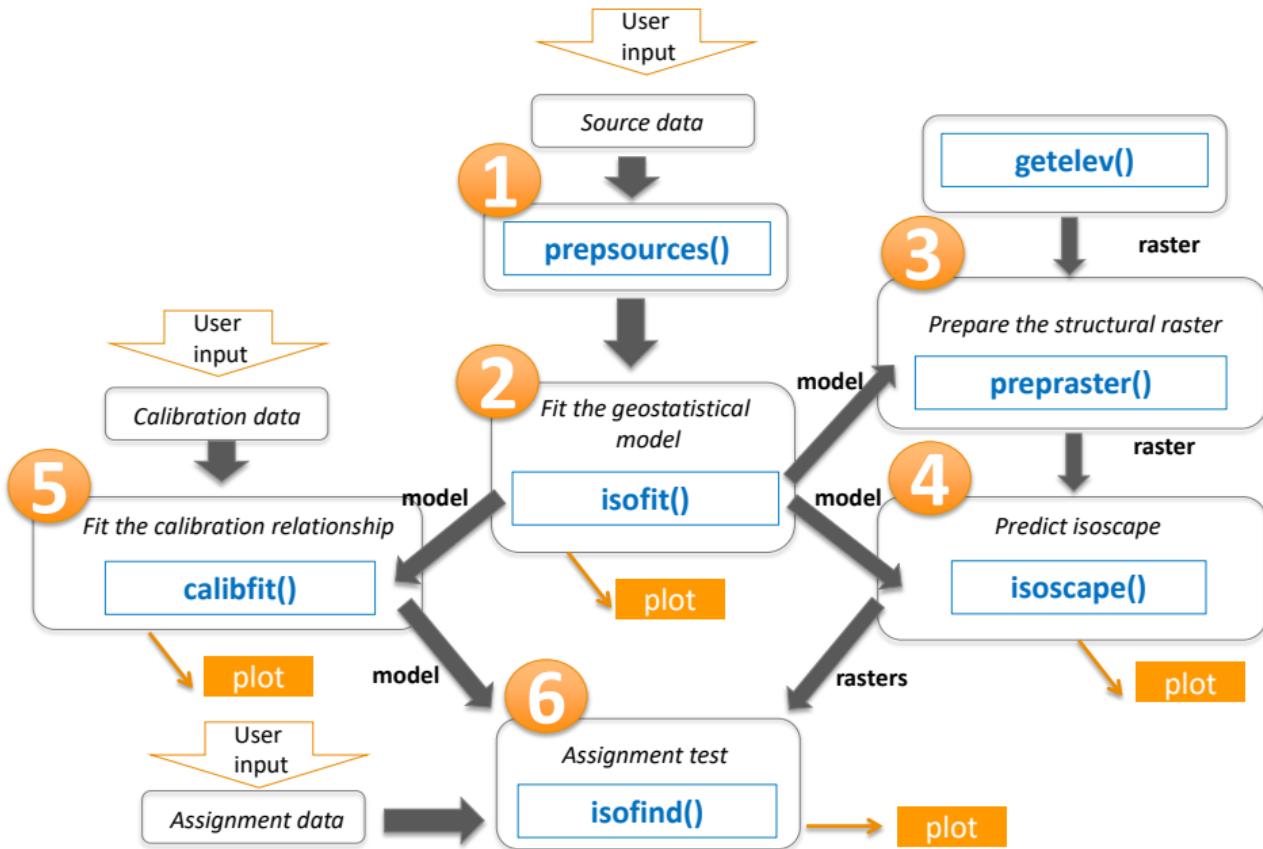
1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

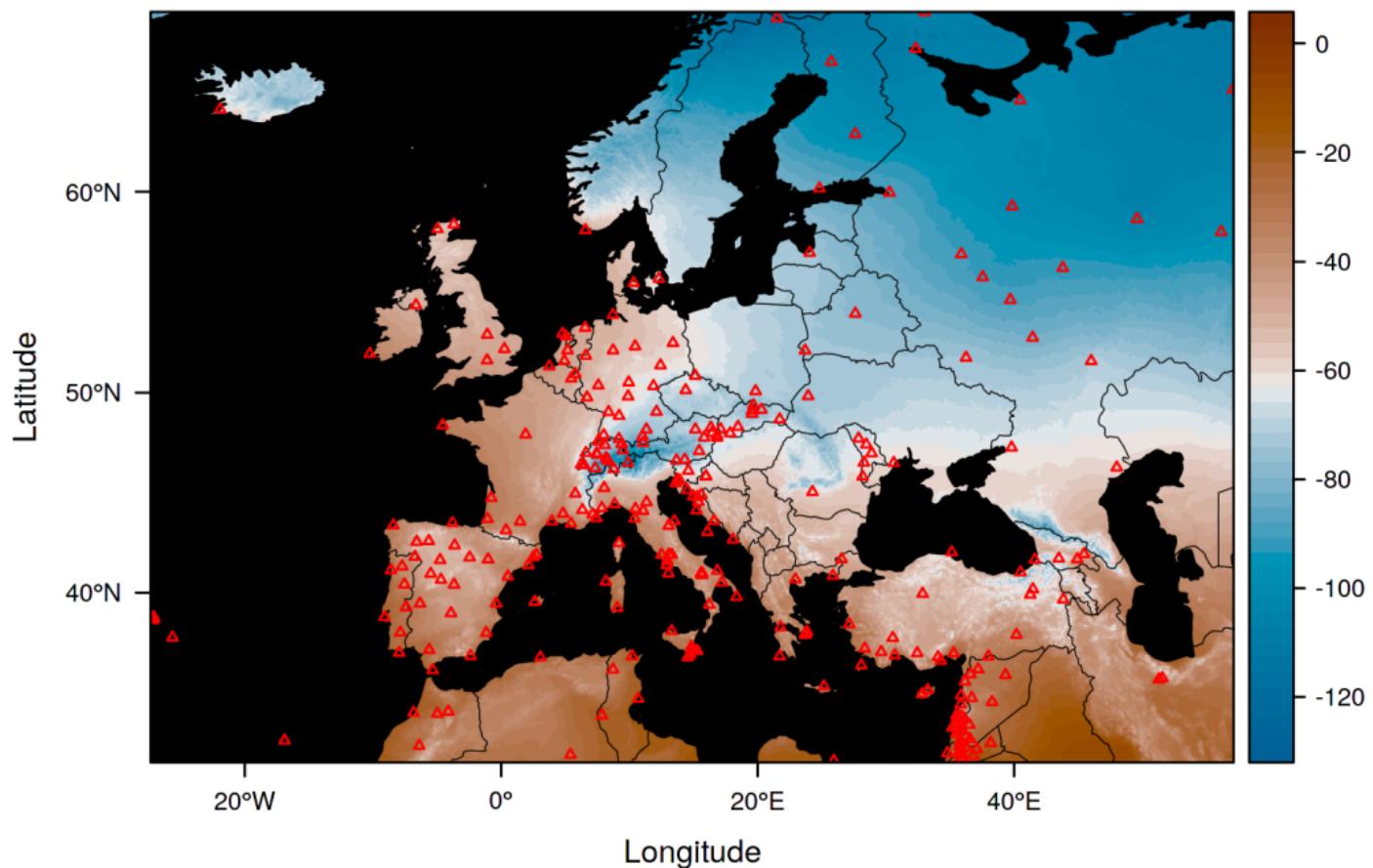
2 Standard workflow

- Step 1. Prepare the source data
- **Step 2. Fit the isoscape**
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX



mean δD_p



Step 2. Fit the isoscape

The statistical model(s) behind the isoscape:

- Mean model (LMM):

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

Step 2. Fit the isoscape

The statistical model(s) behind the isoscape:

- Mean model (LMM):

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

- where

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

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

and

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

Step 2. Fit the isoscape

The statistical model(s) behind the isoscape:

- Mean model (LMM):

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

- where

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

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

and

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

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

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

Step 2. Fit the isoscape

The statistical model(s) behind the isoscape:

- Mean model (LMM):

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

- where

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

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

and

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

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

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

Note:

d_{ab} – the distance between 2 locations – is (by default) measured accounting for the curvature of the Earth.

Step 2. Fit the isoscape

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

```
isofit(  
  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):

```
isofit(  
  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))
```

Step 2. Fit the isoscape

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

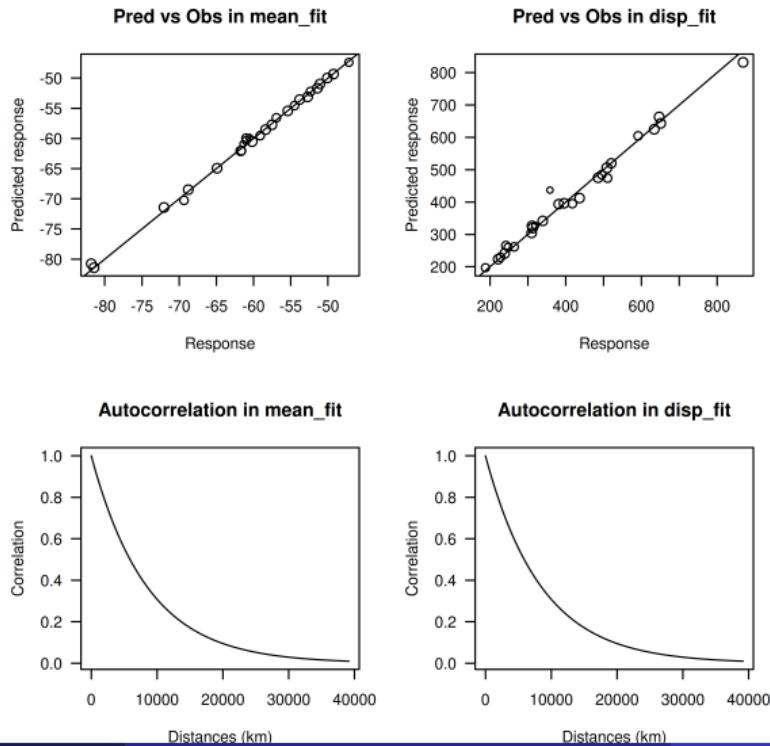
```
names(GermanFit)
## [1] "mean_fit" "disp_fit" "info_fit"

GermanFit$mean_fit
## formula: mean_source_value ~ 1 + elev + lat_abs + (1 | source_ID) + Matern(1 |
##   long + lat)
## REML: Estimation of lambda and corrPars by REML.
##   Estimation of fixed effects by ML.
## family: gaussian( link = identity )
##   ----- Fixed effects (beta) -----
##             Estimate Cond. SE t-value
## (Intercept) -145.78874 65.386512 -2.230
## elev         -0.01163  0.002449 -4.750
## lat_abs      1.68318  1.210012  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 : 1e-08
##   long + lat : 482
##   --- Coefficients for log(lambda):
##   Group      Term Estimate Cond.SE
##   source_ID (Intercept) -18.42    7999
##   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 [through "phi" ~ 0 + offset(pred_disp) ] to 260.015 473.86 304.447 396.193 341.629 ...
##   ----- Likelihood values -----
##             logLik
## logL      (p_v(h)): -71.72787
```

Step 2. Fit the isoscape

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

```
plot(GermanFit)
```



Step 2. Fit the isoscape

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:  
##      157.455740     105.958851     151.485965  
##                                         effective df:  
##                                         4.858699
```

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

```
##      marginal AIC:    conditional AIC:    dispersion AIC:  
##      157.452979     106.076852     155.619901  
##                                         effective df:  
##                                         4.733681
```

Step 2. Fit the isoscape

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:  
##      157.455740     105.958851     151.485965  
##                                         effective df:  
##                                         4.858699
```

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

```
##      marginal AIC:    conditional AIC:    dispersion AIC:  
##      157.452979     106.076852     155.619901  
##                                         effective df:  
##                                         4.733681
```

Note:

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

Step 2. Fit the isoscape

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:  
##      157.455740     105.958851     151.485965  
##                                         effective df:  
##                                         4.858699
```

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

```
##      marginal AIC:    conditional AIC:    dispersion AIC:  
##      157.452979     106.076852     155.619901  
##                                         effective df:  
##                                         4.733681
```

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). (If residual dispersion models are the same then the correction won't change the ranking of the models.)

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 by alternative approaches.
- Some of these uncertainties are huge.

» Back to the future

Table of contents

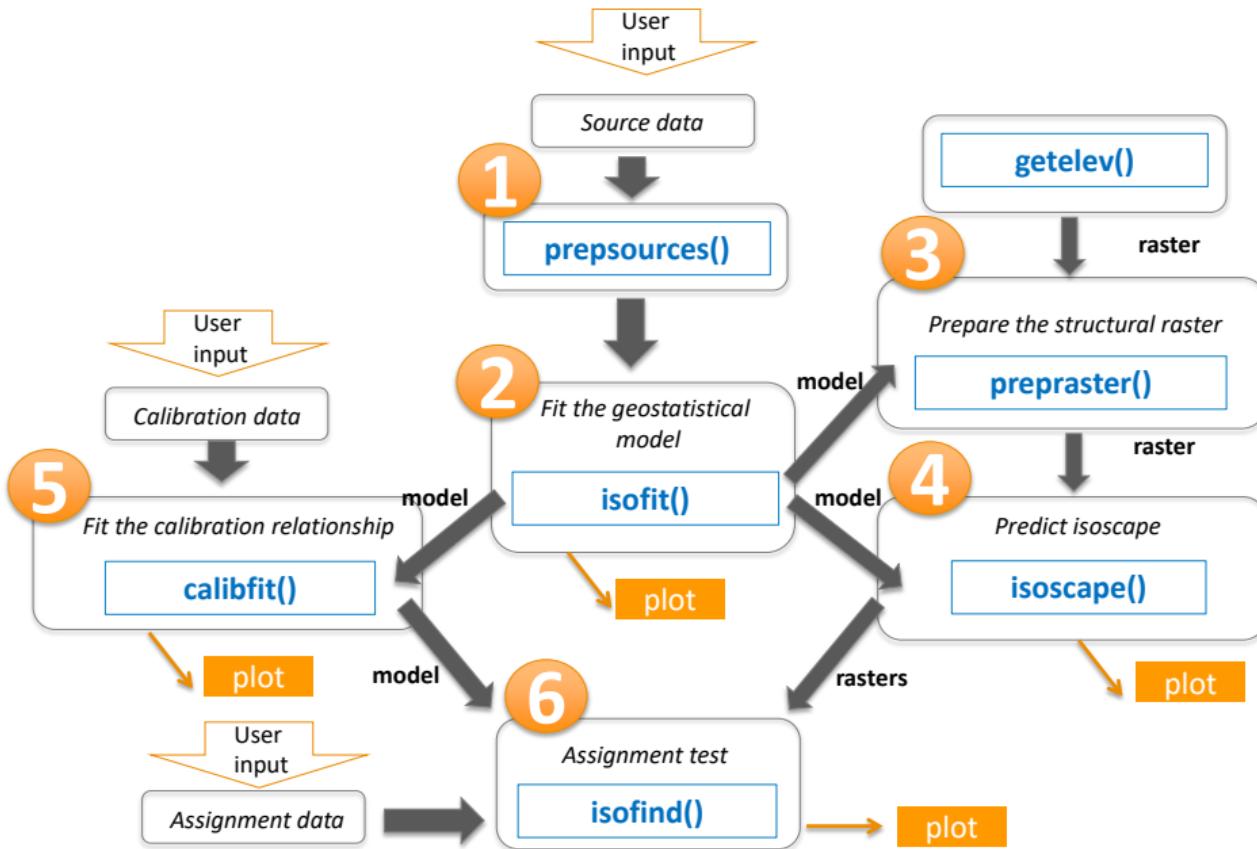
1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

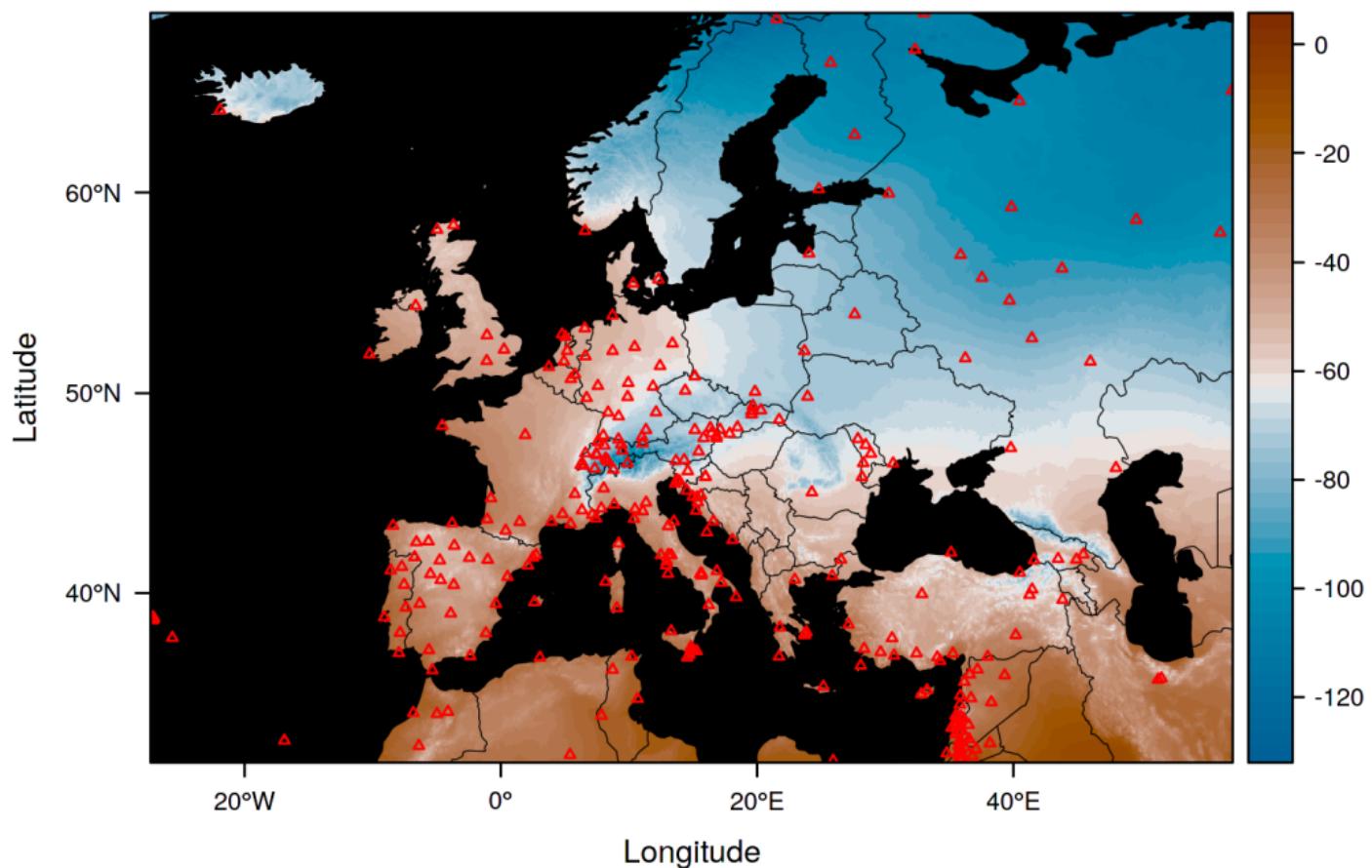
2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster**
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX



mean δD_p



Step 3. Prepare the structural raster

3a. Use `getelev()` to prepare the structural raster providing the locations and covariates for predictions (elevation/bathymetry, latitude, longitude):

```
getelev(  
  file = "~/elevation_world_z5.tif",  
  z = 5, # the higher z is, the higher the resolution (1 = lowest, 14 = highest; default = 5)  
  long_min = -180, long_max = 180, lat_min = -90, lat_max = 90,  
  margin_pct = 5,  
  override_size_check = FALSE,  
  overwrite = FALSE,  
  Ncpu =getOption_IsoriX("Ncpu"),  
  verbose = interactive(),  
  ...  
)
```

Step 3. Prepare the structural raster

3a. Use `getelev()` to prepare the structural raster providing the locations and covariates for predictions (elevation/bathymetry, latitude, longitude):

```
getelev(  
  file = "~/elevation_world_z5.tif",  
  z = 5, # the higher z is, the higher the resolution (1 = lowest, 14 = highest; default = 5)  
  long_min = -180, long_max = 180, lat_min = -90, lat_max = 90,  
  margin_pct = 5,  
  override_size_check = FALSE,  
  overwrite = FALSE,  
  Ncpu =getOption_IsoriX("Ncpu"),  
  verbose = interactive(),  
  ...  
)
```

Example:

```
getelev(file = "elevation_DE_z5.tif",  
        long_min = 5.86, long_max = 15.02, lat_min = 47.23, lat_max = 54.90)
```

Note: you can also just use the function without arguments, but that will download the whole world!

```
getelev()
```

Step 3. Prepare the structural raster

3b. Load the structural raster:

```
ElevDE <- rast("elevation_DE_z5.tif") ## turn the tif into raster (a SpatRaster object)
```

```
ElevDE
## class      : SpatRaster
## dimensions : 453, 542, 1 (nrow, ncol, nlyr)
## resolution : 0.01859565, 0.01859565 (x, y)
## extent     : 5.392739, 15.47158, 46.85066, 55.27449 (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source     : elevation_DE_z5.tif
## name       : elevation_DE_z5
## min value  :          -92
## max value  :         3253
```

The reasons why `getelev()` does not load the raster automatically is that we want to discourage multiple downloads of the same file and we also want people to be able to use their own structural rasters (i.e. by-passing `getelev()`).

Step 3. Prepare the structural raster

3c. Use `preraster()` to remove values below (or above) the surface, aggregate and resize the structural raster:

```
preraster(  
  raster,  
  isofit = NULL,  
  margin_pct = 5,  
  aggregation_factor = 0L,  
  aggregation_fn = mean,  
  manual_crop = NULL,  
  values_to_zero = c(-Inf, 0),  
  verbose = interactive()  
)
```

Step 3. Prepare the structural raster

3c. Use `preraster()` to remove values below (or above) the surface, aggregate and resize the structural raster:

```
preraster(  
  raster,  
  isofit = NULL,  
  margin_pct = 5,  
  aggregation_factor = 0L,  
  aggregation_fn = mean,  
  manual_crop = NULL,  
  values_to_zero = c(-Inf, 0),  
  verbose = interactive()  
)
```

Example:

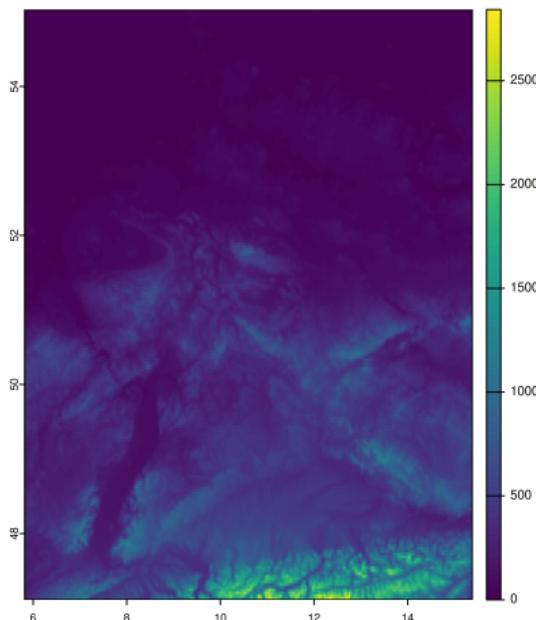
```
ElevRasterDE <- preraster(ElevDE, isofit = GermanFit, aggregation_factor = 2)
```

```
## class      : SpatRaster  
## dimensions : 213, 257, 1  (nrow, ncol, nlyr)  
## resolution : 0.03719131, 0.03719131  (x, y)  
## extent     : 5.820439, 15.3786, 47.111, 55.03275  (xmin, xmax, ymin, ymax)  
## coord. ref. : lon/lat WGS 84 (EPSG:4326)  
## source(s)   : memory  
## name        : elevation_DE_z5  
## min value   :          0.0  
## max value   :        2843.5
```

Step 3. Prepare the structural raster

3d. Check the obtained the structural raster:

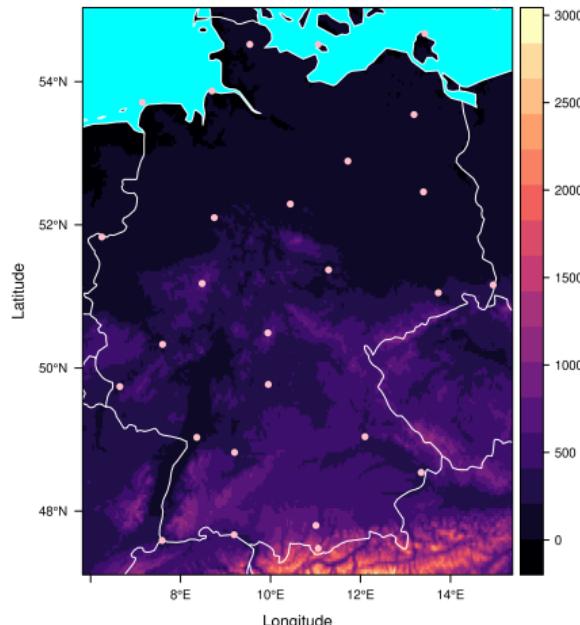
```
plot(ElevRasterDE)
```



Step 3. Prepare the structural raster

3c. Check the obtained the structural raster:

```
levelplot(ElevRasterDE, margin = FALSE) +  
  layer(lpolygon(CountryBorders, border = "white")) +  
  layer(lpolygon(OceanMask, border = "white", col = "cyan")) +  
  xyplot(lat ~ long, data = GNIPDataDEagg, 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 aggregation 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 aggregation 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 aggregation 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, increase the resolution as high as it remains reasonable (below the pixel level makes little sense)

Step 3. Prepare the structural raster

Uncertainty check list:

- ✗ uncertainty behind the raw raster
(structural rasters are the results of some smoothing or interpolation 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
(structural rasters are the results of some smoothing or interpolation 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:
(`getelev()` uses this: <https://github.com/tilezen/joerd/blob/master/docs/data-sources.md>)
 - Which uncertainty? (not just the average one)
 - Which smoothing or aggregation function?
- Increasing the resolution reduces the variance not considered

Table of contents

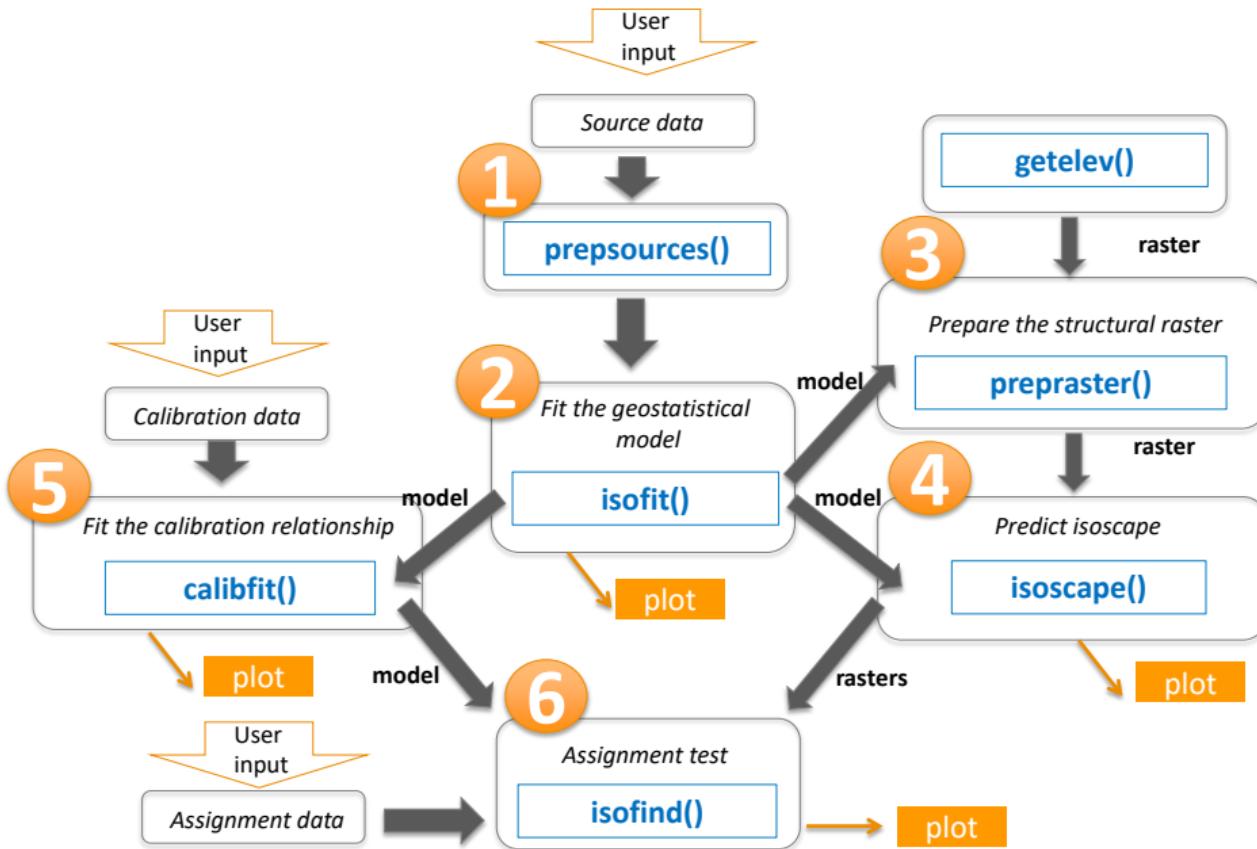
1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

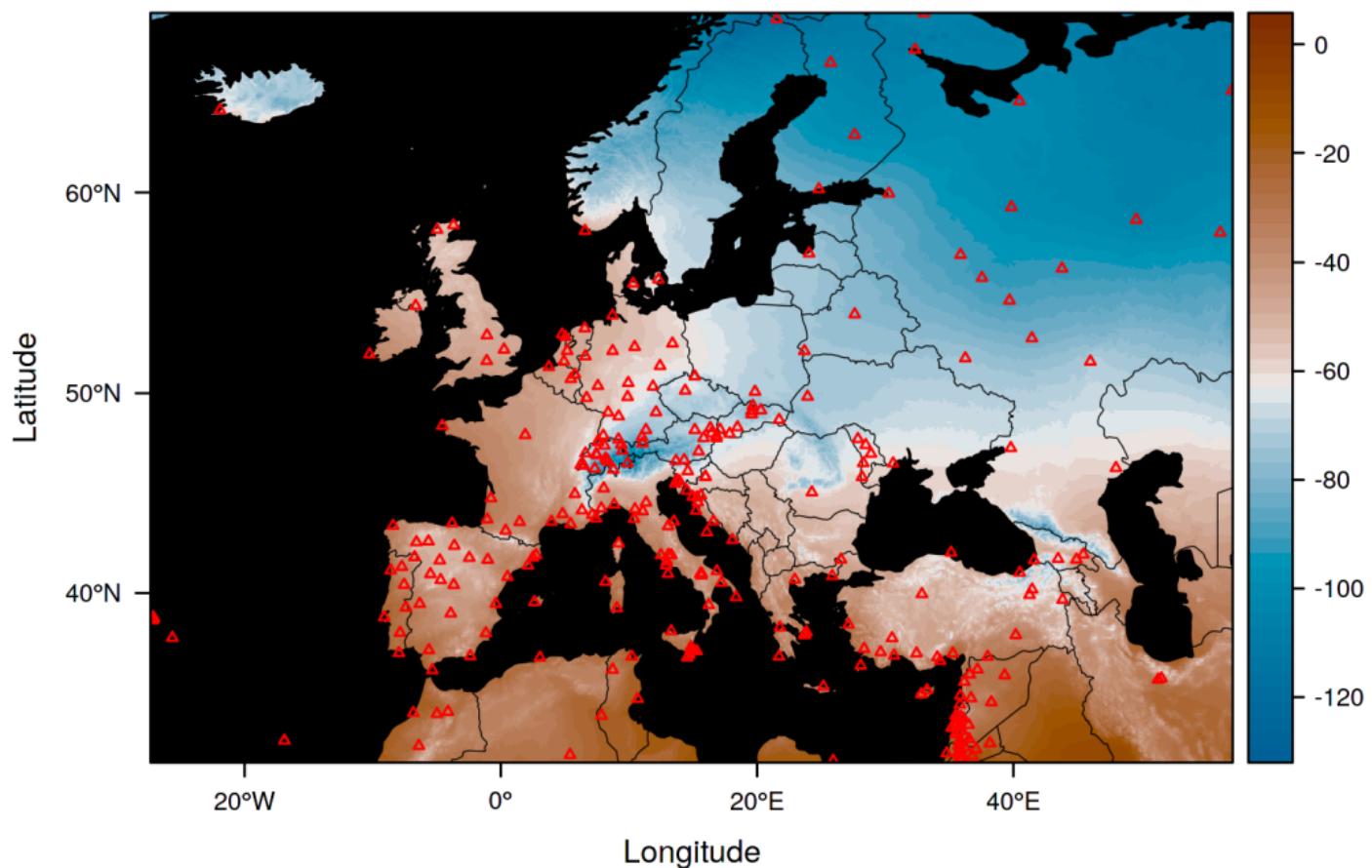
2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape**
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX



mean δD_p



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)
```

```
## The following warnings were produced by the predictions of mean isotopic values:  
## phi dispVar component not yet available for phi model != ~1.  
## The following warnings were produced by the predictions of residual dispersion:  
## Prior weights are not taken in account in residVar computation.
```

```
names(GermanScape)  
## [1] "isoscapes" "sp_points"
```

Step 4. Predict the isoscape

There are 8 rasters behind the isoscape:

```
GermanScape$isoscapes
## class       : SpatRaster
## dimensions : 213, 257, 8  (nrow, ncol, nlyr)
## resolution : 0.03719131, 0.03719131  (x, y)
## extent      : 5.820439, 15.3786, 47.111, 55.03275  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## names       : mean, mean_predVar, mean_idVar, mean_respVar,    disp, disp_predVar, ...
## min values  : -104.75372,    0.5903264,    195.3499,    203.3150, 195.3499,    0.01409377, ...
## max values  :  -46.72956,    37.4781165,    774.7529,    797.1494, 774.7529,    0.08228693, ...
```

Step 4. Predict the isoscape

There are 8 rasters behind the isoscape:

```
GermanScape$isoscapes
## class      : SpatRaster
## dimensions : 213, 257, 8  (nrow, ncol, nlyr)
## resolution : 0.03719131, 0.03719131  (x, y)
## extent     : 5.820439, 15.3786, 47.111, 55.03275  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## names       : mean, mean_predVar, mean_idVar, mean_respVar,    disp, disp_predVar, ...
## min values  : -104.75372,    0.5903264,    195.3499,    203.3150, 195.3499,    0.01409377, ...
## max values  : -46.72956,    37.4781165,    774.7529,    797.1494, 774.7529,    0.08228693, ...
```

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

Step 4. Predict the isoscape

There are 8 rasters behind the isoscape:

```
GermanScape$isoscapes
## class       : SpatRaster
## dimensions : 213, 257, 8  (nrow, ncol, nlyr)
## resolution : 0.03719131, 0.03719131  (x, y)
## extent     : 5.820439, 15.3786, 47.111, 55.03275  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## names       : mean, mean_predVar, mean_idVar, mean_respVar,      disp, disp_predVar, ...
## min values  : -104.75372,    0.5903264,    195.3499,    203.3150, 195.3499,    0.01409377, ...
## max values  : -46.72956,    37.4781165,    774.7529,    797.1494, 774.7529,    0.08228693, ...
```

- **mean** = prediction of the mean isotopic value
- **mean_predVar** = quantify the uncertainty associated to **mean**
- **mean_residVar** = prediction of the variance between observations
- **mean_respVar** = **mean_predVar** + **mean_residVar**
- **disp** = **mean_residVar**
- **disp_predVar** = quantify the uncertainty associated to **disp**
- **disp_residVar** = 2
- **disp_respVar** = $\exp(\text{disp_predVar} + \text{disp_residVar})$

Step 4. Predict the isoscape

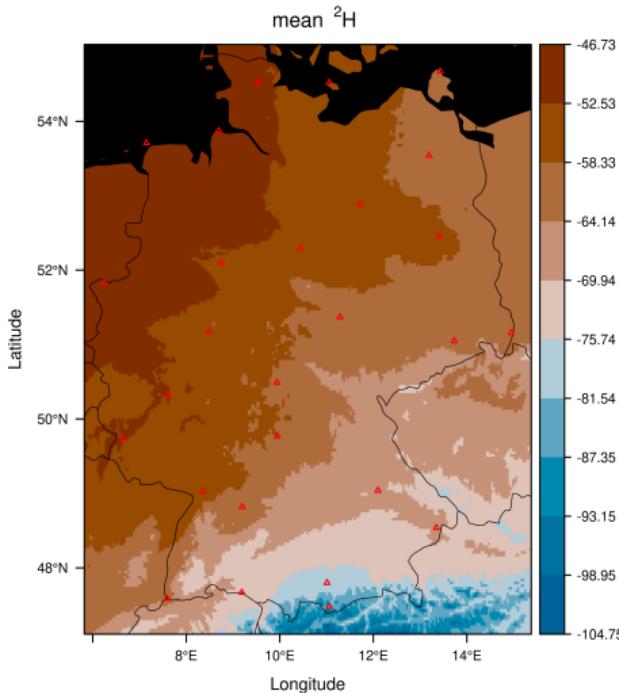
4b. Plot the rasters with `plot()`:

```
plot(  
  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),  
  ...  
)
```

Step 4. Predict the isoscape

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

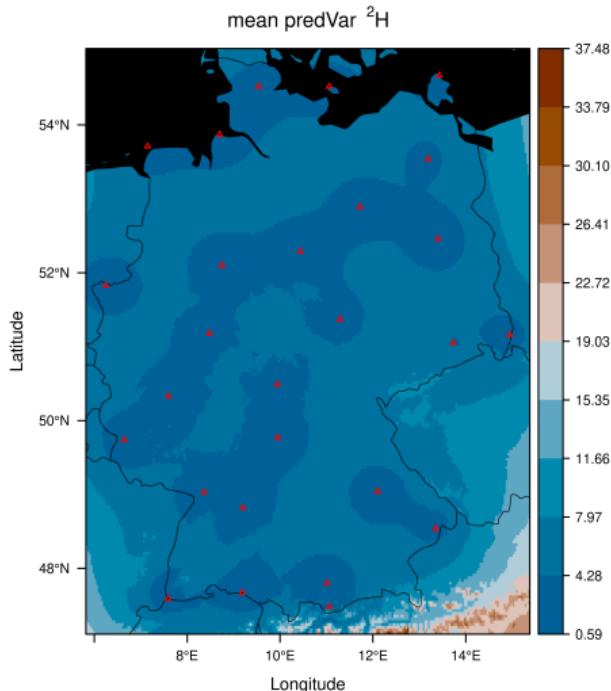
```
plot(GermanScape)
```



Step 4. Predict the isoscape

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

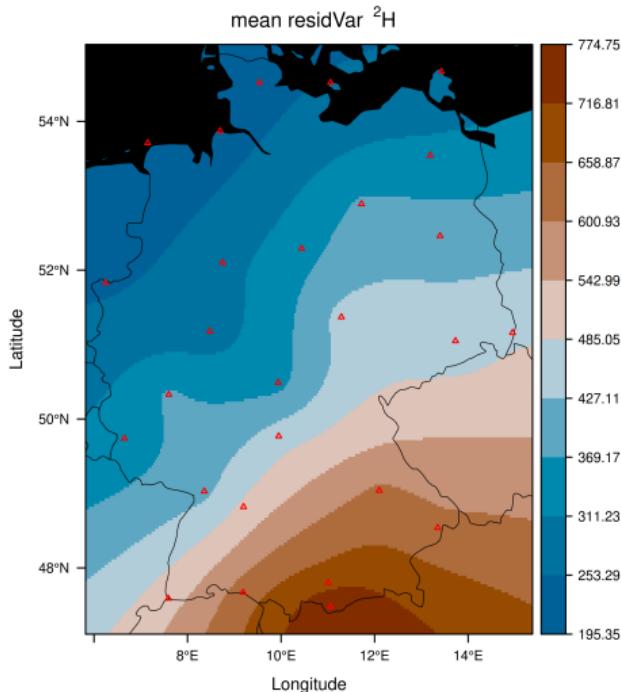
```
plot(GermanScape, which = "mean_predVar")
```



Step 4. Predict the isoscape

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

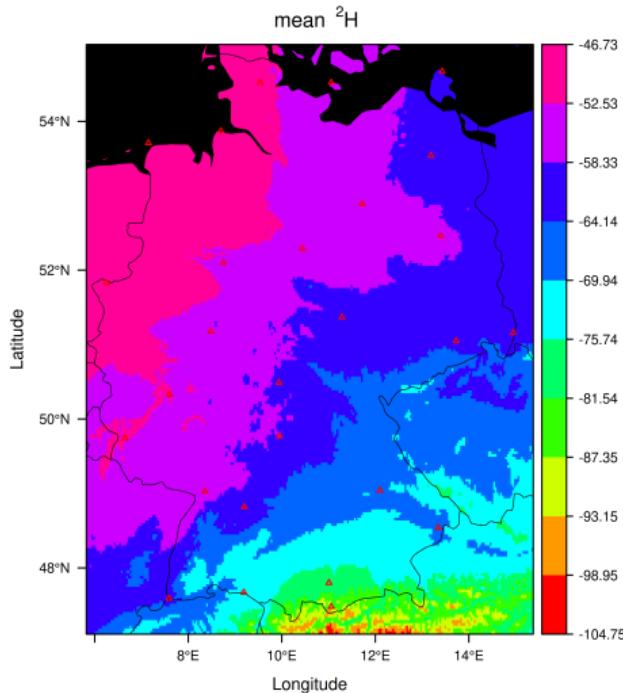
```
plot(GermanScape, which = "mean_residVar")
```



Step 4. Predict the isoscape

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

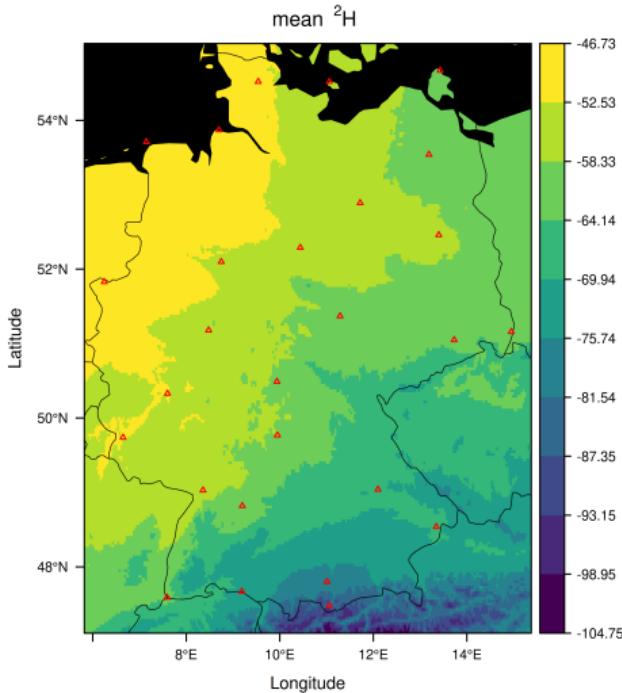
```
plot(GermanScape, palette = list(fn = rainbow))
```



Step 4. Predict the isoscape

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

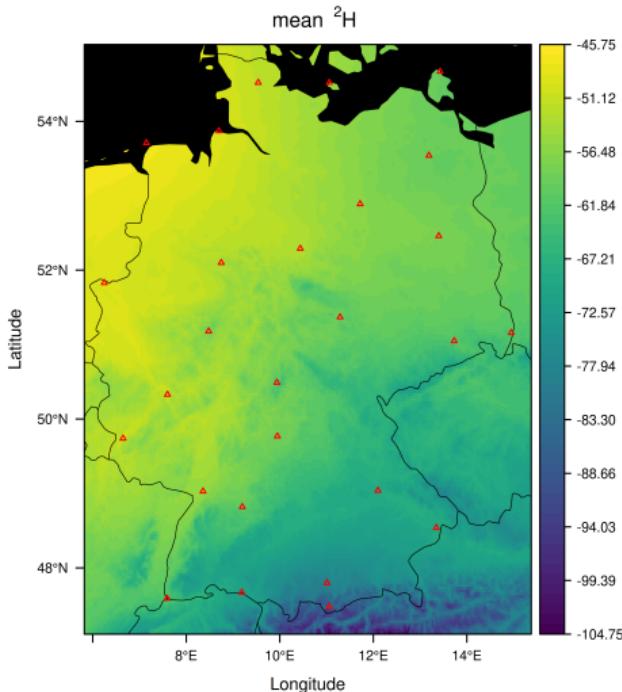
```
plot(GermanScape, palette = list(fn = NULL))
```



Step 4. Predict the isoscape

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

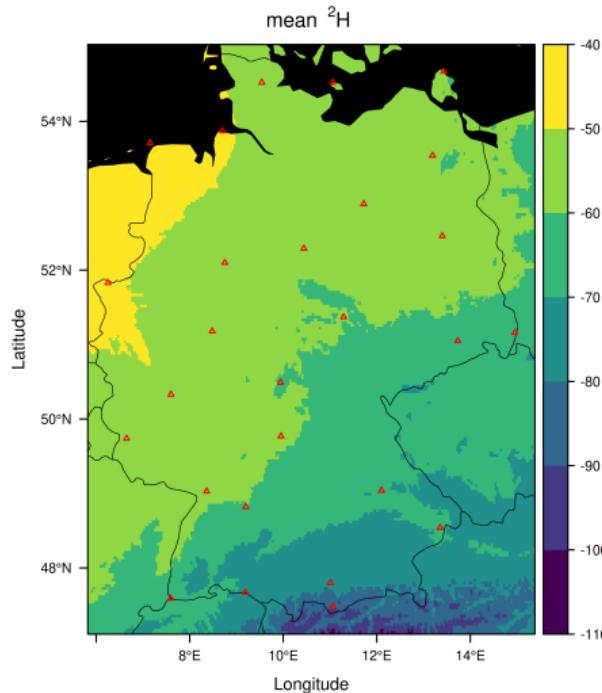
```
plot(GermanScape, palette = list(fn = NULL, step = 1))
```



Step 4. Predict the isoscape

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

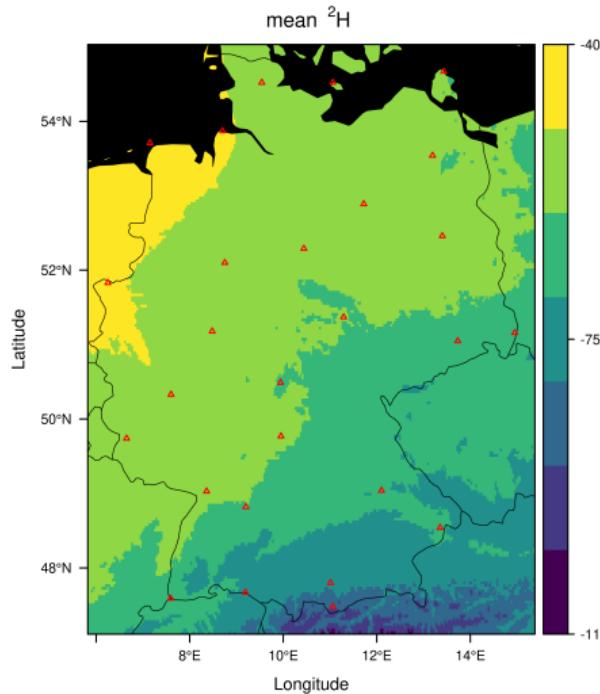
```
plot(GermanScape, palette = list(fn = NULL, step = 10, range = c(-110, -40)))
```



Step 4. Predict the isoscape

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

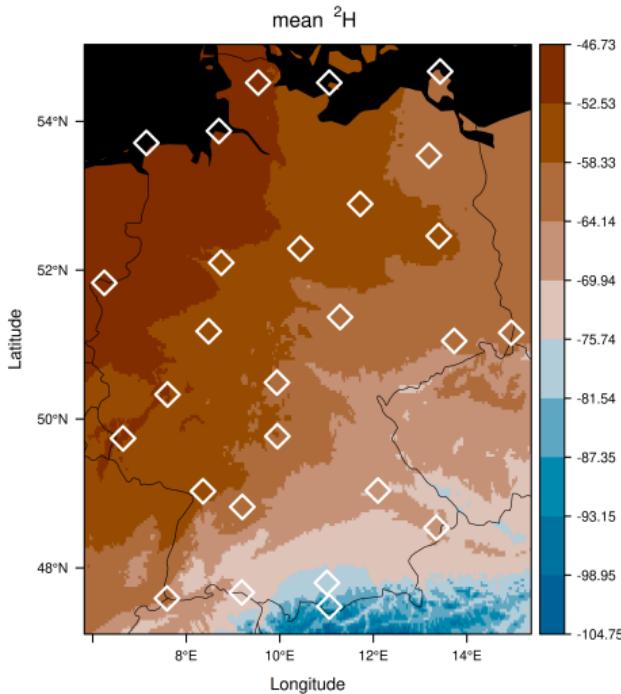
```
plot(GermanScape, palette = list(fn = NULL, step = 10, range = c(-110, -40), n_labels = 3))
```



Step 4. Predict the isoscape

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

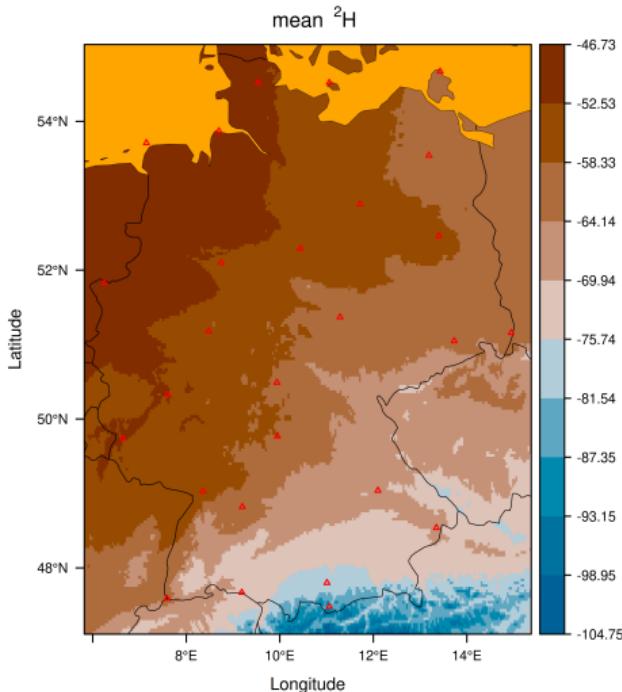
```
plot(GermanScape, sources = list(draw = TRUE, cex = 2, pch = 5, lwd = 2, col = "white"))
```



Step 4. Predict the isoscape

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

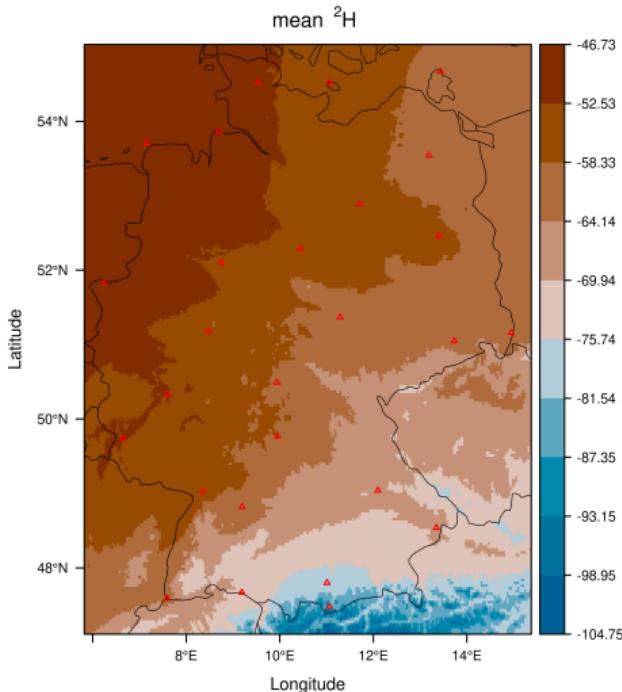
```
plot(GermanScape, mask = list(fill = "orange"))
```



Step 4. Predict the isoscape

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

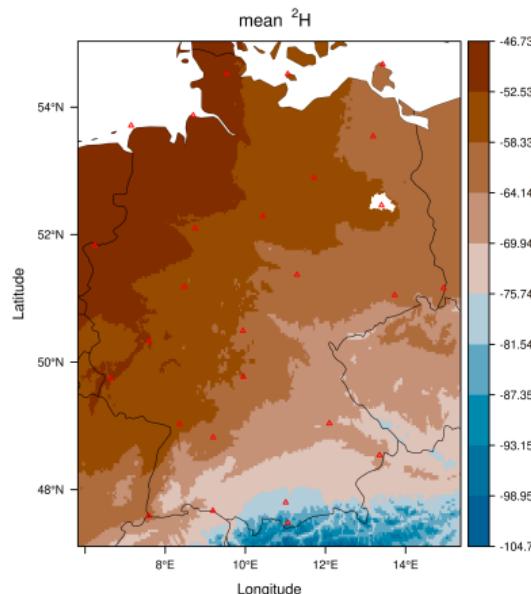
```
plot(GermanScape, mask = list(mask = NULL))
```



Step 4. Predict the isoscape

4b. Plot the rasters with plot():

```
Germany <- rnaturalearth::ne_states(country = "Germany", returnclass = "sv")
          # you can also import any shapefile using terra:vect()
Berlin <- Germany[Germany$name == "Berlin"]
plot(GermanScape, mask = list(mask = rbind(OceanMask, Berlin)), fill = "white")
```



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:

```
terra::writeRaster(GermanScape$isoscapes$mean,  
                   filename = "GermanScape.tif", #many formats are possible  
                   overwrite = TRUE)
```

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

Step 4. Predict the isoscape

Uncertainty check list → same as for **isofit** [▶ Step 2](#).

Step 4. Predict the isoscape

Uncertainty check list → same as for **isofit** [► 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** [► 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

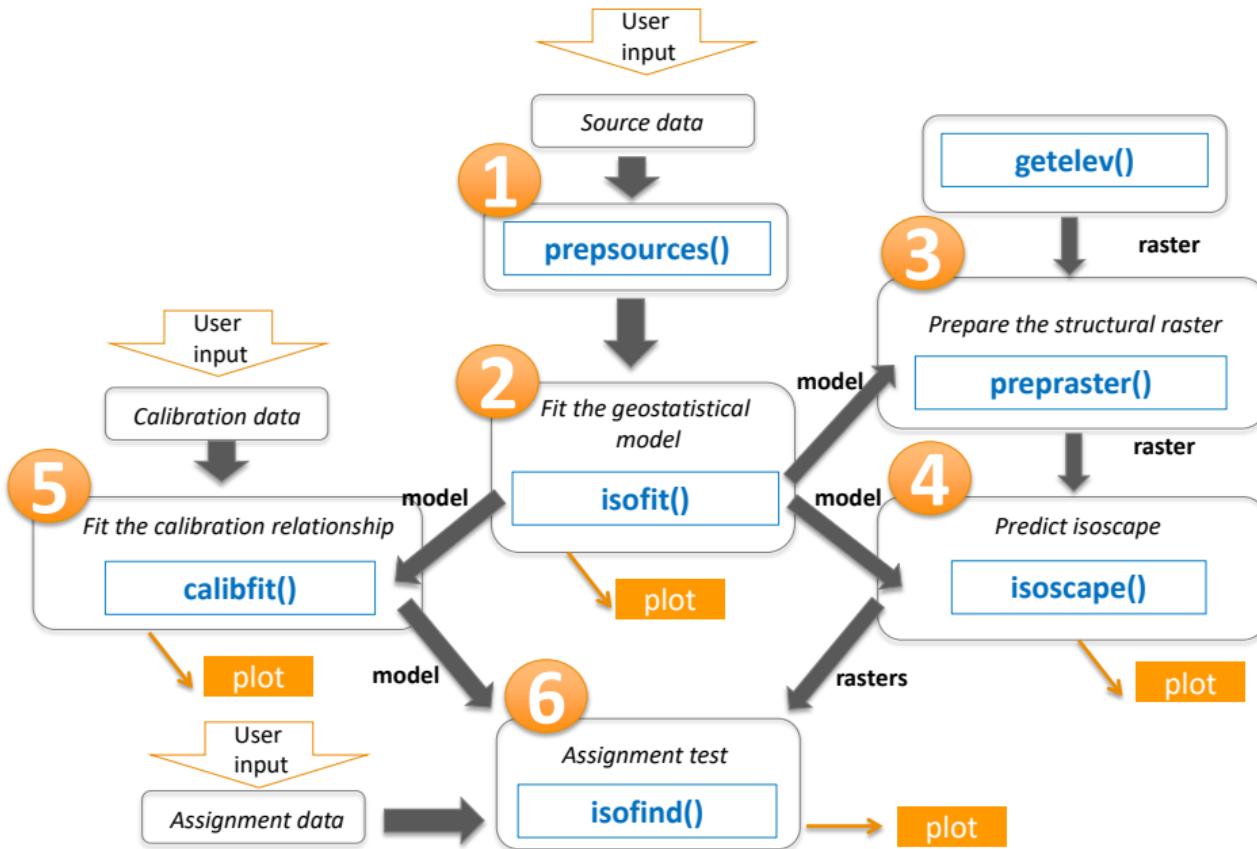
1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

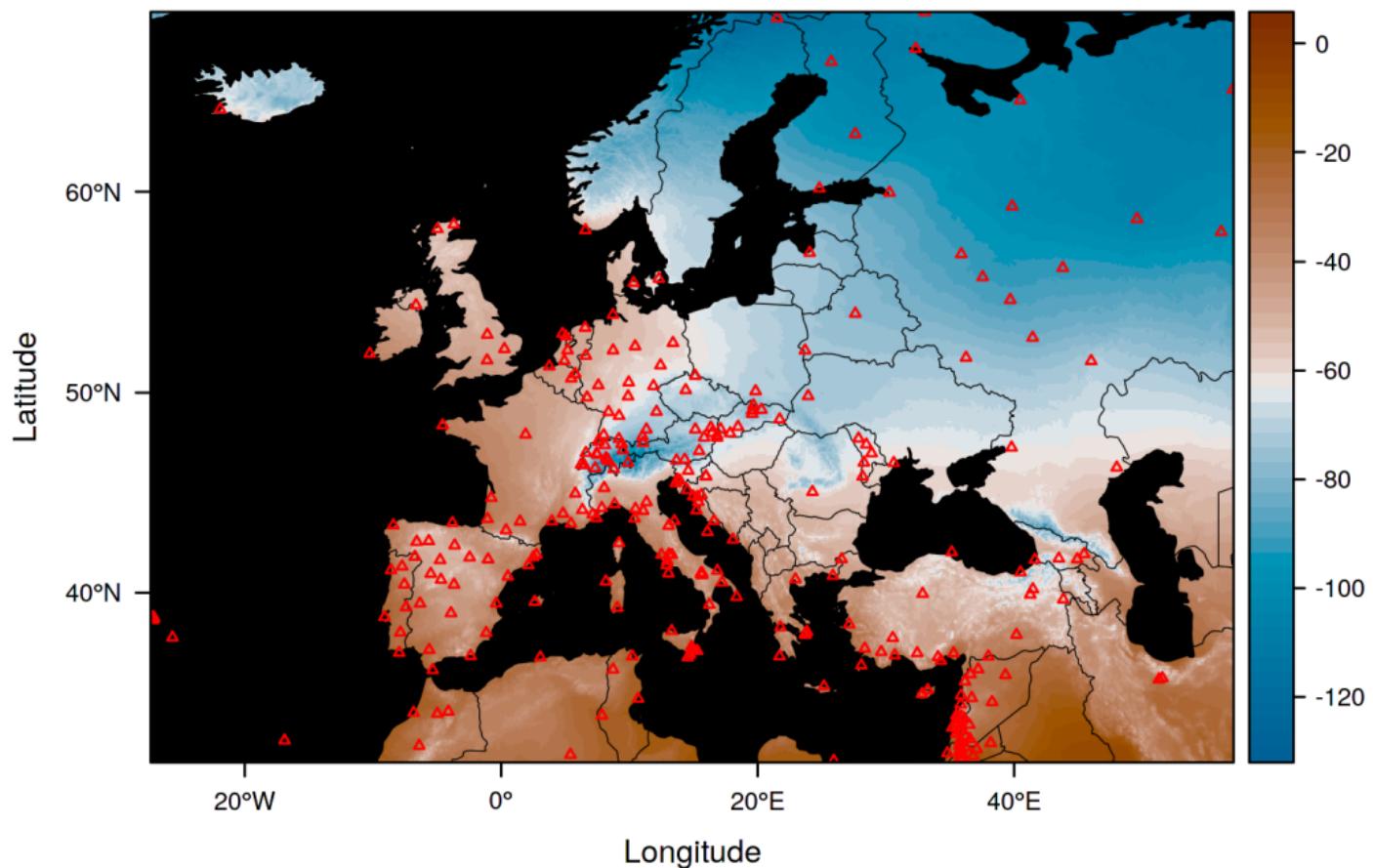
2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- **Step 5. Fit the calibration function**
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX



mean δD_p



Step 5. Fit the calibration function

A calibration procedure is needed whenever the isoscape is not based on the same material as the one you want to assign (e.g. precipitation water vs bats, plancton vs penguins, ...).

The calibration function relates the isotopic composition of samples of known origin to the isotopic composition of the environment at these origin.

It is required to predict what would be the isotopic composition of animals originating from any given position in an isoscape.

Step 5. Fit the calibration function

A calibration procedure is needed whenever the isoscape is not based on the same material as the one you want to assign (e.g. precipitation water vs bats, plancton vs penguins, ...).

The calibration function relates the isotopic composition of samples of known origin to the isotopic composition of the environment at these origin.

It is required to predict what would be the isotopic composition of animals originating from any given position in an isoscape.

IsoriX can handle four different calibration methods (here we will use the "wild" method, the default):

Method	Isotopic composition of the calibration samples	Isotopic composition of the environment associated with the calibration samples
"wild"	known	unknown, estimated from the isoscapes
"lab"	known	known, or assumed to be known
"desk"	unknown, estimated from <code>lm(sample_value ~ source_value)</code>	unknown, estimated from <code>lm(sample_value ~ source_value)</code>
"desk_inverse"	unknown, estimated from <code>lm(source_value ~ sample_value)</code>	unknown, estimated from <code>lm(source_value ~ sample_value)</code>

Step 5. Fit the calibration function

5a. Prepare your calibration data:

```
head(CalibDataBatRev)
##   site_ID     long      lat elev sample_ID species sample_value
## 1 site_29 18.87101 46.19747   90        1       4    -31.00384
## 2 site_29 18.87101 46.19747   90        2       4    -57.50992
## 3 site_29 18.87101 46.19747   90        3       4    -28.57132
## 4 site_29 18.87101 46.19747   90        4       4    -57.42604
## 5 site_32 19.10489 48.53405  522        5       4    -48.19924
## 6 site_32 19.10489 48.53405  522        6       4    -56.33560

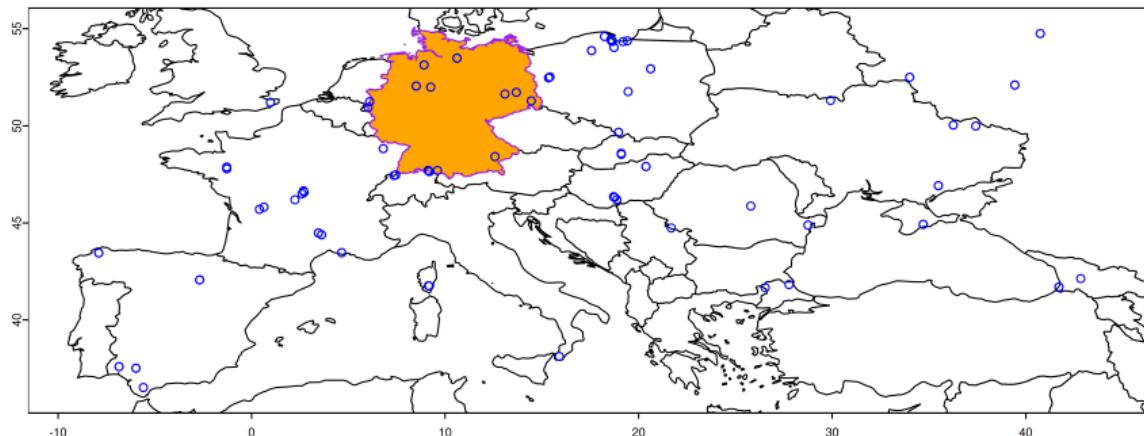
str(CalibDataBatRev)
## 'data.frame': 335 obs. of 7 variables:
## $ site_ID      : Factor w/ 72 levels "site_1","site_10",...: 22 22 22 22 26 26 26 26 4 4 ...
## $ long         : num  18.9 18.9 18.9 18.9 19.1 ...
## $ lat          : num  46.2 46.2 46.2 46.2 48.5 ...
## $ elev         : int  90 90 90 90 522 522 522 20 20 ...
## $ 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  -31 -57.5 -28.6 -57.4 -48.2 ...
```

Note: you do need more than one replicate per location

Step 5. Fit the calibration function

5b. Check your calibration data:

```
Germany <- rnaturalearth::ne_countries(country = "Germany", scale = "large", returnclass = "sv")
with(CalibDataBatRev, plot(CountryBorders, xlim = range_expanded(long), ylim = range_expanded(lat)))
plot(Germany, col = "orange", border = "purple", add = TRUE)
points(jitter(lat, factor = 0.1) ~ jitter(long, factor = 0.1), data = CalibDataBatRev, col = "blue")
```

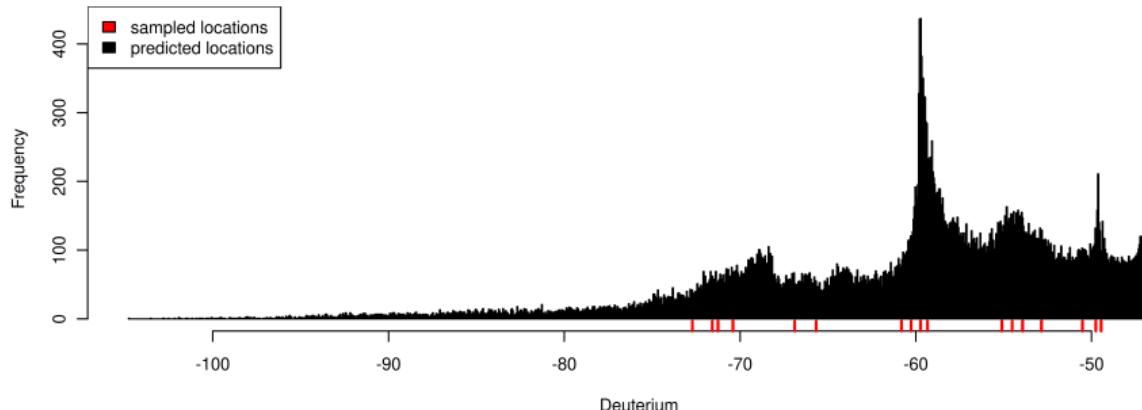


Note: this is NOT great (but we will uses these data for this tutorial)

Step 5. Fit the calibration function

5b. Check your calibration data:

```
IsoPredicted <- values(GermanScape$isoscapes$mean)[, "mean"]
IsoAtBats <- extract(GermanScape$isoscapes$mean, CalibDataBatRev[, c("long", "lat")])[, "mean"]
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"))
```

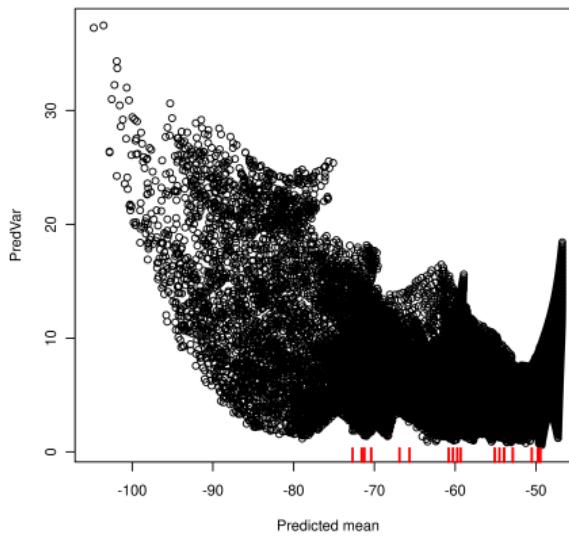


Note: this is quite good but very few points are considered in this plot (only the German ones).

Step 5. Fit the calibration function

5b. Check your calibration data:

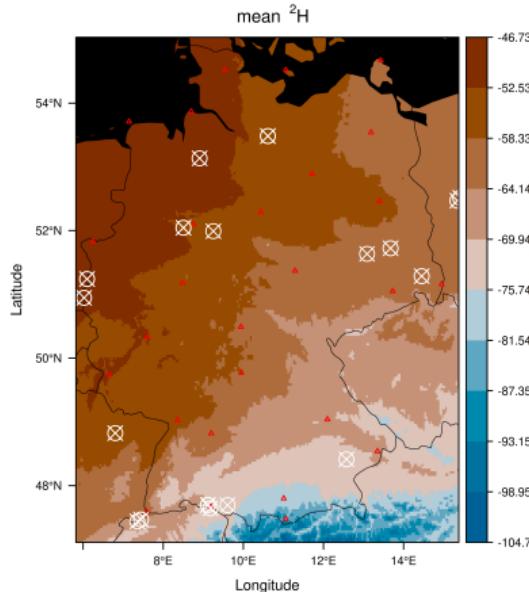
```
IsoPredictedPredVar <- values(GermanScape$isoscapes$mean_predVar)
plot(x = IsoPredicted, y = IsoPredictedPredVar, xlab = "Predicted mean", ylab = "PredVar")
rug(IsoAtBats, col = "red", lwd = 2)
```



Step 5. Fit the calibration function

5b. Check your calibration data:

```
plot(GermanScape, plot = FALSE) + ## plot = FALSE to avoid premature plotting  
xyplot(lat ~ long, data = CalibDataBatRev, pch = 13, col = "white", cex = 2, lwd = 2)
```



The coverage within Germany is quite good but all the location outside Germany will be considered too.

Step 5. Fit the calibration function

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

```
calibfit(  
  data,  
  isofit = NULL,  
  method = c("wild", "lab", "desk", "desk_inverse"),  
  verbose = interactive(),  
  control_optim = list()  
)
```

Step 5. Fit the calibration function

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

```
calibfit(  
  data,  
  isofit = NULL,  
  method = c("wild", "lab", "desk", "desk_inverse"),  
  verbose = interactive(),  
  control_optim = list()  
)
```

Example:

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

```
## Note: extrapolation issues  
## Out of your 335 calibration samples,  
## * 272 correspond to locations outside the area covered by the measurements you used to build your  
isoscapes.  
## -> These cases correspond to extrapolation during the calibration step, which could impede the  
reliability of your assignments.  
## If the proportion of problematic samples is large, you should perhaps rethink the design of your  
isoscapes and/or collect more calibration data within the expected range to avoid any problem.
```

Step 5. Fit the calibration function

5d. Check the fitted 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) = -0.72 +/- 7.16  
##          slope      (+/- SE) = 0.72 +/- 0.12  
##  
## [for more information, use summary()]
```

Step 5. Fit the calibration function

5d. Check the fitted 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) = -0.72 +/- 7.16  
##      slope      (+/- SE) = 0.72 +/- 0.12  
##  
## [for more information, use summary()]
```

The model is quite complex...

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

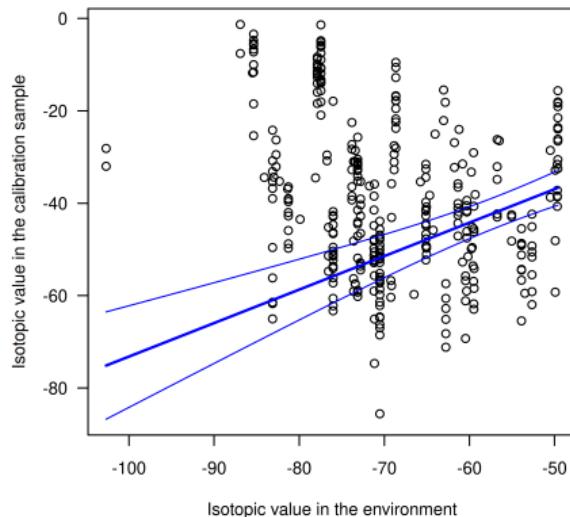
However we often don't have δ_g , but only $\hat{\delta}_g$ and you should not fit $\text{calib}_{gi} = \beta_0^C + \beta_\delta^C \hat{\delta}_g + \varepsilon_{gi}^C$, because there is error on y (usual) but also on x (violating exogeneity assumption).

Which leads us to $\text{calib}_{gi} = \beta_0^C + \beta_\delta^C \hat{\delta}_g + \beta_\delta^C (\delta_g - \hat{\delta}_g) + \epsilon_{gi}^C$, 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).

Step 5. Fit the calibration function

5d. Check the fitted calibration model:

```
plot(CalibBats)
```



Note: this looks bad, but in fact it is incredibly 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

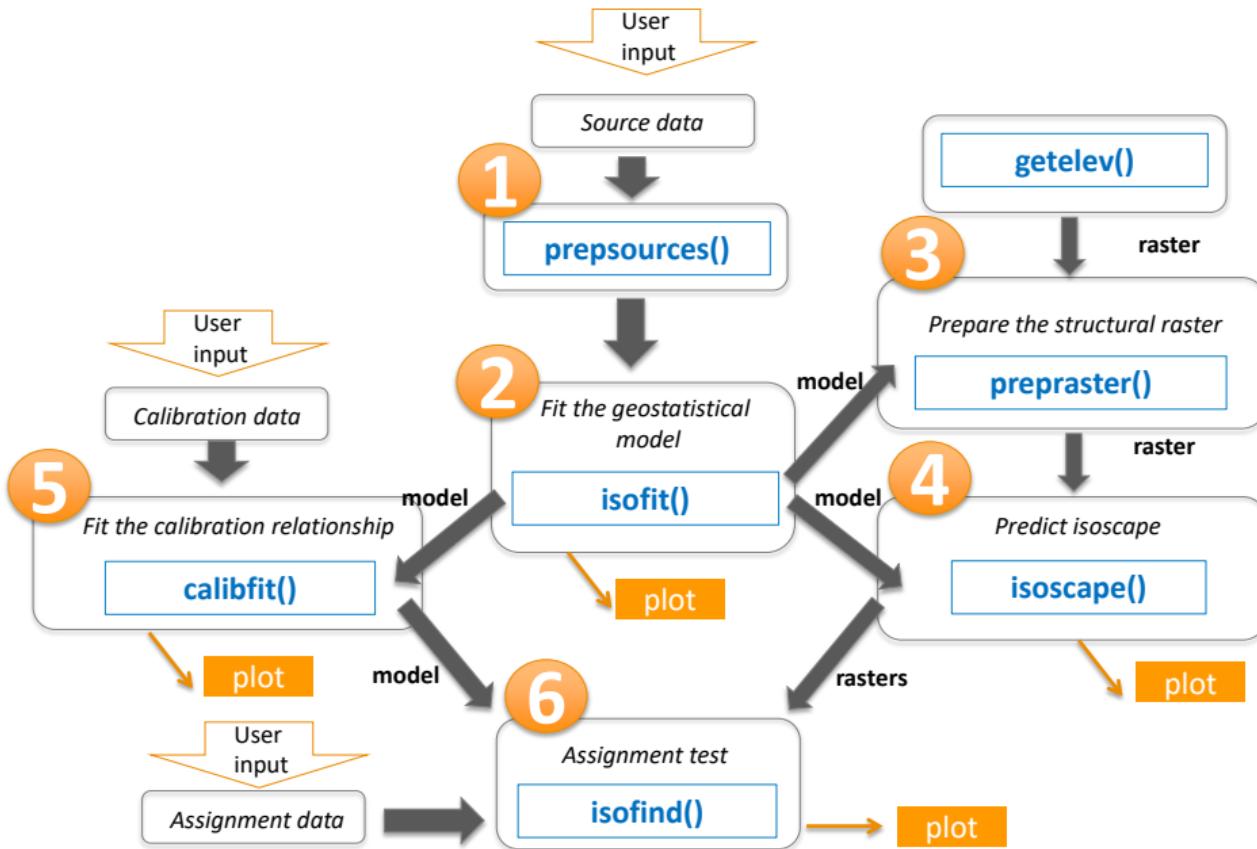
1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment**
- Summary

3 Future of IsoriX



Step 6. Perform the assignment

6a. Prepare your assignment data:

```
AssignDataBatRev
```

```
##   sample_ID      lat      long sample_value
## 1   Nnoc_1 51.72323 13.67630    -43.40550
## 2   Nnoc_2 51.72323 13.67630    -49.59752
## 3   Nnoc_3 51.72323 13.67630    -45.19298
## 4   Nnoc_4 51.69520 14.64945    -41.95605
## 5   Nnoc_5 51.70366 14.63915    -58.19690
## 6   Nnoc_7 51.69520 14.64945    -42.56502
## 7   Nnoc_8 51.70536 14.63289    -46.26581
## 8   Nnoc_9 51.70536 14.63289    -48.47353
## 9   Nnoc_10 51.89348 13.76488   -42.53986
## 10  Nnoc_11 51.84883 13.64525   -43.79554
## 11  Nnoc_12 51.60535 11.92700   -45.72407
## 12  Nnoc_13 51.96695 12.08462   -57.72157
## 13  Nnoc_15 51.96695 12.08462   -72.43155
## 14  Nnoc_16 51.45457 11.50689   -53.39763
```

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()` to perform the assignment tests:

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

Step 6. Perform the assignment

6b. Use `isofind()` to perform the assignment tests:

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

Example:

```
AssignedBats <- isofind(data = AssignDataBatRev, isoscape = GermanScape, calibfit = CalibBats)  
  
names(AssignedBats)  
## [1] "sample"      "group"       "sp_points"
```

Step 6. Perform the assignment

`isofind()` returns the result of the assignment test for each sample:

```
AssignedBats$sample
## $stat
## class      : SpatRaster
## dimensions : 213, 257, 14 (nrow, ncol, nlyr)
## resolution : 0.03719131, 0.03719131 (x, y)
## extent     : 5.820439, 15.3786, 47.111, 55.03275 (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## names       : Nnoc_1, Nnoc_2, Nnoc_3, Nnoc_4, Nnoc_5, Nnoc_7, ...
## min values  : -11.70518, -20.24810, -14.17131, -9.705429, -32.11237, -10.54560, ...
## max values  : 45.86309, 37.32017, 43.39696, 47.862843, 25.45590, 47.02267, ...
##
## $stat_var
## class      : SpatRaster
## dimensions : 213, 257, 14 (nrow, ncol, nlyr)
## resolution : 0.03719131, 0.03719131 (x, y)
## extent     : 5.820439, 15.3786, 47.111, 55.03275 (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## names       : Nnoc_1, Nnoc_2, Nnoc_3, Nnoc_4, Nnoc_5, Nnoc_7, ...
## min values  : 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, ...
## max values  : 156.9414, 160.2009, 157.4513, 156.7849, 171.6925, 156.8227, ...
##
## $pv
## class      : SpatRaster
## dimensions : 213, 257, 14 (nrow, ncol, nlyr)
## resolution : 0.03719131, 0.03719131 (x, y)
## extent     : 5.820439, 15.3786, 47.111, 55.03275 (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## names       : Nnoc_1, Nnoc_2, Nnoc_3, Nnoc_4, Nnoc_5, Nnoc_7, ...
## min values  : 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000, ...
## max values  : 0.9999892, 0.9999807, 0.9999353, 0.9999988, 0.9998048, 0.9999862, ...
```

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.

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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).

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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).
- The null hypothesis of the test is that the sample comes from the cell being examined.

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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).
- 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.

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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).
- 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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).
- 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.

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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).
- 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.
- It is possible that all candidate locations are rejected (unlike relative Bayesian probabilities).

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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).
- 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.
- It is possible that all candidate locations are rejected (unlike relative Bayesian probabilities).
- A lack of rejection may stem from a good match, a large uncertainty, or both.

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 test statistic is the difference between the predicted isoscape value at the origin location (deduced from the calibration fit) and the predicted isoscape value at the location under consideration (read from the isoscape).
- 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.
- It is possible that all candidate locations are rejected (unlike relative Bayesian probabilities).
- 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 easily extract the outcome of the assignment test in the cell corresponding to 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]))  
##      Nnoc_1  
## 1 0.8424342  
  
extract(AssignedBats$sample$pv[[1]], cbind(AssignDataBat$long[1], AssignDataBat$lat[1]))  
##      Nnoc_1  
## 1 0.9398607
```

Step 6. Perform the assignment

You can easily extract the outcome of the assignment test in the cell corresponding to 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]))  
##      Nnoc_1  
## 1 0.8424342  
  
extract(AssignedBats$sample$pv[[1]], cbind(AssignDataBat$long[1], AssignDataBat$lat[1]))  
##      Nnoc_1  
## 1 0.9398607
```

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

Step 6. Perform the assignment

You can easily extract the outcome of the assignment test in the cell corresponding to 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]))  
##      Nnoc_1  
## 1 0.8424342  
  
extract(AssignedBats$sample$pv[[1]], cbind(AssignDataBat$long[1], AssignDataBat$lat[1]))  
##      Nnoc_1  
## 1 0.9398607
```

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]))  
##      Nnoc_15  
## 1 -41.01754  
  
extract(AssignedBats$sample$pv[[13]], cbind(AssignDataBat$long[13], AssignDataBat$lat[13]))  
##      Nnoc_15  
## 1 0.001996747
```

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

Step 6. Perform the assignment

You can easily extract the outcome of the assignment test in the cell corresponding to a given location

Example 3: which bats are unlikely to come from where they have been found?

```
Pvalues <- sapply(1:nrow(AssignDataBat),  
                  function(i) extract(AssignedBats$sample$pv[[i]],  
                                      cbind(AssignDataBat$long[i],  
                                            AssignDataBat$lat[i])))  
  
AssignDataBat$sample_ID[Pvalues <= 0.05]  
## [1] Nnoc_15  
## Levels: Nnoc_1 Nnoc_10 Nnoc_11 Nnoc_12 Nnoc_13 Nnoc_15 Nnoc_16 Nnoc_2 Nnoc_3 Nnoc_4 Nnoc_5 Nnoc_7 Nnoc_8 Nnoc_9 Nnoc_14
```

Answer: only Nnoc_15 seems to be a migrant.

Step 6. Perform the assignment

`isofind()` also returns the result of the assignment test for the whole group of samples:

```
AssignedBats$group
## $pv
## class      : SpatRaster
## dimensions : 213, 257, 1 (nrow, ncol, nlyr)
## resolution : 0.03719131, 0.03719131 (x, y)
## extent     : 5.820439, 15.3786, 47.111, 55.03275 (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## name        : lyr.1
## min value   : 0.0000000
## max value   : 0.7208934
```

Note:

The null hypothesis is here that all samples come from the location being examined.
(so it can often be rejected anywhere)

Step 6. Perform the assignment

6c. Plot the assignment tests with `plot()`:

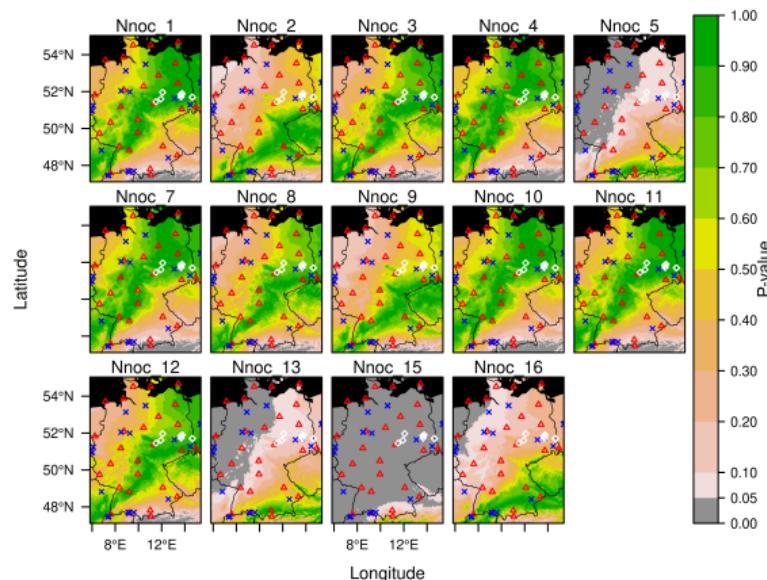
```
plot(  
  x,  
  who      = "group",  
  cutoff   = list(draw = TRUE, level = 0.05, col = "#909090"),  
  sources  = list(draw = TRUE, cex = 0.5, pch = 2, lwd = 1, col = "red"),  
  calibs   = list(draw = TRUE, cex = 0.5, pch = 4, lwd = 1, col = "blue"),  
  assigns  = list(draw = TRUE, cex = 0.5, pch = 5, lwd = 1, col = "white"),  
  borders  = list(borders = NA, lwd = 0.5, col = "black"),  
  mask     = list(mask = NA, lwd = 0, col = "black", fill = "black"),  
  mask2    = list(mask = NA, lwd = 0, col = "purple", fill = "purple"),  
  palette  = list(step = NA, range = c(0, 1), n_labels = 11, digits = 2, fn = NA),  
  plot     = TRUE,  
  sphere   = list(build = FALSE, keep_image = FALSE),  
  ...  
)
```

It is quite similar to the function which plots the isoscapes.

Step 6. Perform the assignment

6c. Plot the assignment tests with `plot()`:

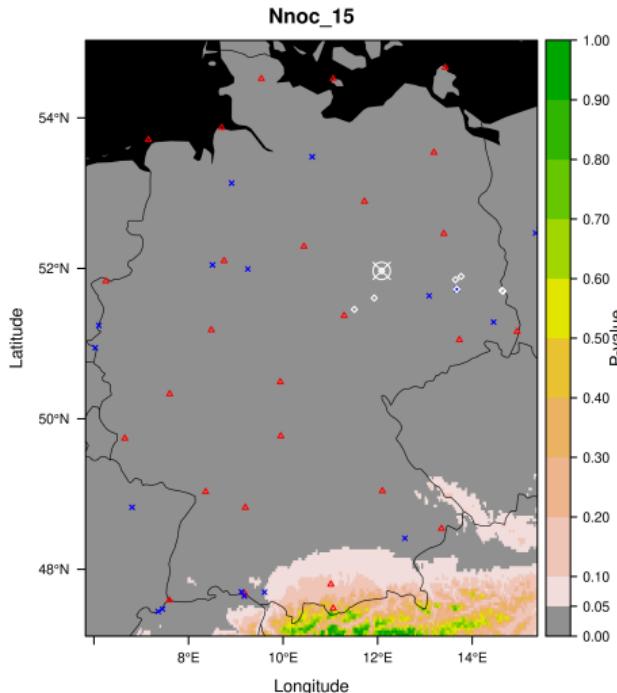
```
plot(AssignedBats, who = 1:14)
```



Step 6. Perform the assignment

6c. Plot the assignment tests with `plot()`:

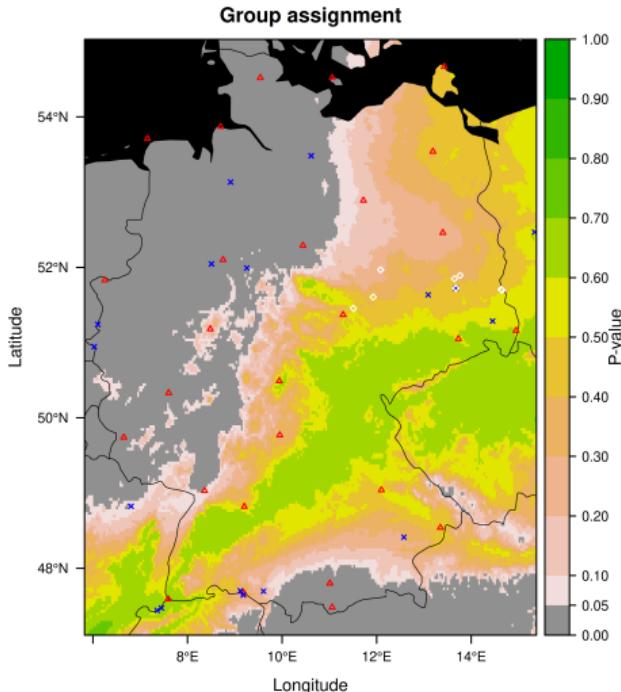
```
plot(AssignedBats, who = "Nnoc_15", plot = FALSE) +  
  xyplot(lat ~ long, data = subset(AssignDataBatRev, sample_ID == "Nnoc_15"),  
         pch = 13, col = "white", cex = 2, lwd = 2)
```



Step 6. Perform the assignment

6c. Plot the assignment tests with `plot()`:

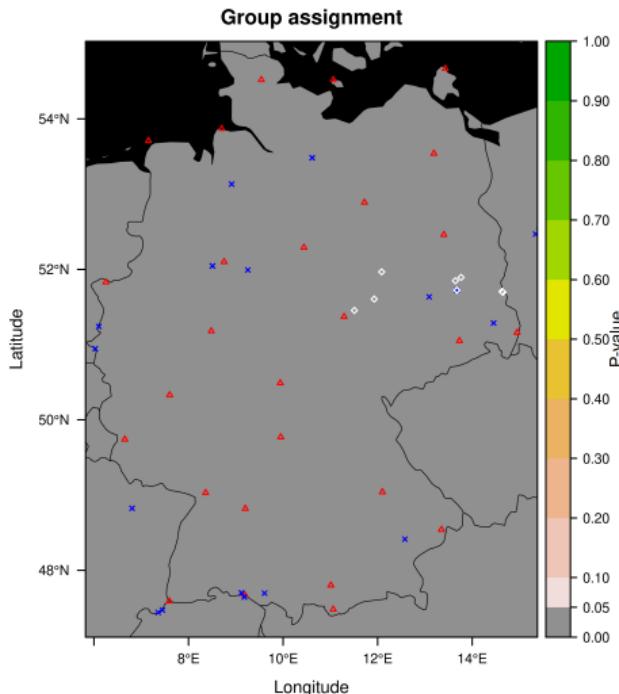
```
plot(AssignedBats) ## plot the group assignment
```



Step 6. Perform the assignment

We can rerun the group assignment without Nnoc_15:

```
AssignDataBat2 <- subset(AssignDataBat, sample_ID != "Nnoc_15")
AssignedBats2 <- isofind(data = AssignDataBat2, isoscape = GermanScape, calibfit = CalibBats)
plot(AssignedBats2)
```



Step 6. Perform the assignment

The most important questions you must answer:

- Are all samples coming from the same location?
- Can the true origin be outside the map?

Step 6. Perform the assignment

The most important questions you must answer:

- Are all samples coming from the same location?
- Can the true origin be outside the map?

Recommendation:

- Always look at the plots per sample before looking at a group assignment.

Step 6. Perform the assignment

Uncertainty check list:

- ✓ prediction variance of the fit of the mean model
- ✓ prediction variance of the inverted calibration fit
- ✓ residual variance of the calibration fit
- ✗/✓¹ covariation between predictions from the mean model and the inverted calibration fit

¹not included as default but included if 'neglect_covPredCalib = FALSE'

Table of contents

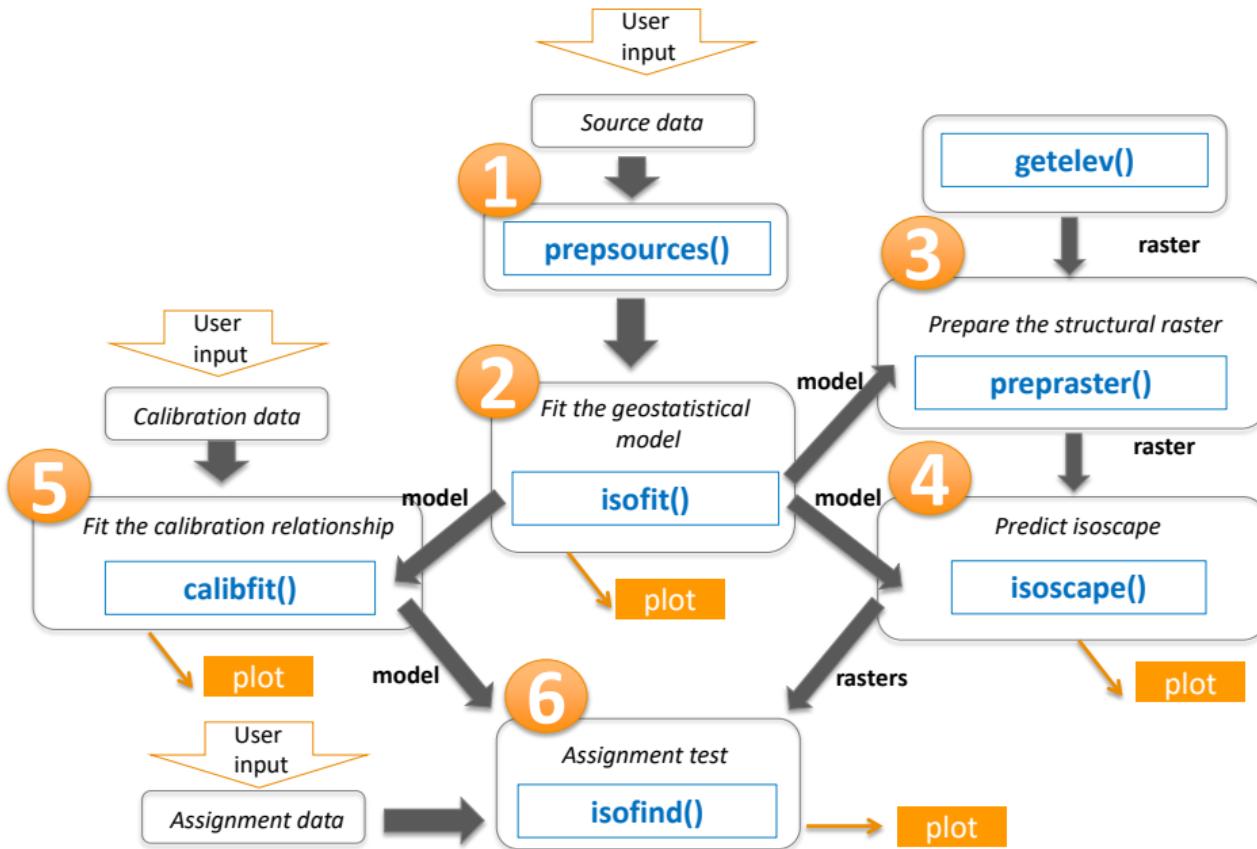
1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX



Where does the bat Nnoc_15 comes from?

An entire workflow can take under 10 lines of code:

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

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

getelev()
ElevDE         <- rast("~/elevation_world_z5.tif")
ElevRasterDE   <- prepraster(ElevDE, isofit = GermanFit, aggregation_factor = 2)

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

CalibBats      <- calibfit(data = CalibDataBatRev, isofit = GermanFit)

Assigned15     <- isofind(data = subset(AssignDataBatRev, sample_ID == "Nnoc_15"),
                           isoscape = GermanScape, calibfit = CalibBats)
```

Note: you should always check all intermediary outputs as we did. 

Where does the bat Nnoc_15 comes from?

```
plot(Assigned15)
```

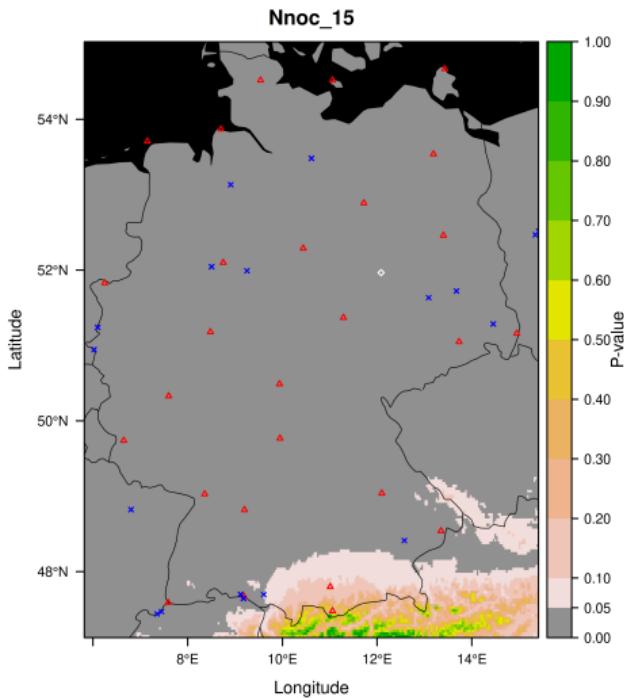


Table of contents

1 Generalities about IsoriX

- Why IsoriX?
- What is IsoriX?
- Help?

2 Standard workflow

- Step 1. Prepare the source data
- Step 2. Fit the isoscape
- Step 3. Prepare the structural raster
- Step 4. Predict the isoscape
- Step 5. Fit the calibration function
- Step 6. Perform the assignment
- Summary

3 Future of IsoriX

Possible future features

- More documentation in the bookdown (<https://bookdown.org/content/782/>).
- More predictors for building isoscapes (e.g. temperature), with data download performed via **R** packages.
- More precipitation isotopic data readily available.
- More calibration data readily available.
- Automatised the weighting the isoscape by precipitation amount.
- DHGLM to capture the uncertainty of the residual dispersion model.
- 3D/4D isoscapes.
- Adding clustering after assignment (with p-value against null model)?

Possible future features

- More documentation in the bookdown (<https://bookdown.org/content/782/>).
- More predictors for building isoscapes (e.g. temperature), with data download performed via **R** packages.
- More precipitation isotopic data readily available.
- More calibration data readily available.
- Automatised the weighting the isoscape by precipitation amount.
- DHGLM to capture the uncertainty of the residual dispersion model.
- 3D/4D isoscapes.
- Adding clustering after assignment (with p-value against null model)?
- What would you prefer? Anything other suggestions?

To follow or contribute to the development of IsoriX

- <https://github.com/courtiol/IsoriX>
- <https://groups.google.com/g/IsoriX>