IsoriX: an R-package to fit isoscapes & perform geographic assignments

Alexandre Courtiol[1], François Rousset[2] & Stephanie Kramer-Schadt[1]

[1] Leibniz Institute of Zoo and Wildlife Research, Berlin (Germany)
[2] Institut des Sciences de l'Évolution, Montpellier (France)

12 February 2015

Table of contents

Table of contents

Why R?

R seems to be today the **most widely used statistical software in the world** (excluding Microsoft Excel).

It is a **free open source** software environment and programming language for statistical computing & graphics that **runs on UNIX** (linux and more), **Windows** & **MacOS**.

Most of the user-visible functions in R are written in R, but **internal procedures can also be writen in C, C++, or FORTRAN**.

Numerous functions are included within the core installation of R and **6,185 packages** are available on the Comprehensive R Archive Network ("CRAN") for a variety of specific purposes.

The user can run a workflow stored in a script file, which allows for **reproducible research** & **easy communication**.

Why IsoriX?

Benefits:

- reproducible workflow
- simple but customizable
- GIS integration
- cutting edge statistical methods

Current limits:

- R knowledge required
- runs locally
- constrained model structure
- one response variable (i.e. 1 type of isotope)
- Maximum Likelihood only

What is IsoriX?

A wrapper! (without help files: only 725 lines of codes)

- direct dependencies:
  - **spaMM** ($\geq$ 1.4.5) (for modelling)
  - **sp**, **raster** (for spatial files)
  - **rasterVis**, **latticeExtra**, **grid** (for plots)
  - **Cairo**, **rgdal** (for exports)

- recursive dependencies:

```
gtools::getDependencies("IsoriX", available=FALSE)
```

```
##  [1] "abind"        "magic"        "RColorBrewer" "hexbin"
##  [5] "zoo"          "lpSolveAPI"   "proxy"        "geometry"
##  [9] "Rcpp"         "RcppEigen"    "sp"           "Cairo"
## [13] "latticeExtra" "raster"       "rasterVis"    "rgdal"
## [17] "spaMM"
```

Content of IsoriX

```
library(help="IsoriX")[[3]][[2]]
```

```
##  [1] "Calibrate                Calibrate isotopic values of organisms into the"
##  [2] "                         isocape scale"
##  [3] "FitCalibrationModel      Fit the calibration model"
##  [4] "FitIsoscapeModel         Fit the isoscape model"
##  [5] "GNIP_Europe              European weather stations data"
##  [6] "IsoriX-package           Isoscape-based Inference of Spatial Origins by"
##  [7] "                         Maximum Likelihood"
##  [8] "Isorix                   Infer spatial origins"
##  [9] "Isoscape                 Construct the isoscape raster"
## [10] "PrepareElevationRaster"
## [11] "                         Prepare the elevation raster"
## [12] "calibrationdata          Calibration dataset"
## [13] "elevationrastersmall     The raster of world elevation"
## [14] "isorixpalette            Colour palettes for plotting"
## [15] "isoscapemodel            The isoscape model fitted on GNIP_Europe"
## [16] "oceanmask                Ocean mask polygon"
## [17] "plot.Isorix              Plotting functions for IsoriX"
## [18] "summary.Isorix           Summary and print methods for IsoriX"
## [19] "worldcountries           World country borders"
```

Table of contents

0. Starting IsoriX

```
setwd("/home/alex/Downloads")  # set working directory

install.packages(pkgs="IsoriX_0.9995.tar.gz", repos=NULL, type="source")

library(IsoriX)
```

```
## Loading required package:  sp
##
## ##########################################
## #  IsoriX (version 0.9995) is loaded!  #
## # Type ?IsoriX for a short introduction #
## # Type example(IsoriX)for a quick demo  #
## ##########################################
```

0. An introduction to IsoriX

```
?IsoriX
```

```
IsoriX-package              package:IsoriX              R Documentation

Isoscape-based Inference of Spatial Origins by Maximum Likelihood

Description:

     Maximum likelihood-based spatial assignment of organisms of
     unknown geographic origin(s) based on their isotopic signature.
     This package can be used for building isoscapes and inferring the
     geographic origin of organisms based on their isotopic ratios.
     This package is essentially a wrapper that uses the package
     'spaMM' for fitting and predicting isoscapes, and assigning an
     organism's origin depending on its isotopic ratio. 'IsoriX' also
     relies heavily on the package 'rasterVis' for plotting the maps
     using 'lattice'.

Details:

     We describe below, step-by-step, the general workflow that aims at
     performing the assignment of organisms of unknown geographic
     origin(s) based on their isotopic signature. Please note that any
     spatial data (locations, rasters) have to be expressed in latitude
```

Workflow

1. Fitting the Isoscape with *FitIsoscapeModel*
2. Preparing the elevation raster with *PrepareElevationRaster*
3. Building the isoscape with *Isoscape*
4. Calibrating the data with *FitCalibrationModel* & *Calibrate* (optional)
5. Performing the assignment with *Isorix*

1. FitIsoscapeModel

Goal: Fits the geostatistical isoscape model using the 'spaMM'.

```
library(spaMM)

## spaMM (version 1.4.5) is loaded.
## Type 'help(spaMM)' for a short introduction.

citation("spaMM")

##
## To cite spaMM in publications use:
##
##   François Rousset and Jean-Baptiste Ferdy (2014) Testing
##   environmental and genetic effects in the presence of spatial
##   autocorrelation. Ecography 37(8): 781-790 URL
##   http://dx.doi.org/10.1111/ecog.00566
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {Testing environmental and genetic effects in the pre
##     author = {François Rousset and Jean-Baptiste Ferdy}
```

1. FitIsoscapeModel

Definition:

```
FitIsoscapeModel(isoscape.data, use.sd = FALSE, matern = TRUE, optimize = TRUE)
```

Input:

```
data(GNIP_Europe)
head(GNIP_Europe)
```

```
##   isoscape.value sd.isoscape.value n.isoscape.value      lat     long elev
## 1      -14.00396         14.357319               111 31.62000 25.95000   24
## 2      -21.15250         20.250122                12 31.66667 35.98333  715
## 3      -20.01882         14.352760                17 31.72000 35.78333  785
## 4      -11.18621         26.586634                29 31.85000 36.81667  533
## 5      -38.16000          9.444205                 5 31.92000  5.40000  150
## 6      -24.63988         17.432109                42 31.95765 35.84828  900
```

```
dim(GNIP_Europe)
```

```
## [1] 296   6
```

Usage:

```
isoscapemodel <- FitIsoscapeModel(isoscape.data=GNIP_Europe, use.sd=TRUE)
```

1. FitIsoscapeModel: outputs

```
summary(isoscapemodel)

## formula: isoscape.value ~ lat.abs + lat.2 + elev + Matern(1 | long + lat)
## REML: Estimation of lambda by REML.
##       Estimation of fixed effects by ML.
## Family: gaussian ( link = identity )
##  ------- Fixed effects (beta) -------
##              Estimate  Cond. SE  t-value
## (Intercept) 55.235932 55.786778   0.9901
## lat.abs     -2.047039  1.981760  -1.0329
## lat.2       -0.001864  0.019790  -0.0942
## elev        -0.010211  0.000825 -12.3773
##  ---------- Random effects ----------
## Family: gaussian ( link = identity )
## Correlation parameters:
##          nu         rho
## 0.255393055 0.001129052
## Coefficients for log[ lambda = var(u) ]:
##       Group        Term Estimate Cond.SE
##  long + lat (Intercept)    7.057 0.08601
## Estimate of lambda:  1161
## # of obs: 296; # of groups: long + lat, 296
##  -------- Residual variance  --------
## phi was fixed to 492.051
##  -------- Likelihood values  --------
##                          logLik
## p_v(h) (marginal L): -1010.209
##   p_beta,v(h) (ReL): -1015.143
```

1. FitIsoscapeModel: outputs (in details)

```
class(isoscapemodel)
```

```
## [1] "HLfit" "list"
```

```
names(isoscapemodel)
```

```
##  [1] "APHLs"          "data"           "y"              "prior.weights"
##  [5] "family"         "X.pv"           "ranFix"         "corrPars"
##  [9] "models"         "predictor"      "ZALMatrix"      "ZAlist"
## [13] "HL"             "fv"             "fixef"          "beta_cov"
## [17] "eta"            "lev_phi"        "std_dev_res"    "lev_lambda"
## [21] "rand.families" "ranef"          "v_h"            "w.resid"
## [25] "w.ranef"        "lambda"         "fittedLambda"   "lambda.object"
## [29] "phi"            "phi.object"     "warnings"       "spaMM.version"
## [33] "call"
```

1. FitIsoscapeModel: outputs (in details)

```
isoscape.value ~ lat.abs + lat.2 + elev + Matern(1 | long + lat)
```

```
 ------- Fixed effects (beta) -------
             Estimate  Cond. SE  t-value
(Intercept) 55.235932 55.786778   0.9901
lat.abs     -2.047039  1.981760  -1.0329
lat.2       -0.001864  0.019790  -0.0942
elev        -0.010211  0.000825 -12.3773
```

1. FitIsoscapeModel: outputs (in details)

```
isoscape.value ~ lat.abs + lat.2 + elev + Matern(1 | long + lat)
```

```
---------- Random effects ----------
Family: gaussian ( link = identity )
Correlation parameters:
         nu         rho
0.255393055 0.001129052
Coefficients for log[ lambda = var(u) ]:
      Group        Term Estimate Cond.SE
 long + lat (Intercept)    7.057 0.08601
Estimate of lambda:  1161
# of obs: 296; # of groups: long + lat, 296
```

1. FitIsoscapeModel: outputs (in details)

```
isoscape.value ~ lat.abs + lat.2 + elev + Matern(1 | long + lat)
```

```
 -------- Residual variance  --------
phi was fixed to 492.051
```

1. FitIsoscapeModel: outputs (in details)

```
isoscape.value ~ lat.abs + lat.2 + elev + Matern(1 | long + lat)
```

```
 -------- Likelihood values  --------
                          logLik
p_v(h) (marginal L): -1010.209
  p_beta,v(h) (ReL): -1015.143

REML: Estimation of lambda and phi by REML.
      Estimation of fixed effects by ML.
```

1. FitIsoscapeModel: behind the curtain

```
weights <- as.numeric(isoscape.data$n.isoscape.value)

phi <- with(isoscape.data,
    sum(sd.isoscape.value^2*(n.isoscape.value-1)) /
    (sum(n.isoscape.value)-length(n.isoscape.value))
    )

phi.stations <- as.numeric(phi/isoscape.data$n.isoscape.value)

formula <- "isoscape.value ~ lat.abs + lat.2 + elev + Matern(1|long+lat)"

model <- spaMM::corrHLfit(
  formula = as.formula(formula),
  prior.weights = weights,
  ranFix=list(phi=phi.stations),
  data = isoscape.data
  )

model$phi.object$phi.Fix <- phi
model$phi <- phi
```

2. PrepareElevationRaster

Goal: prepares the elevation raster for the follow-up analyses.

Definition:

```
PrepareElevationRaster(elevation.raster,
                       isoscape.model=NULL,
                       aggregation.factor=0L,
                       aggregation.fun=mean,
                       manual.crop
                       )
```

Input:

```
data(elevationrastersmall)
elevationrastersmall
```

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

2. PrepareElevationRaster

Usage:

```
elevationraster <- PrepareElevationRaster(elevation.raster=elevationrastersmall,
    isoscape.model=isoscapemodel, aggregation.factor=0)
```

```
## [1] "cropping... (may take a while)"
## [1] "done!"
##    user  system elapsed
##   0.028   0.004   0.031
## class       : RasterLayer
## dimensions  : 45, 102, 4590  (nrow, ncol, ncell)
## resolution  : 0.8333333, 0.8333333  (x, y)
## extent      : -27.50014, 57.49986, 31.49986, 68.99986  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
## data source : in memory
## names       : elevationrastersmall.asc
## values      : -27.1915, 2499.111  (min, max)
```

Output:

```
class(elevationraster)
```

```
## [1] "RasterLayer"
## attr(,"package")
## [1] "raster"
```

3. Isoscape

Goal: builds the predicted isoscape.

Definition:

```
Isoscape(elevation.raster,
         isoscape.model,
         build.sd=TRUE,
         chunk.size.for.predict=150L,
         save.spatial.files=FALSE,
         file.prefix.spatial.files="isoscape",
         overwrite.spatial.files=TRUE
         )
```

Usage:

```
isoscape<-Isoscape(elevation.raster=elevationraster, isoscape.model=isoscapemodel)
```

```
class(isoscape)
```

```
## [1] "Isoscape" "list"
```

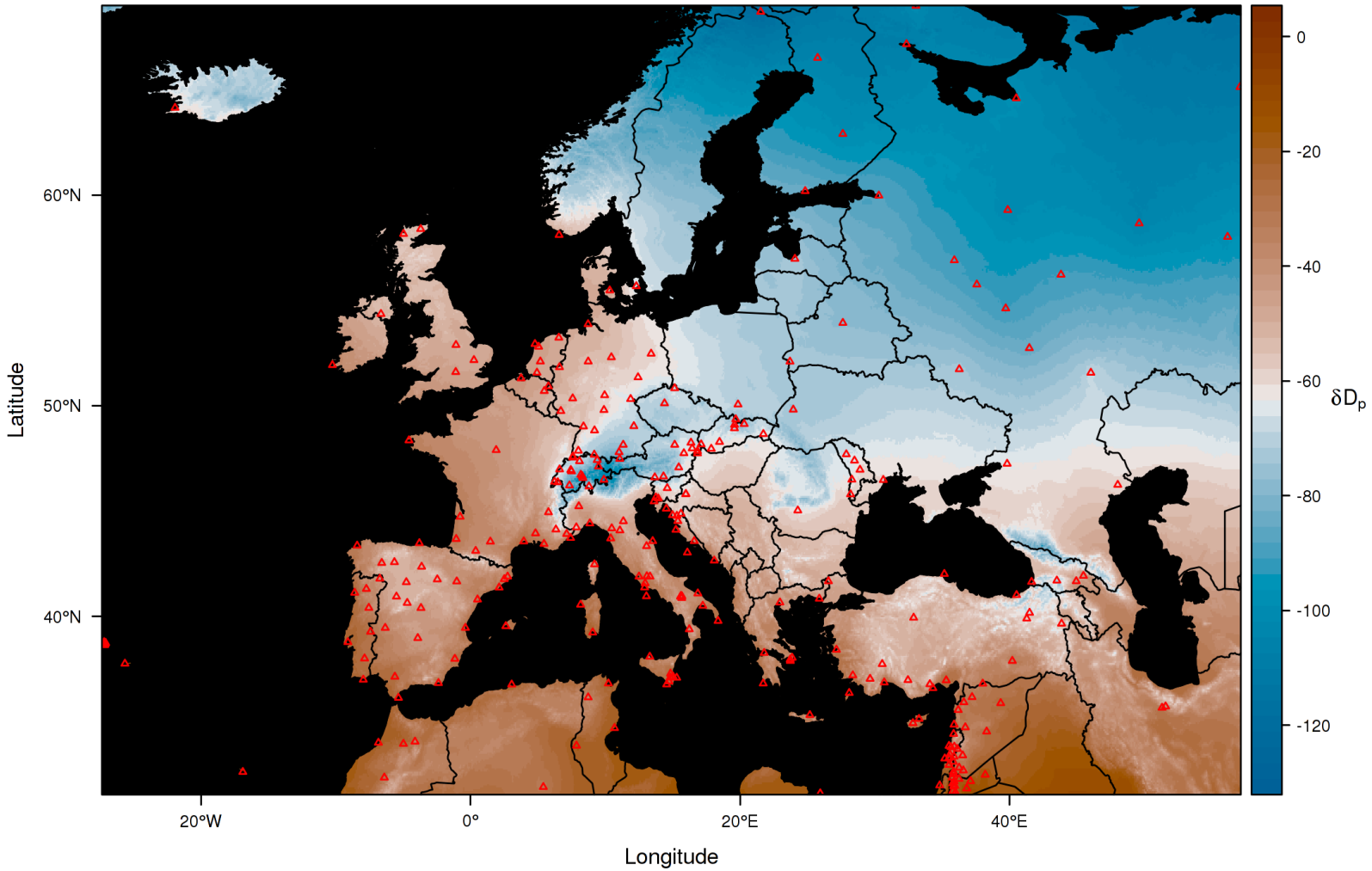
3. Isoscape: outputs

```
isoscape

## pred.raster
## class       : RasterLayer
## dimensions  : 45, 102, 4590  (nrow, ncol, ncell)
## resolution  : 0.8333333, 0.8333333  (x, y)
## extent      : -27.50014, 57.49986, 31.49986, 68.99986  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84
## data source : in memory
## names       : values
## values      : -120.8351, -3.685889  (min, max)
##
##
## sd.raster
## class       : RasterLayer
## dimensions  : 45, 102, 4590  (nrow, ncol, ncell)
## resolution  : 0.8333333, 0.8333333  (x, y)
## extent      : -27.50014, 57.49986, 31.49986, 68.99986  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84
## data source : in memory
## names       : values
## values      : 22.37085, 25.9321  (min, max)
##
##
## source.points.sp
##                  coordinates values
## 1              (25.95, 31.62)  -9999
## 112     (35.98333, 31.66667)  -9999
## 124        (35.78333, 31.72)  -9999
## 141        (36.81667, 31.85)  -9999
## 170             (5.4, 31.92)  -9999
## 175     (35.84828, 31.95765)  -9999
```

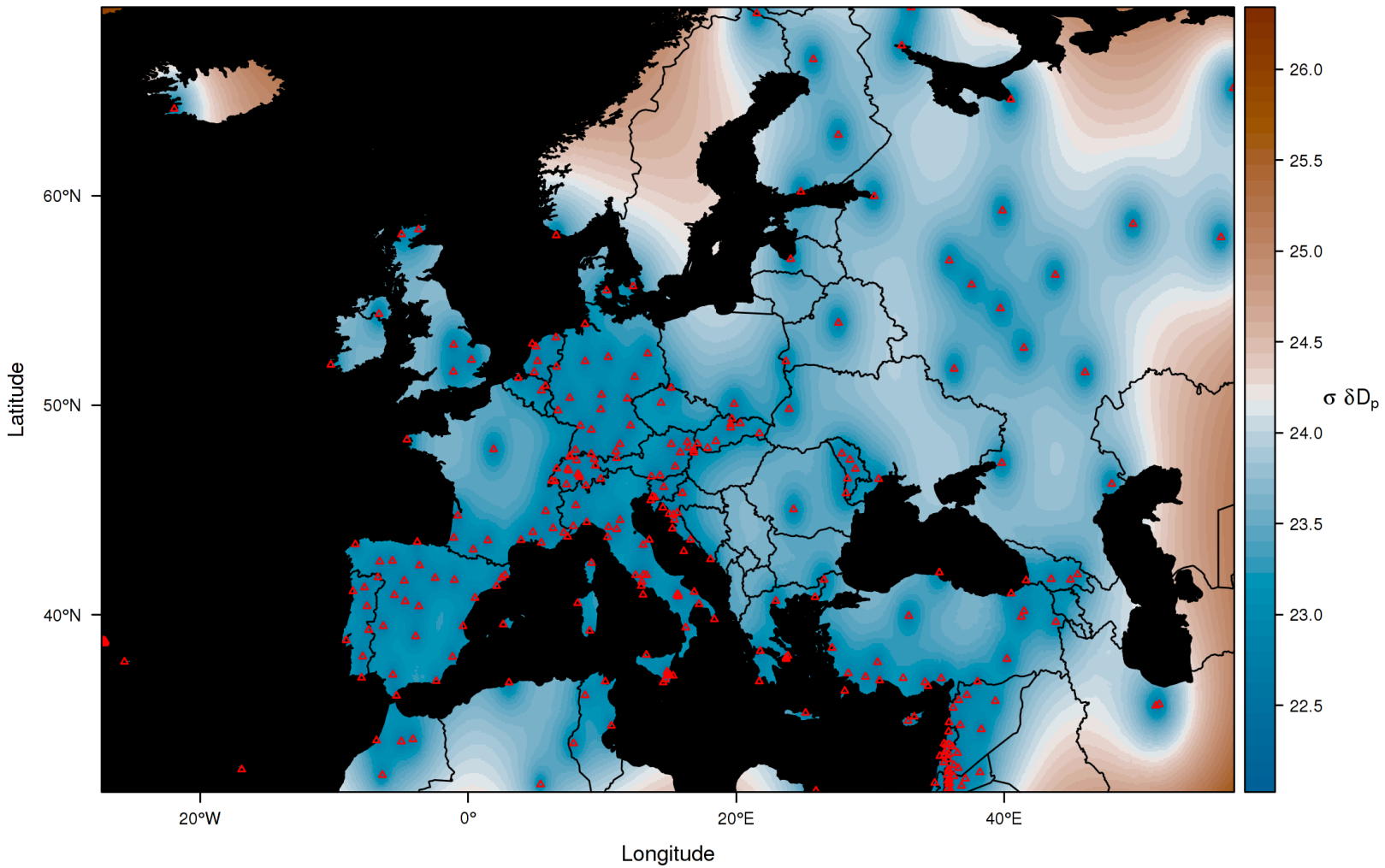
3. Isoscape: outputs

```
plot(isoscape, country.borders=worldcountries, mask=oceanmask,
     palette=isoscapepalette)
```



3. Isoscape: outputs

```
plot(isoscape, plot.sd=TRUE, country.borders=worldcountries, mask=oceanmask,
     palette=isoscapepalette)
```



3. Isoscape: behind the curtain

```
x.for.predictions <- data.frame(
  long=long,
  lat=lat,
  lat.abs=abs(lat),
  lat.2=lat^2,
  elev=extract(elevation.raster, cbind(long, lat))
  )

pred <- spaMM::predict.HLfit(isoscape.model,
                 newX=x.for.predictions,
                 coeffs=coeffs,
                 variances=list(sum=TRUE),
                 binding="fitted"
                 )

predicted.isoscape <- pred$fitted
sd.isoscape <- sqrt(attr(pred, "sumVar"))
```

4a. FitCalibrationModel

Goal: fits the calibration function.

Definition:

```
FitCalibrationModel(calibration.data,
                    isoscape.model,
                    n.simu=100L,
                    poly.degree=1L,
                    plot.diagnostics=TRUE
                    )
```

Input:

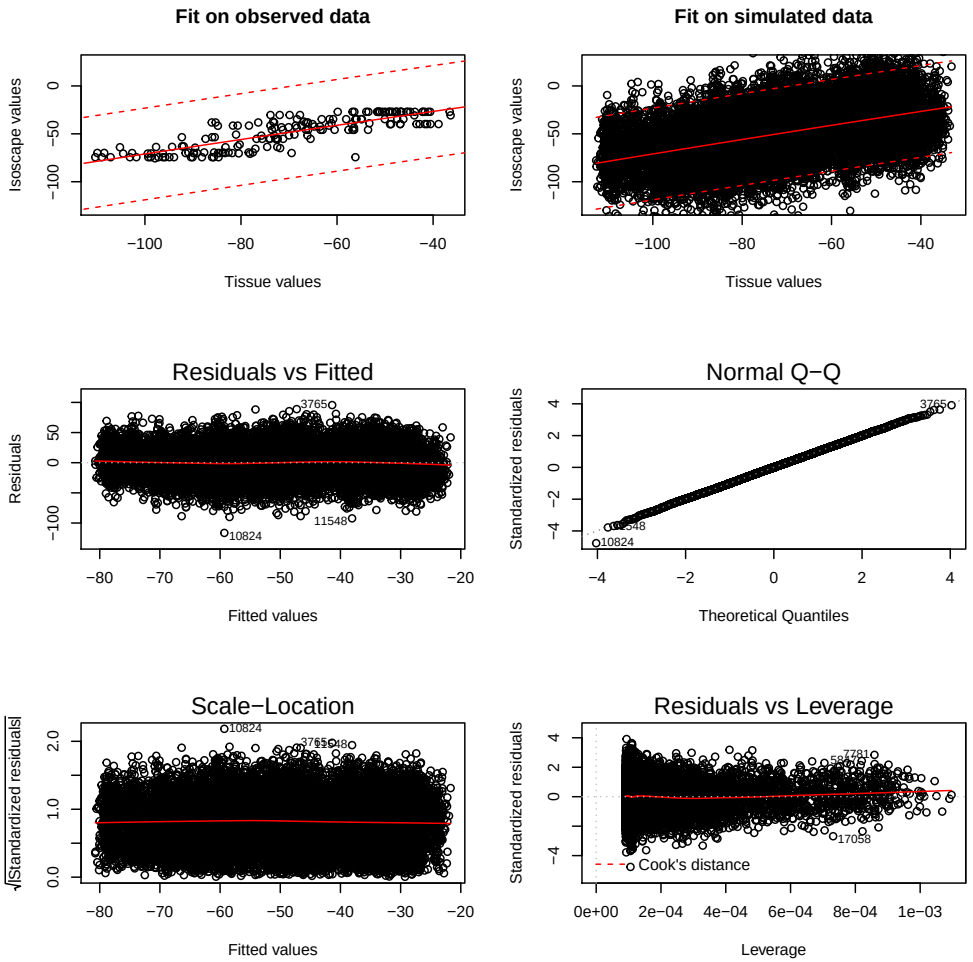
```
data(calibrationdata)
head(calibrationdata)
```

```
##        lat      long elev tissue.value sd.tissue.value
## 1 51.63507 13.090210   88    -80.27262                1
## 2 51.28597 14.452515  142    -97.67615                1
## 3 51.28597 14.452515  142    -72.44561                1
## 4 53.13359  8.909912    6    -81.02448                1
## 5 53.13359  8.909912    6    -68.87410                1
## 6 47.69497  9.607544  493    -97.89426                1
```

4a. FitCalibrationModel

Usage:

```
calibrationmodel <- FitCalibrationModel(calibration.data=calibrationdata,
    isoscape.model=isoscapemodel, poly.degree=2L)
```



4a. FitCalibrationModel: output

```
class(calibrationmodel)

## [1] "lm"

summary(calibrationmodel)

##
## Call:
## lm(formula = isoscape.value ~ poly(tissue.value, poly.degree,
##     raw = TRUE))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -107.702  -16.126    0.385   16.054   93.554
##
## Coefficients:
##                                                 Estimate Std. Error t value
## (Intercept)                                    0.0019639  2.4709105   0.001
## poly(tissue.value, poly.degree, raw = TRUE)1   0.6145897  0.0718522   8.554
## poly(tissue.value, poly.degree, raw = TRUE)2  -0.0011249  0.0004969  -2.264
##                                                Pr(>|t|)
## (Intercept)                                      0.9994
## poly(tissue.value, poly.degree, raw = TRUE)1    <2e-16 ***
## poly(tissue.value, poly.degree, raw = TRUE)2    0.0236 *
## ---
```

4a. FitCalibrationModel: behind the curtain

```
predicts.for.calibration <- spaMM::predict.HLfit(isoscape.model,
                                                 newX=x.for.predicts,
                                                 variances=list(resid=TRUE),
                                                 binding="fitted"
                                                 )

calibration.data$fitted <- predicts.for.calibration$fitted
calibration.data$sd.fitted <- sqrt(attr(predicts.for.calibration, "residVar"))

tissue.value <- as.numeric(replicate(n.simu, rnorm(
      nrow(calibration.data),
      mean=calibration.data$tissue.value,
      sd=calibration.data$sd.tissue.value)))

isoscape.value <- as.numeric(replicate(n.simu, rnorm(
      nrow(calibration.data),
      mean=calibration.data$fitted,
      sd=calibration.data$sd.fitted)))

calibration.model <- lm(isoscape.value ~ poly(tissue.value, poly.degree, raw=TRUE))
```

4b. Calibrate

Goal: performs the calibration.

Definition:

```
Calibrate(noncalibrated.data, calibration.model)
```

Input:

```
toy.dataset <- data.frame(noncalibrated.value=c(-90, -120, -123),
  sd.noncalibrated.value=c(1, 1, 1))
```

Usage:

```
calibrated.dataset <- Calibrate(
  noncalibrated.data=toy.dataset,
  calibration.model=calibrationmodel
  )
```

4b. Calibrate: output

```
head(calibrated.dataset)
```

```
##   noncalibrated.value sd.noncalibrated.value value.to.assign
## 1                 -90                      1       -64.42281
## 2                -120                      1       -89.94739
## 3                -123                      1       -92.61121
##   sd.value.to.assign
## 1           23.94209
## 2           23.96618
## 3           23.97331
```

4b. Calibrate: behind the curtain

```
pred <- predict.lm(calibration.model,
                   newdata=list(tissue.value=noncalibrated.value),
                   se.fit=TRUE
                   )

value.to.assign <- as.numeric(pred$fit)
sd.value.to.assign <- as.numeric(sqrt(pred$se.fit^2+pred$residual.scale^2))
```

5. Isorix

Goal: performs the assignment.

Definition:

```
Isorix(assignment.data, isoscape,
   conf.level = 0.95,
   save.spatial.files = FALSE,
   file.prefix.spatial.files = "assignment_spatial_files",
   overwrite.spatial.files = TRUE)
```

Input:

```
head(calibrated.dataset[, c(3:4)])
```

```
##   value.to.assign sd.value.to.assign
## 1       -64.42281           23.94209
## 2       -89.94739           23.96618
## 3       -92.61121           23.97331
```

5. Isorix

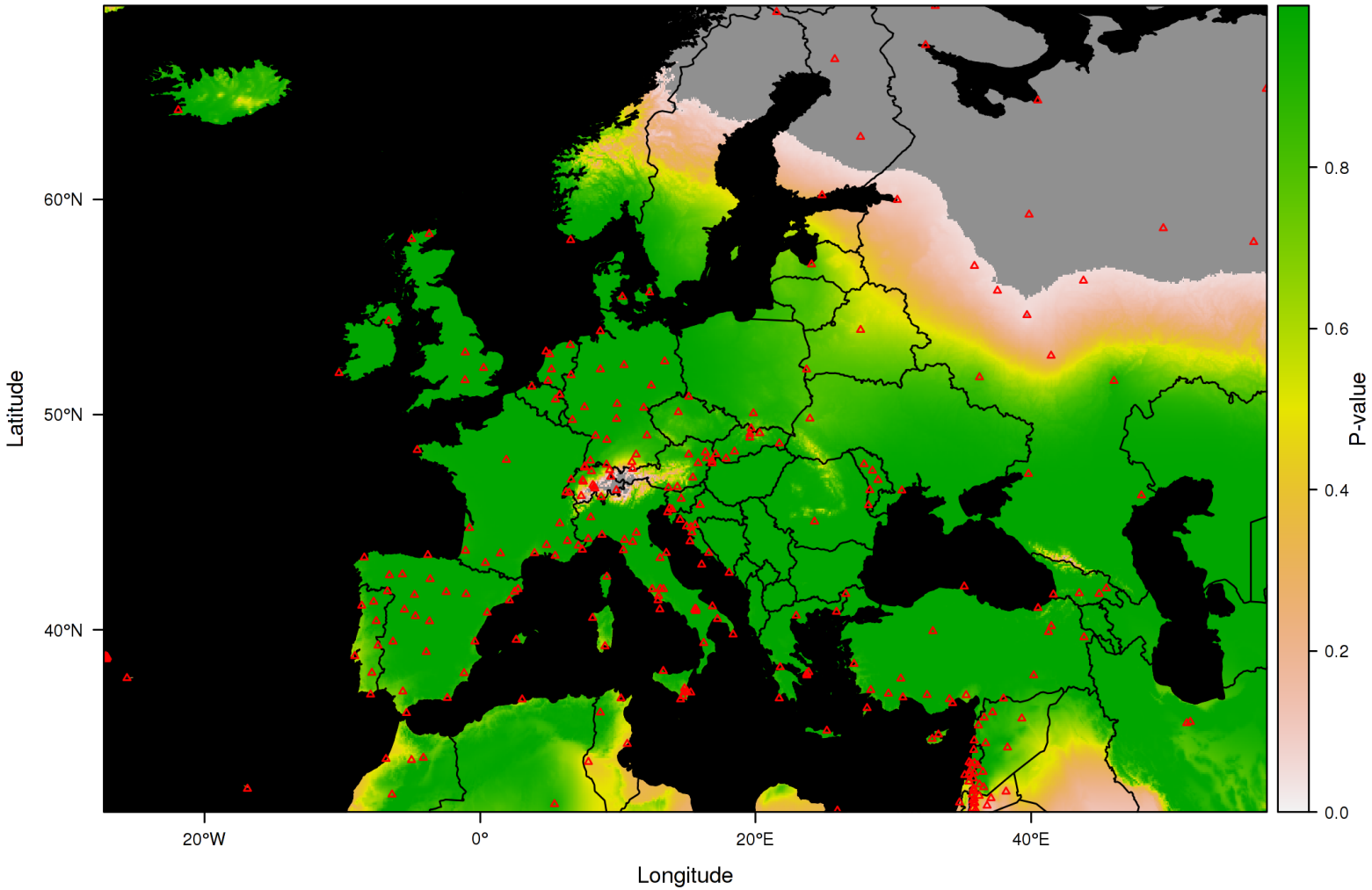
Usage:

```
assignment <- Isorix(assignment.data=my.calibrated.dataset, isoscape=isoscape)

plot(
   assignment,
   plot.predint=TRUE,
   plot.sources=TRUE,
   country.borders=worldcountries,
   mask=oceanmask,
   palette=isorixpalette
   )

plot(
   assignment.fine,
   plot.predint=TRUE,
   plot.sources=TRUE,
   country.borders=worldcountries,
   mask=oceanmask,
   palette=isorixpalette,
   save.png=TRUE,
   filename.png="assignment_test"
   )
```

5. Isorix: output



5. Isorix: behind the curtain

```
for(j in 1:length(assignment.data$value.to.assign)) {
  sd.total <- sqrt(isoscape[["sd.raster"]]@data@values^2+
          assignment.data$sd.value.to.assign[j]^2)

  log.q1 <-  pnorm(
          q=assignment.data$value.to.assign[j],
          mean=isoscape[["pred.raster"]]@data@values,
          sd=sd.total,
          log.p=TRUE)
  log.q2 <-  pnorm(
          q=assignment.data$value.to.assign[j],
          mean=isoscape[["pred.raster"]]@data@values,
          sd=sd.total,
          log.p=TRUE,
          lower.tail=FALSE)

  log.p.PI.values[j, ] <- log(2) + apply(cbind(log.q1, log.q2), 1, min)
}

fisher.stat <- -2*(apply(log.p.PI.values, 2, sum))
fisher.df <- 2*length(assignment.data$value.to.assign)
fisher.pv <- pchisq(q=fisher.stat, df=fisher.df, lower.tail=FALSE)
```

Table of contents

Perspectives

Short term:

- minor changes (earth-based distance, input raster of p-values . . .)
- connectivity with GNIP data (direct/indirect?)
- tests (cross validation), debugging
- host files (e.g. elevation raster) somewhere
- host IsoriX on CRAN
- package presentation in Journal of Statistical Software (IF=3.8)

Middle term:

- more fixed effects?
- ≥ 1 isotope? (independent realization of λ across isotopes?)

Long term:

- maintenance?
- keep it simple vs. swiss army knife?

The assignment workflow in IsoriX