Identifying the flyways boundaries of migratory bird species

Clement Calenge,
Office national de la classe et de la faune sauvage
Saint Benoist – 78610 Auffargis – France.

August 2017

Contents

Introduction

This vignette is the supplementary material of the paper of Guillemain et al. The aim of this paper is to estimate the boundaries of migratory birds flyways. It is supposed throughout this vignette that the reader is familiar with the model developed in this paper. Shortly, this paper uses a dataset describing the recapture locations of common teals initially captured and ringed in two places: (i) in Abberton Reservoir, UK, and (ii) in Camargue, southern France. The aim of the analysis is to estimate the limits of the flyways of the two bird populations (French and British) from these data.

This vignette contains two main sections:

- The first section presents the R code used to fit the model, to calculate the abmigration rates, to test the goodness of fit of the model, and to check the validity of assumptions we made.

- The second section presents a sensitivity analysis for this model. Because this model relies on several subjective decisions (choice of a new coordinate system, number of functions in the B-spline basis), a referee of the paper rightly suggested to perform this sensitivity analysis, to demonstrate the robustness of our results to theses choices.

Remark: to circumvent copyright issues, we provide here an altered version of the dataset used in the paper: we selected a random sample of 75% of the recaptures of the original data, keeping only the location information (i.e. only the x and y coordinates of the recaptures), and we added a random noise to these locations (we moved every bird recapture location randomly by a distance comprised between 0 and 100 km). Note that although the dataset provided in the package is altered, **the other datasets provided in the packages (model fits and sensitivity analysis) are the actual results, obtained from the dataset used in our paper**. In other words, if the reader executes the R code used in this vignette, they will not necessarily find the exact same results, as these results are based on a different dataset. However, these results will be similar (the original dataset has only be slightly altered).

1 Model fit

In this section, we present the R code used for the model fit. The slightly altered dataset is provided with the package `metroponcfs`. We load both the package and the dataset `recteal`:

```r
library(metroponcfs)
data(recteal)
```

The dataset `recteal` has the following structure:

```r
str(recteal)

## List of 5
##  $ recaptures           :'data.frame': 2799 obs. of  4 variables:
##   ..$ date     : chr [1:2799] "Ec1" "Ec3" "Ec1" "Ec1" ...
##   ..$ Abberton: num [1:2799] 1 0 1 0 0 1 1 0 0 1 ...
##   ..$ x        : num [1:2799] -773000 -307589 -624430 150334 -442103 ...
##   ..$ y        : num [1:2799] 9150 -87533 14962 -755649 -935611 ...
##  $ rotationMatrix       : num [1:2, 1:2] 0.573 0.819 0.819 -0.573
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "ry" "rx"
##  $ inverseRotationMatrix: num [1:2, 1:2] 0.819 0.573 -0.573 0.819
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "rx" "ry"
##   .. ..$ : NULL
##  $ knots                : num [1:26] -2136695 -2136695 -2136695 -2136695 -1983111 ...
##  $ lipum                :List of 14
##   ..$ theta1a: num [1:6] 534629 384607 190679 321605 423245 ...
##   ..$ theta1b: num [1:2, 1:2] 2.63e+10 -3.66e+10 -3.66e+10 7.71e+10
##   ..$ theta1c: num [1:2] 289153 164412
##   ..$ theta1d: num [1:2, 1:2] 2.40e+09 -1.19e+09 -1.19e+09 8.88e+08
##   ..$ theta1e: num [1:3, 1:3] 5.56e+08 -1.26e+09 2.14e+09 -1.26e+09 1.11e+10 ...
##   ..$ theta1f: num [1:7] 197387 162688 244091 555261 548994 ...
##   ..$ theta2a: num [1:6] 432826 440466 345341 450084 264407 ...
##   ..$ theta2b: num [1:4, 1:4] 3.28e+10 -2.04e+10 1.91e+10 -1.45e+10 -2.04e+10 ...
##   ..$ theta2c: num 86002
##   ..$ theta2d: num [1:4, 1:4] 3.47e+09 -1.92e+09 2.82e+09 -2.04e+09 -1.92e+09 ...
##   ..$ theta2e: num [1:7] 307634 306893 379750 304340 553514 ...
##   ..$ psi22  : num 0.187
##   ..$ psi21  : num 0.234
##   ..$ delta  : num 0.12
```

This dataset stores the recapture information is a list containing the following elements:

- `$ recaptures`: this element is a data.frame containing the following information for each recapture of common teal: (i) a variable named `date`, storing the number of years elapsed between the initial capture and recapture, (ii) a variable named `Abberton`, indicating whether the recaptured animal was initially captured at Abberton Reservoir, UK (=1) or in Camargue, southern France (=0), and (iii) two variables named `x` and `y` containing the coordinates of the recapture (coordinate system: Lambert azimuthal equal-area).

- `$ rotationMatrix`: this 2×2 matrix $\mathbf{M}$ contains the two vectors $\{\mathbf{m}_1, \mathbf{m}_2\}$ used to rotate the geographical coordinates in a new coordinate system. Thus, if $\mathbf{x}$ is a vector of length two containing the x and y coordinate of a location in the Lambert azimuthal equal-area system, $\mathbf{z} = \mathbf{x}\mathbf{M}$ contains the coordinates of this point in this new system.

- `$ inverseRotationMatrix`: this 2×2 matrix $\mathbf{R}$ allows to transform the coordinates of a point from the new coordinate system to the old one. Thus, if $\mathbf{z}$ is a vector of length two containing the coordinates of a point in the new coordinate system, $\mathbf{x} = \mathbf{z}\mathbf{R}$ contains the coordinates of this point in the original Lambert azimuthal equal-area system.

- `$ knots`: this vector contains the coordinates of the 26 knots in the new coordinate system used to define the B-spline basis in the paper of Guillemain et al.

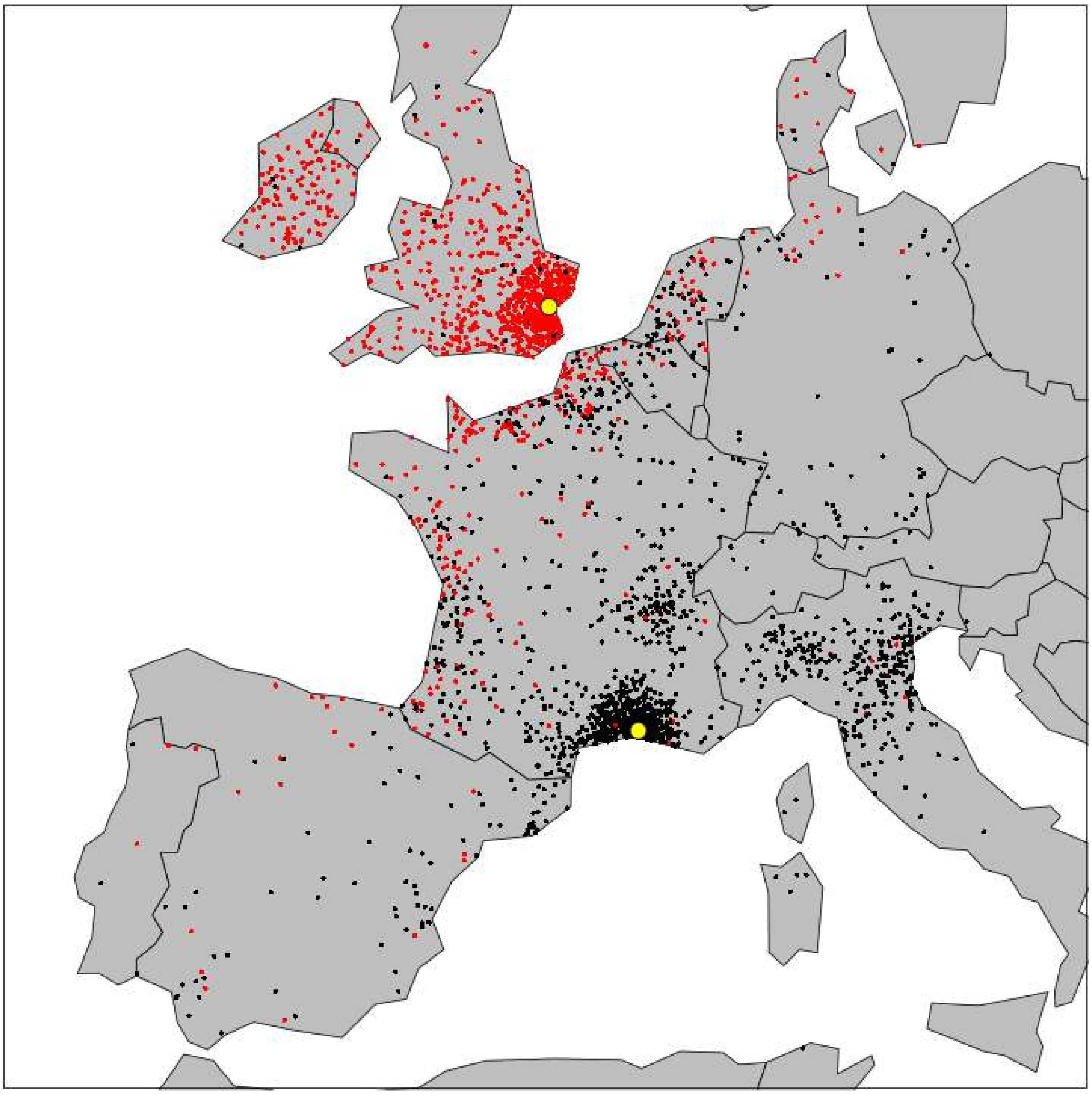
- `$ lipum`: a list containing the parameters of the updating mechanisms used in the Metropolis algorithm (see section 1.1.3).

1.1 Preparation of the data and model fit

1.1.1 Preparation of the data

First, we can display a map of the spatial distribution of recaptures. We need the packages `rworldmap`, `rworldxtra` and `sp` to draw a map of Europe in the correct coordinate system. We draw the map below:

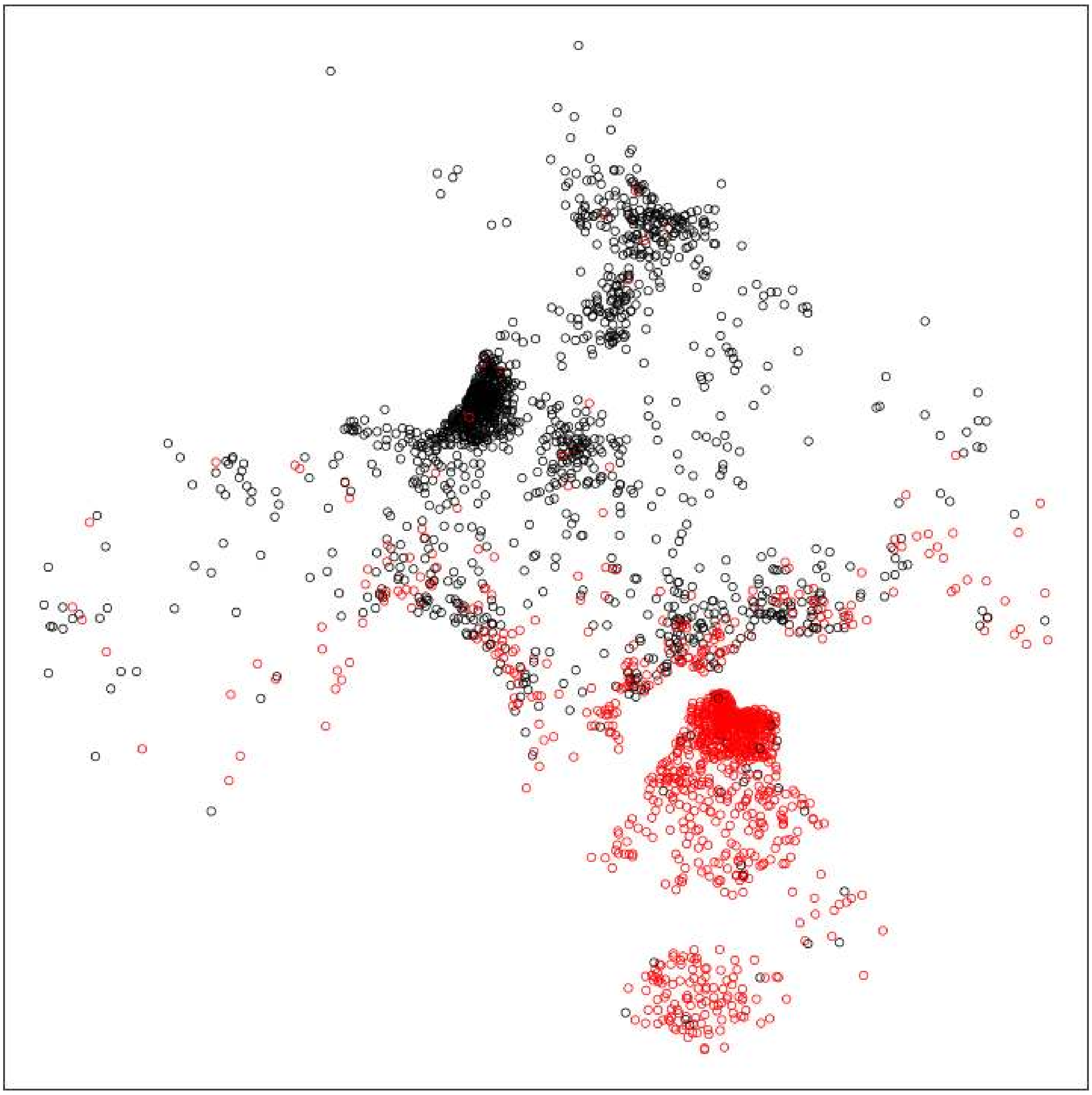
```r
## Load required packages
library(sp)
library(rworldmap)
library(rworldxtra)

## Get the background map
ma <- getMap(resolution = "less islands")

## Selects only the relevant continent (Europe + Russia + North Africa)
ma <- ma[(ma$REGION=="Asia"|ma$REGION=="Europe"|
              ma$REGION=="Africa")&!is.na(ma$REGION),]

## Projects the coordinates system in Lambert Azimuthal Equal Area
ma <- spTransform(ma, CRS("+proj=laea +lat_0=52 +lon_0=10 +x_0=0 +y_0=0
+datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0"))

## Prepare the graphics
par(mar=c(0.1,0.1,0.1,0.1))
plot(recteal$recaptures[,c("x","y")], ty="n", asp=1, axes = FALSE)

## adds the background map
plot(ma,add=TRUE, col="grey")

## The recaptures
points(recteal$recaptures[,c("x","y")],
       col=recteal$recaptures$Abberton+1,
       pch=16, cex=0.6)

## Show the location of the initial capture sites
cap <- structure(c(-627921, -428387,
                   19980, -926623),
                 .Dim = c(2L, 2L))
points(cap, bg="yellow", cex=2, pch=21)
box()
```


The red points correspond to the recapture locations of animals initially captured in Abberton Reservoir (yellow point in England), and the black points correspond to the recapture locations of animals initially captured in Camargue (yellow point in Southern France).

We first rotated the coordinate system, as described in Guillemain et al. We show below the distribution of the recaptures in the new coordinate system (without the background map). This new coordinate system was chosen subjectively. See section 2.1 for a sensitivity analysis of this choice:

```
## The rotation matrix
M <- recteal$rotationMatrix
## The coordinates of the recaptures in the
## new coordinate system
co <- as.matrix(recteal$recaptures[,c("x","y")])
co2 <- co%*%M

## plot the rotated coordinates
par(mar=c(0.1,0.1,0.1,0.1))
plot(co2, col=recteal$recaptures$Abberton+1,
     axes=FALSE)
```

```r
box()
```



This figure corresponds to the figure 1B in Guillemain et al.

After this rotation, we prepare the data for the model fit. For each recapture characterized by a coordinate v_1 in the rotated coordinate system, we calculated the vector of 22 B-spline functions:

$$\mathbf{s_n}(v_1) = \left\{ s_n^{(1)}(v_1), s_n^{(2)}(v_1), ..., s_n^{(22)}(v_1) \right\}^t$$

We use the package `splines` to calculate the value of the 22 functions of the B-spline basis (corresponding to the nodes stored in `recteal$knots`):

```r
library(splines)
spl <- splineDesign(recteal$knots, co2[,1], outer.ok = TRUE)
```

Finally, we store all the data in a list, which will be used to provide the data to the functions of the package `metroponcfs`:

```r
## Minimum and maximum coordinates in the new coordinate system
## limits of uniform prior distribution on the coefficients theta1 and
## theta2 associated to the B-spline basis
maxc <- 1073991
minc <- (-1271611)

## The list that will be used for the fit
lidat <- list(y=recteal$recaptures$Abberton, ## response
              spl=spl,                        ## spline basis
              xy=co2,                         ## original coordinates
              minc=minc,                      ## minimum and...
              maxc=maxc)                      ## ...maximum coordinates
```

1.1.2 Model description

The approach described in Guillemain et al. proposes to model the probability $[d_k = 1|\mathbf{u}_k]$ that an animal k was initially captured in Abberton Reservoir (i.e. $d_k = 1$) given the coordinates $\mathbf{v}_k$ of its recapture in the new coordinate system by:

$$\begin{aligned}
[d_k = 1|\mathbf{v}_k] \quad &= \quad \psi_{2\to 2} \times \{\delta \times [\boldsymbol{\theta}_1 \in \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \in \Theta_2(\mathbf{v}_k)] + [\boldsymbol{\theta}_1 \notin \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \in \Theta_2(\mathbf{v}_k)]\} + \\
&\quad\quad \psi_{2\to 1} \times \{(1-\delta) \times [\boldsymbol{\theta}_1 \in \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \in \Theta_2(\mathbf{v}_k)] + [\boldsymbol{\theta}_1 \in \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \notin \Theta_2(\mathbf{v}_k)]\}
\end{aligned}$$

With $\psi_{2\to 2}$ the probability that an individual belonging to the British subpopulation at recapture time was originally captured in Great-Britain, $\psi_{2\to 1}$ the probability that an individual belonging to the Mediterranean subpopulation at recapture time was originally captured in Great-Britain, and δ the probability that an individual recaptured in the area where the two flyways overlap belongs to the British population at recapture time. We now define the notations $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \Theta_1(\mathbf{v}_k), \Theta_2(\mathbf{v}_k)$.

The vector $\boldsymbol{\theta}_1$ is the vector containing the 22 coefficients (corresponding to the 22 B-spline basis functions) of the spline function defining the boundary of the Mediterranean flyway, and the vector $\boldsymbol{\theta}_2$ is the vector containing the 22 coefficients defining the boundary of the British flyway. Thus, for a given subpopulation j, the boundaries of the flyway in the rotated coordinate system were defined by the equation $v_2 = \mathbf{s}_n(v_1)^t\boldsymbol{\theta}_j$. The Mediterranean flyway was located above the boundary defined by in the rotated coordinate system, whereas the British flyway was located below the boundary defined by in this system. We defined the following functions:

$$\begin{aligned}
h_1(\mathbf{v}, \boldsymbol{\theta}_1) \quad &= \quad \mathbf{s_n}(v_1) - v_2 \\
h_2(\mathbf{v}, \boldsymbol{\theta}_2) \quad &= \quad v_2 - \mathbf{s_n}(v_1)
\end{aligned}$$

Then $\Theta_1(\mathbf{v})$ (resp. $\Theta_2(\mathbf{v})$) is the set of values of $\boldsymbol{\theta}_1$ (resp. $\boldsymbol{\theta_2}$) for which $h_1(\mathbf{v}, \boldsymbol{\theta}_1) > 0$ (resp. $h_2(\mathbf{v}, \boldsymbol{\theta}_2) > 0$), i.e. the set of values of $\boldsymbol{\theta}_1$ (resp. $\boldsymbol{\theta}_2$) which define a Mediterranean (resp. British) flyway including location $\mathbf{u}$.

We defined the following prior distributions for the parameters:

$$\begin{aligned}
\theta_{ij} \quad &\sim \quad \mathcal{U}(-1271611, 11073991) \\
\psi_{2\to 2} \quad &\sim \quad \mathcal{U}(0, 1) \\
\psi_{2\to 1} \quad &\sim \quad \mathcal{U}(0, 1) \\
\delta \quad &\sim \quad \mathcal{U}(0, 1)
\end{aligned}$$

1.1.3 Metropolis algorithm for the model fit

We used the functions of the package `metroponcfs` to fit this model. It is supposed in this section that the reader knows the Metropolis algorithm. An excellent introduction to this method is given in Geyer (2011).

Several technical details of the fitting procedure are given below:

- all the parameters were updated in turn
- The proposal distribution used for the probabilities $\psi_{2\to 2}$, $\psi_{2\to 1}$ and δ was a Gaussian distribution with mean 0 and standard deviation chosen after several trials of the algorithm (see below). These probabilities were logit-transformed prior to the update.
- After some preliminary runs, we realized that there were strong dependencies between some of the 22 coefficients in $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, so that we used a block-updating for some of them. More precisely:
  - For $\boldsymbol{\theta}_1$, we updated each coefficient in turn (with a univariate Gaussian proposal) for the coefficients $\boldsymbol{\theta}_{1.1}$ to $\boldsymbol{\theta}_{1.6}$, $\boldsymbol{\theta}_{1.9}$ and $\boldsymbol{\theta}_{1.10}$, $\boldsymbol{\theta}_{1.16}$ to $\boldsymbol{\theta}_{1.22}$. We used a block-updating mechanism (with a multinormal proposal) for the subvector containing the coefficients $\boldsymbol{\theta}_{1.7}$ and $\boldsymbol{\theta}_{1.8}$, the subvector containing the coefficients $\boldsymbol{\theta}_{1.11}$ and $\boldsymbol{\theta}_{1.12}$, and the subvector containing the coefficients $\boldsymbol{\theta}_{1.13}$ to $\boldsymbol{\theta}_{1.15}$.
  - For $\boldsymbol{\theta}_2$, we updated each coefficient in turn (with a univariate Gaussian proposal) for the coefficients $\boldsymbol{\theta}_{2.1}$ to $\boldsymbol{\theta}_{2.6}$, $\boldsymbol{\theta}_{2.11}$, and $\boldsymbol{\theta}_{2.16}$ to $\boldsymbol{\theta}_{2.22}$. We used a block-updating mechanism (with a multinormal proposal) for the subvector containing the coefficients $\boldsymbol{\theta}_{2.7}$ to $\boldsymbol{\theta}_{2.10}$, and for the subvector containing the coefficients $\boldsymbol{\theta}_{2.12}$ to $\boldsymbol{\theta}_{2.15}$.

Remark: the choice of the updating mechanism required a lot of tuning, and would certainly be different for another migratory species.

We used the functions from the `metroponcfs` package to implement the Metropolis algorithm. The variance of the univariate Gaussian distribution used as proposal distributions, as well as the covariance matrix of the multinormal distribution were chosen after several preliminary runs.

We programmed the function used to calculate the log-posterior distribution (up to a constant):

```r
logposteriorModelFlyways <- function(par, lidat, ctrl)
{
    ## Coefficients
    theta1 <- c(par$theta1a, par$theta1b, par$theta1c, par$theta1d, par$theta1e,
                par$theta1f)
    theta2 <- c(par$theta2a, par$theta2b, par$theta2c, par$theta2d, par$theta2e)

    theta2 <- c(par$theta2a, par$theta2b, par$theta2c, par$theta2d, par$theta2e)

    pa <- c(par$psi21,par$psi22,par$delta)

    ## Prior on the probabilities: logistic distribution on a logit scale
    ## (= uniform distribution on a probability scale)
    pri <- sum(dlogis(pa,location = 0, scale=1, log=TRUE))

    psi22 <- invlogit(par$psi22)
    psi21 <- invlogit(par$psi21)
    delta <- invlogit(par$delta)
    ## Prior distribution of theta
    if (any(theta1<(lidat$minc)))
        return(-Inf)
    if (any(theta2<(lidat$minc)))
        return(-Inf)
    if (any(theta1>(lidat$maxc)))
        return(-Inf)
    if (any(theta2>(lidat$maxc)))
        return(-Inf)

    ## Likelihood.
    p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
    p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
    p <- psi22*(delta*(p1*p2) + (1-p1)*p2)+
        psi21*((1-delta)*(p1*p2) + p1*(1-p2))

    ## Posterior distribution
    return(pri+sum(log(lidat$y*p+(1-lidat$y)*(1-p))))
}
```

We then use the function `findstartingValues()` to find 5 starting values for the 5 MCMC chains. See the help page of this function for its arguments:

```r
## There are 14 parameters. Initialization:
pin <- as.list(1:14)

## Names of the parameters
names(pin) <- c("theta1a", "theta1b", "theta1c", "theta1d",
                "theta1e", "theta1f", "theta2a", "theta2b",
                "theta2c", "theta2d", "theta2e", "psi22",
                "psi21", "delta")

## range (min-max) where to find the starting values
## parameters theta
for (i in 1:11)
    pin[[i]] <- c(lidat$minc, lidat$maxc)
## logit-probabilities
for (i in 12:14)
    pin[[i]] <- c(-10, 10)

## For vector parameters, the number of elements of the vector
nel <- list(theta1a=6, theta1b=2, theta1c=2, theta1d=2,
            theta1e=3, theta1f=7,
            theta2a=6, theta2b=4, theta2d=4, theta2e=7)

## find 5 starting values dispersed in the prior distribution
fs <- findStartingValues(pin, logposteriorModelFlyways, lidat,
                         multiple = nel,
                         method="dispersed", ndispersed = 5,
                         info = FALSE)
```

Finally, we define the objects required for the fit (see the help page of `GeneralSingleMetropolis` for further details on these elements):

```r
## The list of variance and covariance matrices used
lipum <- recteal$lipum
```

```
## generate default updating mechanism using the vector of initial values.
## Then change the default updating mechanism
## for the "blocks" in theta1 and theta2
listUpdating <- defaultListUGSM(fs[[1]])
listUpdating$theta1b <- "mns"
listUpdating$theta1d <- "mns"
listUpdating$theta1e <- "mns"
listUpdating$theta2b <- "mns"
listUpdating$theta2d <- "mns"
```

Finally, we launched the Metropolis algorithm. **WARNING**: the following code is very long (it took 14 days of calculation):

```
## Model fit
for (i in 1:5) {
    parinit <- fs[[i]]
    gsm <- GeneralSingleMetropolis(parinit, lidat=lidat,
                                   logposterior = logposteriorModelFlyways,
                                   lipum = lipum,
                                   listUpdating = listUpdating,
                                   nrepet = 5050000, thinPar = 5000, saveEvery = 100000,
                                   fileSave = paste0("sf-",i,"-backup.Rdata"))
}

## Load the model and store it in an object named rectealmodel
observe <- list()

for (i in 1:5) {
    ## load the data
    load(paste0("sf-",i,"-backup.Rdata"))
    ## increase the Burnin
    libackup <- increaseBurnin(libackup,50010)
    ## store the data
    observe[[i]] <- libackup
}

## create the object
rectealmodel <- do.call(c,observe)
```

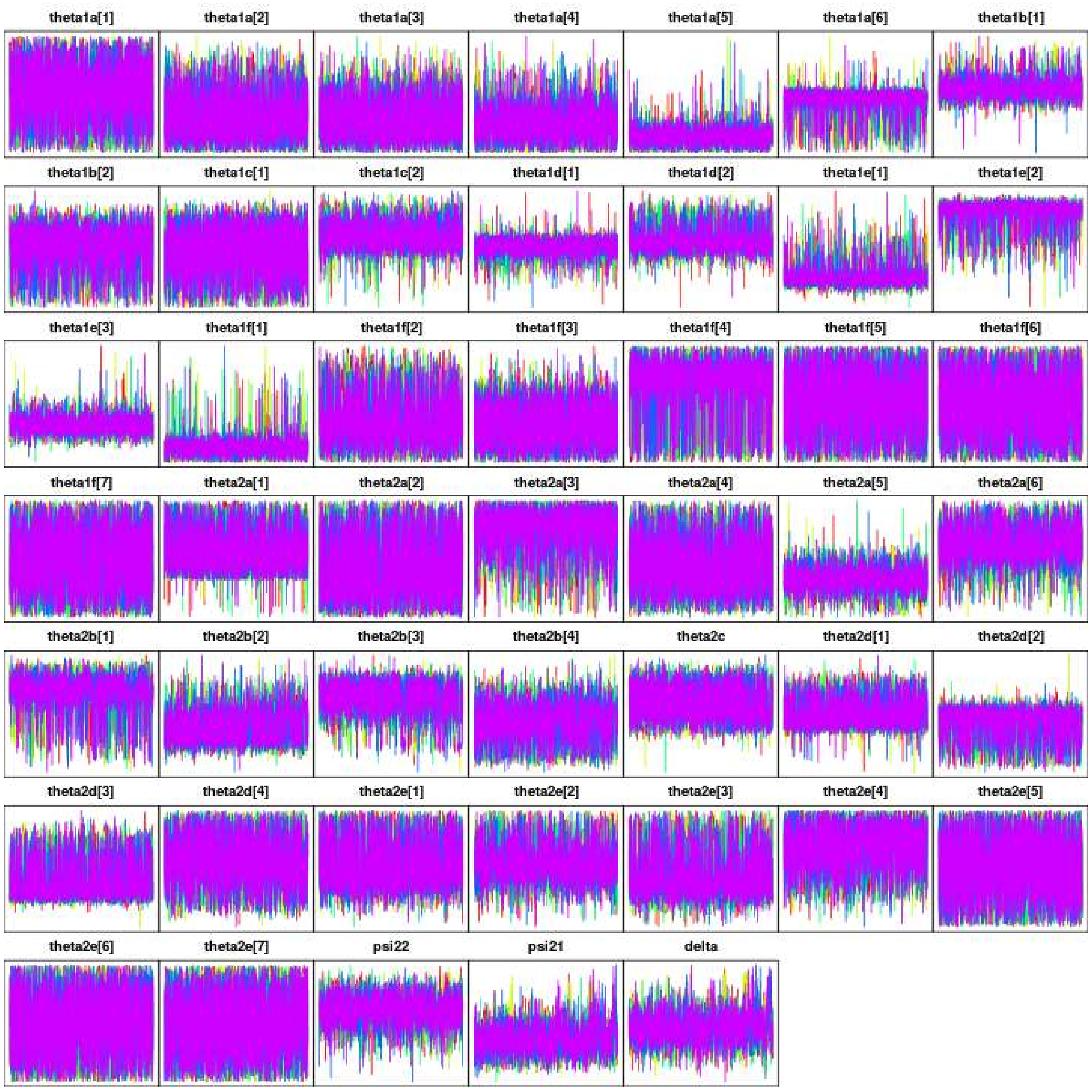
Remark: the code displayed above is very long, if executed as presented above. For the calculations carried out in the paper, Guillemain et al actually launched these calculations using parallel computing, with one MCMC chain per core on a multi-core machine (the model fit took 'only' three days). See Uyttendaele (2015) for additional details.

The results of this fit are stored in the dataset `rectealmodel`:

```
data(rectealmodel)
rectealmodel


## Object of class "CMM"
##
## Number of chains: 5
## 5 parameters or vectors of parameters in the model:
##  [1] "theta1a" "theta1b" "theta1c" "theta1d" "theta1e"
##  [6] "theta1f" "theta2a" "theta2b" "theta2c" "theta2d"
## [11] "theta2e" "psi22"   "psi21"   "delta"
##
## 5050010 iterations per chain (including 50010  burn-in samples)
## Thin parameters every: 5000 iterations
## Number of iterations stored for each chain: 1000
## Calculation took 1216174 seconds (total time over all chains)

## Transformation inverse-logit
rectealmodelb <- rectealmodel
for (i in 1:5) {
    rectealmodelb[[i]]$psi22[,1] <- invlogit(rectealmodelb[[i]]$psi22[,1])
    rectealmodelb[[i]]$psi21[,1] <- invlogit(rectealmodelb[[i]]$psi21[,1])
    rectealmodelb[[i]]$delta[,1] <- invlogit(rectealmodelb[[i]]$delta[,1])
}
```

We illustrate below the mixing properties of the algorithm:

```
plot(rectealmodelb)
```



And we present below the diagnostic of Gelman and Rubin (1992):

```
library(coda)
gelman.diag(tocoda(rectealmodelb))

## Potential scale reduction factors:
##
##             Point est. Upper C.I.
## theta1a[1]       1.000      1.003
## theta1a[2]       1.001      1.003
## theta1a[3]       1.000      1.002
## theta1a[4]       1.001      1.005
## theta1a[5]       1.000      1.000
## theta1a[6]       1.001      1.002
## theta1b[1]       1.010      1.019
## theta1b[2]       1.003      1.007
## theta1c[1]       1.000      1.002
## theta1c[2]       1.001      1.002
## theta1d[1]       1.001      1.002
## theta1d[2]       1.001      1.004
## theta1e[1]       1.001      1.003
## theta1e[2]       1.002      1.004
## theta1e[3]       1.001      1.004
## theta1f[1]       1.002      1.004
## theta1f[2]       0.999      1.000
## theta1f[3]       1.002      1.006
## theta1f[4]       1.003      1.008
## theta1f[5]       0.999      1.000
## theta1f[6]       1.001      1.004
## theta1f[7]       1.001      1.003
## theta2a[1]       1.001      1.003
## theta2a[2]       1.000      1.000
## theta2a[3]       1.001      1.003
## theta2a[4]       1.000      1.002
## theta2a[5]       1.001      1.003
## theta2a[6]       1.000      1.001
## theta2b[1]       1.021      1.043
## theta2b[2]       1.003      1.010
## theta2b[3]       1.014      1.033
## theta2b[4]       1.001      1.005
## theta2c          1.000      1.001
## theta2d[1]       0.999      0.999
## theta2d[2]       1.001      1.003
## theta2d[3]       1.001      1.005
## theta2d[4]       1.000      1.000
## theta2e[1]       1.000      1.001
## theta2e[2]       1.000      1.002
## theta2e[3]       1.001      1.004
## theta2e[4]       1.000      1.002
## theta2e[5]       1.000      1.002
## theta2e[6]       1.001      1.004
## theta2e[7]       0.999      1.000
## psi22            1.000      1.002
## psi21            1.010      1.020
## delta            1.008      1.020
##
## Multivariate psrf
```

```
##
## 1.03
```

Therefore, the mixing properties of the MCMC did not indicate any strong convergence issues.

1.2 Goodness of fit of the model: simulations

We used the approach recommended by Gelman and Meng (1996) to assess the goodness of fit of our model: we sampled 5000 values of the parameter vector from the posterior distribution derived from our fitted model, and for each parameter vector, we simulated a hypothetical replication of the dataset using our fitted model. We used the function below to simulate a vector of "marking site" corresponding to each observation:

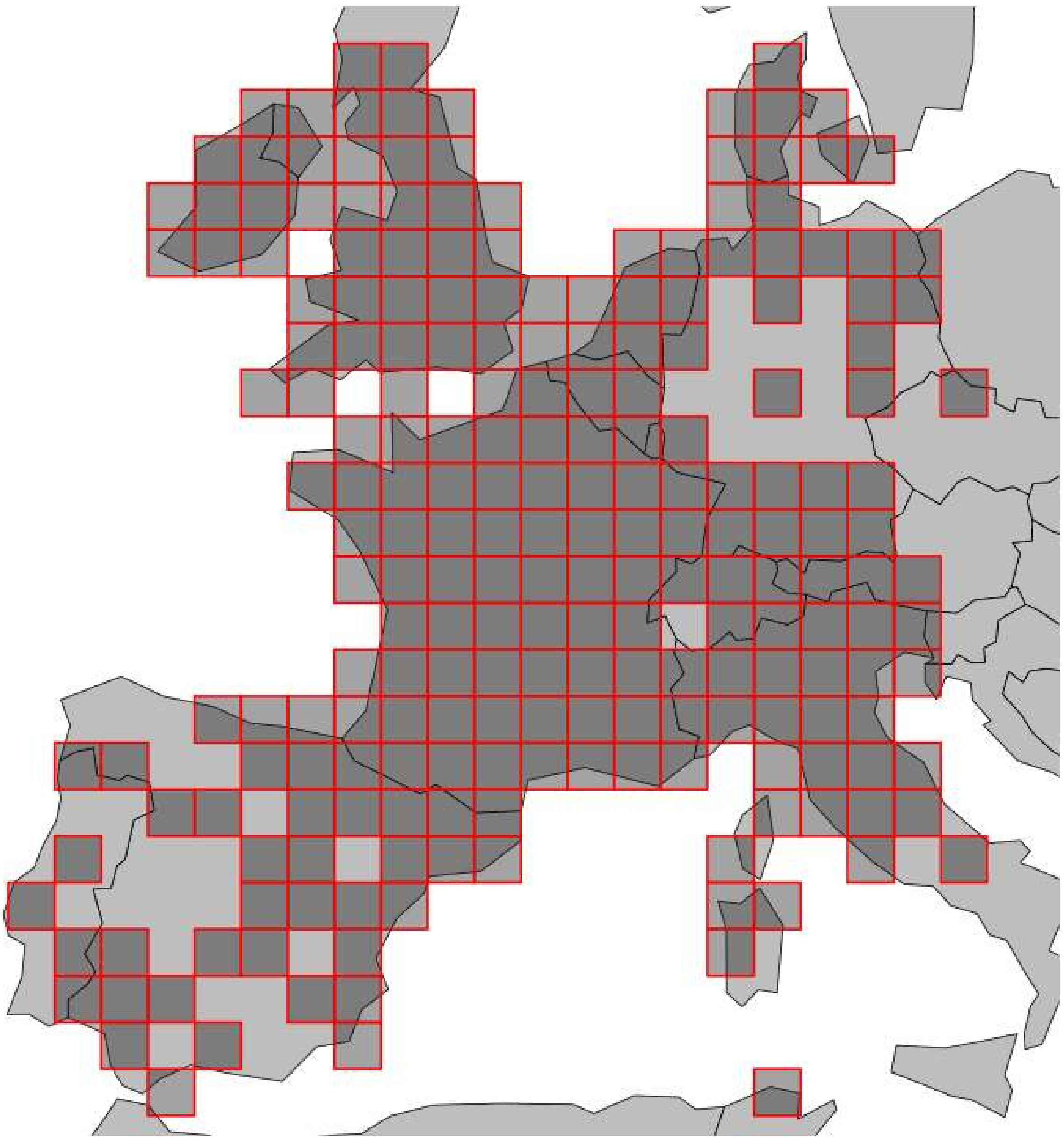
```r
## gs corresponds to the object containing the MCMC chains (i.e. rectealmodel)
## lidat contains the data used for the fit
## no describes which MCMC chain is used
## i describes which sampled vector should be used in this chain to simulate
## the hypothetical replication of the vector of marking sites
simudat <- function(gs, lidat, no, i)
{
    ## Coefficients associated to the functions in the B-spline basis
    ## Mediterranean Boundary
    theta1 <- c(gs[[no]]$theta1a[i,], gs[[no]]$theta1b[i,], gs[[no]]$theta1c[i,],
                gs[[no]]$theta1d[i,], gs[[no]]$theta1e[i,],
                gs[[no]]$theta1f[i,])
    ## British Boundary
    theta2 <- c(gs[[no]]$theta2a[i,], gs[[no]]$theta2b[i,], gs[[no]]$theta2c[i,],
                gs[[no]]$theta2d[i,], gs[[no]]$theta2e[i,])

    ## Probability parameters
    psi22 <- invlogit(gs[[no]]$psi22[i,1])
    psi21 <- invlogit(gs[[no]]$psi21[i,1])
    delta <- invlogit(gs[[no]]$delta[i,1])

    ## Whether, given the values of theta1 and theta2 (i.e. given the boundaries),
    ## each recapture is located in the british or mediterranean flyway
    p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
    p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)

    ## Probability that the recapture was originally captured as british
    p <- psi22*(delta*(p1*p2) + (1-p1)*p2)+
        psi21*((1-delta)*(p1*p2) + p1*(1-p2))

    ## Sample randomly the marking site
    u <- runif(length(p))
    aa <- as.numeric(u<=p)

    ## Result
    return(aa)
}
```

We then used this function to simulate replications of the dataset, for each parameter vector in `rectealmodel`, and combine them in a matrix:

```r
## list to store the simulated datasets
simus <- list()

## Loop to simulate the datasets
k <- 1
for (no in 1:length(rectealmodel)) {
    for (i in 1:nrow(rectealmodel[[1]][[1]])) {
        simus[[k]] <- simudat(rectealmodel, lidat, no, i)
        k <- k+1
    }
}
simus <- do.call(cbind,simus)
```

We then compared summary statistics calculated on the observed data to the distribution of these summary statistics calculated on the simulated datasets. More specifically, we discretized the European land into quadrats of a grid with a resolution of 100 × 100 km. For each quadrat containing at least one recapture (i.e. each quadrat in a set of 221 quadrats), we compared the observed proportion of birds initially ringed in Britain among the birds recaptured in this quadrat, with the distribution of simulated values for this statistic. We present below a map of the quadrats used for this test:

```r
## Load the packages sp and adehabitatMC
library(sp)
library(adehabitatMA)

## transform the recaptures into SpatialPoints and
## generate the grid to count the recaptures
xy <- SpatialPoints(recteal$recaptures[,c("x","y")])
asc <- ascgen(xy, cellsize = 100000)

## keep only the quadrats containing at least one recapture
asc <- asc[asc[[1]]>0,]

## fill the map with a sequence of integers
## (these integers will be used as quadrat ID in the following)
asc[[1]] <- 1:length(asc[[1]])

## Plot:
par(mar=c(0.1,0.1,0.1,0.1))

## Plot the map of Europe
plot(recteal$recaptures[,c("x","y")], ty="n", asp=1, axes = FALSE)
plot(ma,add=TRUE, col="grey")

## Plot the polygons corresponding to the quadrats
coo <- coordinates(asc)
for (i in 1:nrow(asc)) {
    x <- coo[i,1]
    y <- coo[i,2]
    x <- c(x-50000, x+50000, x+50000, x-50000)
    y <- c(y-50000, y-50000, y+50000, y+50000)
    polygon(x,y, col=grey(0.1, 0.4), lwd=2, border="red")
}
```



We calculated the proportion of quadrats for which the prediction was in the 95% credible intervals:

```
## find the quadrat ID corresponding to each recapture
id <- (xy%over%asc)[,1]

## Calculate the observed proportion of English birds
## in each quadrat
pr <- tapply(lidat$y,id,mean)

## Calculate this proportion for the simulated datasets
la <- do.call(cbind, lapply(1:ncol(simus),
                            function(i) {
                                tapply(simus[,i],id,mean)
                            }))

## The 95% credible interval
qu <- apply(la,1,function(x) quantile(x,c(0.025,0.975)))

## Proportion of quadrat where the observed value was in
## the credible interval
mean(pr>=qu[1,]&pr<=qu[2,])

## [1] 0.9773756
```

Thus 98% of the quadrats for which the observed value was in the 95% credible interval, the goodness of fit of our model was correct.

Remark: in this example, we used the slightly altered dataset to calculate this proportion. In the paper, we used the actual dataset, and obtained a proportion of 99% of quadrats where the credible interval contains the true value. Note that if we used 300 $\times$ 300 km quadrats instead of 100 $\times$ 100 km, we would have obtained a similar proportion (98% of the quadrats where the credible interval contains the true value). The interested reader is encouraged to try the code above with quadrats of different sizes to assess this goodness of fit.

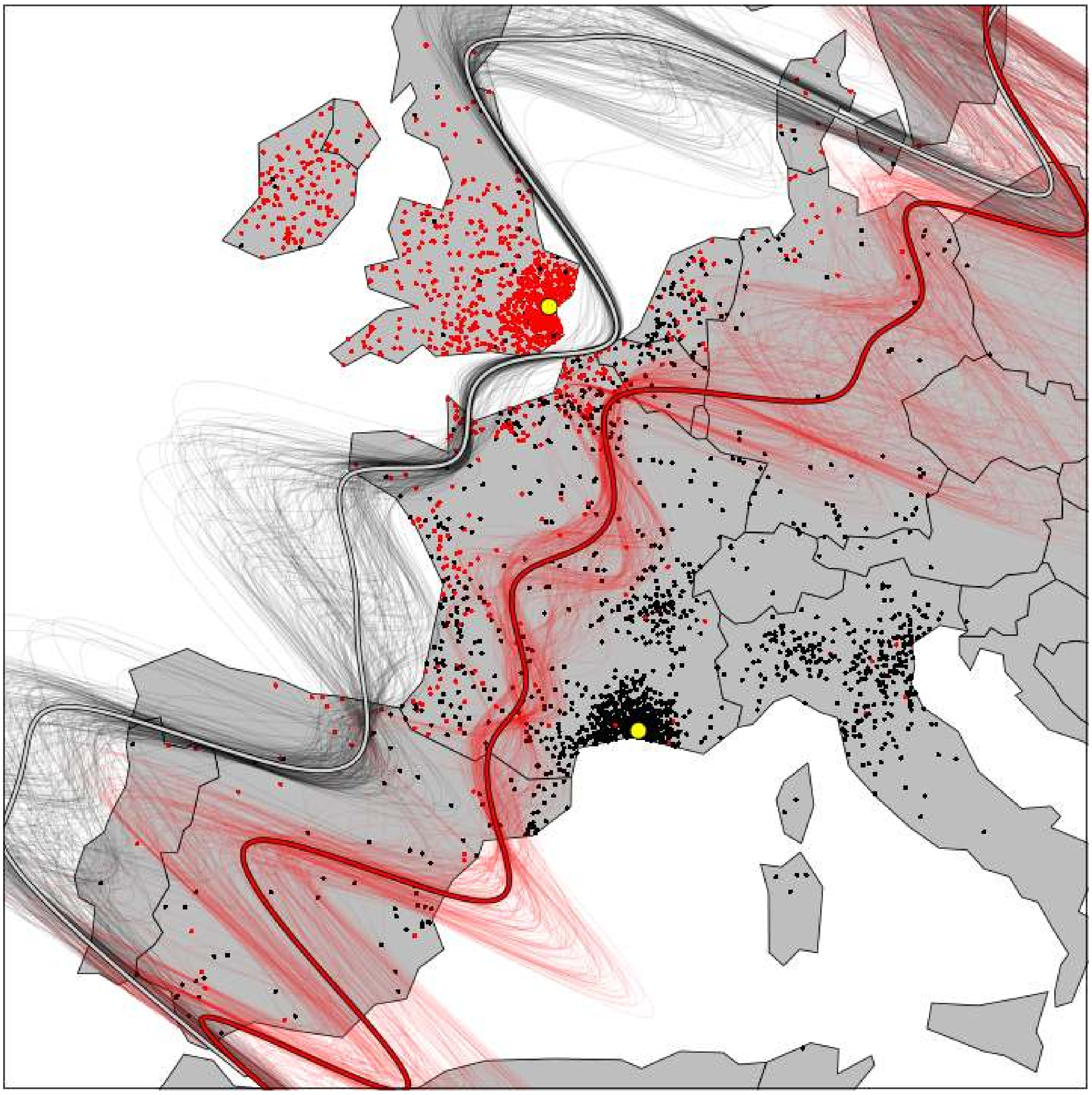
1.3 Model interpretation and other results

1.3.1 Visual display of the boundaries

We extract the values of $\boldsymbol{\theta_1}^{(r)}$ and $\boldsymbol{\theta_2}^{(r)}$ generated at the iteration r by the Metropolis algorithm and store them in a matrix. Every vector generated by the algorithm corresponds to one possible boundary for a flyway, drawn from the posterior distribution. We represent the distribution of boundaries for the two flyways below:

```
## The MCMC "splitted" the parameter vectors theta1 and theta2
## into subcomponents. The function foo "rebinds" these subcomponents
## into matrices.
foo <- function(mod)
{
    theta1 <- cbind(mod$theta1a,mod$theta1b,mod$theta1c,
                    mod$theta1d,mod$theta1e, mod$theta1f)
    theta2 <- cbind(mod$theta2a,mod$theta2b,mod$theta2c,
                    mod$theta2d,mod$theta2e)
    return(list(theta1=theta1, theta2=theta2))
}

## Matrices theta1 and theta2 containing, in each row, a vector generated
## by an iteration of the MCMC
theta1 <- do.call(rbind, lapply(1:length(rectealmodelb),
                                function(i) foo(rectealmodelb[i])$theta1))
theta2 <- do.call(rbind, lapply(1:length(rectealmodelb),
                                function(i) foo(rectealmodelb[i])$theta2))

## plot the spatial distribution of the recapture (same code as before):
par(mar=c(0.1,0.1,0.1,0.1))
plot(recteal$recaptures[,c("x","y")], ty="n", asp=1, axes = FALSE)
plot(ma,add=TRUE, col="grey")
points(recteal$recaptures[,c("x","y")],
       col=recteal$recaptures$Abberton+1,
       pch=16, cex=0.6)
cap <- structure(c(-627921, -428387,
                   19980, -926623),
                 .Dim = c(2L, 2L))
points(cap, bg="yellow", cex=2, pch=21)
```

```r
## spline basis for a sequence of points in the new coordinate system
xtest <- seq(min(recteal$knots), max(recteal$knots), length=1000)
spl2 <- splineDesign(recteal$knots, xtest, outer.ok = TRUE)

## simulated boundaries (we sample here 200 values to keep the
## time to compile this vignette reasonnable, but the reader
## can plot all the boundaries if they want to see all the data)
tovisualize <- sample(1:nrow(theta1), 200)
tmp <- sapply(tovisualize, function(i) {
    mo <- theta1[i,]
    gg <- spl2%*%mo
    ## revert to original coordinate system
    front1 <- cbind(gg, xtest)%*%recteal$inverseRotationMatrix
    lines(front1, col=rgb(0.0,0.0,0,0.1))
})
tmp <- sapply(tovisualize, function(i) {
    mo <- theta2[i,]
    gg2 <- spl2%*%mo
    ## revert to original coordinate system
    front2 <- cbind(gg2, xtest)%*%recteal$inverseRotationMatrix
    lines(front2, col=rgb(1,0,0,0.1))
})

## Median boundaries
mo1 <- apply(theta1,2,median)
mo2 <- apply(theta2,2,median)
gg <- spl2%*%mo1
gg2 <- spl2%*%mo2
front1 <- cbind(gg, xtest)%*%recteal$inverseRotationMatrix
front2 <- cbind(gg2, xtest)%*%recteal$inverseRotationMatrix
lines(front1, col="black", lwd=5)
lines(front1, col="lightgrey", lwd=3)
lines(front2, col="black", lwd=5)
lines(front2, col="red", lwd=3)
box()
```



Remark: to keep the time to compile this vignette reasonnable, we plot here only a subsample of the boundaries simulated by MCMC. However, the reader is encouraged to plot all of them by replacing the vector `tovisualize` in the previous code with:

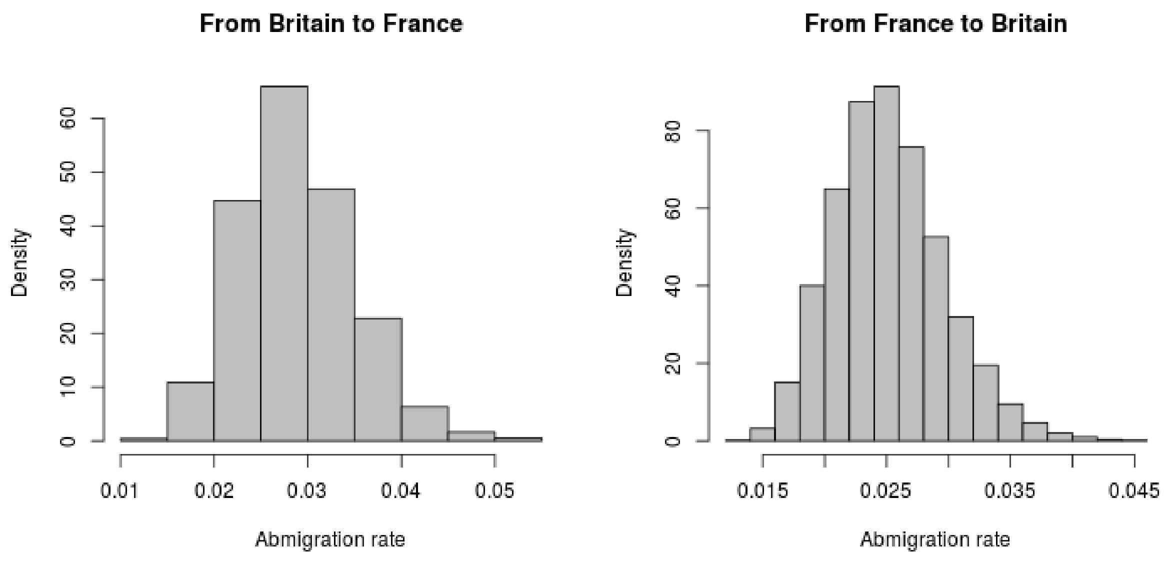
```
tovisualize <- 1:nrow(theta1)
```

1.3.2 Abmigration rates

We now calculate abmigation rates. We use equation (4) of Guillemain et al.:

```
## values generated by the MCMC for psi22, psi21 and delta
psi22 <- unlist(lapply(rectealmodelb, function(x) x$psi22))
psi21 <- unlist(lapply(rectealmodelb, function(x) x$psi21))
delta <- unlist(lapply(rectealmodelb, function(x) x$delta))

## corresponding values for the two probabilities of abmigration
## use of equation (4)
f12 <- 1-(psi22*delta)/(psi22*delta + psi21*(1-delta))
```

```r
f21 <- ((1-psi22)*delta)/(1-(psi22*delta + psi21*(1-delta)))

## Posterior distribution of these abmigration rate
par(mfrow = c(1,2))
hist(f12, xlab="Abmigration rate", probability = TRUE,
     main="From Britain to France", col="grey")
hist(f21, xlab="Abmigration rate", probability = TRUE,
     main="From France to Britain", col="grey")
```



1.3.3 Distribution of abmigrating animals

We show here how we implemented the test of the random spatial distribution of abmigrating animals in their new population. We used the bivariate K_{12} function with a random labelling approach to test this hypothesis (see Guillemain et al.).

Abmigrating animals were identified by the following approach:

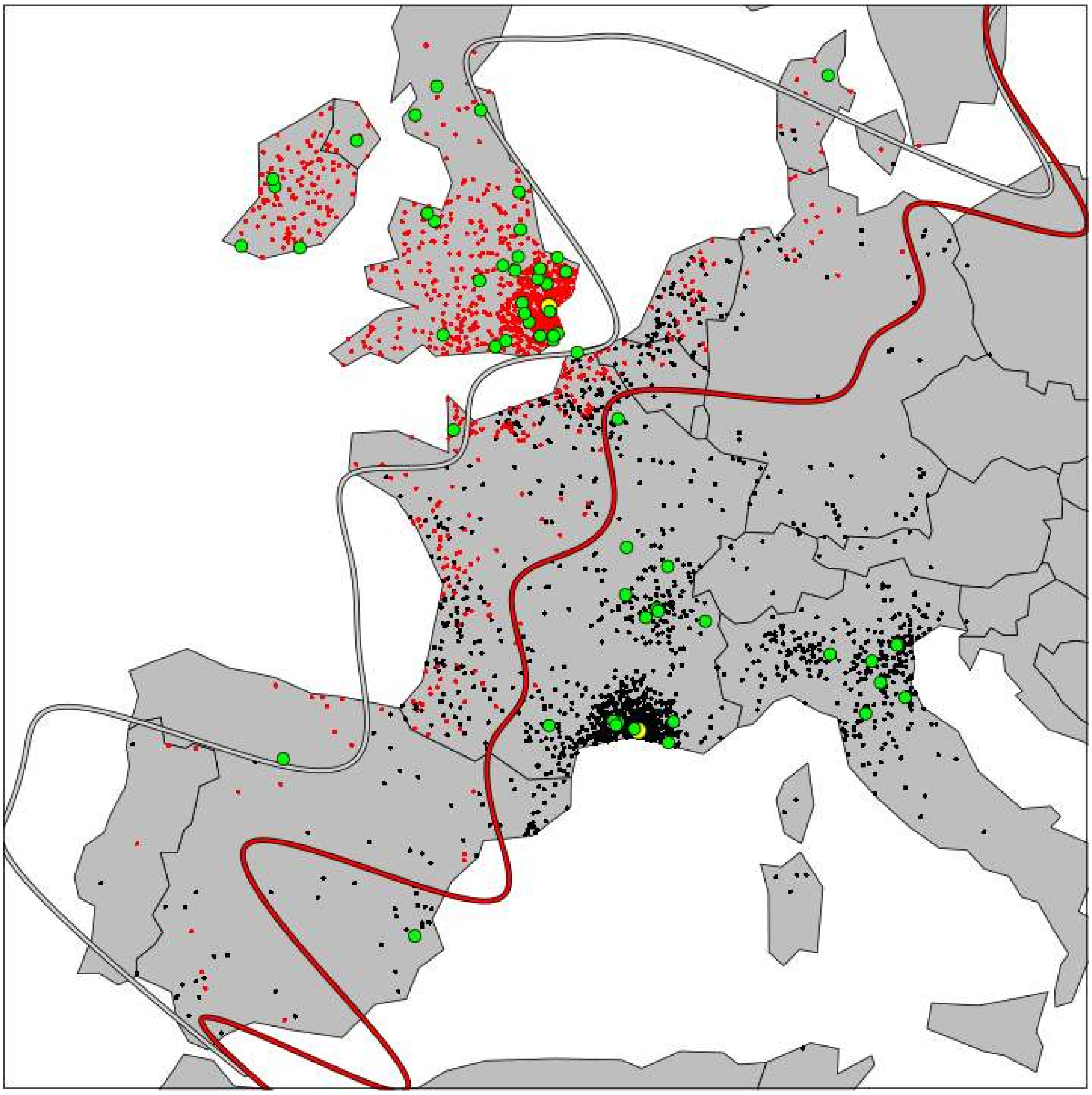
- Each generated vector $\boldsymbol{\theta}_1^{(r)}$ (resp. $\boldsymbol{\theta}_2^{(r)}$) corresponds to a possible boundary of the Mediterranean (resp. British) flyway, drawn from the posterior distribution. Therefore, for every recapture and every pair of generated vectors $\boldsymbol{\theta}_1^{(r)}$ and $\boldsymbol{\theta}_2^{(r)}$, we can determine if the recapture occurred in the British and/or Mediterranean Flyway. Therefore, for every pair of generated vectors $\boldsymbol{\theta}_1^{(r)}$ and $\boldsymbol{\theta}_2^{(r)}$, we can identify the set of animals captured in one flyway and recaptured in the other (i.e. abmigrating animals).

- Therefore, for a given recapture, considering all the generated pairs of vectors $\boldsymbol{\theta}_1^{(r)}$ and $\boldsymbol{\theta}_2^{(r)}$ generated by the Metropolis algorithm, we can determine the probability that this recapture actually occurred in the same flyway as its original capture (and therefore the probability that this animal was abmigrating).

- We considered as abmigrating animals all animals for which this probability was greater than 2/3.

We display the abmigrating animals below (green points):

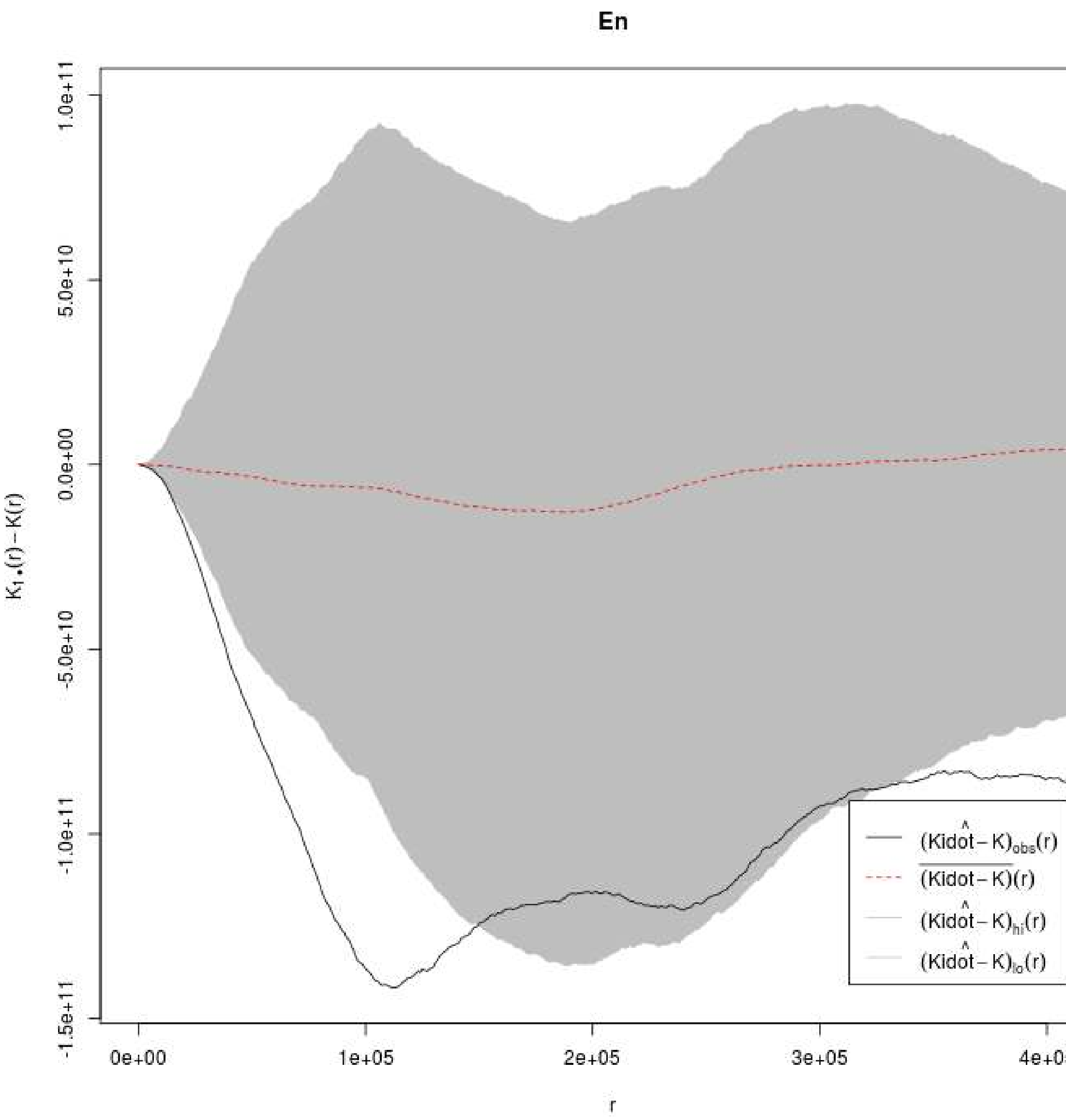
```r
## Identification of abmigrating France -> Britain
T1 <- do.call(cbind, lapply(1:nrow(theta1), function(i) {
    (co2[,2]-spl%*%theta1[i,])<0&recteal$recaptures$Abberton==0
}))
## Identification of abmigrating Britain -> France
T2 <- do.call(cbind, lapply(1:nrow(theta1), function(i) {
    (co2[,2]-spl%*%theta2[i,])>0&recteal$recaptures$Abberton==1
}))

## Abmigrating animals (defined as animals with
## probability > 2/3 to be abmigrators)
abmfb <- apply(T1,1,mean)> (2/3)
abmbf <- apply(T2,1,mean)> (2/3)


###################
##
## Plot

par(mar=c(0.1,0.1,0.1,0.1))
plot(recteal$recaptures[,c("x","y")], ty="n", asp=1, axes = FALSE)
plot(ma,add=TRUE, col="grey")
points(recteal$recaptures[,c("x","y")],
       col=recteal$recaptures$Abberton+1,
       pch=16, cex=0.6)
cap <- structure(c(-627921, -428387,
                   19980, -926623),
                 .Dim = c(2L, 2L))
points(cap, bg="yellow", cex=2, pch=21)
lines(front1, col="black", lwd=5)
lines(front1, col="lightgrey", lwd=3)
lines(front2, col="black", lwd=5)
lines(front2, col="red", lwd=3)

## Abmigrating animals:
points(recteal$recaptures[abmfb,c("x","y")], pch=21, bg="green", cex=1.5)
points(recteal$recaptures[abmbf,c("x","y")], pch=21, bg="green", cex=1.5)

box()
```



We provide below the code used to test the random distribution of the abmigrating animals in their new population. Because this vignette is compiled when the package is built, we carried out only 20 iterations of the test, but the reader is encouraged to increase this number to 99:

```
library(spatstat)

## We remove the points located in the overlap area
## Identification of flyways
FC <- do.call(cbind, lapply(1:nrow(theta1), function(i) {
    (co2[,2]-spl%*%theta1[i,])>0
}))
## Identification of abmigrating Britain -> France
FB <- do.call(cbind, lapply(1:nrow(theta1), function(i) {
    (co2[,2]-spl%*%theta2[i,])<0
}))

## keep only the recaptures located outside the overlap area.
ov <- apply(FC+FB<2,1,mean)>0.8
daov <- recteal$recaptures[ov,c("x","y")]
```

```
abm <- as.numeric((abmfb+abmbf)>0)[ov]

## Test the random distribution
po <- ppp(jitter(coordinates(daov)[,1]), jitter(coordinates(daov)[,2]),
         window=owin(range(coordinates(daov)[,1])+c(-1000,1000),
         range(coordinates(daov)[,2])+c(-1000,1000)),
         marks=factor(as.character(abm)))

Jdif <- function(X, ..., i) {
    Kidot <- Kdot(X, ..., i = i)
    K <- Kest(X, ...)
    dif <- eval.fv(Kidot - K)
    return(dif)
}

## increase nsim to at least 99 if you want to test this code
En <- envelope(po, Jdif, nsim = 20, i = "1", simulate = expression(rlabel(po)))

## Generating 20 simulations by evaluating expression  ...
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
## 16, 17, 18, 19,  20.
##
## Done.

## plot the result
plot(En)

##       lty col   key                  label
## obs     1   1   obs hat((Kidot-K))[obs](r)
## mmean   2   2 mmean       bar((Kidot-K))(r)
## hi      1   8    hi  hat((Kidot-K))[hi](r)
## lo      1   8    lo  hat((Kidot-K))[lo](r)
##                                                            meaning
## obs               observed value of K["1" ~ dot](r) - K(r) for data pattern
## mmean                sample mean of K["1" ~ dot](r) - K(r) from simulations
## hi    upper pointwise envelope of K["1" ~ dot](r) - K(r) from simulations
## lo    lower pointwise envelope of K["1" ~ dot](r) - K(r) from simulations
```

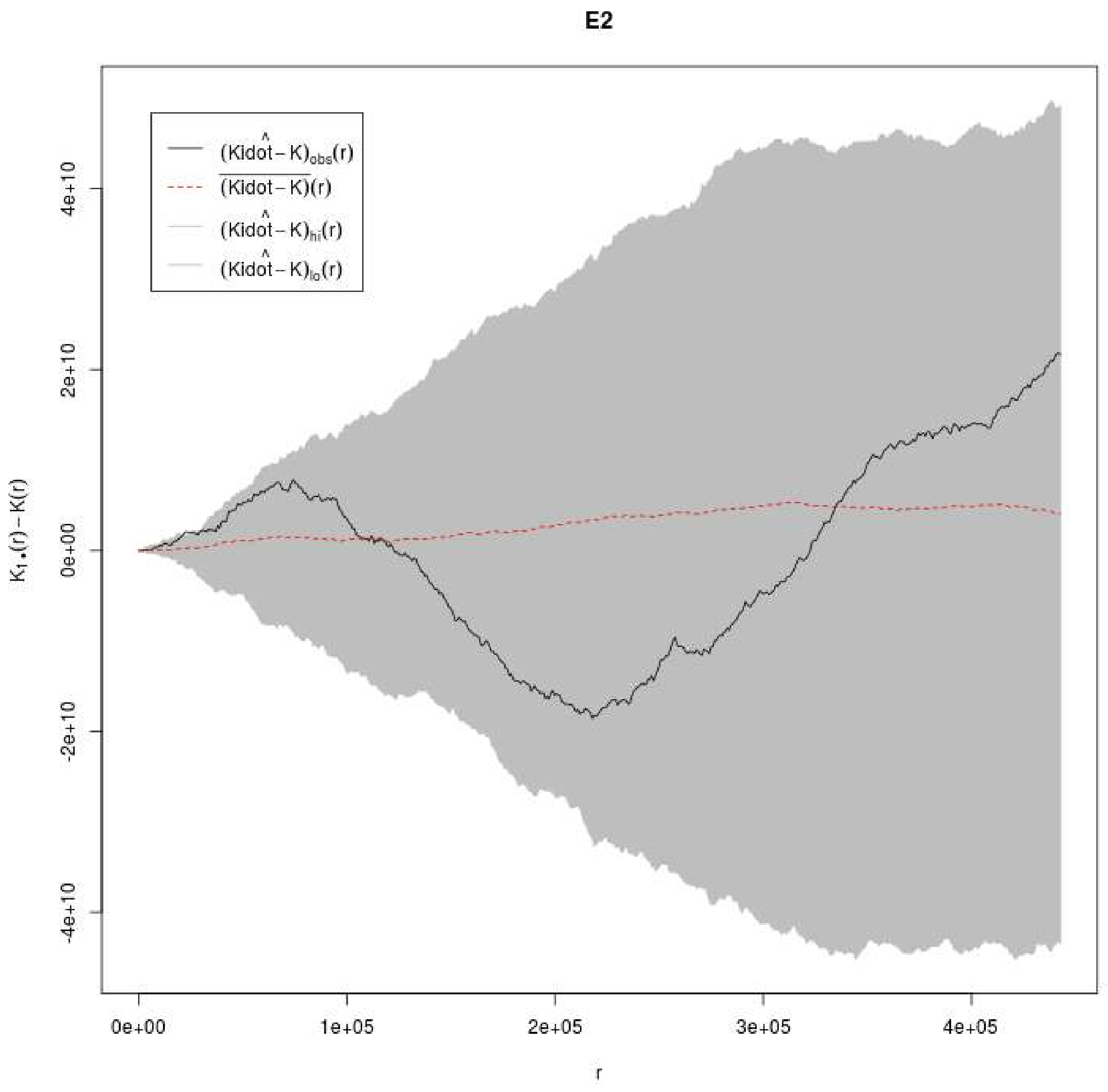


1.3.4 Random distribution of the two types of animals within the overlap area

As in the previous section, we provide below the R code used to test the random distribution of the animals of the two populations within the overlap area. As in the previous section, because this vignette is compiled when the package is built, we carried out only 20 iterations of the test, but the reader is encouraged to increase this number:

```
## relocations within the overlap area
ov <- apply(FC+FB>1,1,mean)>0.8
daov <- recteal$recaptures[ov,c("x","y")]
captori <- recteal$recaptures[ov,"Abberton"]

## Test of the random distribution
p2 <- ppp(jitter(coordinates(daov)[,1]), jitter(coordinates(daov)[,2]),
          window=owin(range(coordinates(daov)[,1])+c(-1000,1000),
          range(coordinates(daov)[,2])+c(-1000,1000)),
          marks=factor(as.character(captori)))
```

```r
## increase nsim to at least 99 if you want to test this code
E2 <- envelope(p2, Jdif, nsim = 20, i = "1", simulate = expression(rlabel(p2)))

## Generating 20 simulations by evaluating expression  ...
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
## 16, 17, 18, 19,  20.
##
## Done.

plot(E2)

##       lty col   key                  label
## obs     1   1   obs hat((Kidot-K))[obs](r)
## mmean   2   2 mmean      bar((Kidot-K))(r)
## hi      1   8    hi  hat((Kidot-K))[hi](r)
## lo      1   8    lo  hat((Kidot-K))[lo](r)
##                                                                  meaning
## obs             observed value of K["1" ~ dot](r) - K(r) for data pattern
## mmean              sample mean of K["1" ~ dot](r) - K(r) from simulations
## hi    upper pointwise envelope of K["1" ~ dot](r) - K(r) from simulations
## lo    lower pointwise envelope of K["1" ~ dot](r) - K(r) from simulations
```

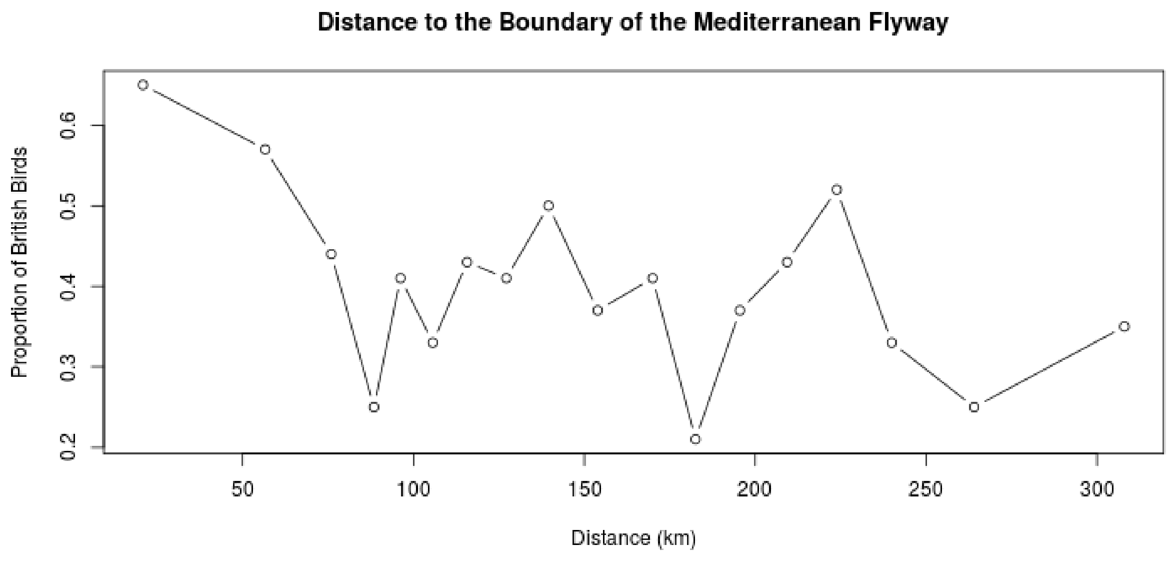


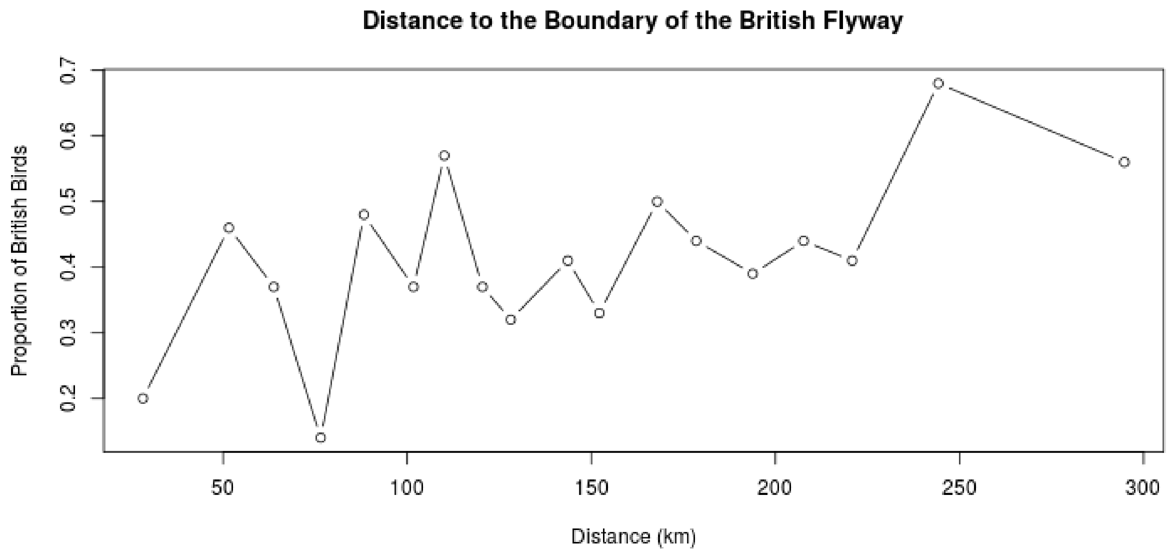
One referee of the paper suggested an additional test of the random distribution hypothesis: they proposed to test whether the proportion of birds initially ringed at a place – say, in Britain – was larger in the overlap area close to the boundary the closest to Britain than far from it. We carry out this test here.

We focused on the birds recaptured in the “overlap” area. We calculated the perpendicular distance between the recapture location and the two boundaries. We then defined 20 categories of distance to these boundaries and we calculated the proportion of British birds within each class:

```
## median distribution of the coefficients used to define the boundaries
mo1 <- apply(theta1,2,median)
mo2 <- apply(theta2,2,median)

## Calculation of the coordinates of the boundaries
xtest <- seq(min(recteal$knots), max(recteal$knots), length=1000)
spl2 <- splineDesign(recteal$knots, xtest, outer.ok = TRUE)
gg <- spl2%*%mo1
gg2 <- spl2%*%mo2
front1 <- cbind(gg, xtest)%*%recteal$inverseRotationMatrix
front2 <- cbind(gg2, xtest)%*%recteal$inverseRotationMatrix
```

```r
## Conversion of the recaptures as spatialPoints
library(sp)
library(rgeos)
xy <- SpatialPoints(as.matrix(recteal$recaptures[,c("x","y")]))

## Conversion of the boundaries as Spatial Lines
sl1 <- SpatialLines(list(Lines(Line(front1), ID="fr1")))
sl2 <- SpatialLines(list(Lines(Line(front2), ID="fr2")))

## Distance between each recapture and each boundary
gd1 <- as.vector(gDistance(xy, sl1,byid=TRUE))/1000
gd2 <- as.vector(gDistance(xy, sl2,byid=TRUE))/1000

## Keep only the recaptures located in the overlap area
xyr <- coordinates(xy)%*%recteal$rotationMatrix
spl3 <- splineDesign(recteal$knots, xyr[,1] , outer.ok = TRUE)
sup1 <- xyr[,2]-spl3%*%mo1>0
sup2 <- xyr[,2]-spl3%*%mo2>0
oo <- sup1+sup2
gd1o <- gd1[oo==1]
gd2o <- gd2[oo==1]
repo <- recteal$recaptures$Abberton[oo==1]

## For each boundary, cut the vector of distances in 20 classes
cu1 <- cut(gd1o,c(-10,quantile(gd1o, seq(0.1,0.9,by=0.05)), 1e10))
## Median distance in each class:
mdi1 <- tapply(gd1o, cu1, median)
moy1 <- round(tapply(repo, cu1, mean),2)
cu2 <- cut(gd2o,c(-10,quantile(gd2o, seq(0.1,0.9,by=0.05)), 1e10))
## Median distance in each class:
mdi2 <- tapply(gd2o, cu2, median)
moy2 <- round(tapply(repo, cu2, mean),2)

## Plot the results
par(mfrow = c(2,1))
plot(mdi1, moy1, ty="b", xlab="Distance (km)",
     ylab = "Proportion of British Birds",
     main = "Distance to the Boundary of the Mediterranean Flyway")
plot(mdi2, moy2, ty="b", xlab="Distance (km)",
     ylab = "Proportion of British Birds",
     main = "Distance to the Boundary of the British Flyway")
```





We can see that the proportion of British birds does not vary a lot according to the distance to the boundaries. Note however that there is a slight increase in proportion of birds ringed from the closer ringing location at the immediate proximity of one boundary (and a "mirror" slight decrease at close to the opposite boundary, at a distance close to 300 km), due to the uncertainty in the exact boundary location. This proportion was constant elsewhere. This confirms the results that were found with the K-function.

1.3.5 Change in time of the parameters

Finally, we fitted another model where the parameters $\psi_{22}(t), \psi_{21}(t)$ and $\delta(t)$ were supposed to vary according to the number t of years separating the initial capture and the recapture events. We modify accordingly the parameters required by `GeneralSingleMetropolis` below:

```r
## The list of variance and covariance matrices used
lipum <- recteal$lipum
lipum$psi22 <- rep(lipum$psi22,4)
lipum$psi21 <- rep(lipum$psi21,4)
lipum$delta <- rep(lipum$delta,4)
```

```
#############
## Changing the updating mechanism

## psi22
listUpdating$psi22 <- "mis"

## psi21
listUpdating$psi21 <- "mis"

## delta
listUpdating$delta <- "mis"

## add the year in the dataset
lidat$year <- as.numeric(substr(recteal$recaptures$date, 3,3))
```

We also change the log-posterior of the model:

```
logposteriorModelFlywaysTime <- function(par, lidat, ctrl)
{
    ## Coefficients of the B-spline functions
    theta1 <- c(par$theta1a, par$theta1b, par$theta1c, par$theta1d, par$theta1e,
                par$theta1f)
    theta2 <- c(par$theta2a, par$theta2b, par$theta2c, par$theta2d, par$theta2e)

    ## The probability parameters
    pa <- c(par$psi21,par$psi22,par$delta)

    ## Prior on probabilities (logistic distribution on a logistic scale
    ## = uniform distribution on a probability scale)
    pri <- sum(dlogis(pa,location = 0, scale=1, log=TRUE))

    ## back to probability scale
    psi22 <- invlogit(par$psi22)
    psi21 <- invlogit(par$psi21)
    delta <- invlogit(par$delta)

    ## Prior distribution on theta1 and theta2
    if (any(theta1<(lidat$minc)))
        return(-Inf)
    if (any(theta2<(lidat$minc)))
        return(-Inf)
    if (any(theta1>(lidat$maxc)))
        return(-Inf)
    if (any(theta2>(lidat$maxc)))
        return(-Inf)

    ## Likelihood.
    p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
    p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
    p <- psi22[lidat$year]*(delta[lidat$year]*(p1*p2) + (1-p1)*p2)+
        psi21[lidat$year]*((1-delta[lidat$year])*(p1*p2) + p1*(1-p2))

    ## Posterior distribution
    return(pri+sum(log(lidat$y*p+(1-lidat$y)*(1-p))))
}
```

We generate five starting values for this model:

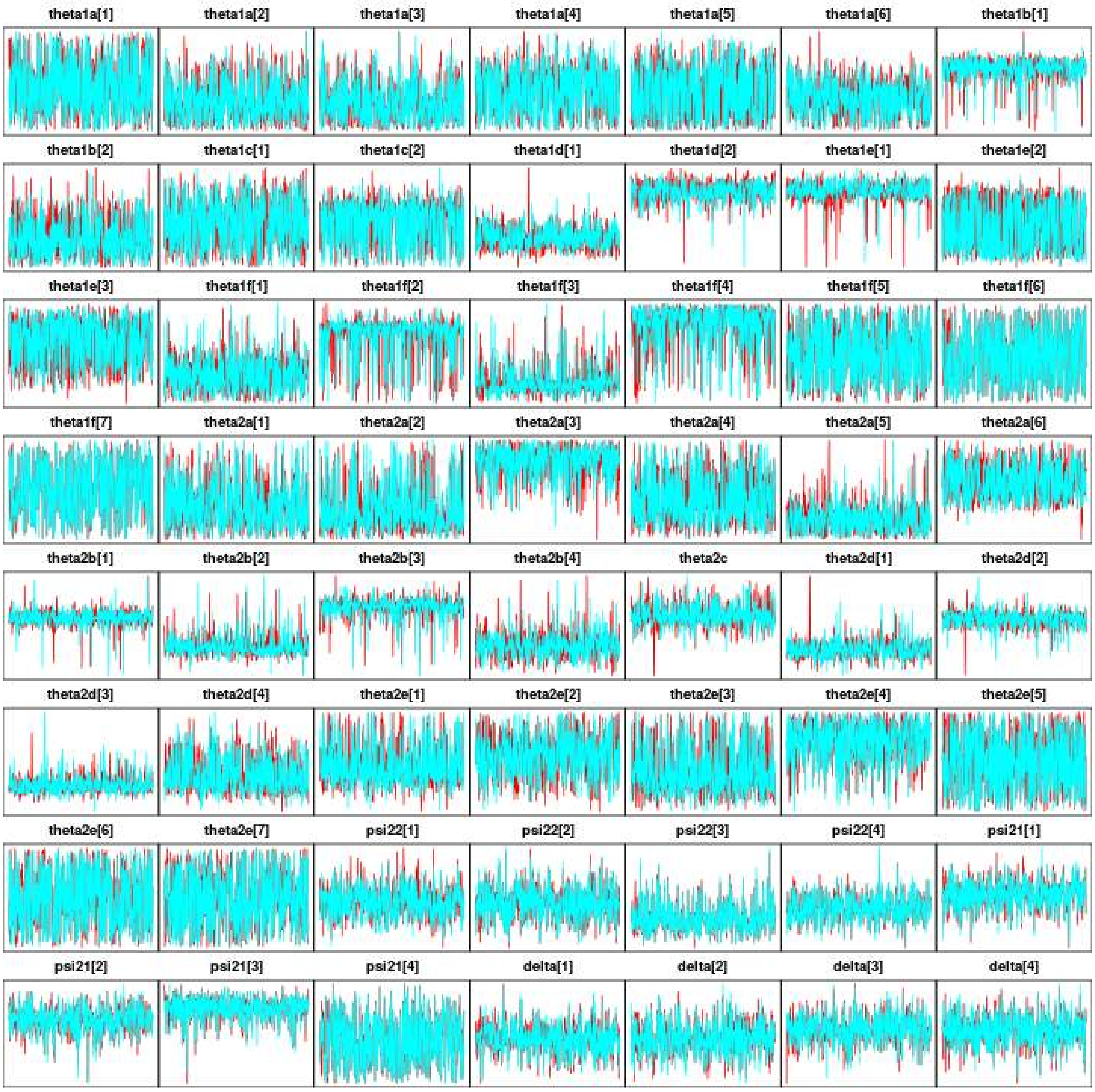
```
nelt <- list(theta1a=6, theta1b=2, theta1c=2, theta1d=2, theta1e=3, theta1f=7,
            theta2a=6, theta2b=4, theta2d=4, theta2e=7, psi22=4, psi21 = 4, delta = 4)
fs <- findStartingValues(pin, logposteriorModelFlywaysTime, lidat, multiple = nelt,
                         method="dispersed", ndispersed = 5, info=FALSE)
```

And finally, we launched the Metropolis algorithm. As before, the following code is very long (it also took a cumulated time of 14 days of calculation, which we reduced to three days by parallelizing the calculations):

```
for (i in 1:5) {
    parinit <- fs[[i]]
    gsm <- GeneralSingleMetropolis(parinit, lidat=lidat,
                                   logposterior = logposteriorModelFlywaysTime,
                                   lipum = lipum,
                                   listUpdating = listUpdating,
                                   nrepet = 5050000, thinPar = 5000, saveEvery = 100000,
                                   fileSave = paste0("sft-",i,"-backup.Rdata"))
}


## Load the model and store it in an object named rectealmodel
observe <- list()

for (i in 1:5) {
    ## load the data
    load(paste0("sft-",i,"-backup.Rdata"))
    ## increase the Burnin
    libackup <- increaseBurnin(libackup,50010)
    ## store the data
    observe[[i]] <- libackup
}

## generates the MCMC object
rectealmodeltime <- do.call(c,observe)
```

The results are stored in the object `rectealmodeltime`. We can plot these chains:

```
data(rectealmodeltime)
plot(rectealmodeltime)
```



Finally, we compared the DIC of the model accounting for time changes in the parameters and the DIC of the original model:

```
## Here, the log-posterior is the same as the log-likelihood.
(DIC_notime <- DIC(rectealmodel, logposteriorModelFlyways))


## [1] 1900.149


(DIC_time <- DIC(rectealmodeltime, logposteriorModelFlywaysTime))


## [1] 1411.611
```

XXThe numbers are incorrect here (previous model with uniform prior on the logit scale). I am waiting for the calculations to end.

Note that here, the two DIC are similar, though the model accounting for time is slightly better. The DIC difference is too small to justify the additional fit of 9 additional parameters, so that the original model is better.

2 Sensitivity analysis

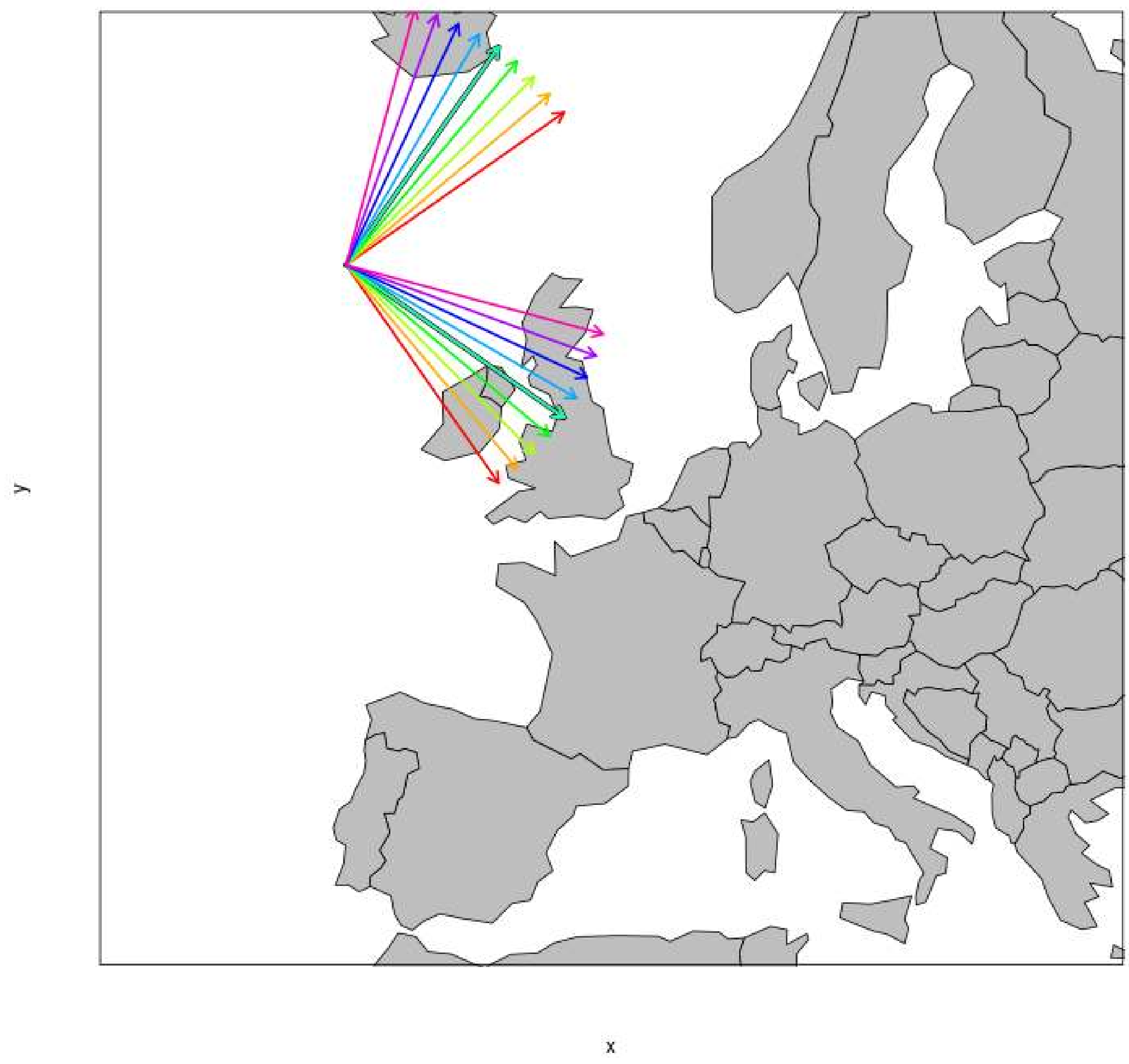
Our approach relies on two subjective choices (choice of a new coordinate system, number of functions in the B-spline basis), a referee of the paper rightly suggested to perform this sensitivity analysis, to demonstrate the robustness of our results to theses choices. We describe this analysis in this section.

2.1 The choice of the coordinate system

To model the boundary of the flyways, we fitted a regression spline model in a rotated coordinate system. We chose this coordinate system subjectively. We give, in the paper, several reasons why this choice should not strongly affect the results. In this section, we also try to fit our model with different choices for the new coordinate system, to assess its influence on our results.

We tried several rotation angles comprised between -20° and +20° in comparison to the chosen system (by steps of 5°), and we fitted our model in these different system. The figure below show the different rotations corresponding to these different systems:

```r
## x and y coordinates of the origin of the system
xx <- -1640187
yy <- 861165.9

## x and y coordinates of the vectors for this system
rx <- recteal$rotationMatrix[,1]
ry <- recteal$rotationMatrix[,2]

## Various m2 in comparison to the chosen system
## (angles between -20 and +20 by steps of 5£^\circ£)
## The chosen m2 is characterized by an angle of -35£^\circ£
anglori <- -35
## angles in radian by steps of 5£^\circ£
testesx <- pi*seq(anglori-20, anglori+20, by=5)/180
plot(rbind(c(xx,yy),c(xx+rx[1]*1000000,yy+rx[2]*1000000),
           recteal$recaptures[,c("x","y")]), ty="n", asp=1, axes = FALSE)
testesy <- testesx+pi/2

## adds the background map
plot(ma,add=TRUE, col="grey")
box()

## Original system
arrows(xx, yy, xx+rx[1]*1000000, yy+rx[2]*1000000, lwd = 4, length=0.1)
arrows(xx, yy, xx+rx[1]*1000000, yy+rx[2]*1000000, lwd = 2, col="white", length=0.1)
arrows(xx, yy, xx+ry[1]*1000000, yy+ry[2]*1000000, lwd = 4, length=0.1)
arrows(xx, yy, xx+ry[1]*1000000, yy+ry[2]*1000000, lwd = 2, col="white", length=0.1)

## the tested systems
for (whi in 1:length(testesx)) {
    rxb <- c(cos(testesx[whi]), sin(testesx[whi]))
    ryb <- c(cos(testesy[whi]), sin(testesy[whi]))
    arrows(xx, yy, xx+rxb[1]*1000000, yy+rxb[2]*1000000,
           col=rainbow(9)[whi], lwd = 2, length=0.1)
    arrows(xx, yy, xx+ryb[1]*1000000, yy+ryb[2]*1000000,
           col=rainbow(9)[whi], lwd = 2, length=0.1)
}
```



We fitted the model for every angle, using the following code. **WARNING:** the following code is very long (the cumulated time spent in these calculations was 29 days. We parallelized the calculations, so that they took “only” 7 days):

```
for (whi in 1:length(testesx)) {
    ## Rotation matrix corresponding to the tested angle
    M <- cbind(c(cos(testesy[whi]), sin(testesy[whi])),
               c(cos(testesx[whi]), sin(testesx[whi])))

    ## The coordinates of the recaptures in the
    ## new coordinate system
    co <- as.matrix(recteal$recaptures[,c("x","y")])
    co2 <- co%*%M

    ## Spline definition
    library(splines)
    spl <- splineDesign(recteal$knots, co2[,1], outer.ok = TRUE)

    ## theta2 associated to the B-spline basis
    minc <- min(co2[,2])-65.11
    maxc <- max(co2[,2])+162.4

    ## The list that will be used for the fit
    lidat <- list(y=recteal$recaptures$Abberton,
                  spl=spl,
                  xy=co2,
                  minc=minc,
                  maxc=maxc)

    ## To save some time, we fitted only one MCMC chain. Starting
    ## value:
    fs <- findStartingValues(pin, logposteriorModelFlyways, lidat, multiple = nel,
                             method="onebest")

    ## MCMC
    gsm1 <- GeneralSingleMetropolis(fs, lidat=lidat,
                                    logposterior = logposteriorModelFlyways,
                                    lipum = lipum,
                                    listUpdating = listUpdating,
                                    nrepet = 5050000, thinPar = 5000,
                                    saveEvery = 100000,
                                    fileSave = paste0("simulations-",whi,
                                    "-coosys.Rdata"))
}

## The list listeangles, storing the results
listeangle <- list()
for (i in 1:9) {
    load(paste0("simulations-",i,"-coosys.Rdata"))
    libackup <- increaseBurnin(libackup,50010)
    listeangle[[i]] <- libackup
}
```

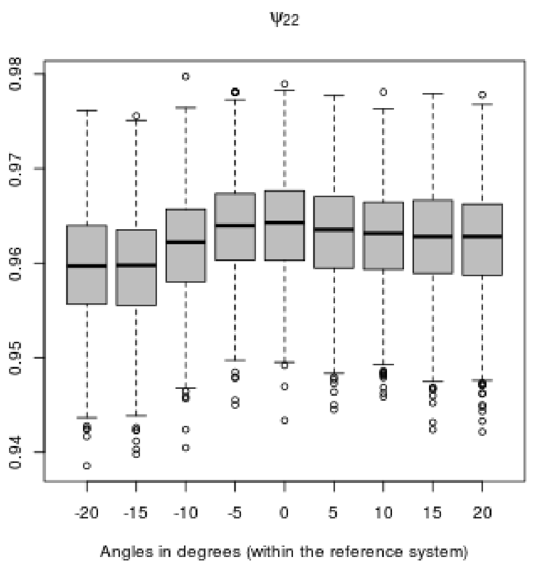
We stored the results of these calculations in the component `fit` of the dataset `sensitivityAngles`. Now load this dataset, and store the results in the list `listeangle`:

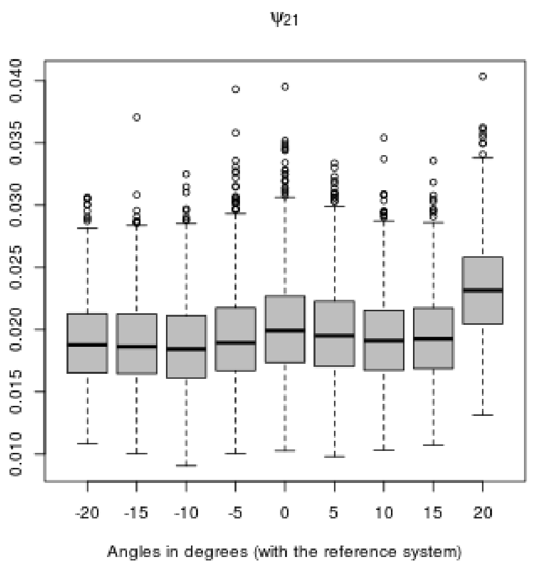
```
data(sensitivityAngles)
listeangle <- sensitivityAngles$fit
```

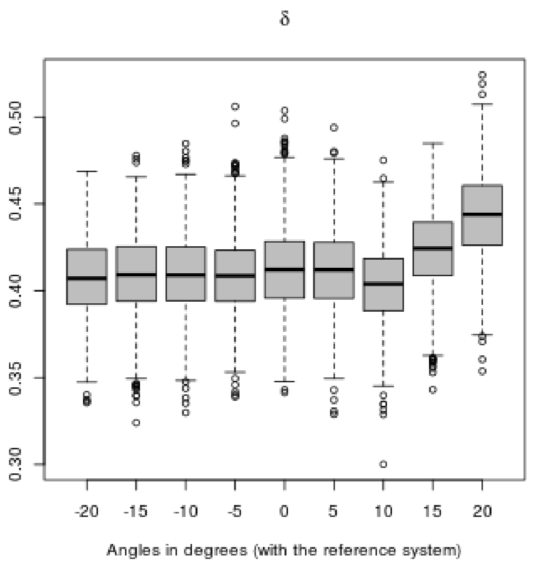
We can compare the posterior distribution of the parameters $\psi_{1\leftarrow 2}, \psi_{2\leftarrow 2}, \delta$ for the different choices:

```
par(mfrow = c(2,2))

## The angles, for x axis labelling
anglestestes <- seq(-20, +20, by=5)

## the posterior distribution of psi22 for different rotation angles
psi22 <- do.call(cbind,lapply(listeangle, function(x) invlogit(x$psi22)))
boxplot(psi22, axes=FALSE, col="grey",
        main=expression(psi[22]),
        xlab="Angles in degrees (within the reference system)")
axis(1, at=1:9, labels = anglestestes)
axis(2)
box()
```

```r
## the posterior distribution of psi21 for different rotation angles
psi21 <- do.call(cbind,lapply(listeangle, function(x) invlogit(x$psi21)))
boxplot(psi21, axes=FALSE, col="grey",
        main=expression(psi[21]),
        xlab="Angles in degrees (with the reference system)")
axis(1, at=1:9, labels = anglestestes)
axis(2)
box()

## the posterior distribution of delta for different rotation angles
delta <- do.call(cbind,lapply(listeangle, function(x) invlogit(x$delta)))
boxplot(delta, axes=FALSE, col="grey",
        main=expression(delta),
        xlab="Angles in degrees (with the reference system)")
axis(1, at=1:9, labels = anglestestes)
axis(2)
box()
```







The choice of the coordinate system did not have a large effect on the posterior distribution of these parameters, except when the rotation angle was large (+20°). As explained in the paper, it is not surprising that the results differ significantly when the rotation angle is strongly different from the reference system: a function in the reference system has a low probability to remain a function in a strongly rotated system (one x value can have multiple y values). Note that this difference happens only when the rotation angle is close to +20° in comparison to the reference system (i.e. only for positive rotation; there is no noticeable difference for a rotation angle of -20°): with a rotation angle of +20°, the rotated system is the closest to the latitude-longitude system among all the tested angles. This rotation angle would be a better choice if the flyways (and therefore, their boundaries) were oriented according to a North-South direction. We know that this is not the case: the birds migrate frow South-Western Europe to Western Russia/Finland, so we expected more a North-East/South-West orientation of the flyways (and therefore their boundaries).

Note that even when the rotation angle is of +20°, the mean of the posterior distribution of δ and $\psi_{2\leftarrow 1}$ (characterized by the highest difference) is not strongly different from the mean of the distribution with the reference system.

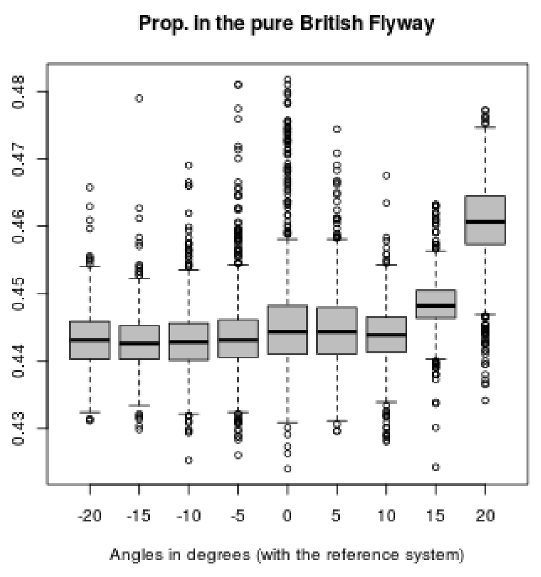
On the other hand, even the strongest negative rotation in comparison to our reference system (i.e. -20°) remains reasonable to model a flyway oriented according to a Northeast/Southwest direction; this explains why our results do not differ significantly from the results presented in our paper in such cases.

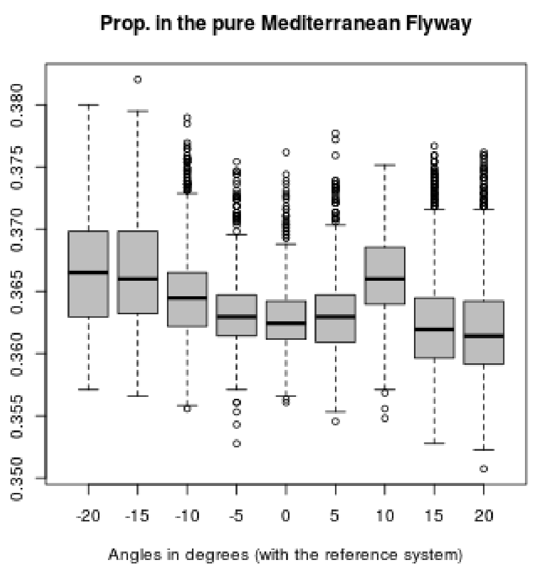
We also evaluated the effect of the choice of the rotation angle on the results with other metrics: for each possible choice of coordinate system, we obtained an estimation of the posterior distribution on the boundaries of the two flyways. We therefore calculated, for each possible system, the proportion of birds recaptured in each of the three areas (pure British flyway, pure Mediterranean flyway, overlap area). We present the code below (note that because we work on an altered dataset – see introduction – the result that the reader can obtain with this code will be slightly different from our results):

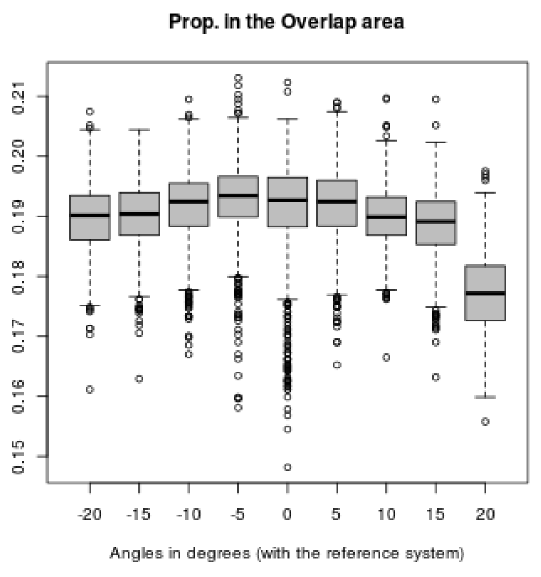
```
lip12 <- lip1 <- lip2 <- list()
for (whi in 1:9) {
    ## rotation matrix
    M <- cbind(c(cos(testesy[whi]), sin(testesy[whi])),
               c(cos(testesx[whi]), sin(testesx[whi])))

    ## Coordinates of the recaptures in the rotated system
    co <- as.matrix(recteal$recaptures[,c("x","y")])
    co2 <- co%*%M

    ## B-spline basis
    library(splines)
    spl <- splineDesign(recteal$knots, co2[,1], outer.ok = TRUE)

    ## Data used for the fit
    maxc <- 1073991
    minc <- (-1271611)
    lidat <- list(y=recteal$recaptures$Abberton,
                  spl=spl,
                  xy=co2,
                  minc=minc,
                  maxc=maxc)

    ## for each simulated vector of parameters, mean proportion
    ## of animals in the three zones
    rre <- sapply(1:nrow(listeangle[[whi]]$theta1), function(i) {
               theta1 <- listeangle[[whi]]$theta1[i,]
               theta2 <- listeangle[[whi]]$theta2[i,]
               p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
               p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
               p12 <- p1*p2
               return(c(mean(p1*(1-p2)),mean(p2*(1-p1)),mean(p12)))
           })

    ## stored in the three lists
    lip1[[whi]] <- rre[1,]
    lip2[[whi]] <- rre[2,]
    lip12[[whi]] <- rre[3,]
}
lip1 <- do.call(cbind,lip1)
lip2 <- do.call(cbind,lip2)
lip12 <- do.call(cbind,lip12)
```

The results of these calculations with the true dataset (not the slightly altered dataset as above) are stored in the component `proportions` from the dataset `sensitivityAngles`. We display how these proportions vary according to the chosen system below:

```
par(mfrow = c(2,2))
proportions <- sensitivityAngles$proportions
lip1 <- proportions$lip1
lip2 <- proportions$lip2
lip12 <- proportions$lip12

boxplot(lip1, axes=FALSE, col="grey",
        main="Prop. in the pure British Flyway",
        xlab="Angles in degrees (with the reference system)")
axis(1, at=1:9, labels = anglestestes)
axis(2)
box()
boxplot(lip2, axes=FALSE, col="grey",
        main="Prop. in the pure Mediterranean Flyway",
        xlab="Angles in degrees (with the reference system)")
axis(1, at=1:9, labels = anglestestes)
axis(2)
box()
boxplot(lip12, axes=FALSE, col="grey",
        main="Prop. in the Overlap area",
        xlab="Angles in degrees (with the reference system)")
axis(1, at=1:9, labels = anglestestes)
axis(2)
box()
```







We find results similar to those discussed above: the choice of the coordinate system has a small effect on these proportions, unless the rotation is both positive and very different from the reference system (+20°), and even in this case, the differences between the new reference system (0°) and the heavily rotated system are small (a difference of proportions of ≈ 0.02). This confirms our previous interpretation.

Finally, we considered the coordinate system used in the paper. We defined three sets of recaptures: (i) the set of recaptures located in the pure British flyway according to the model fitted in this system, (ii) the set of recaptures located in the pure Mediterranean flyway according to this model, and (iii) the set of recaptures located in the overlap area according to this model. For each set of recaptures, we then calculated the proportion of recaptures of the set that was still in this set in each alternative rotated coordinate system.

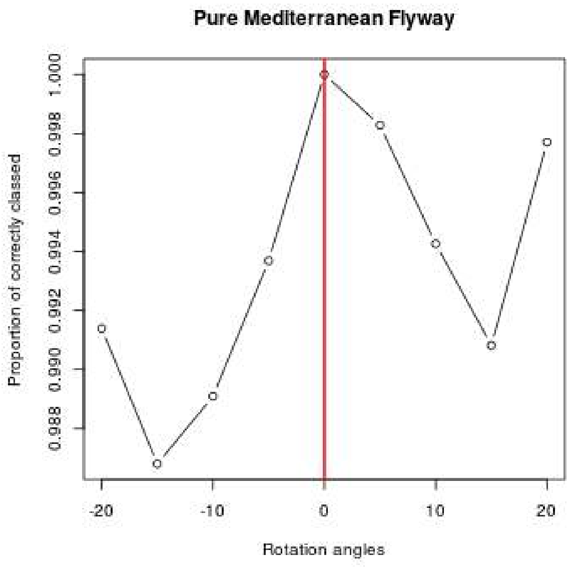
```
## Our reference system is the fifth rotation
whi <- 5

## We calculate, for each set and each recapture, a vector defining
## whether the recapture is in the set or not.
## The rotation matrix for this reference system
M <- cbind(c(cos(testesy[whi]), sin(testesy[whi])),
            c(cos(testesx[whi]), sin(testesx[whi])))

## The coordinates in this system
co <- as.matrix(recteal$recaptures[,c("x","y")])
co2 <- co%*%M

## The dataset
library(splines)
spl <- splineDesign(recteal$knots, co2[,1], outer.ok = TRUE)
maxc <- 1073991
minc <- (-1271611)
lidat <- list(y=recteal$recaptures$Abberton,
              spl=spl,
              xy=co2,
              minc=minc,
              maxc=maxc)

## Whether each recapture is in the pure British flyway
p1ref <- apply(sapply(1:nrow(listeangle[[whi]]$theta1), function(i) {
                    theta1 <- listeangle[[whi]]$theta1[i,]
                    theta2 <- listeangle[[whi]]$theta2[i,]
                    p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
                    p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
                    return(p1*(1-p2))
                }),1,mean)>0.5

## Whether each recapture is in the pure Mediterranean flyway
p2ref <- apply(sapply(1:nrow(listeangle[[whi]]$theta1), function(i) {
                    theta1 <- listeangle[[whi]]$theta1[i,]
                    theta2 <- listeangle[[whi]]$theta2[i,]
                    p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
                    p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
                    return(p2*(1-p1))
                }),1,mean)>0.5

## Whether each recapture is in the overlap area
p12ref <- apply(sapply(1:nrow(listeangle[[whi]]$theta1), function(i) {
                           theta1 <- listeangle[[whi]]$theta1[i,]
                           theta2 <- listeangle[[whi]]$theta2[i,]
                           p1 <- ((as.vector(lidat$xy[,2]-
                                                  tcrossprod(theta1,lidat$spl)))>0)
                           p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-
                                        lidat$xy[,2])>0)
                           p12 <- p1*p2
                           return(p12)
                       }),1,mean)>0.5




## Then, for each rotation angle
lip12c <- lip1c <- lip2c <- list()
for (whi in 1:9) {

    ## Rotation matrix
    M <- cbind(c(cos(testesy[whi]), sin(testesy[whi])),
               c(cos(testesx[whi]), sin(testesx[whi])))
```

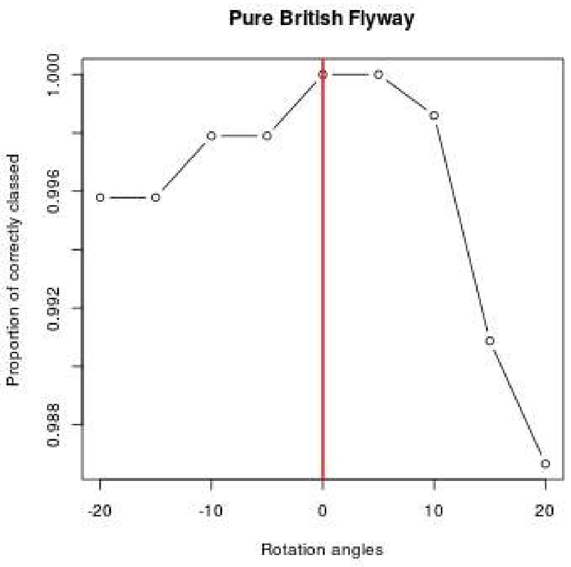
```r
    co <- as.matrix(recteal$recaptures[,c("x","y")])
    co2 <- co%*%M

    ## dataset used for the fit
    library(splines)
    spl <- splineDesign(recteal$knots, co2[,1], outer.ok = TRUE)
    maxc <- 1073991
    minc <- (-1271611)
    lidat <- list(y=recteal$recaptures$Abberton,
                  spl=spl,
                  xy=co2,
                  minc=minc,
                  maxc=maxc)

    ## Proportion of "pure British" recaptures still in the
    ## estimated pure British Flyway
    rre1 <- apply(sapply(1:nrow(listeangle[[whi]]$theta1), function(i) {
               theta1 <- listeangle[[whi]]$theta1[i,]
               theta2 <- listeangle[[whi]]$theta2[i,]
               p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
               p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
               p12b <- p1*p2
               p1b <- p1*(1-p2)
               p2b <- p2*(1-p1)
               return(p1b)
           }),1,mean)

    ## Proportion of "pure Mediterranean" recaptures still in the
    ## estimated pure Mediterranean Flyway
    rre2 <- apply(sapply(1:nrow(listeangle[[whi]]$theta1), function(i) {
               theta1 <- listeangle[[whi]]$theta1[i,]
               theta2 <- listeangle[[whi]]$theta2[i,]
               p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
               p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
               p12b <- p1*p2
               p1b <- p1*(1-p2)
               p2b <- p2*(1-p1)
               return(p2b)
           }),1,mean)

    ## Proportion of overlap recaptures still in the
    ## estimated overlap area
    rre12 <- apply(sapply(1:nrow(listeangle[[whi]]$theta1), function(i) {
               theta1 <- listeangle[[whi]]$theta1[i,]
               theta2 <- listeangle[[whi]]$theta2[i,]
               p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
               p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
               p12b <- p1*p2
               p1b <- p1*(1-p2)
               p2b <- p2*(1-p1)
               return(p12b)
           }),1,mean)

    lip1c[[whi]] <- rre1
    lip2c[[whi]] <- rre2
    lip12c[[whi]] <- rre12
}
```

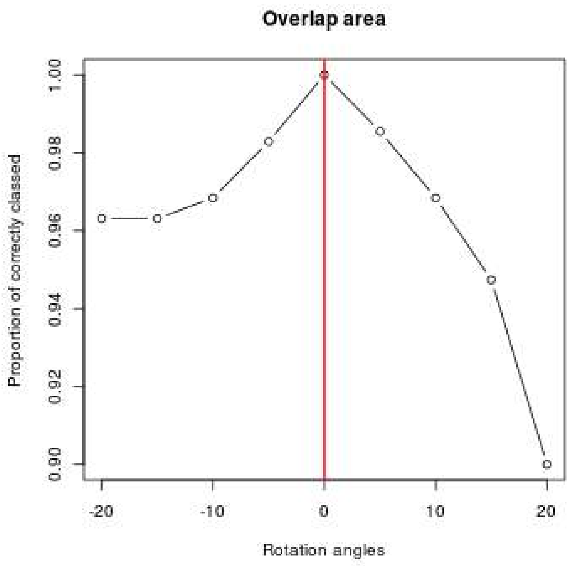
```
## British
lip1c <- do.call(cbind,lip1c)
## Mediterranean
lip2c <- do.call(cbind,lip2c)
## Overlap
lip12c <- do.call(cbind,lip12c)

## The required proportion
df <- data.frame(camarguepure=apply(lip1c, 2, function(x) mean(c(x>0.5)[p1ref])),
                 anglaisepure=apply(lip2c, 2, function(x) mean(c(x>0.5)[p2ref])),
                 overlap = apply(lip12c, 2, function(x) mean(c(x>0.5)[p12ref])))
df$angles <- anglestestes
```

We have stored the results in the component `differenceWithRef` in the list `sensitivityAngles`:

```
df <- sensitivityAngles$differenceWithRef
par(mfrow = c(2,2))
plot(df$Angles, df$PureMediterranean, ty="b",
     xlab = "Rotation angles",
     ylab = "Proportion of correctly classed",
     main="Pure Mediterranean Flyway")
abline(v=0, col="red", lwd=2)
plot(df$Angles, df$PureBritish, ty="b",
     xlab = "Rotation angles",
     ylab = "Proportion of correctly classed",
     main="Pure British Flyway")
abline(v=0, col="red", lwd=2)
plot(df$Angles, df$Overlap, ty="b",
     xlab = "Rotation angles",
     ylab = "Proportion of correctly classed",
     main="Overlap area")
abline(v=0, col="red", lwd=2)
```







As we can see, unless the rotations angles is very high (+20°), the proportions of correctly classed recaptures does not vary a lot. And even for high rotation angles, the results are not *very* different from those obtained in the reference system: 99.7% of the recaptures of the pure Mediterranean flyways are correctly classed, 98.8% of the recaptures of the pure British flyways are correctly classed, and 90% of the overlap area are correctly classed (reaching 95% for a rotation angle of 15°).

This consistent pattern for all metrics again confirms our previous interpretation: The choice of the coordinate system does not strongly affect the results of the analysis, except when it is strongly different from a system allowing the modelling of a flyway oriented according to North-East/South-West direction. Our results are therefore robust to the choice of the coordinate system.

2.2 The choice of the number of functions in the B-spline basis

To model the boundary of the flyways, we fitted a regression spline model based on a B-spline basis defined with 26 B-spline functions. The choice of this number of functions was subjective, and controls the smoothness of the boundary. In this section, we try to fit our model with different choices for this number (seven choices comprised between 16 and 36 functions), to assess its influence on our results.

We fitted our model using these different numbers of functions, using the following code. **WARNING:** the following code is very long (the cumulated time spent in these calculations was 24 days. We parallelized the calculations, so that they took “only” 6 days):

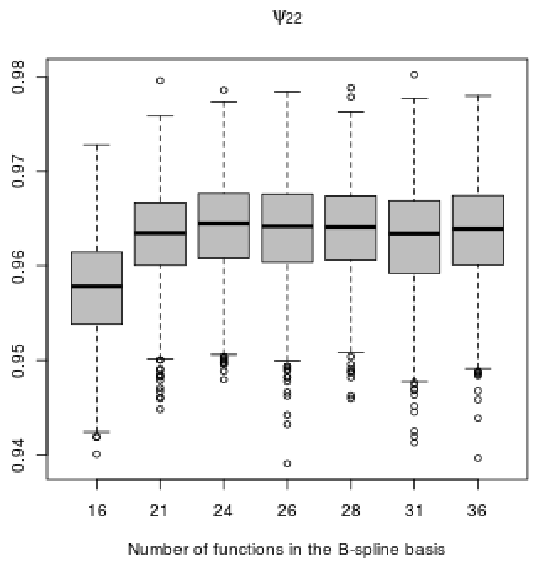
```
for (whi in 1:length(testesx)) {

    ## The coordinates in the new system
    co <- as.matrix(recteal$recaptures[,c("x","y")])
    co2 <- co%*%M

    ## The B-spline basis
    library(splines)
    knots <- recteal$knots
    ra <- range(knots[4:23])
    ## The number of unique knots; the number of B-spline
    ## functions is this number + 6 (the end knots
    ## are repeated 4 times)
    uu <- c(10, 15, 18, 20, 22, 25, 30)[whi]
    kn <- seq(ra[1],ra[2], length=uu)
    knots <- c(rep(ra[1],3), kn, rep(ra[2],3))
    spl <- splineDesign(knots, co2[,1], outer.ok = TRUE)

    ## min and max limits
    minc <- min(co2[,2])-65.11
    maxc <- max(co2[,2])+162.4

    ## The list that will be used for the fit
    lidat <- list(y=recteal$recaptures$Abberton,
                  spl=spl,
                  xy=co2,
                  minc=minc,
                  maxc=maxc)

    ## Lipum parameter (sd and variance of the proposal)
    lipum <- list(theta1=rep(310000,ncol(spl)),
                  theta2=rep(310000,ncol(spl)),
                  psi22=0.187,
                  psi21=0.234,
                  delta=0.12)

    ## Multiple values: theta1 and theta1
    mu <- list(theta1=ncol(spl), theta2=ncol(spl))


    ## New function to calculate posterior:
    ## La logposterior
    logposteriorModelFlywaysSim <- function(par, lidat, ctrl)
    {
        ## Coefficients
        theta1 <- par$theta1
        theta2 <- par$theta2

        pa <- c(par$psi21,par$psi22,par$delta)
        ## Prior on probabilities
        pri <- sum(dlogis(pa,location = 0, scale=1, log=TRUE))
        psi22 <- invlogit(par$psi22)
        psi21 <- invlogit(par$psi21)
        delta <- invlogit(par$delta)
        ## Prior distribution on  theta
        if (any(theta1<(lidat$minc)))
            return(-Inf)
        if (any(theta2<(lidat$minc)))
            return(-Inf)
        if (any(theta1>(lidat$maxc)))
            return(-Inf)
        if (any(theta2>(lidat$maxc)))
            return(-Inf)

        ## Likelihood.
        p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
        p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
        p <- psi22*(delta*(p1*p2) + (1-p1)*p2)+
            psi21*((1-delta)*(p1*p2) + p1*(1-p2))
        ## Posterior distribution
        return(pri+sum(log(lidat$y*p+(1-lidat$y)*(1-p))))
    }


    ## To save some time, we fitted only one MCMC chain. Starting
    ## value:
    ## Changing the parameter support
    pin <- as.list(1:5)
    names(pin) <- c("theta1","theta2","psi22","psi21","delta")
    for (i in 1:2)
        pin[[i]] <- c(lidat$minc, lidat$maxc)
    for (i in 3:5)
        pin[[i]] <- c(-10, 10)

    fs <- findStartingValues(pin, logposteriorModelFlywaysSim, lidat, multiple = mu,
                             method="onebest", info = FALSE)

    ## The updating mechanisms
    listUpdating <- defaultListUGSM(fs)

    gsm <- GeneralSingleMetropolis(fs, lidat=lidat,
                                   logposterior = logposteriorModelFlywaysSim,
                                   lipum = lipum,
                                   listUpdating = listUpdating,
                                   nrepet = 5050000, thinPar = 5000, saveEvery = 100000,
                                   fileSave = paste0("simusNSpline-",whi,"-sim.Rdata"))
}
```

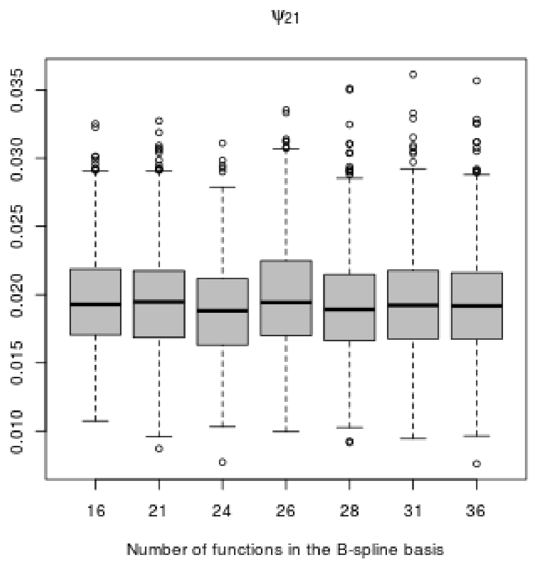
We stored the results of these calculations in the component `fit` of the dataset `sensitivityBsplines`. Now load this dataset, and store the results in the list `listesplines`:

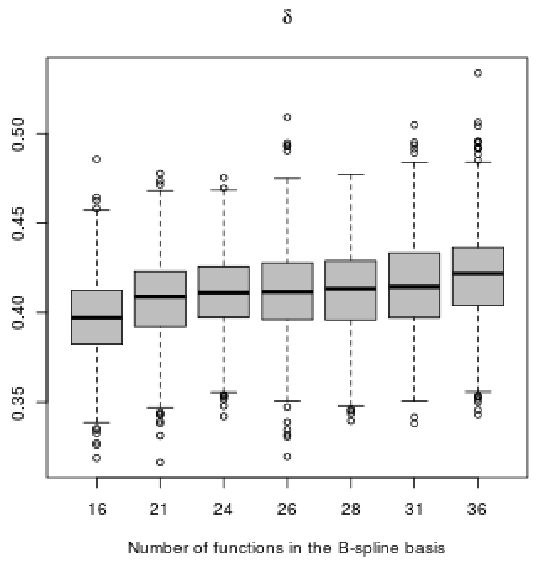
```
data(sensitivityBsplines)
listesplines <- sensitivityBsplines$fit
```

We can compare the posterior distribution of the parameters $\psi_{1\leftarrow 2}, \psi_{2\leftarrow 2}, \delta$ for the different choices:

```r
par(mfrow = c(2,2))
nfonct <- c(10, 15, 18, 20, 22, 25, 30)+6
psi22 <- do.call(cbind,lapply(listesplines, function(x) invlogit(x$psi22)))
boxplot(psi22, axes=FALSE, col="grey",
        main=expression(psi[22]),
        xlab="Number of functions in the B-spline basis")
axis(1, at=1:7, labels = nfonct)
axis(2)
box()

psi21 <- do.call(cbind,lapply(listesplines, function(x) invlogit(x$psi21)))
boxplot(psi21, axes=FALSE, col="grey",
        main=expression(psi[21]),
        xlab="Number of functions in the B-spline basis")
axis(1, at=1:7, labels = nfonct)
axis(2)
box()

delta <- do.call(cbind,lapply(listesplines, function(x) invlogit(x$delta)))
boxplot(delta, axes=FALSE, col="grey",
        main=expression(delta),
        xlab="Number of functions in the B-spline basis")
axis(1, at=1:7, labels = nfonct)
axis(2)
box()
```







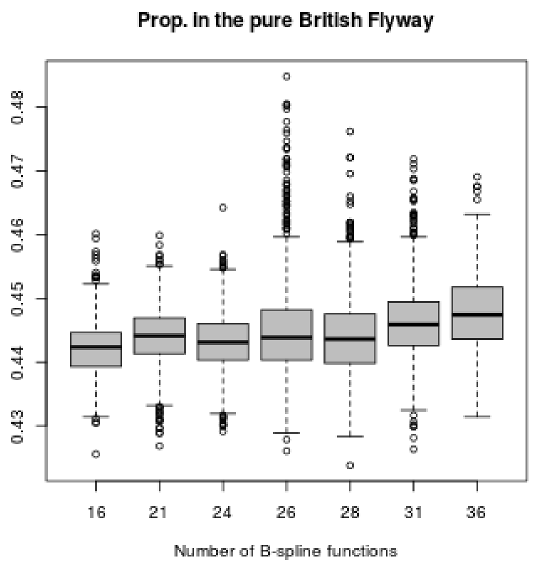
The choice of the coordinate system did not have a large effect on the posterior distribution of these parameters. The parameter $\psi_{2\leftarrow 2}$ is slightly smaller when the number of functions in the basis was equal to 16 (see below for an explanation).

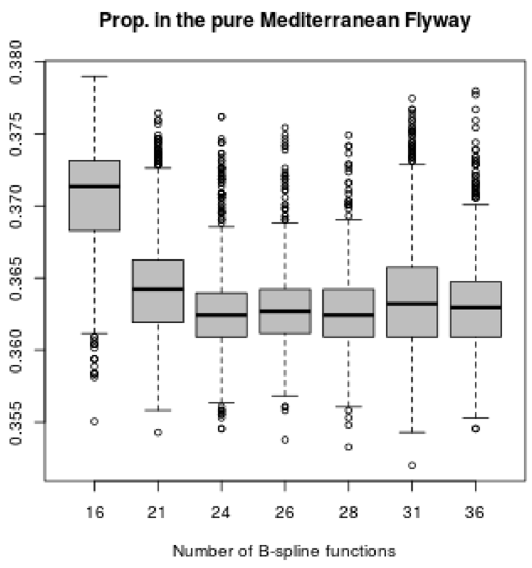
Moreover, for each possible choice of number of functions in the B-spline basis, we obtained an estimation of the posterior distribution of the location of the boundaries of the two flyways. We therefore calculated, for each possible number of functions, the proportion of birds recaptured in each of the three areas (pure British flyway, pure Mediterranean flyway, overlap area). We present the code below (note that because we work on an altered dataset – see introduction – the result that the reader can obtain with this code will be slightly different from our results):

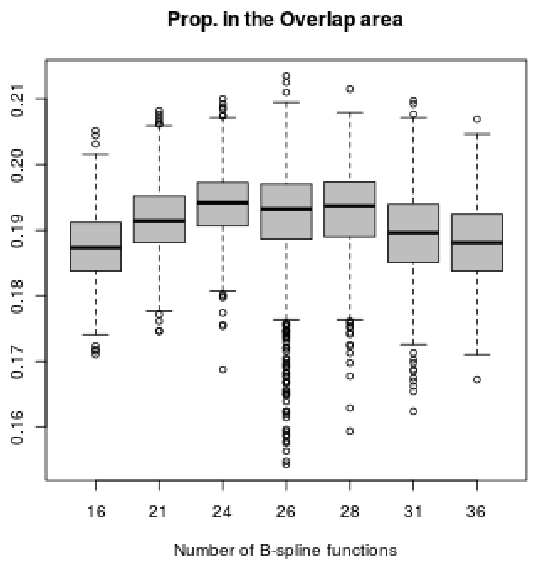
```
## On calcule la proportion d'individus anglais en zone anglaise/française/overlap
lip12 <- lip1 <- lip2 <- list()
for (whi in 1:7) {

    ## The B-spline basis
    knots <- recteal$knots
    ra <- range(knots[4:23])
    uu <- c(10, 15, 18, 20, 22, 25, 30)[whi]
    kn <- seq(ra[1],ra[2], length=uu)
    knots <- c(rep(ra[1],3), kn, rep(ra[2],3))
    spl <- splineDesign(knots, co2[,1], outer.ok = TRUE)
    minc <- min(co2[,2])-65.11
    maxc <- max(co2[,2])+162.4

    ## The list that will be used for the fit
    lidat <- list(y=recteal$recaptures$Abberton,
                  spl=spl,
                  xy=co2,
                  minc=minc,
                  maxc=maxc)

    ## for each simulated vector of parameters, mean proportion
    ## of animals in the three zones
    rre <- sapply(1:nrow(listesplines[[whi]]$theta1), function(i) {
               theta1 <- listesplines[[whi]]$theta1[i,]
               theta2 <- listesplines[[whi]]$theta2[i,]
               p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
               p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
               p12 <- p1*p2
               return(c(mean(p1*(1-p2)),mean(p2*(1-p1)),mean(p12)))
           })

    ## stored in the three lists
    lip1[[whi]] <- rre[1,]
    lip2[[whi]] <- rre[2,]
    lip12[[whi]] <- rre[3,]
}
lip1 <- do.call(cbind,lip1)
lip2 <- do.call(cbind,lip2)
lip12 <- do.call(cbind,lip12)
```

The results of these calculations with the true dataset (not the slightly altered dataset as above) are stored in the component `proportions` from the dataset `sensitivityBsplines`. We display how these proportions vary according to the chosen system below:

```
par(mfrow = c(2,2))
proportions <- sensitivityBsplines$proportions
lip1 <- proportions$lip1
lip2 <- proportions$lip2
lip12 <- proportions$lip12

boxplot(lip1, axes=FALSE, col="grey",
        main="Prop. in the pure British Flyway",
        xlab="Number of B-spline functions")
axis(1, at=1:7, labels = nfonct)
axis(2)
box()
boxplot(lip2, axes=FALSE, col="grey",
        main="Prop. in the pure Mediterranean Flyway",
        xlab="Number of B-spline functions")
axis(1, at=1:7, labels = nfonct)
axis(2)
box()
boxplot(lip12, axes=FALSE, col="grey",
        main="Prop. in the Overlap area",
           xlab="Number of B-spline functions")
axis(1, at=1:7, labels = nfonct)
axis(2)
box()
```







As for the parameter $\psi_{2\leftarrow 2}$, the proportion of birds in the Mediterranean flyway was in average greater than with the chosen basis of 26 functions (again, see below for an explanation). However, even in this case, the differences between the new reference system (26 functions) and the heavily rotated system are small (less than 1% in average).

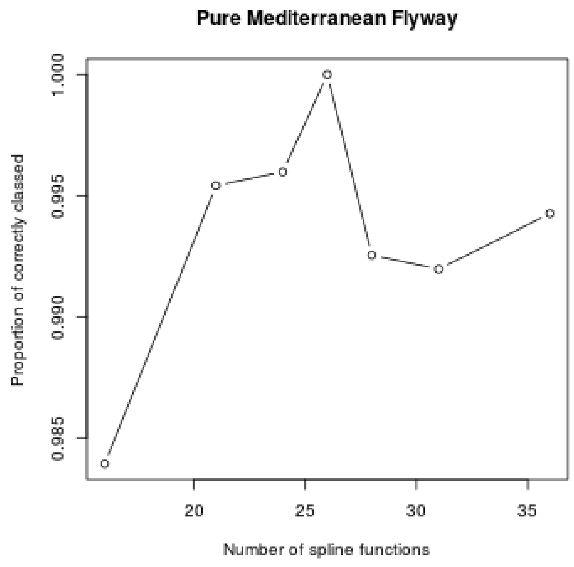
Finally, we considered the number of functions used in the paper. We defined three sets of recaptures: (i) the set of recaptures located in the pure British flyway according to the model fitted with 26 B-spline functions, (ii) the set of recaptures located in the pure Mediterranean flyway according to this model, (iii) the set of recaptures located in the overlap area according to this model. For each set of recaptures, we then calculated the proportion of recaptures of the set that was still in this set for each possible number of functions.

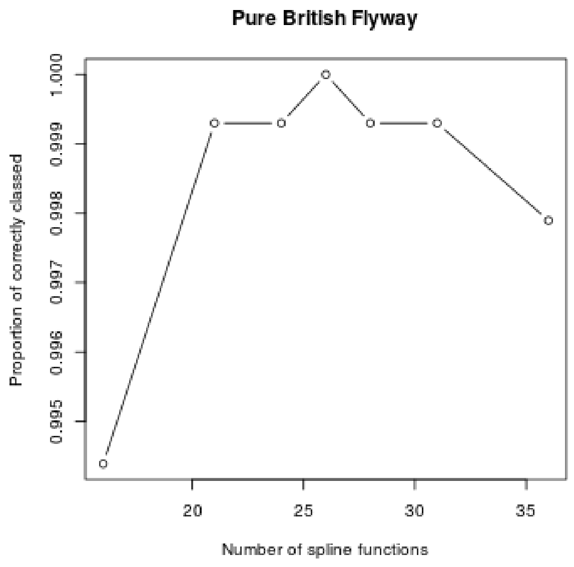
```r
## Actual number on functions: 20 different knots
## We define the B-spline basis
whi <- 4
knots <- recteal$knots
ra <- range(knots[4:23])
uu <- c(10, 15, 18, 20, 22, 25, 30)[whi]
kn <- seq(ra[1],ra[2], length=uu)
knots <- c(rep(ra[1],3), kn, rep(ra[2],3))
spl <- splineDesign(knots, co2[,1], outer.ok = TRUE)
minc <- min(co2[,2])-65.11
maxc <- max(co2[,2])+162.4

## The list that will be used for the fit
lidat <- list(y=recteal$recaptures$Abberton,
              spl=spl,
              xy=co2,
              minc=minc,
              maxc=maxc)

## Whether each recapture is in the pure British flyway
p1ref <- apply(sapply(1:nrow(listesplines[[whi]]$theta1), function(i) {
                    theta1 <- listesplines[[whi]]$theta1[i,]
                    theta2 <- listesplines[[whi]]$theta2[i,]
                    p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
                    p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
                    return(p1*(1-p2))
                }),1,mean)>0.5

## Whether each recapture is in the pure Mediterranean flyway
p2ref <- apply(sapply(1:nrow(listesplines[[whi]]$theta1), function(i) {
                    theta1 <- listesplines[[whi]]$theta1[i,]
                    theta2 <- listesplines[[whi]]$theta2[i,]
                    p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
                    p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
                    return(p2*(1-p1))
                }),1,mean)>0.5

## Whether each recapture is in the overlap area
p12ref <- apply(sapply(1:nrow(listesplines[[whi]]$theta1), function(i) {
                           theta1 <- listesplines[[whi]]$theta1[i,]
                           theta2 <- listesplines[[whi]]$theta2[i,]
                           p1 <- ((as.vector(lidat$xy[,2]-
                                                 tcrossprod(theta1,lidat$spl)))>0)
                           p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-
                                       lidat$xy[,2])>0)
                           p12 <- p1*p2
                           return(p12)
                       }),1,mean)>0.5


## Then, for each number of functions
lip12c <- lip1c <- lip2c <- list()
for (whi in 1:7) {

    ## B-spline basis
    knots <- recteal$knots
    ra <- range(knots[4:23])
    uu <- c(10, 15, 18, 20, 22, 25, 30)[whi]
    kn <- seq(ra[1],ra[2], length=uu)
    knots <- c(rep(ra[1],3), kn, rep(ra[2],3))
    spl <- splineDesign(knots, co2[,1], outer.ok = TRUE)
    minc <- min(co2[,2])-65.11
    maxc <- max(co2[,2])+162.4

    ## The list that will be used for the fit
    lidat <- list(y=recteal$recaptures$Abberton,
                  spl=spl,
                  xy=co2,
                  minc=minc,
                  maxc=maxc)

    ## Proportion of "pure British" recaptures still in the pure British Flyway
    rre1 <- apply(sapply(1:nrow(listesplines[[whi]]$theta1), function(i) {
              theta1 <- listesplines[[whi]]$theta1[i,]
              theta2 <- listesplines[[whi]]$theta2[i,]
              p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
              p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
              p12b <- p1*p2
              p1b <- p1*(1-p2)
              p2b <- p2*(1-p1)
              return(p1b)
       }),1,mean)

    ## Proportion of "pure Mediterranean" recaptures still in the
    ## estimated pure Mediterranean Flyway
    rre2 <- apply(sapply(1:nrow(listesplines[[whi]]$theta1), function(i) {
              theta1 <- listesplines[[whi]]$theta1[i,]
              theta2 <- listesplines[[whi]]$theta2[i,]
              p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
              p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
              p12b <- p1*p2
              p1b <- p1*(1-p2)
              p2b <- p2*(1-p1)
              return(p2b)
       }),1,mean)

    ## Proportion of overlap recaptures still in the
    ## estimated overlap area
    rre12 <- apply(sapply(1:nrow(listesplines[[whi]]$theta1), function(i) {
              theta1 <- listesplines[[whi]]$theta1[i,]
              theta2 <- listesplines[[whi]]$theta2[i,]
              p1 <- ((as.vector(lidat$xy[,2]-tcrossprod(theta1,lidat$spl)))>0)
              p2 <- ((as.vector(tcrossprod(theta2,lidat$spl))-lidat$xy[,2])>0)
              p12b <- p1*p2
              p1b <- p1*(1-p2)
              p2b <- p2*(1-p1)
              return(p12b)
       }),1,mean)

    lip1c[[whi]] <- rre1
    lip2c[[whi]] <- rre2
    lip12c[[whi]] <- rre12
}
## British
```

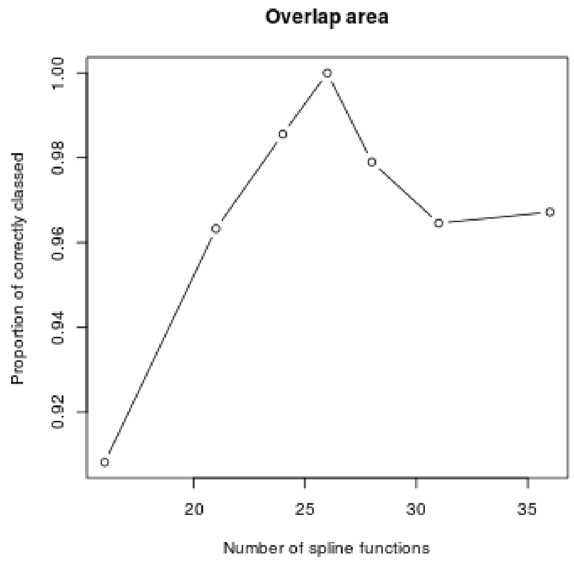
```r
lip1c <- do.call(cbind,lip1c)
## Mediterranean
lip2c <- do.call(cbind,lip2c)
## Overlap
lip12c <- do.call(cbind,lip12c)

## Proportion des classés en zone anglaise effectivement en anglaise
df2 <- data.frame(camarguepure=apply(lip1c, 2, function(x) mean(c(x>0.5)[p1ref])),
                  anglaisepure=apply(lip2c, 2, function(x) mean(c(x>0.5)[p2ref])),
                  overlap = apply(lip12c, 2, function(x) mean(c(x>0.5)[p12ref])))
df2$nsplines <- nfonct
```

We have stored the results in the component `differenceWithRef` in the list `sensitivityBsplines`:

```r
df2 <- sensitivityBsplines$differenceWithRef
par(mfrow = c(2,2))
plot(df2$nsplines, df2$PureMediterranean, ty="b",
     xlab = "Number of spline functions",
     ylab = "Proportion of correctly classed",
     main="Pure Mediterranean Flyway")
abline(v=0, col="red", lwd=2)
plot(df2$nsplines, df2$PureBritish, ty="b",
     xlab = "Number of spline functions",
     ylab = "Proportion of correctly classed",
     main="Pure British Flyway")
abline(v=0, col="red", lwd=2)
plot(df2$nsplines, df2$Overlap, ty="b",
     xlab = "Number of spline functions",
     ylab = "Proportion of correctly classed",
     main="Overlap area")
abline(v=0, col="red", lwd=2)
```







As we can see, for a very small number of B-spline functions (16 functions), the proportions of correctly classed recaptures decreases, but does not vary a lot otherwise. Thus, as noted in the paper, when the number of functions is small, the modelled boundary is not flexible enough and is too close to a linear boundary. With 26 functions in the basis, the boundary closely fits the data. Note that even for a number of functions as small as 16, the results are not very different from those obtained with 26 bases: 98.5% of the recaptures of the pure Mediterranean flyways are correctly classed, 99.5% of the recaptures of the pure British flyways are correctly classed, and 91% of the overlap area are correctly classed (reaching 96% for 20 functions). This confirms that the choice of the number of B-spline functions does not strongly affect the results of the analysis.

References

Geyer C. 2011. Introduction to Markov Chain Monte Carlo. pp. 3–49 *In:* Brooks S., Gelman A., Jones G.L. and Meng X.L. *Handbook of Markov Chain Monte Carlo: Methods and Applications.* Chapman & Hall/CRC.

Gelman, A. and Rubin, D. 1992. Inference from iterative simulation using multiple sequences. Statistical science 7, 457-472.

Gelman, A. and Meng, X. 1996. Model checking and model improvement. pp. 189–211 *in* Gilks, W. and Richardson, S. (Eds.) Markov chain Monte Carlo in practice,Chapman & Hall/CRC.

Guillemain M., Calenge C., Champagnon J. and Hearn R. in prep. Determining the boundaries and plasticity of migratory bird flyways: a Bayesian model for Common Teal *Anas crecca* in Western Europe.

Uyttendaele, N. 2015. How to speed up R code: an introduction. arXiv:1503.00855