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

1 Introduction

In this vignette, we provide the R code used by Guillemain et al. 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.

To circumvent copyright issues, we provide here an altered version of the dataset used by these authors: 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).

This dataset is provided with the package `metroponcfs`. We load both the package and the dataset `recteal`:

```
library(metroponcfs)
data(recteal)
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 2.3).

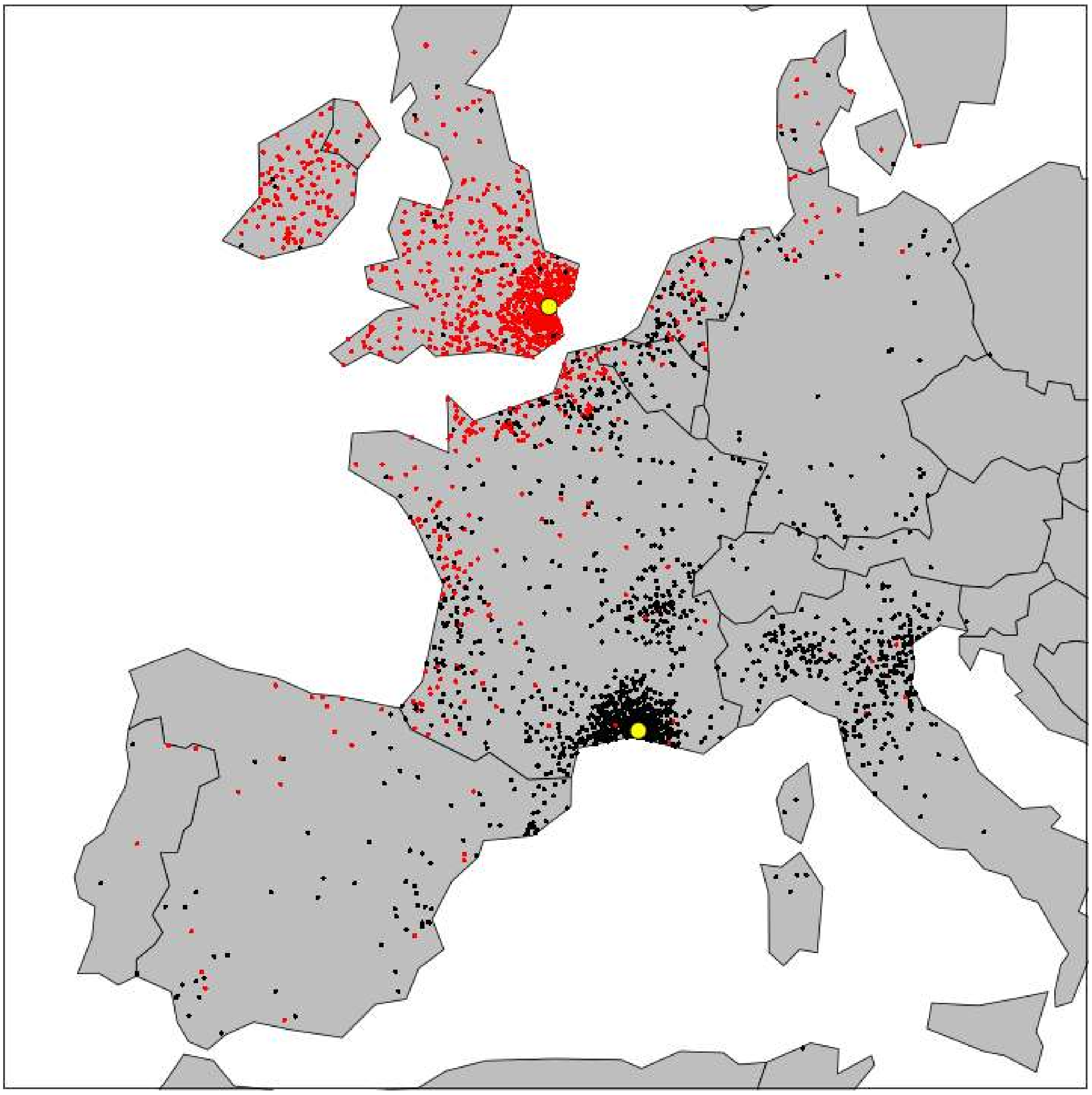
2 Preparation of the data and model fit

2.1 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
```

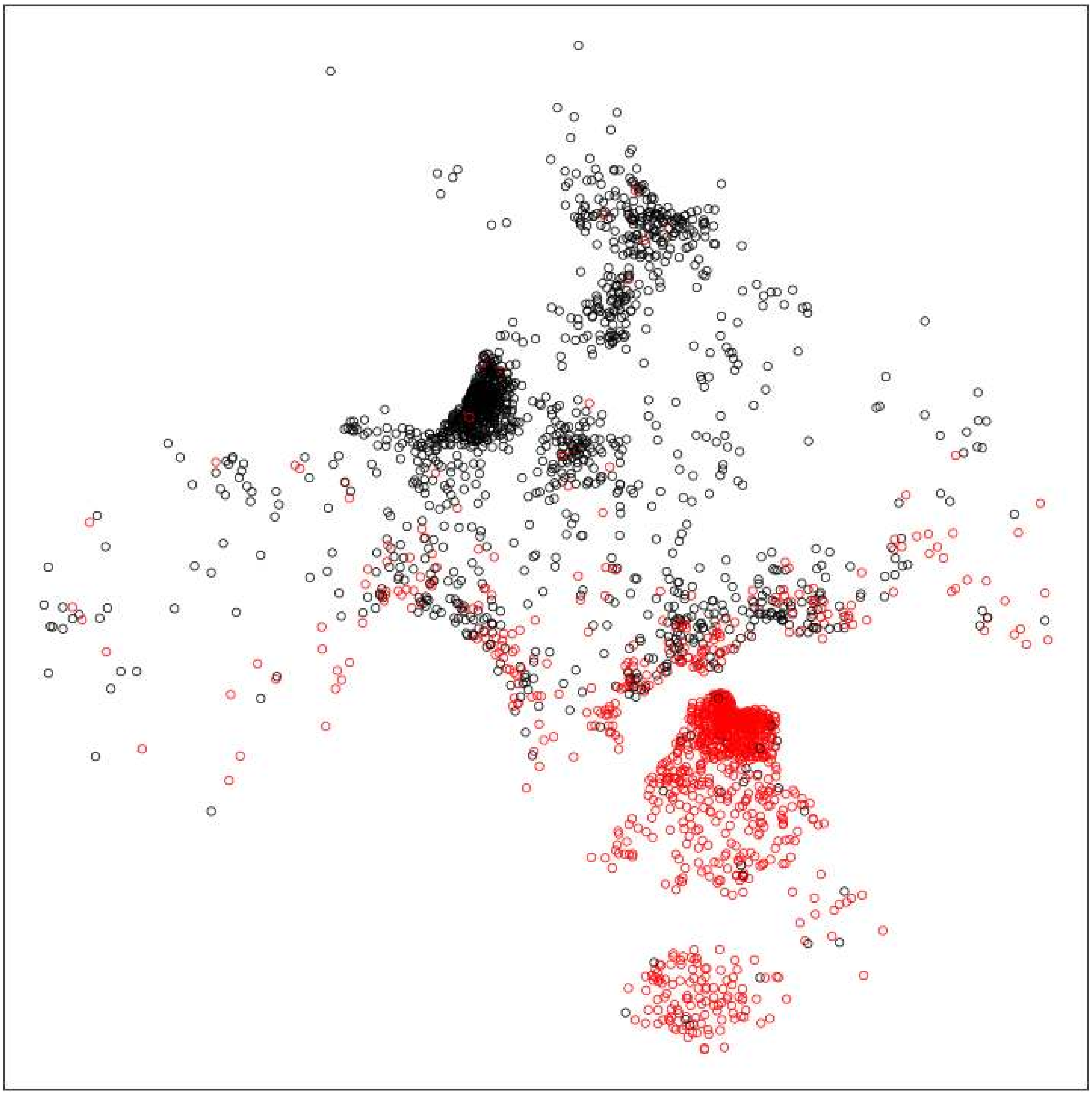
```
points(recteal$recaptures[,c("x","y")],
       col=recteal$recaptures$Abberton+1,
       pch=16, cex=0.6)

## 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):

```
## 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)
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
```

2.2 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:

$$\begin{aligned}[d_k = 1|\mathbf{v}_k] \quad = \quad & \psi_{2\to2} \times \{\delta \times [\boldsymbol{\theta}_1 \in \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \in \Theta_2(\mathbf{v}_k)] + [\boldsymbol{\theta}_1 \notin \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \in \Theta_2(\mathbf{v}_k)]\} + \\ & \psi_{2\to1} \times \{(1-\delta) \times [\boldsymbol{\theta}_1 \in \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \in \Theta_2(\mathbf{v}_k)] + [\boldsymbol{\theta}_1 \in \Theta_1(\mathbf{v}_k), \boldsymbol{\theta}_2 \notin \Theta_2(\mathbf{v}_k)]\}\end{aligned}$$

With $\psi_{2\to2}$ the probability that an individual belonging to the British subpopulation at recapture time was originally captured in Great-Britain, $\psi_{2\to1}$ 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:

$$\begin{aligned}\theta_{ij} &\sim \mathcal{U}(-1271611, 11073991)\\ \text{logit}(\psi_{2\to 2}) &\sim \mathcal{U}(-15, 15)\\ \text{logit}(\psi_{2\to 1}) &\sim \mathcal{U}(-15, 15)\\ \text{logit}(\delta) &\sim \mathcal{U}(-15, 15)\end{aligned}$$

2.3 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.

In the article, Guillemain et al. ran 5 chains of 2 000 000 iterations each. In this vignette, we just want to illustrate the calculations, so that we ran only two chains of 500 000 iterations each (which took about 12 hours of calculations). However, the reader is free to test this code with a larger number of chains and a larger number of iterations per chain. After some exploration, we chose the following vectors of starting values:

```r
parInit1 <- list(theta1a = c(-429848, -1225539,
                 -1058897, -299476, -1185661, -339494),
                 theta1b = c(-290987, -370117),
                 theta1c = c(-691426, -1190062),
                 theta1d = c(-286011, -624738),
                 theta1e = c(-264845, -404173, -459037),
                 theta1f = c(-732016, 211003, -704255,
                 943347, -736270, 27310, -244915),
                 theta2a = c(-1261815, -544118, 215340,
                 585184, -1242381, 820451),
                 theta2b = c(-810987, 456770, -574762, 166028),
                 theta2c = -497628,
                 theta2d = c(663207, -852554, 487832,
                 -383177),
                 theta2e = c(-79686, 656123, -748390,
                 920375, -999008, -999285, -18276),
                 psi22 = 0.96, psi21 = 0.018,
                 delta = 0.3126)

parInit2 <- list(theta1a = c(331777, 287922, -1172087, -324465,
                 129919, -1116989),
                 theta1b = c(-105664, -545644),
                 theta1c = c(-1141727, -881130),
                 theta1d = c(-374846, -560183),
                 theta1e = c(-302361, -353503, -688120),
                 theta1f = c(-933325, 448871, -1050266, -348321,
                 358430, 10629, -727748),
                 theta2a = c(-1051310, -919978, 585145,
                 -374519, -473335, 32921),
                 theta2b = c(156275, -261841, 65729, -275447),
                 theta2c = -180564,
                 theta2d = c(69644, -482812, -127223, -213950),
                 theta2e = c(-80815, 79801, -89061, 203133, -696576,
                 -418571, 598242),
                 psi22 = 0.953, psi21 = 0.014, delta = 0.401)
```

We define below 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(parInit1)
listUpdating$theta1b <- "mns"
listUpdating$theta1d <- "mns"
listUpdating$theta1e <- "mns"
listUpdating$theta2b <- "mns"
listUpdating$theta2d <- "mns"
```

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)

    pa <- c(par$psi21,par$psi22,par$delta)
    if (any(pa<(-15)|pa>15))
        return(-Inf)

    psi22 <- invlogit(par$psi22)
    psi21 <- invlogit(par$psi21)
    delta <- invlogit(par$delta)

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

Finally, we launched the Metropolis algorithm. **WARNING**: the following code is very long (12 hours of calculation on a PC equipped with an intel core i5):

```
gsm1 <- GeneralSingleMetropolis(parInit1, lidat=lidat,
                                logposterior = logposteriorModelFlyways,
                                lipum = lipum,
                                listUpdating = listUpdating,
                                nrepet = 500000, thinPar = 1000, saveEvery = 10000,
                                fileSave = "saveFlyways-1.rda")

gsm2 <- GeneralSingleMetropolis(parInit2, lidat=lidat,
                                logposterior = logposteriorModelFlyways,
                                lipum = lipum,
                                listUpdating = listUpdating,
                                nrepet = 500000, thinPar = 1000, saveEvery = 10000,
                                fileSave = "saveFlyways-2.rda")

rectealmodel <- c(gsm1, gsm2)
```

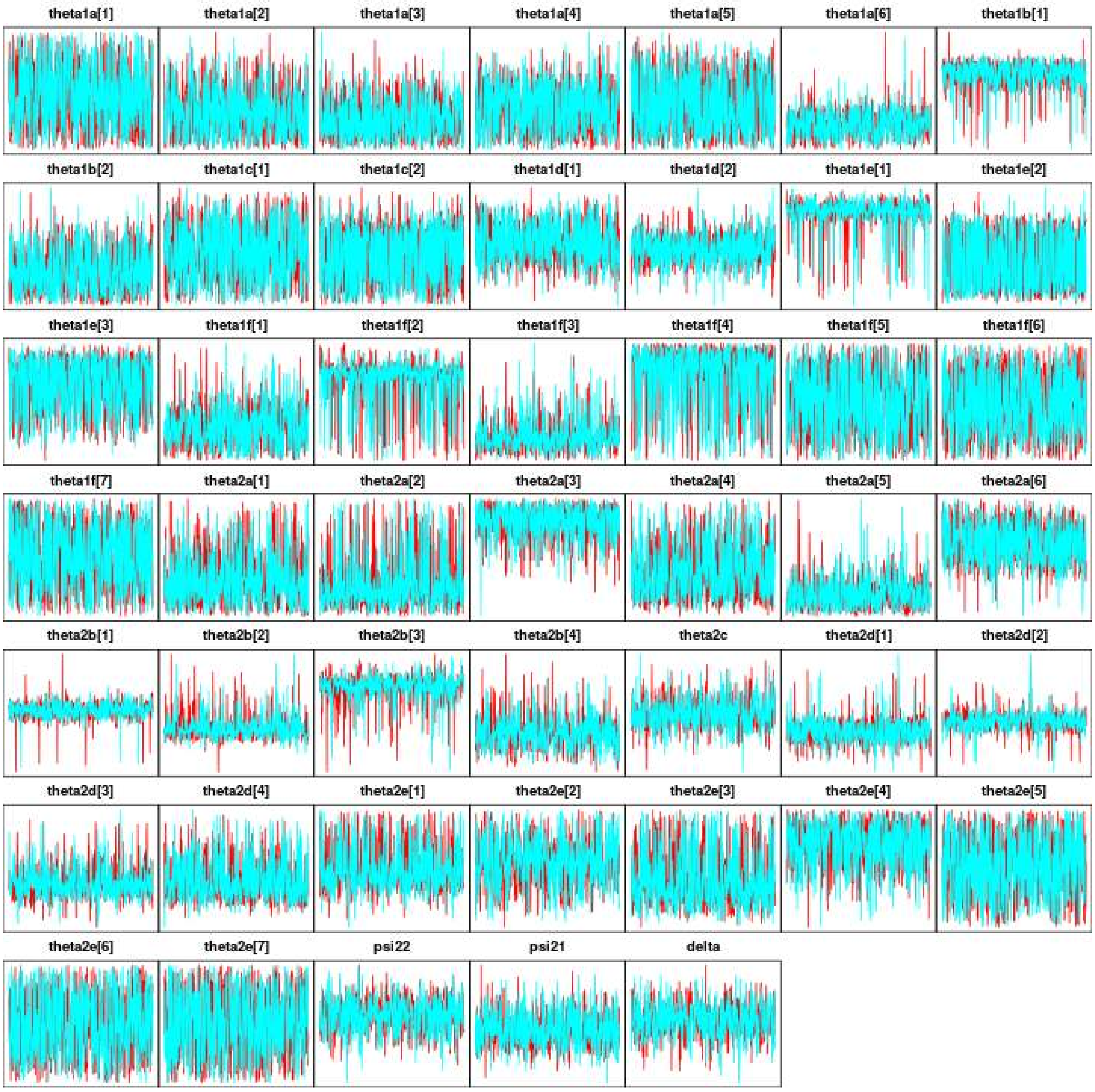
Remark: the code displayed above is very long, if executed as presented above. For the calculations carried out in the paper, Guillemain et al actually launched these calculations using parallel computing. See Uyttendaele (2015) for additional details.

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

```
data(rectealmodel)
rectealmodel

## Object of class "CMM"
##
## Number of chains: 2
## 2 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"
##
## 500010 iterations per chain (including 10  burn-in samples)
## Thin parameters every: 1000 iterations
## Number of iterations stored for each chain: 500
## Calculation took 43251 seconds (total time over all chains)

## Transformation inverse-logit
gsm1 <- rectealmodel[1]
gsm1$psi22[,1] <- invlogit(gsm1$psi22[,1])
gsm1$psi21[,1] <- invlogit(gsm1$psi21[,1])
gsm1$delta[,1] <- invlogit(gsm1$delta[,1])
gsm2 <- rectealmodel[2]
gsm2$psi22[,1] <- invlogit(gsm2$psi22[,1])
gsm2$psi21[,1] <- invlogit(gsm2$psi21[,1])
gsm2$delta[,1] <- invlogit(gsm2$delta[,1])
rectealmodelb <- c(gsm1, gsm2)
```

We illustrate below the mixing properties of the algorithm:

```
plot(rectealmodelb)
```



The mixing properties are poor here: we used only two runs of a small number of iterations here to illustrate the method (mixing is much better with 2 million iterations and 5 chains).

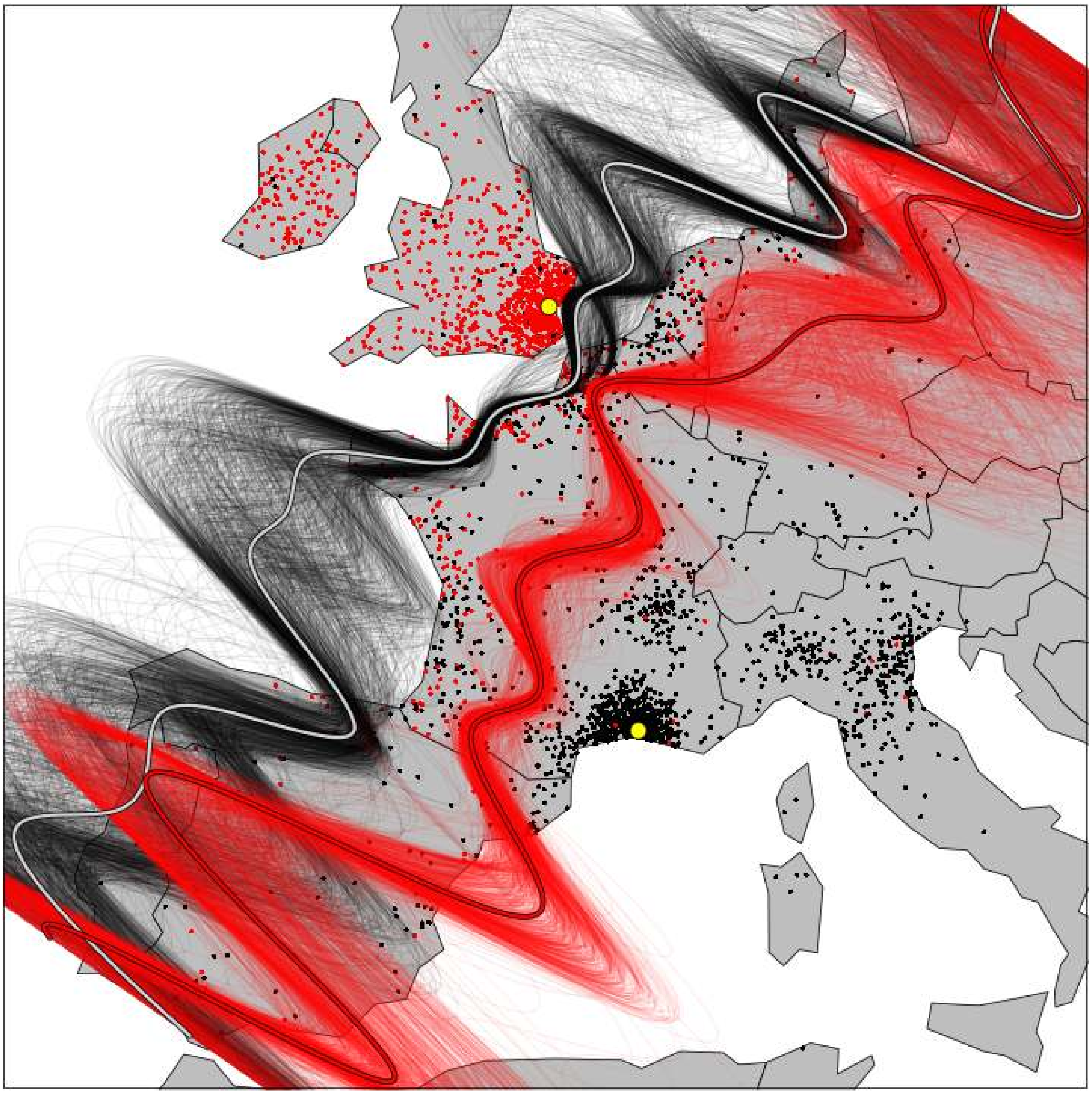
3 Model interpretation and other results

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:

```
## We store the generated vectors theta1 and theta2 in 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,
```

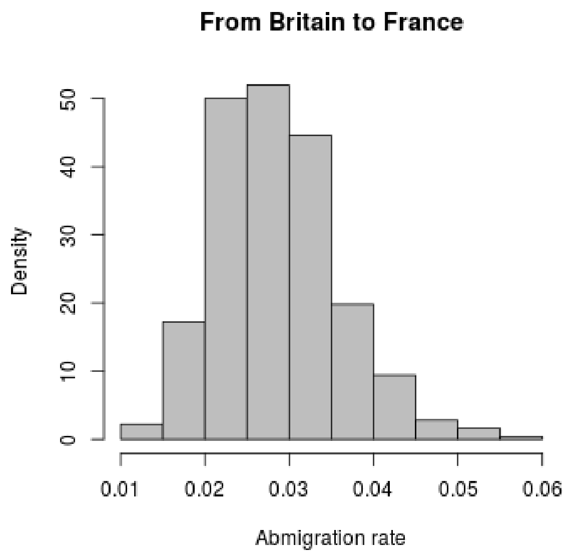
```r
                     mod$theta2d,mod$theta2e)
    return(list(theta1=theta1, theta2=theta2))
}

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)

## adds the boundaries
## 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)

## All simulated boundaries
tmp <- sapply(1:nrow(theta1), function(i) {
    mo <- theta1[i,]
    gg <- spl2%*%mo
    ## revert to original coordinate system
    front1 <- cbind(gg, xtest)%*%recteal$inverseRotationMatrix
    lines(front1, col=rgb(0.0,0.0,0,0.1))
})
tmp <- sapply(1:nrow(theta2), function(i) {
    mo <- theta2[i,]
    gg2 <- spl2%*%mo
    ## revert to original coordinate system
    front2 <- cbind(gg2, xtest)%*%recteal$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()
```

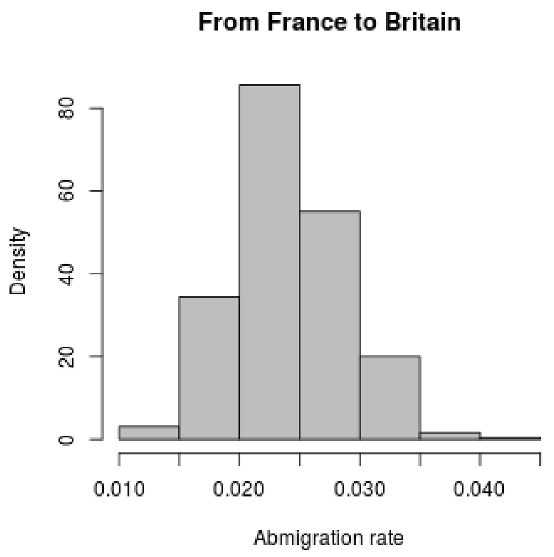


3.2 Abmigration rates

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

```r
psi22 <- c(rectealmodelb[1]$psi22, rectealmodelb[2]$psi22)
psi21 <- c(rectealmodelb[1]$psi21, rectealmodelb[2]$psi21)
delta <- c(rectealmodelb[1]$delta, rectealmodelb[2]$delta)

f12 <- 1-(psi22*delta)/(psi22*delta + psi21*(1-delta))
f21 <- ((1-psi22)*delta)/(1-(psi22*delta + psi21*(1-delta)))

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")
```





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).

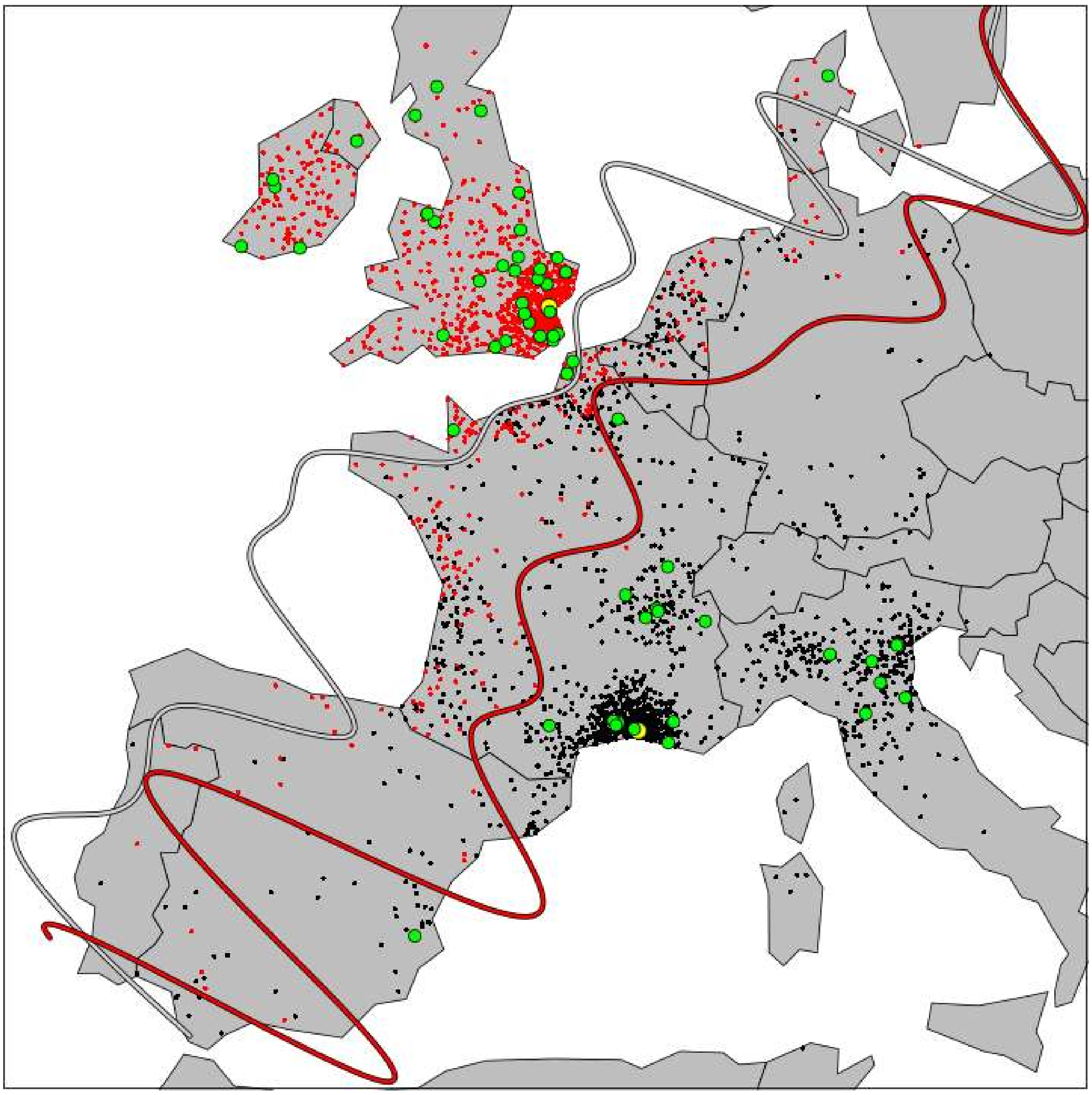
- Therefore, for a given recapture, considering all the generated pairs of vectors $\boldsymbol{\theta}_1^{(r)}$ and $\boldsymbol{\theta}_2^{(r)}$ generated by the Metropolis algorithm, we can determine the probability that this recapture actually occurred in the same flyway as its original capture (and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

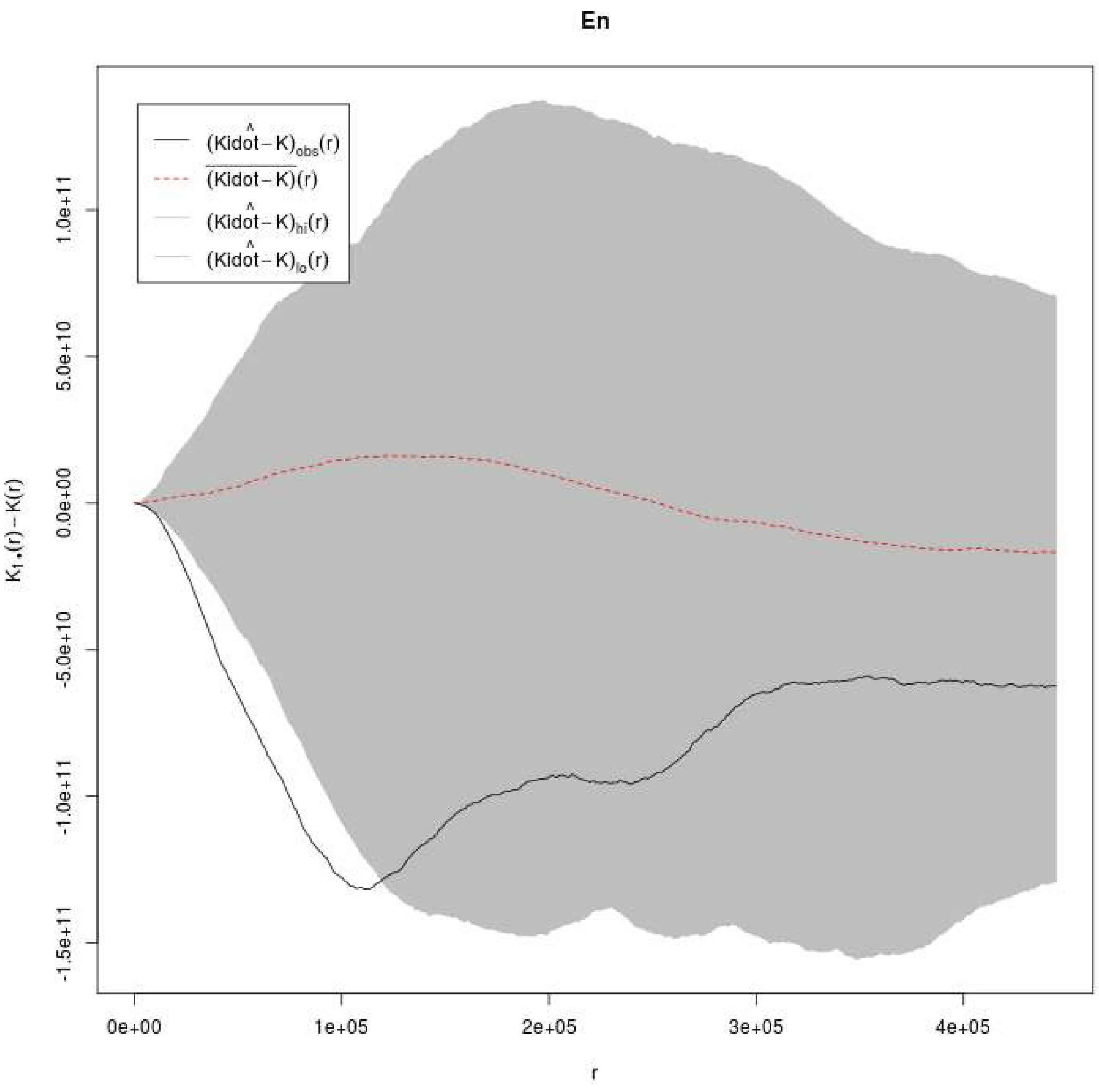
```r
## Abmigrating animals
abmfb <- apply(T1,1,mean)> (2/3)
abmbf <- apply(T2,1,mean)> (2/3)


##################
##
## 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:

```
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(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
```

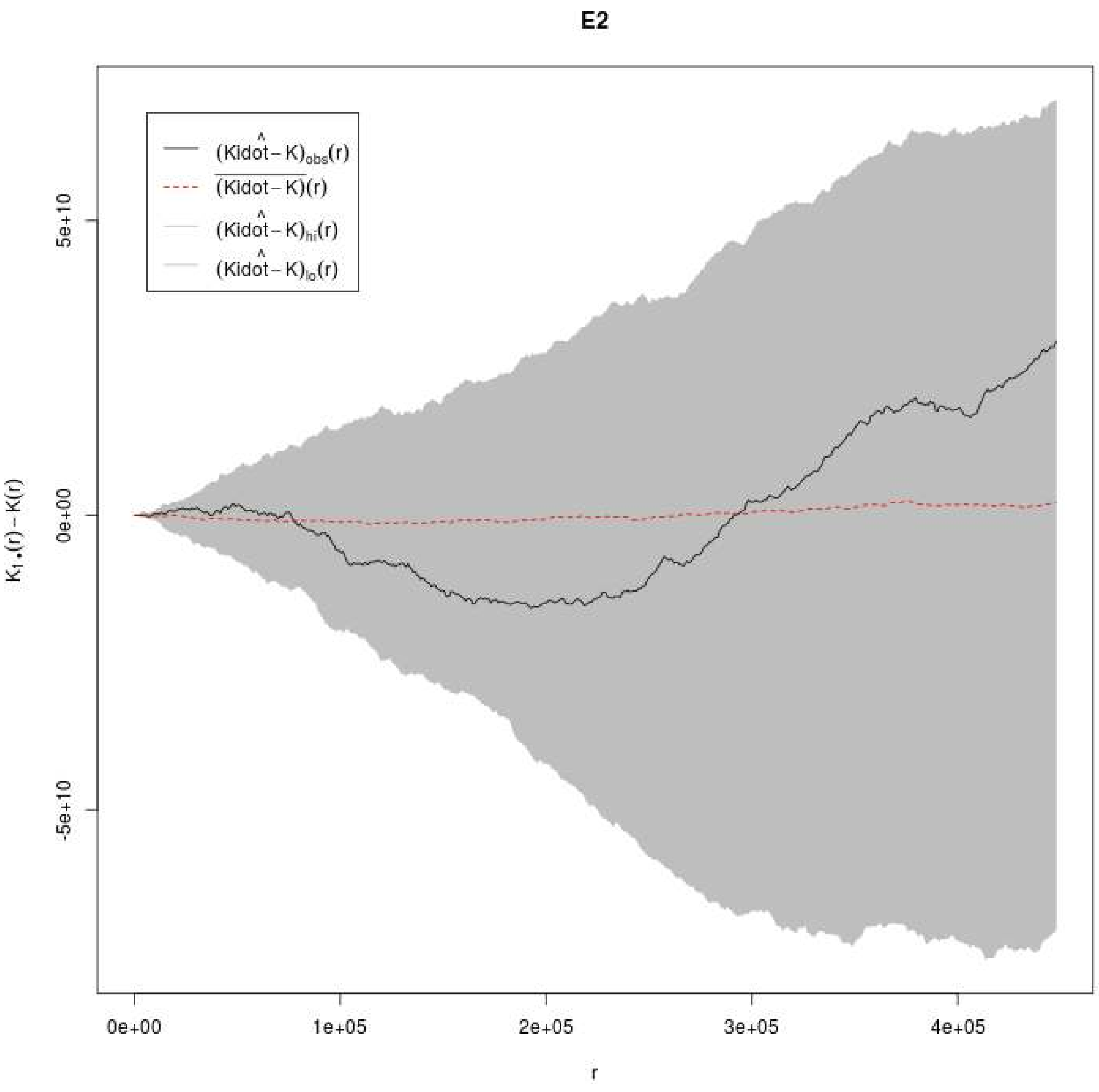


3.4 Random distribution of the two types of animals within the overlap area

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)))
```

```
## 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
```



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"
```

```r
## 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 modify the starting values of the parameters:

```r
parInit1t <- list(theta1a = c(-429848, -1225539,
                  -1058897, -299476, -1185661, -339494),
                  theta1b = c(-290987, -370117),
                  theta1c = c(-691426, -1190062),
                  theta1d = c(-286011, -624738),
                  theta1e = c(-264845, -404173, -459037),
                  theta1f = c(-732016, 211003, -704255,
                  943347, -736270, 27310, -244915),
                  theta2a = c(-1261815, -544118, 215340,
                  585184, -1242381, 820451),
                  theta2b = c(-810987, 456770, -574762, 166028),
                  theta2c = -497628,
                  theta2d = c(663207, -852554, 487832,
                 -383177),
                  theta2e = c(-79686, 656123, -748390,
                  920375, -999008, -999285, -18276),
                  psi22 = rep(0.96,4), psi21 = rep(0.018,4),
                  delta = rep(0.3126,4))

parInit2t <- list(theta1a = c(331777, 287922, -1172087, -324465,
                  129919, -1116989), theta1b = c(-105664, -545644),
                  theta1c = c(-1141727, -881130),
                  theta1d = c(-374846, -560183),
                  theta1e = c(-302361, -353503, -688120),
                  theta1f = c(-933325, 448871, -1050266, -348321,
                  358430, 10629, -727748),
                  theta2a = c(-1051310, -919978, 585145,
                  -374519, -473335, 32921),
                  theta2b = c(156275, -261841, 65729, -275447),
                  theta2c = -180564,
                  theta2d = c(69644, -482812, -127223, -213950),
                  theta2e = c(-80815, 79801, -89061, 203133, -696576,
                  -418571, 598242),
                  psi22 = rep(0.953,4), psi21 = rep(0.014,4), delta = rep(0.401,4))
```

We also change the log-posterior of the model:

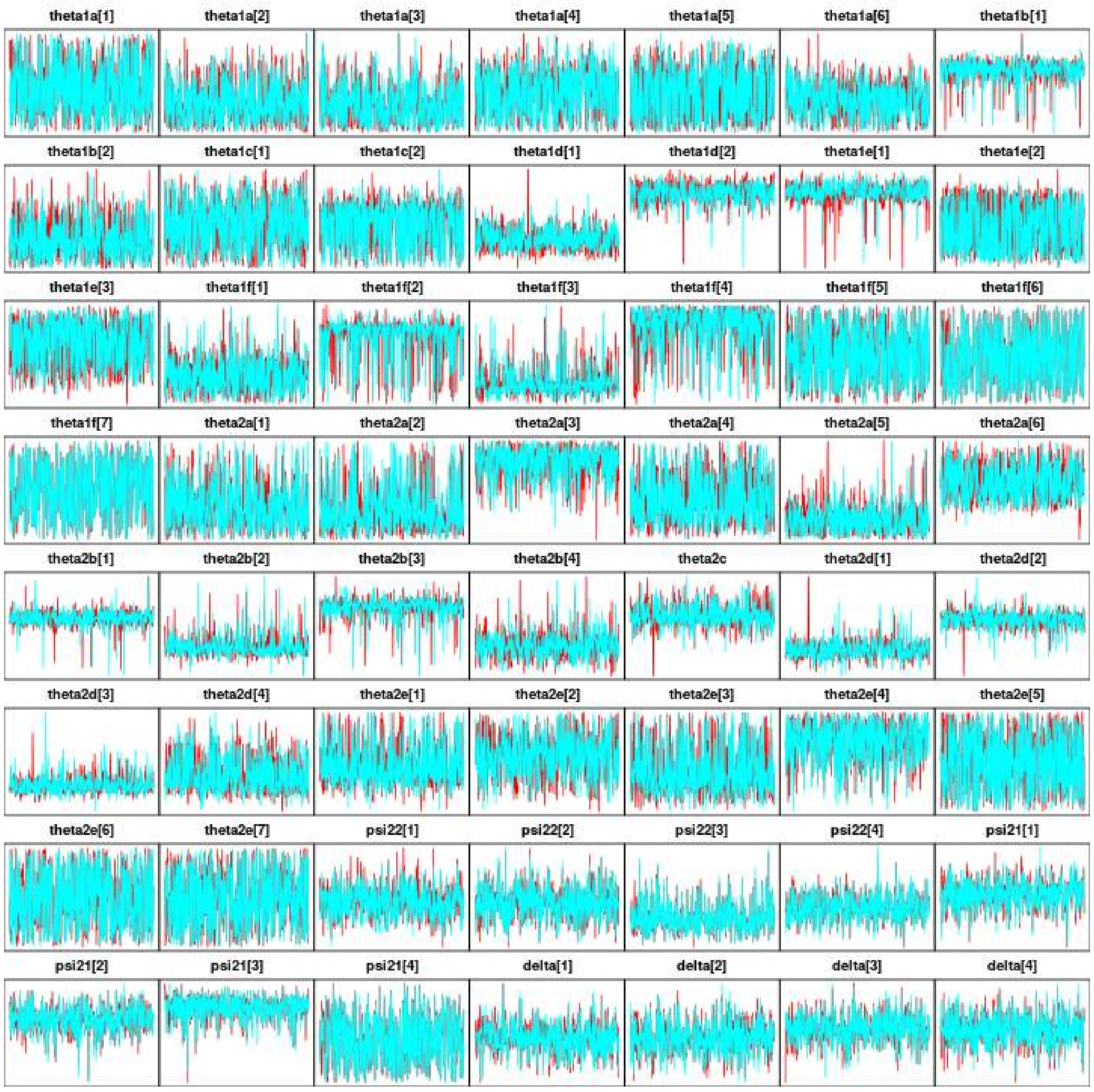
```r
logposteriorModelFlywaysTime <- function(par, lidat, ctrl)
{
    ## Coefficients
    theta1 <- c(par$theta1a, par$theta1b, par$theta1c, par$theta1d, par$theta1e,
                par$theta1f)
    theta2 <- c(par$theta2a, par$theta2b, par$theta2c, par$theta2d, par$theta2e)

    pa <- c(par$psi21,par$psi22,par$delta)
    if (any(pa<(-15))|any(pa>15))
        return(-Inf)

    psi22 <- invlogit(par$psi22)
    psi21 <- invlogit(par$psi21)
    delta <- invlogit(par$delta)

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

And finally, we launched the Metropolis algorithm. As before, since we just want to illustrate the calculations, we ran only two chains of 500 000 iterations each (which took about 12 hours of calculations):

```
gsm1b <- GeneralSingleMetropolis(parInit1t, lidat=lidat,
                                 logposterior = logposteriorModelFlywaysTime,
                                 lipum = lipum,
                                 listUpdating = listUpdating,
                                 nrepet = 500000, thinPar = 1000, saveEvery = 10000,
                                 fileSave = "saveFlywaysTime-1.rda")

gsm2b <- GeneralSingleMetropolis(parInit1t, lidat=lidat,
                                 logposterior = logposteriorModelFlywaysTime,
                                 lipum = lipum,
                                 listUpdating = listUpdating,
                                 nrepet = 500000, thinPar = 1000, saveEvery = 10000,
                                 fileSave = "saveFlywaysTime-2.rda")

rectealmodeltime <- c(gsm1b, gsm2b)
```

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

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



As before, the mixing properties are poor here: we used only two runs of a small number of iterations here to illustrate the method (mixing is much better with 2 million iterations and 5 chains).

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] 1309.47

(DIC_time <- DIC(rectealmodeltime, logposteriorModelFlywaysTime))

## [1] 1307.26
```

Note that here, the two DIC are similar; the model accounting for time is here slightly better, but the DIC difference is too small to justify the additional fit of 9 additional parameters, so that the original model is better. Also, recall that the dataset used to fit this model is not the same as the one used by

Guillemain et al., which explains the slight difference of results. Finally, the poor mixing in this last case also accounts for underestimation of the variance in deviance for this last model, and therefore to the underestimation of DIC.

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.

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