Assessing spatio-temporal variation in abundance: a flexible framework accounting for sampling bias and an application to Vulnerable Common Pochard (*Aythya ferina*)

Benjamin Folliot, Clement Calenge, Alain Caizergue, Adrien Tableau, Guillaume Souchay, Matthieu Guillemain, Jocelyn Champagnon.

February 2022

Contents

1 Introduction

This vignette describes how Folliot et al. (in prep.) estimated the population trend of Common pochard (*Aythya ferina*) in Europe from 2002 to 2012 (Folliot et al. *Quantifying spatio-temporal variation in abundance: a flexible framework accounting for sampling bias*). The package `pochardTrend` contains all the functions used in this paper, and is required to reproduce the calculations in this document. Note that due to copyright issues, we could not include the original dataset in this package, and we replaced the original dataset with a simulated dataset. However, note that the R object storing the results of the model fit on the original dataset is available in the package. Therefore, if the reader reproduces the calculations on the simulated dataset, they will not obtain the exact same model fit as the one stored in the dataset `fittedModel`.

This vignette also describes checks carried out to validate the model. To install this package, first install the package `devtools` and use the function `install_github` to install `pochardTrend`:

```
## If devtools is not yet installed, type
install.packages("devtools")

## Install the package caperpyogm
devtools::install_github("ClementCalenge/pochardTrend", ref="main")
```

Remark: on Windows, it is required to also install the Rtools (https://cran.r-project.org/bin/windows/Rtools/) on your computer to have a working `devtools` package (see https://www.r-project.org/nosvn/pandoc/devtools.html).

The package can then be loaded with:

```
library(pochardTrend)
```

2 Description of the data and model

2.1 The datasets

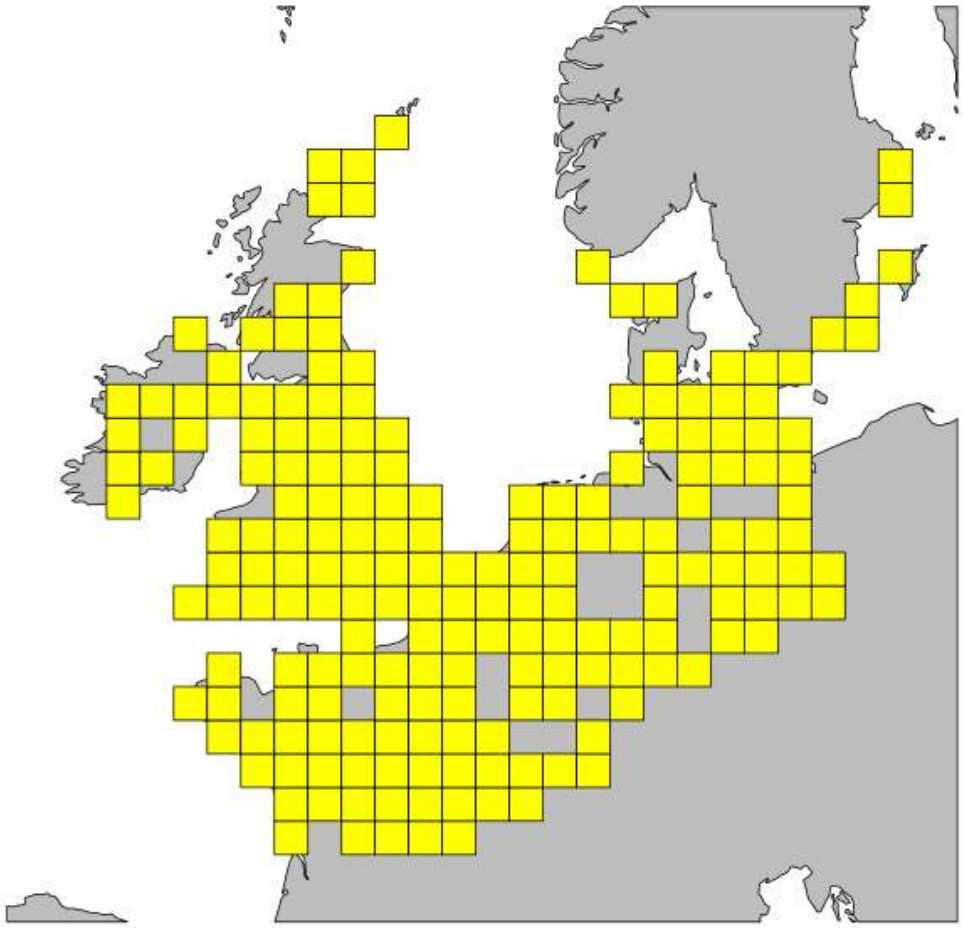
We work on the dataset pochard, which contains the results of counts carried out on 981 sites thoughout northern Europe:

```
head(pochard)

##   site count year idg    longq     latq
## 1    1     0 2010 112 8.366138 49.08187
## 2    1     0 2002 112 8.366138 49.08187
## 3    1     0 2004 112 8.366138 49.08187
## 4    1     0 2009 112 8.366138 49.08187
## 5    1     0 2006 112 8.366138 49.08187
## 6    1     7 2003 112 8.366138 49.08187
```

This dataset is a data.frame containing for each `year` and `site`, the result of the `count`. Due to the confidentiality policy of Wetlands International, we could not include the coordinates of the sites. Moreover, for the same reasons, we could not deliver the actual values of `count`: we had to simulate this vector of values. However, the location of the 75×75 grid cells containing the sites could be provided. We give in the column `idg` the ID of the grid cell containing the site, and in the columns `longq` and `latq` the longitude/latitude of the centroid of the grid cells. Note that only the latitude is used in the model of pochard trends, to assess possible differences in trends over the species range on a North-South

axis, due to migratory short-stopping (shortening of migration distance leading to increasing numbers at high latitudes and decreasing numbers at low latitudes) in response to increasing winter temperatures. Longitude is also included in the dataset to allow the graphical display of maps of cell centers. Note that the distribution of the grid cells over Europe can be displayed with the datasets `quadratMap` and `euromap`:

```
plot(euromap, col="grey")
plot(quadratMap, col="yellow", add=TRUE)

## old-style crs object detected; please recreate object with a recent sf::st_crs()
```



2.2 The model

It is supposed throughout this vignette that the reader is familiar with the model developed in this paper. Nevertheless, we will present a brief reminder on this model in this introduction.

Let N_{it} be the number of animals counted in the site i in year t. We suppose that N_{it} follows a Poisson

distribution with expectation λ_{it}:

$$N_{it} \sim \mathcal{P}(\lambda_{it})$$

We model this expectation with:

$$\log \lambda_{it} = \alpha_i + (\beta_0 + d_i) \times t + e_{it}$$

with e_{it} a gaussian residual characterizing site i and year t, with mean 0 and standard deviation σ_e, α_i is a site-specific intercept, β_0 is the fixed slope of the year, and d_i is a site random effect on this slope. This random effect follows a Gaussian distribution with a mean $b_{q(i)}$ characterizing the grid cell $q(i)$ containing the site i, and a standard deviation σ_d:

$$d_i \sim \mathcal{N}(b_{q(i)}, \sigma_d)$$

The grid cell mean random effect is itself modelled by a Gaussian distribution:

$$b_{q(i)} \sim \mathcal{N}(\gamma \times L_{q(i)}, \sigma_b)$$

with γ the slope of the latitude $L_{q(i)}$ of the grid cell $q(i)$.

3 Model fit and model checks

3.1 Model fit

The model can be fitted to the dataset `pochard` with the function `fitModel`. WARNING: this function is very long and may take several minutes to complete. Impatient users can skip this command, as the result is available in the dataset `fittedModel` of the package:

```
fittedModel <- fitModel(pochard$count, pochard$site, pochard$year,
                        pochard$idg, pochard$latq)
```

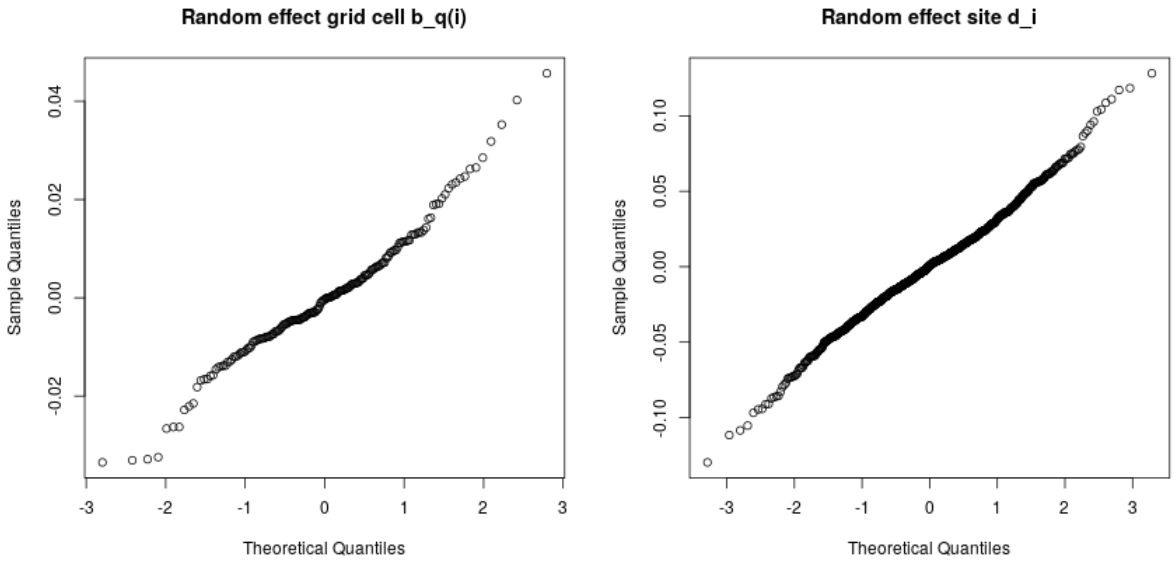
Note that the dataset `fittedModel` of the package contains the results of the model fit on the original (i.e., not simulated) dataset. In other words, if the reader executes this command line, the resulting object will not be identical to the object `fittedModel` of the package. We interpret this model in a later section.

3.2 Model checks

3.2.1 Gaussian distribution of the random effects

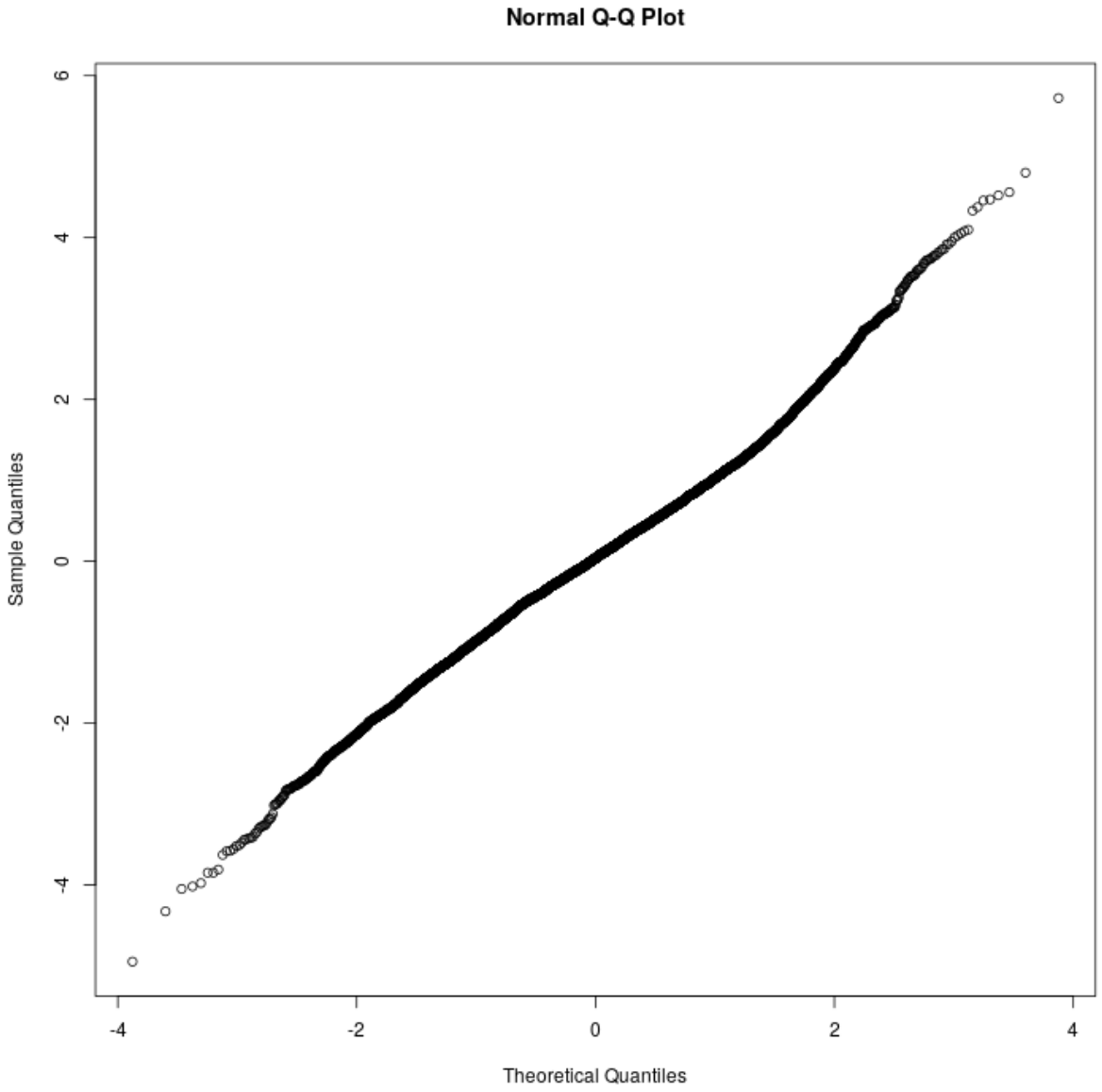
We first check that the distribution of the random effects d_i and $b_{q(i)}$ is Gaussian as expected with a quantile-quantile plot:

```
par(mfrow = c(1,2))
qqnorm(fittedModel$random$bqi[,2], main="Random effect grid cell b_q(i)")
qqnorm(fittedModel$random$di[,2], main="Random effect site d_i")
```



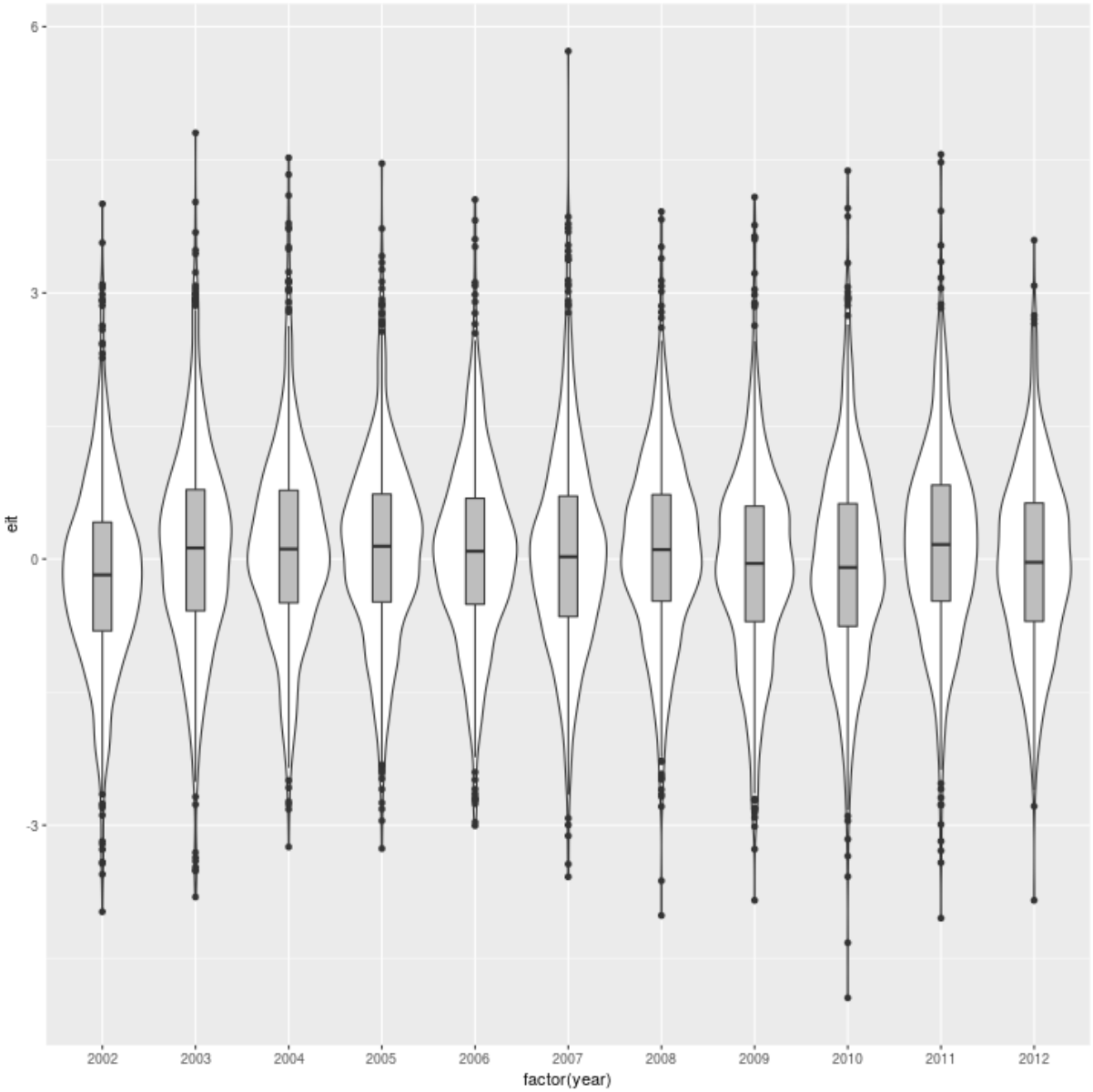
These plots are ok. We also show the distribution of the residuals e_{it}:

```
qqnorm(fittedModel$random$eit)
```



Note that these residuals seem randomly distributed as a function of the year:

```
library(ggplot2)
dae <- data.frame(year=pochard$year, eit=fittedModel$random$eit)
ggplot(dae,aes(x=factor(year), y=eit))+geom_violin()+
    geom_boxplot(width=0.2, fill="grey")
```



No problematic pattern can be highlighted regarding the distribution of random effects.

3.2.2 Absence of spatial autocorrelation in the random effects

We carried out a Moran test of the spatial autocorrelation of the grid cells random effects. We first define a relative neighbor graph based on the coordinates of the grid cell centroid (see the help page of `relativeneigh` in the package spdep):

```
## Coordinates of the centroids of the grid cells
xy <- st_coordinates(st_centroid(quadratMap))

## Warning in st_centroid.sf(quadratMap): st_centroid assumes attributes are constant over
geometries of x

## old-style crs object detected; please recreate object with a recent sf::st_crs()
## old-style crs object detected; please recreate object with a recent sf::st_crs()

## calculate the neighbourhood graph from the coordinates
## of the centroids
```

```
library(spdep)
gnb <- graph2nb(relativeneigh(xy), sym=TRUE)
## We remove a weird neighbouring relationship
gnb[[185]] <- 184L
gnb[[184]] <- 185L
```

We then carry out the Moran test:

```
## Moran test
moran.test(fittedModel$random$bqi[,2], nb2listw(gnb))
```

```
##
##  Moran I test under randomisation
##
## data:  fittedModel$random$bqi[, 2]
## weights: nb2listw(gnb)
##
## Moran I statistic standard deviate = -0.36695,
## p-value = 0.6432
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      -0.027363991      -0.005208333       0.003645436
```

This test is non-significant. We have removed all the major autocorrelation from this dataset. Note that another Moran test was carried out on the site effects in our paper. Since we could not include the coordinates of the sites in the package `pochardTrend` (due to confidentiality reasons), we cannot perform this test in this vignette (though the code is identical to the code used at the scale of the grid cell).

4 Model interpretation

We can show the main estimates from the model with:

```
fittedModel
```

```
##
## ******
## ** Object of class 'modelPochard'
##
##
## *** Fixed parameters:
##
##              Name    Parameter          SE
## 1           beta0  0.382289458 0.117768951
## 2 latitude_effect -0.008427484 0.002290245
##
## *** Standard deviation random effects:
##
##           Name ParamLogScale          SE ParamLinScale
## 1 log_sigmabqi    -3.4111830 0.279954378    0.03300214
## 2  log_sigmadi    -2.6061887 0.108505261    0.07381534
## 3   log_sigmae     0.1690289 0.009238973    1.18415433
##
##
```

```
## *** Quantiles of the distribution of the random effects:
##
##              Min          25%        Median         75%
## di  -0.12970939 -0.020239075  0.0008296851 0.019344593
## bqi -0.03350279 -0.007539205 -0.0003388064 0.006423657
## eit -4.94531260 -0.608935715  0.0376467952 0.692692014
##             Max
## di  0.12810309
## bqi 0.04567916
## eit 5.72387495
##
##
## This object is a list. Results returned by the function sdreport()
##  of the package TMB are available in the component $rawTMB
```

4.1 Estimate of the population decline

Therefore, the median slope of the relationship between year and log-count is:

$$0.3823 - 0.0084 \times \text{Latitude}$$

The latitude varies between 46° and 60°. Therefore the median slope of the relationship between year and log-count varies between these two values:

```
beta0 <- fittedModel$fixed$beta0
latitude_effect <- fittedModel$fixed$latitude_effect

beta0 + latitude_effect*c(46,60)


## [1] -0.005374797 -0.123359570
```

However, this variation on a log-scale is difficult to interpret. We can back transform these values to the linear scale, but this would only give the point estimates for the change rate.

We can calculate mean and standard error rate of change with simulations. Indeed, with a mixed effects model, the vector of estimated parameters $(\hat{\beta}_0, \hat{\gamma})$ is expected to follow a bivariate Gaussian distribution with a covariance matrix returned by `fitModel`. We can then simulate random vectors from this distribution, and back-transform the resulting slopes on a linear scale to calculate both the point estimate and the standard error of these rate of changes at the two latitudes:

```
## We load MASS to simulate a multivariate
library(MASS)

## Covariance matrix for the two coefficients
co <- cf <- fittedModel$rawTMB$cov.fixed[c("beta0","latitude_effect"),
                                        c("beta0","latitude_effect")]

## Simulate 100 000 values from the bivariate Gaussian distribution
simv <- mvrnorm(100000, c(beta0, latitude_effect), co)

## Decrease per year
## at 60 degrees latitude
dimpa60 <- (1-exp(apply(simv,1,function(x) x[2]*60+x[1])))

## at 46 degrees latitude
```

```r
dimpa46 <- (1-exp(apply(simv,1,function(x) x[2]*46+x[1])))


## point estimate and SE at 60 degrees
1-exp(beta0+latitude_effect*60)

##     beta0
## 0.1160542

## SE
sd(dimpa60)

## [1] 0.01838449

## point estimate and SE at 46 degrees
1-exp(beta0+latitude_effect*46)

##       beta0
## 0.005360379

sd(dimpa46)

## [1] 0.0136122
```

The counts decrease of 11.6% per year at 60° (SE = 1.8%), and of 0.54% per year at 46° (SE = 1.4%).

We can similarly estimate the decrease over the 11 years of the study period 2002-2012:

```r
dimpe60 <- (1-exp(apply(simv,1,function(x) 11*(x[2]*60+x[1]))))
dimpe46 <- (1-exp(apply(simv,1,function(x) 11*(x[2]*46+x[1]))))

## 60 degrees
1-exp(11*(beta0+latitude_effect*60))

##     beta0
## 0.7425566

sd(dimpe60)

## [1] 0.06142887

## 46 degrees
1-exp(11*(beta0+latitude_effect*46))

##      beta0
## 0.05740896

sd(dimpe46)

## [1] 0.1440494
```

The population decline averages 74% (SE=6%) over the whole period at 60° and 5.7% (SE=14%) at 46°.

Finally, we can calculate the yearly decline at the average latitude:

```
## Point estimate
1-exp(beta0+latitude_effect*mean(pochard$latq))

##      beta0
## 0.04872636

## SE
dimens <- (1-exp(apply(simv,1,function(x) x[2]*mean(pochard$latq)+x[1])))
sd(dimens)

## [1] 0.005754989
```

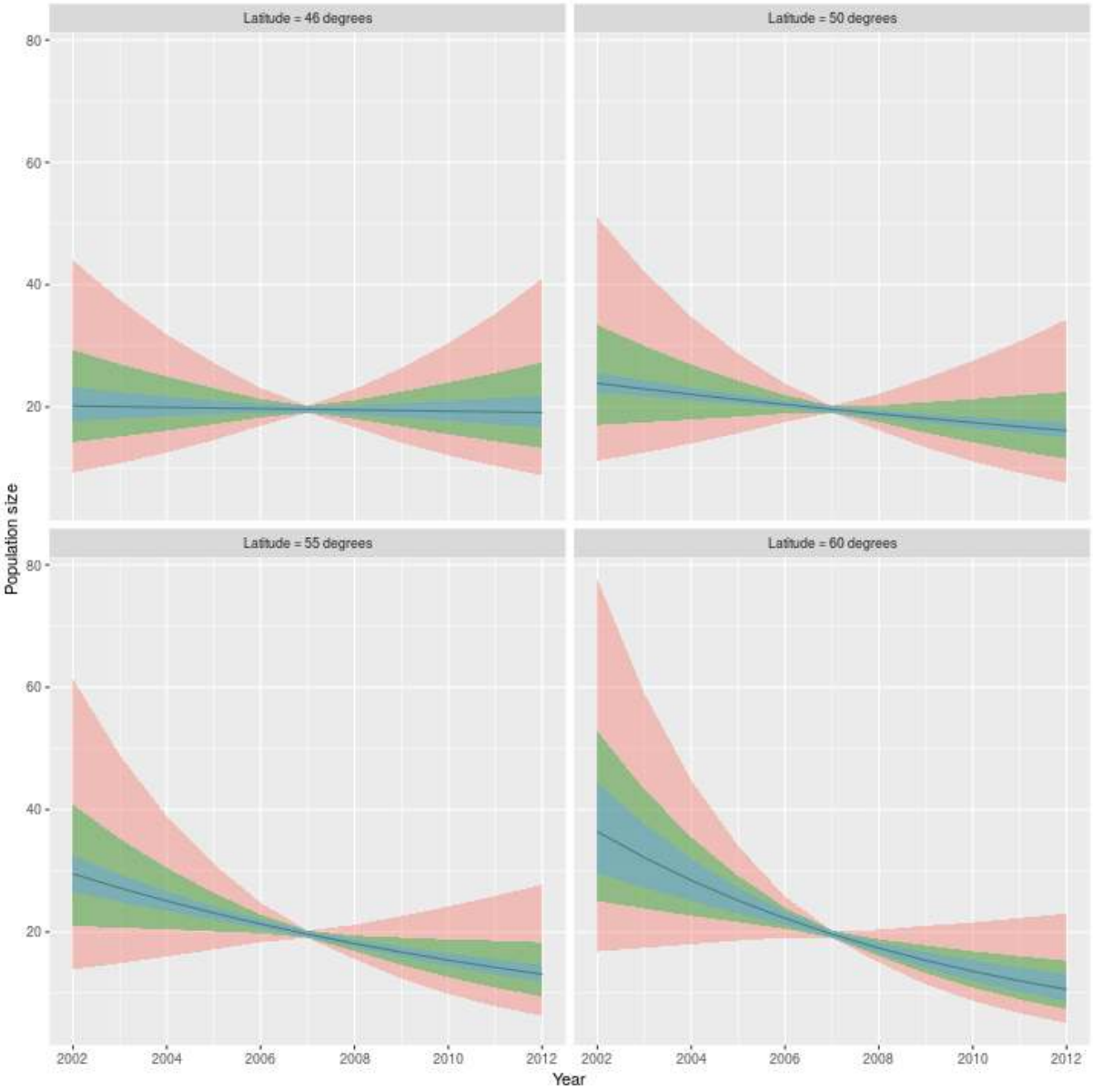
The yearly decline is equal to 4.9% (SE=0.6%), which represents an overall decline of:

```
## Point estimate
1-exp(11*(beta0+latitude_effect*mean(pochard$latq)))

##     beta0
## 0.4227551

## SE
dimenso <- (1-exp(apply(simv,1,function(x) 11*(x[2]*mean(pochard$latq)+x[1]))))
sd(dimenso)

## [1] 0.03854098
```

42% (SE = 4%).

We can show the estimated trend with:

```
lat <- seq(45, 60, by=5)
lat[1] <- 46

showTrends(fittedModel, 2002:2012,lat)+theme(legend.position = "none")
```



See the paper for a description of this figure.

4.2 Estimate of the variance parameters (and their SE)

To increase the stability of the estimation process, the function `fitModel` log-transforms the standard deviation parameters (i.e. the standard deviation of random effects). Therefore, the results below present the value of the logarithm of the standard deviation parameters:

```
fittedModel

##
## ******
## ** Object of class 'modelPochard'
##
##
## *** Fixed parameters:
##
##              Name    Parameter          SE
```

```
## 1           beta0  0.382289458 0.117768951
## 2 latitude_effect -0.008427484 0.002290245
##
## *** Standard deviation random effects:
##
##           Name ParamLogScale          SE ParamLinScale
## 1 log_sigmabqi    -3.4111830 0.279954378    0.03300214
## 2  log_sigmadi    -2.6061887 0.108505261    0.07381534
## 3   log_sigmae     0.1690289 0.009238973    1.18415433
##
##
## *** Quantiles of the distribution of the random effects:
##
##             Min          25%        Median         75%
## di  -0.12970939 -0.020239075  0.0008296851 0.019344593
## bqi -0.03350279 -0.007539205 -0.0003388064 0.006423657
## eit -4.94531260 -0.608935715  0.0376467952 0.692692014
##            Max
## di  0.12810309
## bqi 0.04567916
## eit 5.72387495
##
##
## This object is a list. Results returned by the function sdreport()
##  of the package TMB are available in the component $rawTMB
```

The results show the point estimate of the standard deviation, but as in the last section, it is difficult to back-transform the standard errors of these SD – which are measured on a log-scale – to a linear scale. We therefore use an approach similar to the one used in the last section: we simulate values of the log-SD from a Gaussian distribution on the log-scale and back-transform them to the linear scale to calculate the standard errors on the linear scale (these SE are the values displayed in the paper):

```
## sigma_bqi
## point estimate
mean(exp(rnorm(100000, fittedModel$rawTMB$par.fixed["log_sigmabqi"],
               sqrt(fittedModel$rawTMB$cov.fixed["log_sigmabqi","log_sigmabqi"]))))

## [1] 0.03427011

## SE
sd(exp(rnorm(100000, fittedModel$rawTMB$par.fixed["log_sigmabqi"],
             sqrt(fittedModel$rawTMB$cov.fixed["log_sigmabqi","log_sigmabqi"]))))

## [1] 0.009818696

## sigma_di
## point estimate
mean(exp(rnorm(100000, fittedModel$rawTMB$par.fixed["log_sigmadi"],
               sqrt(fittedModel$rawTMB$cov.fixed["log_sigmadi","log_sigmadi"]))))

## [1] 0.07422764

## SE
sd(exp(rnorm(100000, fittedModel$rawTMB$par.fixed["log_sigmadi"],
             sqrt(fittedModel$rawTMB$cov.fixed["log_sigmadi","log_sigmadi"]))))
```

```
## [1] 0.008078016
```

```
## sigma_e
## point estimate
mean(exp(rnorm(100000, fittedModel$rawTMB$par.fixed["log_sigmae"],
               sqrt(fittedModel$rawTMB$cov.fixed["log_sigmae","log_sigmae"]))))
```

```
## [1] 1.184217
```

```
## SE
sd(exp(rnorm(100000, fittedModel$rawTMB$par.fixed["log_sigmae"],
             sqrt(fittedModel$rawTMB$cov.fixed["log_sigmae","log_sigmae"]))))
```

```
## [1] 0.01094829
```