Estimation of the seroprevalence and prevalence of active infection of the brucellosis in a population of alpine ibex in the Bargy Mountains (French Alps), from 2012 to 2018

Clement Calenge, Sebastien Lambert, Elodie Petit, Anne Thebault
Emmanuelle Gilot-Fromont, Carole Toigo, Sophie Rossi.

November 2020

Contents

1 Introduction

The package BIB contains the R code used to fit the model described in the paper of Calenge et al. (in prep.). The aim of this paper is to estimate the seroprevalence and prevalence of active infection of Alpine ibexes by the brucellosis in the Bargy mountains (French Alps). BIB stands for "Brucellosis of Ibexes in Bargy mountains".

To install this package, first install the package `devtools` and use the function `install_github` to install `BIB`:

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

## Install the package caperpyogm
devtools::install_github("ClementCalenge/BIB")
```

It is supposed throughout this vignette that the reader is familiar with the model developed in this paper, as well as with the results. This vignette just aims at giving the code needed to reproduce the results, tables and figures of the paper and its supplementary material.

Note that since the data used for the model are sensitive, we did not have the authorization to include the data used to fit the model (though discussions are still underway to include this dataset in the package). We therefore simulated a dataset with similar properties, so that the user can try the code on this simulated dataset. The results presented in the package are those obtained with this simulated dataset.

We first load the library `BIB`, which contains the data and the functions that we used to fit the model:

```
library(BIB)
```

All the datasets of the package are "lazy-loaded", i.e. they are immediately available:

```
str(BruData)

## List of 23
##  $ CensusesNmark             : int [1:220] 0 2 1 2 0 1 0 0 1 0 ...
##  $ CensusesNind              : int [1:220] 1 4 6 9 9 27 7 3 3 2 ...
##  $ CensusesYear              : num [1:220] 1 1 1 1 1 1 1 1 1 1 ...
##  $ CensusesNdays             : int 220
##  $ CapturesN                 : int 118
##  $ CapturesYear              : num [1:118] 1 2 1 1 1 1 1 1 1 1 ...
##  $ CapturesSeroplus          : num [1:118] 1 0 1 1 1 1 1 0 1 1 ...
##  $ CaptureNumber             : int [1:6] 14 14 9 5 6 5
##  $ RecapturesN               : int 14
##  $ RecapturesSeroplus        : num [1:14] 0 0 1 0 1 0 0 0 0 0 ...
##  $ RecapturesTimeSinceCapt   : num [1:14] 3.682 5.674 3.455 4.663 0.943 ...
##  $ RecapturesDurationContamin: num [1:14, 1:6] 0 1 1 0 0 0 0 0 0 0 ...
##  $ BacterialNanalyzed        : num [1:51] 2 6 1 6 2 5 1 6 5 2 ...
##  $ BacterialNpositive        : int [1:51] 0 0 0 2 0 4 1 5 4 0 ...
##  $ BacterialLogFC            : num [1:51] -2 -2.148 -2.714 0.573 -2.471 ...
##  $ BacterialNanimals         : int 51
##  $ BacterialAge              : num [1:51] 3.5 -1.5 6.5 4.5 0.5 1.5 5.5 -3.5 -1.5 3.5 ...
##  $ CenSeroplusAge            : num [1:64] -1.5 0.5 0.5 -1.5 -0.5 7.5 0.5 0.5 0.5 0.5 ...
##  $ CenSeroplusLogFC          : num [1:64] 0.492 1.186 1.186 0.492 1.186 ...
##  $ CenSeroplusYear           : num [1:64] 1 3 3 1 2 1 3 3 4 3 ...
```

```
##  $ CenSeroplusNanimals       : int 64
##  $ PopSize                   : num [1:6] 543 293 355 248 254 388
##  $ Nyears                    : num 6
```

This list contains the simulated data with a similar structure to those that we used to fit the model. The following elements are available:

- `CensusesNmark`: Number of marked animals during each census day.
- `CensusesNind`: Total number of marked animals during each census day.
- `CensusesYear`: Year corresponding to each census day.
- `CensusesNdays`: Number of census days.
- `CapturesN`: Number of captured animals.
- `CapturesYear`: Year corresponding to each captured animal.
- `CapturesSeroplus`: Whether each captured animal is seropositive.
- `CaptureNumber`: Number of captured animals during each year.
- `RecapturesN`: Number of recaptured animals.
- `RecapturesSeroplus`: Whether each recaptured animal is seropositive.
- `RecapturesTimeSinceCapt`: For each recaptured animal, the time spent since the first capture.
- `RecapturesDurationContamin`: A matrix describing for each recaptured animal (rows), whether the year in column belonged to the period between the capture and the recapture.
- `BacterialNanalyzed`: The number of analyzed organs for each animal in the bacterial cultures dataset.
- `BacterialNpositive`: The number of organs detected as infected for each animal in the bacterial cultures dataset.
- `BacterialLogFC`: The logarithm of the CFT titer for each animal in the bacterial cultures dataset.
- `BacterialNanimals`: The number of animals in the bacterial cultures dataset.
- `BacterialAge`: The age of each animal in the bacterial dataset.
- `CenSeroplusAge`: The age of each animal for which the CFT titer was available in the capture dataset.
- `CenSeroplusLogFC`: The log of the CFT titer for each animal for which the CFT titer was available in the capture dataset.
- `CenSeroplusYear`: The year for each animal for which the CFT titer was available in the capture dataset.
- `CenSeroplusNanimals`: The number of animals for which the CFT titer was available in the capture dataset.
- `PopSize`: Point estimates of the population size during each year of the study.
- `Nyears`: The number of year of the monitoring (6 years in this study).

2 Model fit

2.1 Fit with JAGS

We have fitted 4 models in the papers. These models have all been programmed in JAGS and their code is available as datasets in the package. For example, for the main model used in the paper, the code can be retrieved by:

```
cat(ModelCatalyticYear)
```

We do not show the result of this command to save some space in this vignette, but the reader is encouraged to execute this command to read the code of the model. Other models are coded similarly (see the help page `?ModelCatalyticYear` for a description of the model codes). The model can be fitted with the package `rjags`:

```
library(rjags)

obmcy <- jags.model(textConnection(ModelCatalyticYear),
                    data=BruData, n.chains = 5,
                    inits =
                        list(e=as.numeric(BruData$BacterialNpositive>0),
                             eta=0.1))

SamplesModelCatalyticYear <-
    coda.samples(obmcy, n.iter=5000, thin=5,
                 variable.names = c("pitu","pitm","pit", "gammat",
                                    "eta","beta0","beta1","beta2","beta3",
                                    "sigmaf","probad","probai",
                                    "wt","lambdat", "omegat"))
```

The other models have been fitted similarly, and the resulting MCMC samples are available as datasets of the package (see the help page `SamplesModelCatalyticYear`).

2.2 Mixing

Once the MCMC samples have been obtained, we can plot the chain for a visual examination of the mixing:

```
library(bayesplot)

## This is bayesplot version 1.7.2
## - Online documentation and vignettes at mc-stan.org/bayesplot
## - bayesplot theme set to bayesplot::theme_default()
##    * Does _not_ affect other ggplot2 plots
##    * See ?bayesplot_theme_set for details on theme setting

library(MCMCvis)
cm <- MCMCchains(SamplesModelCatalyticYear,
                 params=c("pitu","pitm","pit", "gammat",
                          "eta","beta0","beta1","beta2","beta3",
                          "sigmaf","probad","probai",
                          "wt","lambdat", "omegat"),
                 mcmc.list=TRUE)
mcmc_trace(cm)
```

We also calculate the Gelman diagnostic for the parameters. Again, to save some space, we do not show the results of this diagnostic in this vignette and leave it to the reader to check the correct mixing based on this diagnostic:

```
gelman.diag(SamplesModelCatalyticYear, multivariate=FALSE)
```

2.3 Goodness of fit

We then checked the goodness of fit of the model, using the approach recommended by Gelman and Meng (1996, see paper). Each MCMC iteration generated a sampled value $\boldsymbol{\theta}_r$ of the vector of parameters $\boldsymbol{\theta}$ of the model. For each simulated value $\boldsymbol{\theta}r$, we simulated a replication of the captures, bacterial cultures and censuses datasets. We then compared summary statistics calculated on the observed datasets to the distribution of these summary statistics calculated on the simulated datasets. More specically, we used as summary statistics: (i) the number of marked females detected during each day of census in the censuses dataset, (ii) the proportion of unmarked females detected as seropositive each year in the seroprevalence dataset, (iii) the same proportion calculated only on marked females, (iv) the log-titer of the CFT for each seropositive female. This check is programmed in the function `simulateGOF`:

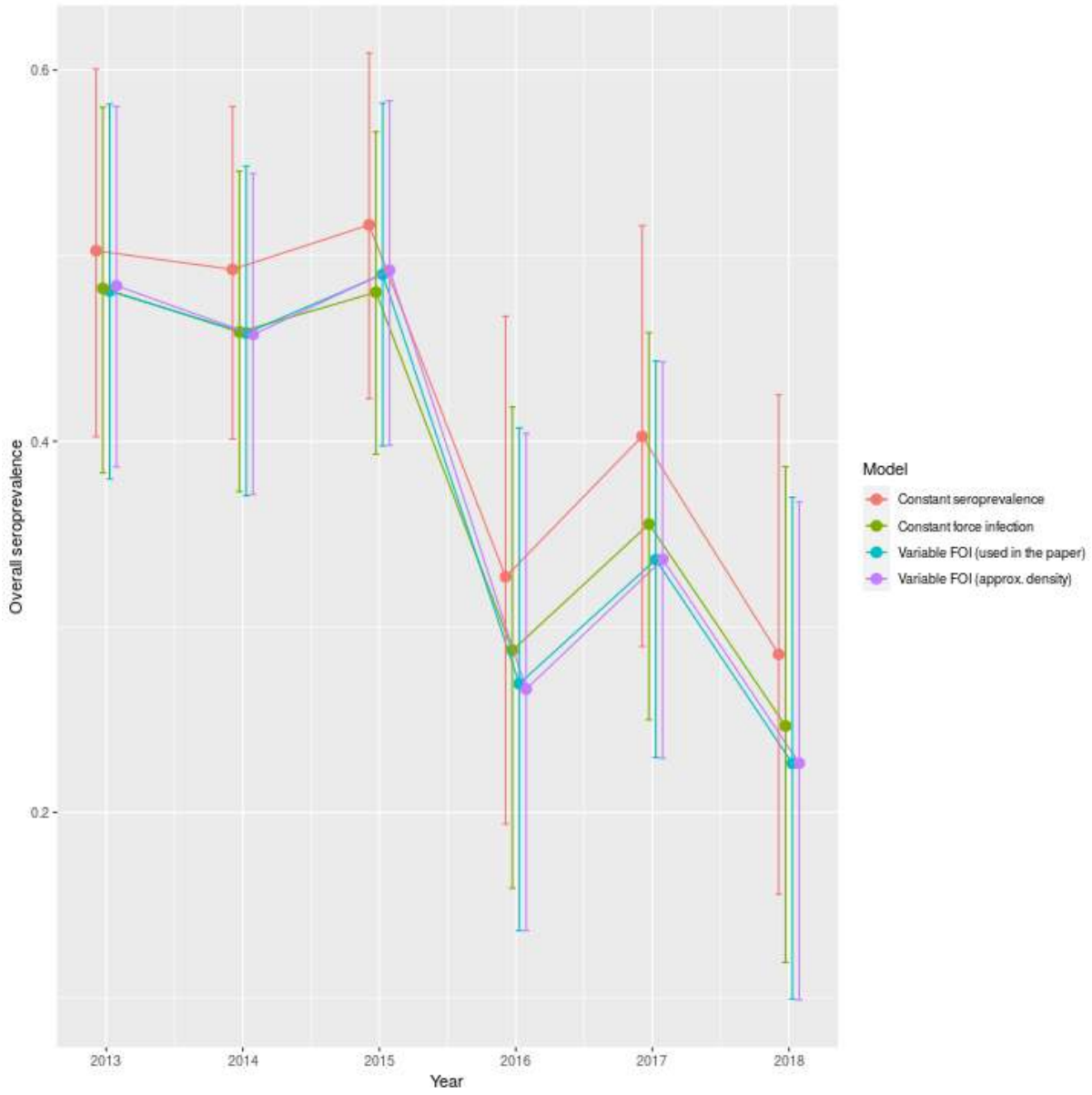
```
simg <- simulateGOF(SamplesModelCatalyticYear, BruData)
simg

## ######################################
## ##
## ## Goodness of fit of the BIB model
## ##
## Number of simulations: 5000
## Probability used for the credible intervals in this check: 80 %
## ##
## Percentage of the simulate CI including
## the obs. value of number of marked animals: 92 %
## ##
## Comparison of the observed percentage of seropositive animals among
## captured unmarked animals with the simulated CI for each year:
##   obs   ICsim
## 1  55 [39-71]
## 2  58 [39-73]
## 3  65 [46-81]
## 4  38 [12-62]
## 5  54 [31-77]
## 6  29  [0-71]
## ##
## ##
## Comparison of the observed percentage of seropositive animals among
## captured marked animals with the simulated CI (all years pooled):
##   obs  ICsim
## 1  14 [7-36]
## ##
## ##
## Percentage of logFC values in the simulated CI for analysed animals: 92 %
```

2.4 Model comparison

As explained before, we have fitted similarly the four alternative models, and the resulting MCMC samples are available as datasets of the package (all the datasets of the package with a name starting with `SampleModel`). We have programmed the function `totalPrevalence` to estimate the overall prevalence with each model based on these MCMC sample datasets. This is the code we used to generate the figure in the supplementary material to compare the models:

```
library(ggplot2)
df <- do.call(rbind, lapply(list(SamplesModelCatalyticYear,
                                 SamplesModelConstPrev,
                                 SamplesModelCatalytic,
                                 SamplesModelDensity),
                            totalPrevalence))

df$Model <- factor(c(rep("Variable FOI (used in the paper)", 6),
                                 rep("Constant seroprevalence", 6),
                                 rep("Constant force infection", 6),
                                 rep("Variable FOI (approx. density)", 6)),
                                levels=c("Constant seroprevalence",
                                         "Constant force infection",
                                         "Variable FOI (used in the paper)",
                                         "Variable FOI (approx. density)"))
```

```
pd <- position_dodge(0.2)
ggplot(df,aes(x=Year, y=Prevalence, col=Model, group=Model))+
     geom_point(size=3, position=pd)+geom_line(position=pd)+
     geom_errorbar(aes(ymin=LowerBound, ymax=UpperBound), width=0.2, position=pd)+
     xlab("Year")+ylab("Overall seroprevalence")
```



The equivalent of this figure, obtained with the actual dataset, is interpreted in the supplementary material of the paper.

3 Results

3.1 Summary tables

We have programmed the function `getTable2and3` to present the point estimates and 90% credible intervals for the key parameters of the model, as displayed in table 2 and 3 of the paper. We can therefore obtain the table 2, showing the results for the overall seroprevalence, the seroprevalence of

marked animals, the seroprevalence of unmarked animals, and the proportion of marked animals in the population:

```
getTable2and3(SamplesModelCatalyticYear)
```

```
##   Year Overall.Prevalence Seroprev.marked Seroprev.unmarked
## 1 2013   0.48 [0.35-0.61]         0 [0-0]   0.54 [0.4-0.69]
## 2 2014   0.46 [0.34-0.57]   0.05 [0-0.15]   0.57 [0.43-0.7]
## 3 2015   0.49 [0.37-0.61]    0.1 [0-0.23]  0.64 [0.49-0.78]
## 4 2016   0.27 [0.11-0.44]   0.06 [0-0.16]  0.38 [0.15-0.63]
## 5 2017    0.34 [0.2-0.47]   0.04 [0-0.12]  0.52 [0.31-0.73]
## 6 2018   0.23 [0.08-0.41]   0.01 [0-0.04]   0.33 [0.11-0.6]
##        Prop.Marked
## 1 0.12 [0.09-0.14]
## 2 0.21 [0.18-0.24]
## 3 0.28 [0.22-0.34]
## 4 0.34 [0.31-0.37]
## 5 0.38 [0.35-0.41]
## 6 0.32 [0.29-0.35]
```

And similarly, we can calculate table 3, showing the results for the proprtion of seropositive animals with active infection and the proportion of the whole population with active infection:

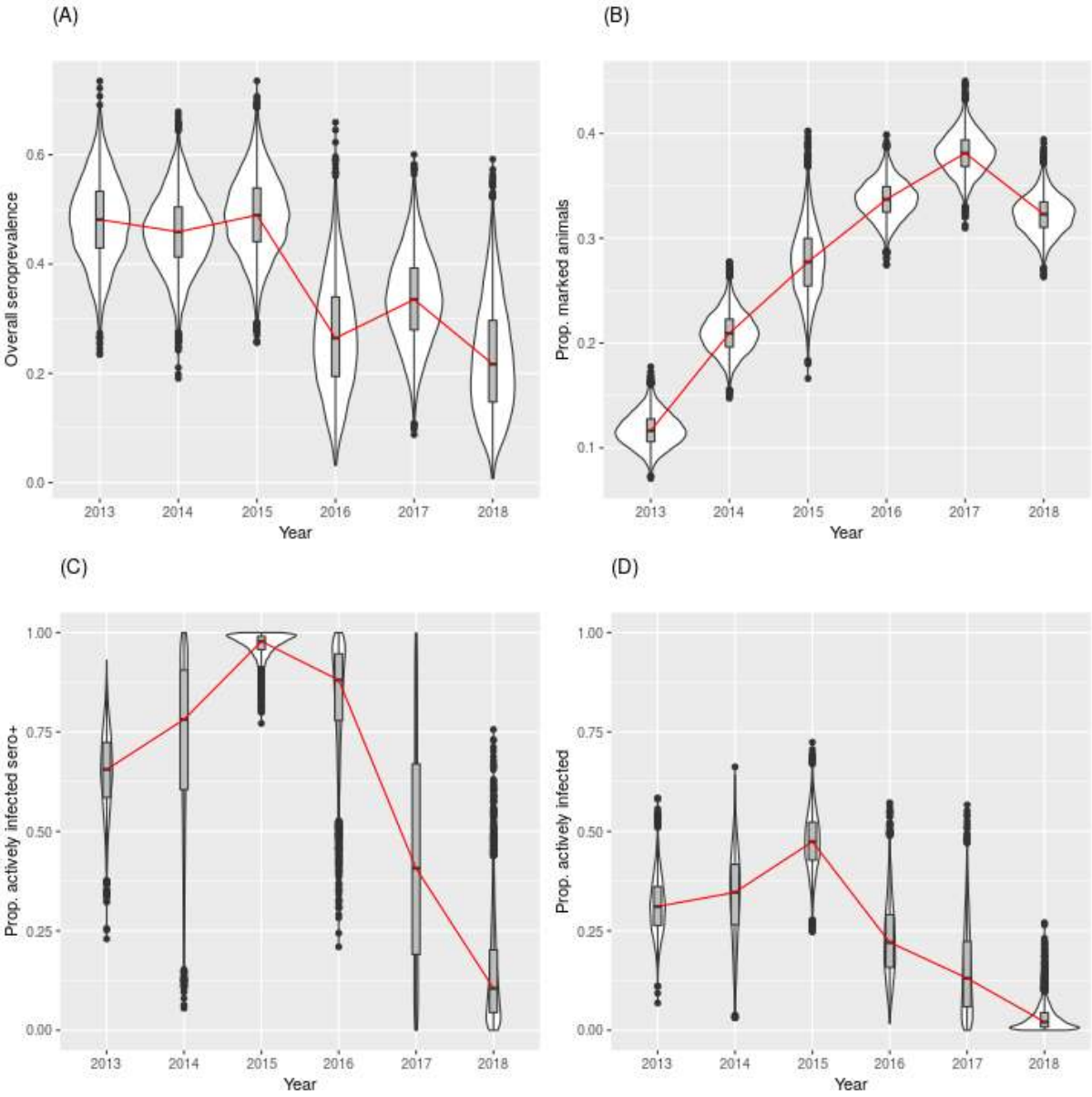
```
getTable2and3(SamplesModelCatalyticYear, table2=FALSE)
```

```
##   Year Prop.sero.active.inf Overall.prop.active.inf
## 1 2013     0.65 [0.48-0.81]         0.31 [0.2-0.43]
## 2 2014     0.74 [0.34-0.98]        0.34 [0.14-0.51]
## 3 2015        0.97 [0.91-1]        0.48 [0.36-0.59]
## 4 2016     0.85 [0.59-0.99]         0.23 [0.09-0.4]
## 5 2017     0.44 [0.03-0.93]        0.15 [0.01-0.35]
## 6 2018     0.14 [0.01-0.39]            0.03 [0-0.1]
```

3.2 Figures of the paper

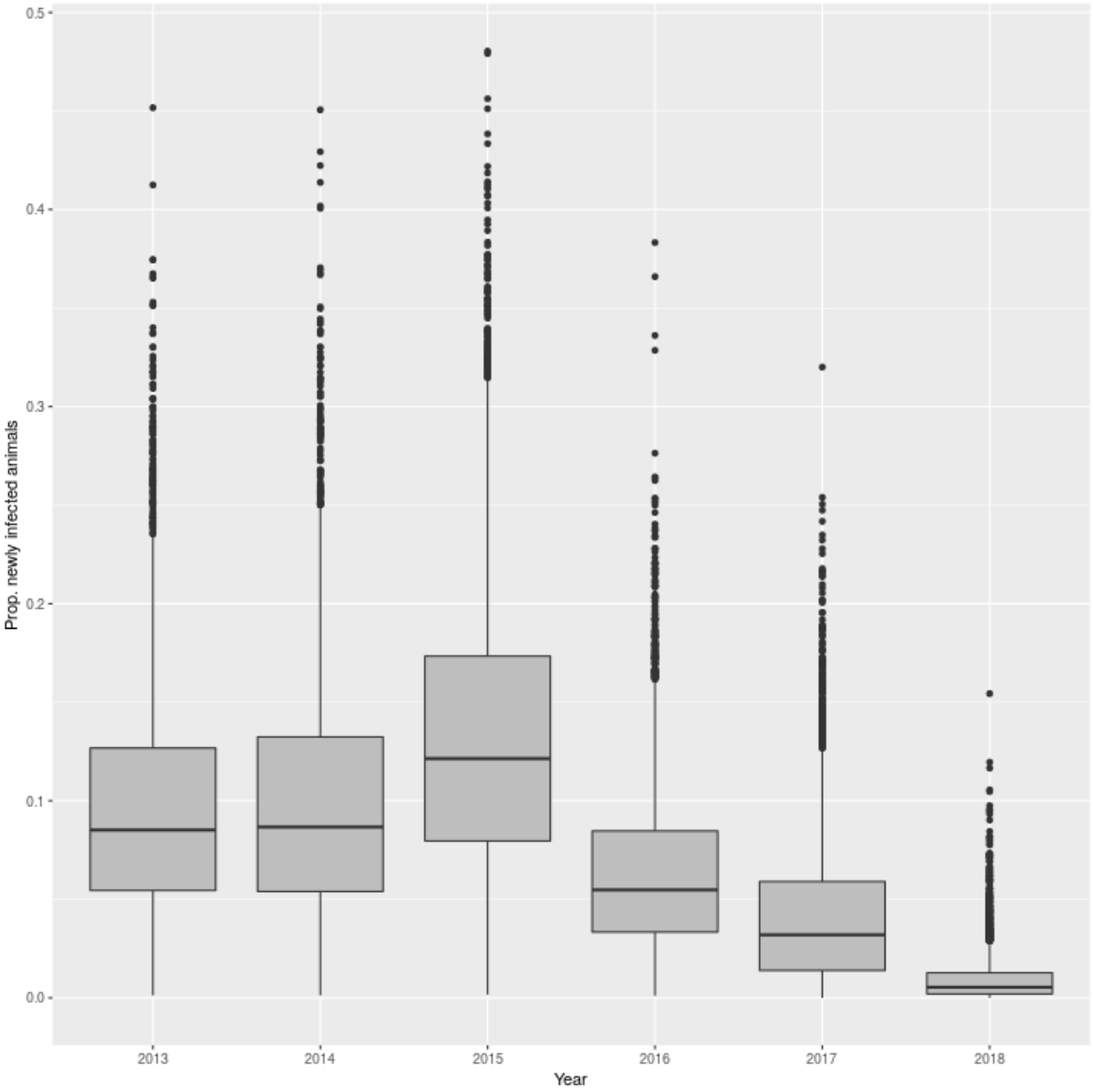
Similarly, we have programmed the function `showSummary` to draw the figure 3 of the paper. This function displays the posterior distribution of (A) the overall seroprevalence of the disease, (B) the proportion of marked animals in the population, (C) the proportion of actively infected animals among seropositive animals, and (D) the proportion of actively infected animals:

```
showSummary(SamplesModelCatalyticYear)
```



The function `showFOI` can be used to draw the figure 4 of the paper, showing how the force of infection is estimated to vary among years:

```
showFOI(SamplesModelCatalyticYear)
```



3.3 Constant capture effort over the core area of the massif

As noted in the paper, we assessed the assumption of constant capture effort over the core area of the massif (i.e., no spatial variation): we divided our study area into two halves: a northeastern part and a southwestern part. We focused on the censuses dataset, and we used our model to test whether the proportion of marked animals observed in these two areas could have been expected under our non-spatialized model. The total numbers of animals observed during the censuses, as well as the number of marked animals among these animals, are available for each year and zone in the dataset `captureSpatial`:

```
capturesSpatial
```

```
##        Year Ntot Nmarked      Zone
## 2013_1 2013  499      24 Southwest
## 2013_2 2013  299       8 Northeast
## 2014_1 2014  249      72 Southwest
## 2014_2 2014  187      15 Northeast
## 2015_1 2015   61      28 Southwest
## 2015_2 2015   54       6 Northeast
```

```
## 2016_1 2016  285     152 Southwest
## 2016_2 2016  218      36 Northeast
## 2017_1 2017  303     163 Southwest
## 2017_2 2017  266      62 Northeast
## 2018_1 2018  326     184 Southwest
## 2018_2 2018  217      46 Northeast
```

The test of constant capture effort is implemented in the function `testSpatial`:

```
testSpatial(SamplesModelCatalyticYear, capturesSpatial)

##    Year Ntot     Nmarked      Zone             CI
## 1  2013  499   24 (5\\%) Southwest  [8\\%--16\\%]
## 2  2013  299    8 (3\\%) Northeast  [7\\%--17\\%]
## 3  2014  249  72 (29\\%) Southwest [15\\%--28\\%]
## 4  2014  187   15 (8\\%) Northeast [14\\%--28\\%]
## 5  2015   61  28 (46\\%) Southwest [15\\%--43\\%]
## 6  2015   54   6 (11\\%) Northeast [15\\%--43\\%]
## 7  2016  285 152 (53\\%) Southwest [27\\%--40\\%]
## 8  2016  218  36 (17\\%) Northeast [27\\%--41\\%]
## 9  2017  303 163 (54\\%) Southwest [32\\%--45\\%]
## 10 2017  266  62 (23\\%) Northeast [31\\%--45\\%]
## 11 2018  326 184 (56\\%) Southwest [26\\%--39\\%]
## 12 2018  217  46 (21\\%) Northeast [25\\%--40\\%]
```

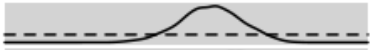
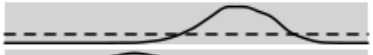
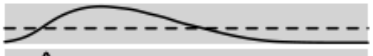
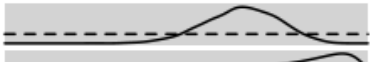
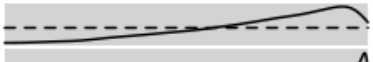
This table shows, for each year and each zone, the credible interval on the proportion of marked animals expected under our model supposing a constant capture effort.

3.4 Sensitivity of the estimation to the prior distribution

We checked the sensitivity of the estimation to the prior distribution. We used the functions `overlapPriorPost` and `plotOverlap` to calculate the coefficient τ measuring the overlap between the prior and the posterior distribution for each parameter of interest in the model. We show below the code used to generate the figure in the appendix:

```
## Use the MCMC samples from the model ModelCatalyticYear
rs <- do.call(rbind, SamplesModelCatalyticYear)

## Detects the name of the parameters for which tau should be calculated
library(stringr)
nam <- c("pitu","gammat","probad","wt","beta","^eta","sigmaf")
nam <- unlist(lapply(nam, function(z) colnames(rs)[str_detect(colnames(rs),z)]))


## For each parameter, calculates the value of tau:

lb <- sapply(nam, function(na) {
    if (any(sapply(c("pitu","gammat","probad","probai","wt"), function(x) str_detect(na, x))))
        tau <- overlapPriorPost(SamplesModelCatalyticYear, na, prior=1, from=0, to=1)
    if (str_detect(na, "beta"))
        tau <- overlapPriorPost(SamplesModelCatalyticYear, na)
    if (str_detect(na, "eta"))
        tau <- overlapPriorPost(SamplesModelCatalyticYear, na, prior=1/100, from=0, to=100)
    if (str_detect(na, "sigmaf"))
        tau <- overlapPriorPost(SamplesModelCatalyticYear, na,
```

```
                                prior="dgamma(den$x[j],0.001, 0.001)", from=0, to=1000)
    return(tau)
})


## Builds the data.frame required by plotOverlap
df <- data.frame(Parameter=nam, tau=lb)
namo <- paste0("expression(",c(paste0("pi[",1:6,"]^(u)"), paste0("gamma[",1:6,"]"), "p[d]", paste0("
df$namo <- namo
df$prior <- "dunif(z)"
df$prior[20:23] <- "dnorm(z,0,sqrt(1000))"
df$prior[24] <- "dunif(z,0,100)"
df$prior[25] <- "dgamma(z,0.001,scale=0.001)"
df$from <- 0
df$to <- 1
df$from[20:23] <- -4
df$to[20:23] <- 4
df$from[24] <- 0
df$to[24] <- 30
df$from[25] <- 0.01
df$to[25] <- 5


plotOverlap(SamplesModelCatalyticYear, df, cex=1)
```

Parameter	τ	Distributions
$\pi_1^{(u)}$	0.387	
$\pi_2^{(u)}$	0.38	
$\pi_3^{(u)}$	0.395	
$\pi_4^{(u)}$	0.578	
$\pi_5^{(u)}$	0.519	
$\pi_6^{(u)}$	0.582	
γ_1	0.094	
γ_2	0.112	
γ_3	0.177	
γ_4	0.104	
γ_5	0.11	
γ_6	0.104	
p_d	0.214	
w_1	0.434	
w_2	0.644	
w_3	0.134	
w_4	0.437	
w_5	0.887	
w_6	0.413	
β_0	0	
β_1	0.01	
β_2	0.001	
β_3	0.002	
η	0.014	
σ_f	0.001	

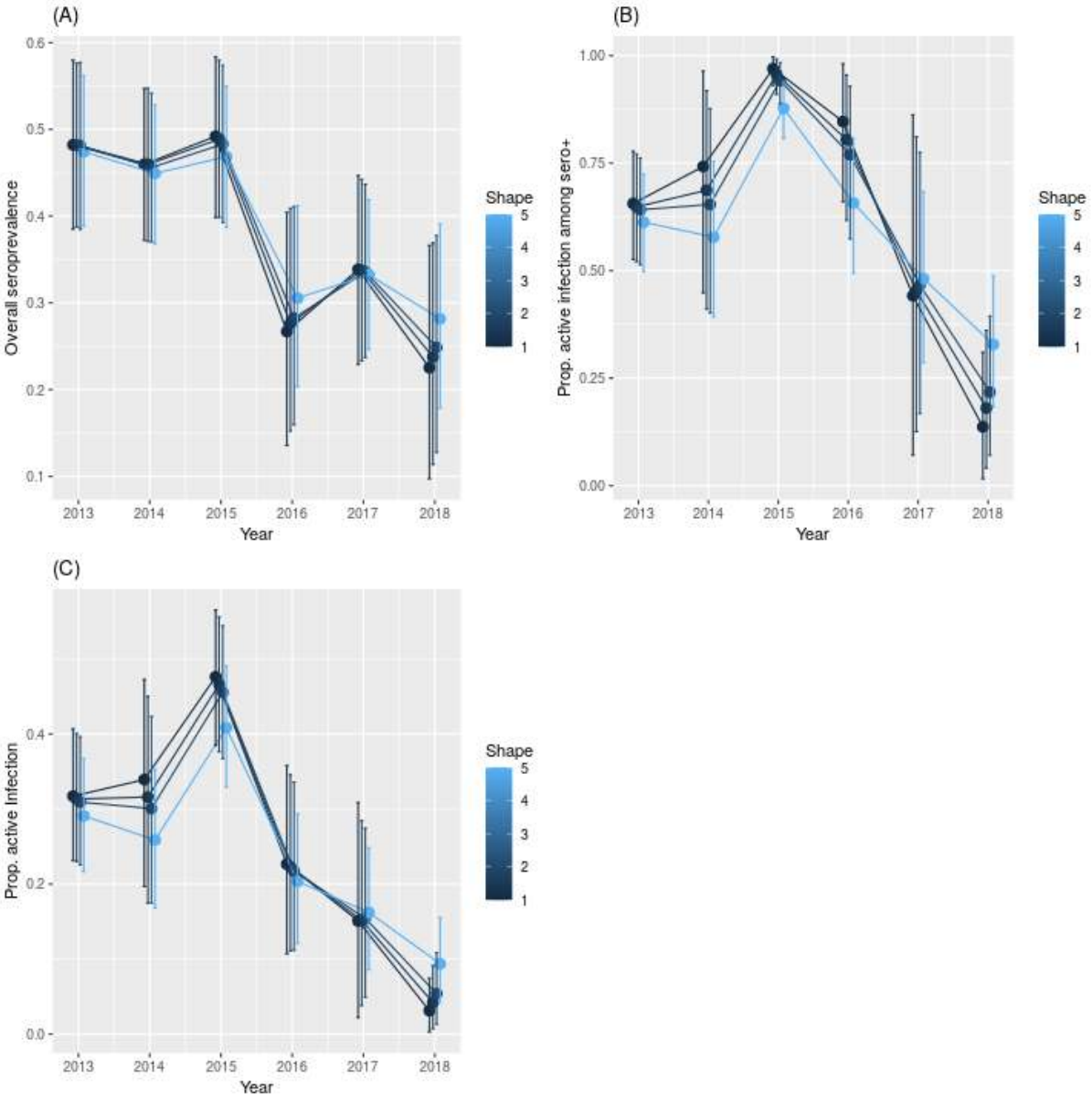
As can be seen on this figure, the coefficients that are the most sensitive are the yearly seroprevalence among unmarked animals $\pi_t^{(u)}$ and the yearly proportion of actively infected animals in the population w_t. As explained in the appendices of the paper, we tried different prior distributions for these parameters to assess the sensitivity of the parameters of interest of the model (overall seroprevalence and proportion of the population actively infected) to these prior distribution. We tried beta distributions characterized by two equal parameters, and equal to 1 (uniform distribution), 1.5, 2 and 5. Larger shape parameters correspond to prior distributions more concentrated around 0.5. We fitted the models for the differente possible beta distributions below, and calculated the quantities of interest below. Note that this code can take some time to execute, and that the resulting object `listRobustness` is available as a dataset in the package:

```
library(stringr)

listebeta <- c(1,1.5,2,5)

so <- lapply(listebeta, function(x) {
    samb <- robustnessPitWit(x,x,1,1, BruData)
    totalPrevalence(samb)
})
```

```
dfso <- do.call(rbind, so)
dfso$beta <- rep(listebeta,each=6)
sow <- lapply(listebeta, function(x) {
    samb <- robustnessPitWit(1,1,x,x, BruData)
    rs <- do.call(rbind,samb)
    r1 <- rs[,str_detect(colnames(rs), "wt")]
    data.frame(Year=2013:2018, wtm=colMeans(r1),
               lb=apply(r1,2,quantile,c(0.1)),
               ub=apply(r1,2,quantile,c(0.9)))
})
dfsow <- do.call(rbind, sow)
dfsow$beta <- rep(listebeta,each=6)
soom <- lapply(listebeta, function(x) {
    samb <- robustnessPitWit(x,x,x,x, BruData)
    rs <- do.call(rbind,samb)
    r1a <- rs[,str_detect(colnames(rs), "wt")]
    r1b <- rs[,str_detect(colnames(rs), "pit\\[")]
    r1 <- r1a*r1b
    data.frame(Year=2013:2018, omegam=colMeans(r1),
               lb=apply(r1,2,quantile,c(0.1)), ub=apply(r1,2,quantile,c(0.9)))
})
dfsoom <- do.call(rbind, soom)
dfsoom$beta <- rep(listebeta,each=6)

listRobustness <- list(pit=dfso, wit=dfsow, pitwit=dfsoom)
```

We can plot the effect of the prior on the estimates below:

```
library(gridExtra)
pd <- position_dodge(0.2)
pl1 <- ggplot(listRobustness$pit,aes(x=Year, y=Prevalence, col=beta, group=beta))+
    geom_point(size=3, position=pd)+geom_line(position=pd)+
    geom_errorbar(aes(ymin=LowerBound, ymax=UpperBound), width=0.2, position=pd)+
    xlab("Year")+ylab("Overall seroprevalence")+ggtitle("(A)")+
    labs(col = "Shape")

pd <- position_dodge(0.2)
pl2 <- ggplot(listRobustness$wit,aes(x=Year, y=wtm, col=beta, group=beta))+
    geom_point(size=3, position=pd)+geom_line(position=pd)+
    geom_errorbar(aes(ymin=lb, ymax=ub), width=0.2, position=pd)+
    xlab("Year")+ylab("Prop. active infection among sero+")+
    ggtitle("(B)")+labs(col = "Shape")

pd <- position_dodge(0.2)
pl3 <- ggplot(listRobustness$pitwit,aes(x=Year, y=omegam, col=beta, group=beta))+
    geom_point(size=3, position=pd)+geom_line(position=pd)+
    geom_errorbar(aes(ymin=lb, ymax=ub), width=0.2, position=pd)+
    xlab("Year")+ylab("Prop. active Infection")+
    ggtitle("(C)")+labs(col = "Shape")

grid.arrange(pl1, pl2, pl3, nrow=2)
```



This figure is discussed in the appendices of the paper.

Bibliography

- Calenge C., Lambert S., Petit E., Thebault A., Gilot-Fromont E., Toigo C., Rossi S. (in prep.) Estimating disease prevalence and temporal dynamics using biased capture serological data in a wildlife reservoir: the example of brucellosis in Alpine ibex (Capra ibex). Submitted to Preventive Veterinary Medicine.