pestim - an R package to estimate population counts from aggregated mobile phone data

Bogdan Oancea

27-feb-2018

Abstract

This package has been developed in the context of a European research project within the European Statistical System called ESSnet on Big Data. It provides an implementation for a hierarchical model to combine both aggregated mobile phone data and external official (administrative or survey) data to produce estimates of population counts in each cell of a division of a territory using a Bayesian approach.

1 Introduction

`pestim` R package implements a hierarchical model to estimate the population counts of different territorial cells combining the information from aggregated mobile phone data and a population register or survey data, both at a given time instant and along a sequence of time periods. The theoretical model implemented by the `pestim` R package follows the ecological sampling techniques to estimate population counts (see e.g. [2] and [3]) which are based on a bayesian approach. The complete methodology is described in [1]. The proposed model rests on two working assumptions:

- Given that mobile phone data and official data operate at different time scales, we assume that there exists an initial time instant in which we can equate population figures from both sources.
- The mobility patterns of individuals do not depend on the mobile network operator which they are subscribed to.

The inference of the population counts from mobile phone data and official data is achieved in a two step process:

- at a given time instant t_0 both mobile phone and official data are used to infer the population counts in each territorial cell;
- at later moments, $t_1, t_2, ...$ the transition probabilities are inferred from mobile phone data and are used to estimate the spatial and time evolution of the population.

The package is structured on three layers of functions:

- **Auxiliary functions**, providing computation of mathematical functions such as the ratio of two beta functions, the confluent hypergeometric function, an optimization routine for a concrete probability distribution, etc. Examples of these functions are `ratioBeta, kummer, Phi, modeLambda`;

- **Distribution-related functions**, providing computation regarding the generation of random deviates according to different probability distributions comprising both priors, posteriors, and the generation of parameter specifications for these distributions. Examples of these functions are `dtriang, rtriang, ptriang, qtriang, dlambda, rlambda, rmatProb, rN0, rNt, rNtcondN0, rg, rp, alphaPrior, genAlpha, genUV`.

- **Estimation-related functions**, providing computation of estimates based upon the populations generated with the preceding functions. Examples of these functions are `postN0, postNt, postNtcondN0`.

`pestim` package is freely available under the `GPL3` and `EUPL` licenses at the following address: `https://github.com/MobilePhoneESSnetBigData/pestim`. It requires at least R version 3.3.0, but upgrading R to the newest version is highly recommended. It can be installed using `install_github()` function from `devtools` package:

```
library(devtools)
install_github("MobilePhoneESSnetBigData/pestim", build_vignettes=TRUE)
```

The Reference Manual [14] is available at the following address: `https://github.com/MobilePhoneESSnet`
`pestim/raw/master/doc/pestim_Reference_Manual.pdf`.

Since it contains `C++` code, the user need to install `Rtools` under Windows enviroment or to have a `C++` compiler under Linux or Mac OS X environments. We also provide Windows binaries and Mac OS X binaries for this package that can be downloaded from:

- for Windows binary package: `https://github.com/MobilePhoneESSnetBigData/Estimation_Population/blob/master/pestim_0.1.0.zip`.

- for Mac OS X binary package: `https://github.com/MobilePhoneESSnetBigData/Estimation_Population/blob/master/pestim_0.1.0.tgz`.

These binary packages can be downloaded and installed with the normal command, such as:

```
install.package("pestim_0.1.0.zip")
```

for Windows but in this case the user should also install all the dependencies.

2 The hierarchical model

In this section we will give a short description of the theoretical model underlaying the `pestim` R package for estimation of the population counts at a given time moment. More detailed documents about this model can be downloaded from `https://github.com/MobilePhoneESSnetBigData/Estimation_Population`.

A first input of the model is $\mathbf{N}^{\text{MNO}} = (N_1^{\text{MNO}}, \ldots, N_I^{\text{MNO}})^T$ which represents the population counts reported by the mobile network operator in each territorial cell $i \in \mathcal{I} = \{1, \ldots, I\}$ (i.e. the aggregated mobile phone data). The second input of the model is the official population counts in each territorial cell denoted by $\mathbf{N}^{\text{REG}} = (N_1^{\text{REG}}, \ldots, N_I^{\text{REG}})^T$. These official population counts could come from administrative data sources or from statistical surveys. `pestim` implements a function to estimate the actual population counts $\mathbf{N} = (N_1, \ldots, N_I)^T$ combining both data sources. It also computes the posterior probability distribution $\mathbb{P}\left(\mathbf{N}|\mathbf{N}^{\text{MNO}}; \mathbf{N}^{\text{REG}}\right)$ that

Figure 1: The diagram of the process for population estimation using mobile phone and official population data.

can be used to assess the uncertainty in the output estimates. This process can be represented schematically as shown in figure 1.

The main idea of the model is to emulate the ecological sampling setting in which the number of detected individuals in each cell follows a binomial distribution $Bin(N_i, p_i)$ whose parameter N_i is the target of the model and is assigned a weakly informative prior and the detection probability p_i is also assigned a weakly informative prior based upon both data sources. In our case, if we have N_i individuals in cell i and we have an independent detection probability p_i for each individual through the mobile telecommunication network, then we will detect N_i^{MNO} individuals according to the aggregated mobile phone data naturally following a binomial distribution. N_i can be modeled as Poisson random variables with independent parameters λ_i, variables which are pairwise independent, while p_i are the proportions of individuals detected by the MNO at time t_0 in each cell i. It is important to note here that p_i is not simply the market share of the MNO in cell i but the actual proportion of individuals detected by the network. As an example, a call between a subscriber in a cell i and a non-subscriber in another cell j of a given MNO is certainly detected by the network in ***both cells***, thus potentially being part of the aggregated data N_i^{MNO} and N_j^{MNO}. This example emphasizes the importance of the knowledge of the preprocessing and aggregation procedures from microdata for the final result.

The detection probabilities p_i are modeled as $Beta$ distributed independently random variables with α_i and β_i parameters in each cell. The prior distribution of the parameters α_i and β_i comes from the following reasoning. We assume that $\frac{\alpha_i}{\alpha_i+\beta_i}$ and $\alpha_i + \beta_i$ are independently distributed according to $\frac{\alpha_i}{\alpha_i+\beta_i} \simeq f_1(\frac{\alpha_i}{\alpha_i+\beta_i}; \mathbf{N}^{\text{REG}}, \mathbf{z})$ and $\alpha_i + \beta_i \simeq f_2(\alpha_i + \beta_i; \mathbf{N}^{\text{REG}}, \mathbf{z})$. Here f_1 and f_2 are weakly informative prior distributions for $\frac{\alpha}{\alpha+\beta}$ and $\alpha + \beta$ and they made use of the information from the population register ($\mathbf{N}^{\text{REG}}$) and any other auxiliary information $\mathbf{z}$. The parameters $\alpha_i + \beta_i$ can be understood as the population size of each cell N_i (thus with support in $(0, \infty)$) upon which the detection is executed at that time instant and $\alpha_i/(\alpha_i + \beta_i)$ can be understood as a priori proportions of individuals detected by the MNO in cell i. If we assume f_2 to be a gamma distribution with parameters $(N_i^{\text{MNO}} + 1, \frac{N_i^{\text{MNO}}}{N_i^{\text{REG}}})$ the most probable value for the sample size is N_i^{REG} which is consistent with the preceding hypothesis for N_i. With no prior information about the detection probability we may assume $f_1 = \text{Unif}[0, 1]$. the parameters λ_i are modeled with another weakly information prior f_3 which may incorporate the information we have from the population register or similar sources.

Thus, the model can be summarized as follows:

$$
\begin{aligned}
N_i^{\mathrm{MNO}} &\simeq \mathrm{Bin}\left(N_i, p_i\right), \qquad N_i^{\mathrm{MNO}} \perp N_j^{\mathrm{MNO}}, \quad i \neq j = 1, \ldots, I \\
N_i &\simeq \mathrm{Po}\left(\lambda_i\right), \qquad N_i \perp N_j, \quad i \neq j = 1, \ldots, I \\
p_i &\simeq \mathrm{Beta}\left(\alpha_i, \beta_i\right), \qquad p_i \perp p_j \quad i \neq j = 1, \ldots, I \\
(\alpha_i, \beta_i) &\simeq \frac{f_1(\frac{\alpha_i}{\alpha_i+\beta_i}; \mathbf{N}^{\mathrm{REG}}, \mathbf{z}) \cdot f_2(\alpha_i + \beta_i; \mathbf{N}^{\mathrm{REG}}, \mathbf{z})}{\alpha_i + \beta_i} \quad (\alpha_i, \beta_i) \perp (\alpha_j, \beta_j), \quad i \neq j = 1, \ldots, I \\
\lambda_i &\simeq f_3(\lambda_i; N_i^{\mathrm{REG}}, \mathbf{z}) \quad (\lambda_i > 0, \lambda_i \perp \lambda_j), \quad i = 1, \ldots, I.
\end{aligned} \tag{1}
$$

The quantity of interest here is the target population counts $\mathbf{N} = (N_1, \ldots, N_I)^T$ in each cell i. We will follow a Bayesian approach to compute the posterior distribution of the target population. This approach allows us to account for the inference and the assessment of the uncertainty, hence of quality of estimations. We can leverage the prior information we have at our disposal by choosing the probability distribution f_1, f_2 and f_3. The posterior distribution $\mathbb{P}\left(\mathbf{N} \middle| \mathbf{N}^{\mathrm{MNO}}; \mathbf{N}^{\mathbf{REG}}\right)$ is given by (we dropped the subscripts since each cell is treated independently of each other) [1]:

$$
\mathbb{P}\left(N \middle| N^{\mathrm{MNO}}; N^{\mathbf{REG}}\right) \propto \int_0^\infty d\lambda \quad \mathbb{P}\left(\lambda \middle| N^{\mathrm{MNO}}; N^{\mathbf{REG}}\right) \cdot \mathrm{Po}(N; \lambda), \tag{2}
$$

N being a Poisson random variable, the most probable value for N is given by $\lfloor \lambda \rfloor$ and the posterior distribution for the hyperparameter λ will allow us to provide a point estimator for N (mode, mean, median, ...).

The posterior $\mathbb{P}\left(\lambda \middle| \mathbf{N}^{\mathrm{MNO}}; \mathbf{N}^{\mathbf{REG}}\right)$ is given by [4] [1]:

$$
\mathbb{P}\left(\lambda \middle| N^{\mathrm{MNO}}; N^{\mathbf{REG}}\right) \propto \mathbb{P}\left(\lambda\right) \cdot \mathrm{Po}(N^{\mathrm{MNO}}; \lambda) \cdot S\left(\lambda, N^{\mathrm{MNO}}, N^{REG}\right), \tag{3}
$$

where we have defined

$$
S(\lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}) = \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} I_{N^{\mathrm{MNO}},n}(N^{\mathrm{REG}}), \tag{4}
$$

$$
I_{N^{\mathrm{MNO}},n}(N^{\mathrm{REG}}) = \int_0^\infty \int_0^\infty d\alpha d\beta \; \frac{f_1(\frac{\alpha}{\alpha+\beta}; N^{\mathrm{REG}}) \cdot f_2(\alpha+\beta; N^{\mathrm{REG}}}{\alpha+\beta} \frac{B\left(\alpha + N^{\mathrm{MNO}}, \beta + n - N^{\mathrm{MNO}}\right)}{B\left(\alpha, \beta\right)}. \tag{5}
$$

In order to compute the integral $I_{N^{\mathrm{MNO}},n}(N^{\mathrm{REG}})$ we use a change of variables $u = \frac{\alpha}{\alpha+\beta}, v = \alpha + \beta$. The integral becomes [1]:

$$
I_{n,m}(N^{\mathrm{REG}}) = \int_0^\infty dv f_2(v) \int_0^1 du \; f_1(u) \frac{B(u \cdot v + n, (1-u) \cdot v + m)}{B(u \cdot v, (1-u) \cdot v)} \tag{6}
$$

This form of the integral allows us to use Monte Carlo technique. We consider the function $g_{n,m}(\mathbf{x}) = \frac{B(x_1 \cdot x_2 + n, (1-x_1) \cdot x_2 + m)}{B(x_1 \cdot x_2, (1-x_1) \cdot x_2)}$ and generate M bidimensional random variables $\mathbf{x} \in [0,1] \times \mathbf{R}^+$ according to the bidimensional distribution $f_1 \times f_2$. Then, using $f(\mathbf{x}) = f_1(x_1) f_2(x_2)$ as importance function, we can write:

$$
I_{n,m}(N^{\mathrm{REG}}) = \lim_{M \to \infty} \frac{1}{M} \sum_{i=1}^{M} g_{n,m}(\mathbf{x}_i). \tag{7}
$$

We made use of stratified importance sampling and introduced the stratification as follows: we defined $H_1 \cdot H_2$ strata as the rectangular domains $[a_{h_1-1}, a_{h_1}] \times [b_{h_2-1}, b_{h_2}]$, where $a_{h_1} = F_1^{-1}(h_1/H_1)$ $(h_1 = 1, \ldots, H_1)$ and $b_{h_2} = F_2^{-1}(h_2/H)$ $(h_2 = 1, \ldots, H_2)$, and F_i stands for the distribution function corresponding to the density function f_i. Defining the importance function in each stratum by $f_{h_1 h_2} = H_1 \cdot H_2 \cdot f_1 \cdot f_2$ truncated at $[a_{h_1-1}, a_{h_1}] \times [b_{h_2-1}, b_{h_2}]$ and taking equal-size strata $M_{h_1 h_2} = \frac{M}{H_1 H_2}$, then we obtained:

$$I_{n,m}(N^{\mathrm{REG}}) = \lim_{M\to\infty} \frac{1}{M} \sum_{h_1=1}^{H_1} \sum_{h_2=1}^{H_2} \sum_{i_{h_1}=1}^{M/H_2} \sum_{i_{h_2}=1}^{M/H_1} g_{n,m}(\mathbf{x}_{i_{h_1} i_{h_2}}) \tag{8}$$

where the random values $\mathbf{x}_{i_{h_1} i_{h_2}}$ are generated with the corresponding density function $f_{h_1 h_2}$.

Once computed the integrals $I_{N^{\mathrm{MNO}},n}(N^{\mathrm{REG}})$, the series $S(\beta_0, N^{\mathrm{MNO}}, N^{\mathrm{REG}})$ is summed up with standard procedures.

Another approach of computing $S(\beta_0, N^{\mathrm{MNO}}, N^{\mathrm{REG}})$ gives [1]:

$$S(\lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}) = \int_0^\infty \int_0^\infty d\alpha d\beta \; \frac{f_1(\frac{\alpha}{\alpha+\beta}; \mathbf{N}^{\mathrm{REG}}, \mathbf{z}) \cdot f_2(\alpha+\beta; \mathbf{N}^{\mathrm{REG}}, \mathbf{z})}{\alpha+\beta} \Phi(\alpha, \beta; \lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}), \tag{9}$$

where we have defined

$$\Phi(\alpha, \beta; \lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}) = \frac{B(\alpha + N^{\mathrm{MNO}}, \beta)}{B(\alpha, \beta)} \cdot {}_1F_1(\lambda; \beta, \alpha + \beta + N^{\mathrm{MNO}}) \tag{10}$$

(see the corresponding functions `Phi()` and `kummer()` from `pestim` package). With the same change of variables we get:

$$\begin{aligned} S(\lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}) &= \int_0^\infty dv \; f_2(v) \int_0^1 du \; f_1(u) \cdot \Phi(u \cdot v, (1-u) \cdot v; \lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}) \\ &= \int_0^\infty dv \; f_2(v) \int_0^1 du \; f_1(u) \cdot \bar{\Phi}(u, v; \lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}) \end{aligned} \tag{11}$$

Considering $\mathbf{x} = (u, v)^T$ we can apply the same Monte Carlo technique and we obtained:

$$S(\lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}) = \lim_{M\to\infty} \frac{1}{M} \sum_{h_1=1}^{H_1} \sum_{h_2=1}^{H_2} \sum_{i_{h_1}=1}^{M/H_2} \sum_{i_{h_2}=1}^{M/H_1} \bar{\Phi}(\mathbf{x}_{i_{h_1} i_{h_2}}; \lambda, N^{\mathrm{MNO}}, N^{\mathrm{REG}}), \tag{12}$$

where the random values $\mathbf{x}_{i_{h_1} i_{h_2}}$ are generated with the corresponding density function $f_{h_1 h_2}$.

To estimate the number of individuals per cell we need a method to generate random variables according to the posterior distribution $\mathbb{P}\left(N_i | \mathbf{N}^{\mathrm{MNO}}; \mathbf{N}^{\mathrm{REG}}\right)$. For this we need to generate random values for hyperparameter λ according to its posterior distribution (**??**):

$$\mathbb{P}\left(\lambda \middle| N^{\mathrm{MNO}}; N^{\mathbf{REG}}\right) \propto \mathbb{P}(\lambda) \cdot \mathrm{Po}(N^{\mathrm{MNO}}; \lambda) \cdot S\left(\lambda, N^{\mathrm{MNO}}, N^{REG}\right).$$

The unnormalized posterior density $\mathbb{P}\left(\lambda | \mathbf{N}^{\mathrm{MNO}}; \mathbf{N}^{\mathrm{REG}}\right)$ does not allow us to find easily the corresponding posterior distribution function to apply the inverse method to generate random variables [10] and we made use of the acceptance-rejection method. We considered a candidate distribution $g(x) = \mathrm{Cauchy}(x; x_0 = \lambda^*, \sigma)$ truncated at $\mathbb{R}^+$ with λ^* being the mode of $f(\lambda)$. For

the family of prior distributions $\lambda \simeq \Gamma(\alpha+1, \alpha/N^{\mathrm{REG}})$ we choose $\sigma = N^{\mathrm{REG}}/\sqrt{\alpha}$. To generate random values λ according to $\mathbb{P}\left(\lambda|\mathbf{N}^{\mathrm{MNO}}; \mathbf{N}^{\mathrm{REG}}\right)$ we generate values according to $g(\lambda)$, and values v according to Unif$(0,1)$ so that we accept those λ such that $v \leq \frac{f(\lambda)}{c \cdot g(\lambda)}$.

The process of generating random values for the hyperparameter λ is implemented as `rlambda()` function in the `pestim` package.

To generate random values N according to $\mathbb{P}\left(N|N^{\mathrm{MNO}}; N^{\mathrm{REG}}\right)$ we generate values λ and then the corresponding values N according to the Poisson distribution with parameter λ.

3 Examples how to use `pestim` package to estimate the population at a time instant

In the following will we show how to use functions provided by `pestim` package to compute population estimations. In our examples we will use some synthetic generated data but the same computations can be used with real data.

3.1 Prior distribution of the hyperparameters

`pestim` implements functions for the following prior distributions:

- uniform distribution;
- triangular distribution;
- gamma distribution;

The uniform distribution is well know and functions that implement the density, distribution function, quantile function and random variable generation are provided by the standard `R` distribution.

The triangular distribution can be used to for modelling the local market shares u, the cell size v and the hyperparameter λ. The corresponding functions for the density, distribution function, quantile function and random variable generation are `ptriang`, `dtriang`, `qtriang` and `rtriang`. Below is an example of how to use the triangular distribution function (see figure 2).

```
library(ggplot2)
library(pestim)
x <- seq(0.10, 0.65, by = 0.01)
y <- dtriang(x, xMin = 0.10, xMax = 0.65, xMode = 0.32)
df <- data.frame(x = x, y= y)
ggplot(df, aes(x, y)) + geom_line() + scale_x_continuous(limits = c(0, 1)) +
           xlab('u') + ylab('Probability Density')
```

The gamma distribution is another choice for modelling both the cell size v and the hyperparameter λ. Functions for density, distribution function, quantile function and random generation for the gamma distribution are included in the standard `R` distribution.

We can assume a parameterization $\Gamma(\alpha+1, \xi^*/\alpha)$, where ξ^* stands for the mode of the modeled variable (v or λ) and $\alpha > 0$ determines the degree of concentration around the mode ξ^* (see figure 3). In the following example we used 5 values for α, 1, 5, 10, 100, and 1000. The following code shows how to use this distribution.

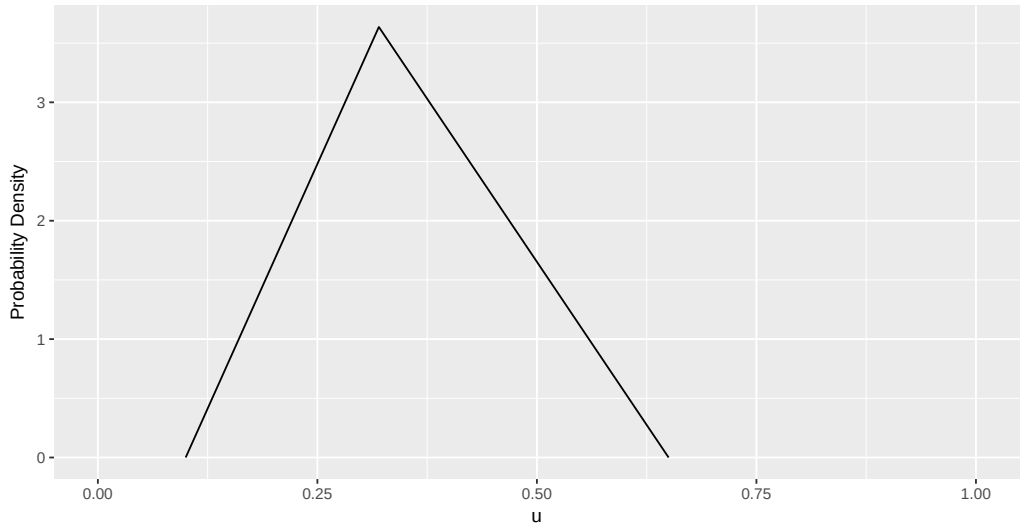

Figure 2: The triangular distribution.

```
alphas <- c(1, 5, 10, 100, 1000)
mode <- 35
df <- lapply(alphas, function(alpha){
    x <- 0:100
    y <- dgamma(x, shape = alpha + 1, scale = mode / alpha)
    z <- as.character(alpha)
    output <- data.frame(x = x, y = y, alpha = z)
    return(output)
})

df <- Reduce(rbind, df)
ggplot(df, aes(x, y, col = alpha, group = alpha)) +
          geom_line(aes(linetype = alpha)) +
          scale_x_continuous(limits = c(0, 100)) + xlab('') + ylab('')
```

3.2 Estimation for a single cell

We can assume that there is a high correlation between the actual size and official population data [12]. In our examples, for the simulated data that we will generate, for a given actual true value N_i^0 we will simulate a population register value $N^{\mathrm{REG}_i} \simeq \lfloor N(\mu = N_i^0, \sigma = 10\% \cdot N_i^0) \rfloor$. For the corresponding number of individuals detected through the mobile network we will assume a proportion of detected individuals randomly between 15% and 40% as realistic figures (see [13] to compare with market shares as an approximation to these figures).

Since the treatment of all cells is independent of each other, we will start by showing the estimation process for a single cell. We investigate different combinations of priors and numerical regimes for N^{MNO} and N^{REG}. In all cases we assume a priori $f_3 \simeq \Gamma\left(\alpha + 1, \frac{N^{\mathrm{Reg}}}{\alpha}\right)$ for λ. Let us consider a true population size of $N^{(0)} = 100$, a population given by some administrative register $N^{\mathrm{Reg}} = 97$ assuming an error of 3%. Let us also consider the number of individuals detected by the mobile network as $N^{\mathrm{MNO}} = 19$ assuming a proportion of detected individuals of around 20%.

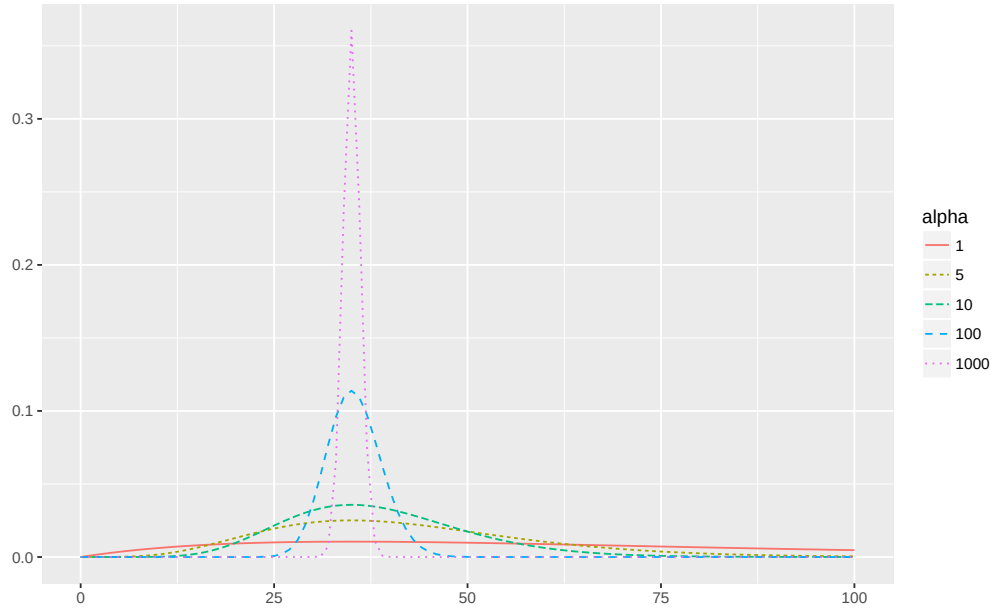

Figure 3: The gamma distribution.

3.2.1 $f_1 \simeq \mathbf{Unif}(u_m, u_M)$, $f_2 \simeq \mathbf{triang}(v_m, v_M, v_*)$

For the prior distribution of the proportion of detected individuals we will assume a weakly informative distribution $f_u = \text{Unif}(u_m, u_M)$ with $u_m = 0$ and $u_M = 0.50$. For the prior distribution of the cell size we will assume a triangular distribution with parameters $v_m = 87$, $v_M = 107$, and $v^* = 97$, assuming an (unknown) error of 10% over the population register size.

Computing the estimates for values of $\alpha = 1, 10, 100, 1000$ and observing the effect of the amount of uncertainty in the population size (see figure 4) can be achieved as follows:

```
library(pestim)
library(data.table)
library(ggplot2)
nReg <- 97
nMNO <- 19
fu <- list('unif', xMin = 0, xMax = 0.50)
fv <- list('triang', xMin = 87, xMax = 107, xMode = 97)
alphaSeq <- c(1, 10, 100, 1000)
flambdaList <- list()
for (alpha in alphaSeq){
    flambdaList[[as.character(alpha)]] <-
         list('gamma', shape = 1 + alpha, scale = nReg / alpha)
}
nSim <- 100
results <- lapply(alphaSeq, function(alpha){
    flambda <- flambdaList[[as.character(alpha)]]
    output <- replicate(nSim, postN0(nMNO, nReg, fu, fv, flambda))
    output <- as.data.table(t(matrix(unlist(output), nrow = 3)))
    setnames(output, c('postMean', 'postMedian', 'postMode'))
    output[, sim := 1:nSim]
```

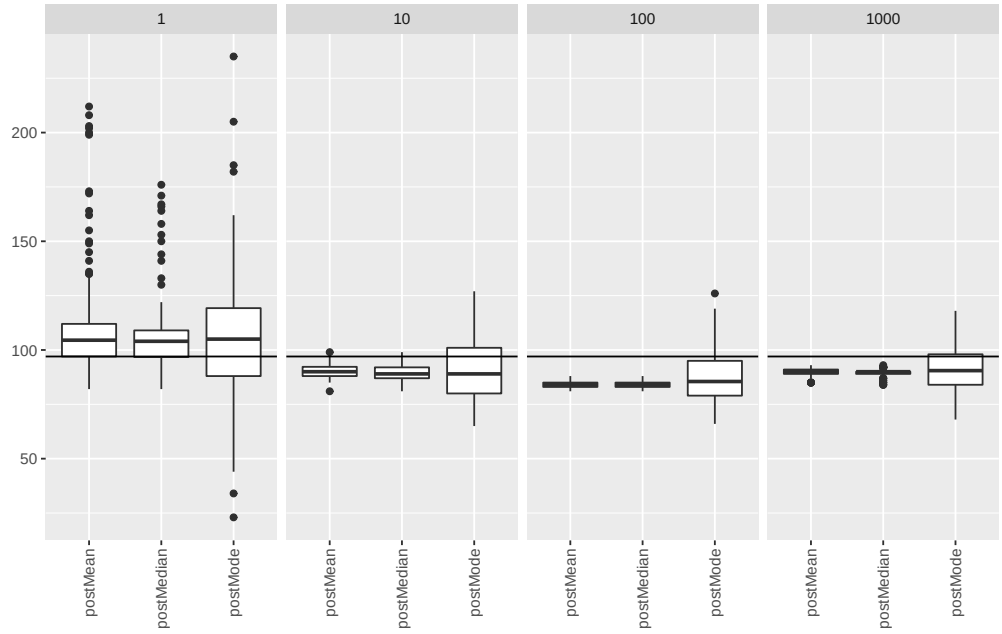

Figure 4: The uncertainty in population estimation.

```
    output <- melt(output, id.vars = 'sim')
    output[, 'alpha' := alpha]
    return(output)
})
names(results) <- alphaSeq
results <- rbindlist(results)
ggplot(results, aes(x = variable, y = value)) +
           geom_boxplot() + facet_grid(. ~ alpha) +
           xlab('') + ylab('') +  geom_hline(yintercept = nReg) +
           theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = .5))
```

The actual computation of the population estimation is done by `postN0()` function that takes the following parameters:

- nMNO - the number of the individuals detected in the actual cell according to the mobile network operator
- nReg - the number of individuals from the register
- fu and fv - named lists with the prior marginal distributions of the two-dimensional points for the Monte Carlo integration
- flambda - named list with the prior distribution of the lambda parameter
- n - the number of points to generate in the posterior distribution for the computation. Default value is 1e3
- scale - a numeric vector with the scale to count the number of individuals. Default value is 1

- relTol - relative tolerance in the computation of the kummer function (default value is 1e-6)
- nSim - number of two-dimensional points to generate to compute the integral (default value is 1e4)
- nStrata - integer vector of length 2 with the number of strata in each dimension (default value is c(1, 1e2))

In this example we used the default values for `n, scale, relTol, nSim, nStrata` parameters.

This function generates a n random values for population according to the posterior distribution by calling `rN0()` function. The posterior distribution used to generates random numbers is a Poisson distribution (see the second row from model 1) . In turn, `rN0()` calls `rlambda()` to generate these points.

`rlambda()` generates the points according to the accept-reject method using as candidate distribution a Cauchy distribution whose parameters are taken from the prior distributions and the mode of the posterior distribution of the λ parameter as is described at the end of section 2. `rlambda()` first computes the mode for the posterior distribution of λ using `modeLambda()` function and then apply the acception-rejection method. This latter function uses the posterior density function of the parameter λ in the hierarchical model implemented in our package by `dlambda()` function. This computes the unnormalized posterior density function of the parameter λ as $f(\lambda\big|N^{\text{MNO}}; N^{\text{Nreg}}) \propto f(\lambda) \cdot \text{dpois}(N^{\text{MNO}}; \lambda) \cdot S(\lambda; N^{\text{MNO}}, N^{\text{Nreg}})$, where `dpois` is the probability density function of a Poisson distribution and is implemented i n the standard `R` distribution and S is defined in equation 3, 4.

S is computed using the Monte Carlo method described in section 2. The points needed by this method (see eq.12) are generated using `genUV()` function and Φ function from eq. 12 is evaluated by a call to `Phi()` from `pestim`. `Phi()` multiplies a ratio of two *Beta* functions (eq. 10) computed by `ratioBeta()` function and the confluent hypergeometric function which is given by a call to `kummer()` function. `ratioBeta()` computes the ratio of two two *Beta* functions using the difference between their logarithms and then exponentiating the result to avoid numerical overflow. The logarithms of the *Beta* functions are computed using the `lbeta()` from the base R library.

`kummer()` was implemented in C++ and called using `Rcpp` package because it is numerically intensive and the performance of a pure R implementation is far from the C++ implementation in terms of computing time. It is a partial implementation of the confluent hypergeometric function ${}_1F_1(z; a; b)$. Since is one of the most time consuming function from `pestim` package we proived here few details about the implementation that we used in the current version of the package. The confluent hypergeometric function is defined as:

$$\mathbf{M}(z; a; b) = \sum_{j=0}^{\infty} \frac{(a)_j}{\Gamma(b+j)} \times \frac{z^j}{j!} \tag{13}$$

where $(a)_j$ is the Pochhammer symbol defined by:

$$a_0 = 1, \quad (a)_j = a \times (a+1) \times \cdots (a+j-1), \qquad j = 1, 2, \cdots \tag{14}$$

The sum in equation 13 always converge, function $\mathbf{M}$ being analytic throughout the complex plane $\mathbb{C}$. Next, we define:

$${}_1F_1(z; a; b) = \Gamma(b)\mathbf{M}(z; a; b) = \sum_{j-0}^{\infty} \frac{(a)_j}{(b)_j} \times \frac{z^j}{j!} \tag{15}$$

which is also denoted by M(a;b;z) and is referred to as the confluent hypergeometric function.

We followed the work described in [11] and divided the computation of ${}_1F_1(z;a;b)$ according to the value of z in two different cases: for $z < 80$ we used a Taylor series expantion approach, while for $z \geq 80$ we've made use of WatsonâĂŹs lemma [15]. Thus, in the first case we computed the confluent hypergeometric function as:

$$ {}_1F_1(z;a;b) \approx S_N = \sum_{j=0}^{N} \frac{(a)_j}{(b)_j} \frac{z^j}{j!} = \sum_{j=0}^{N} A_j \tag{16} $$

The actual C++ implementation is based on the following equations:

$$ A_0 = 1 \tag{17} $$

$$ S_0 = A_0 \tag{18} $$

$$ A_{j+1} = A_j \frac{a+j}{b+j} \frac{z}{j+1} \tag{19} $$

$$ S_{j+1} = S_j + A_{j+1} \quad j = 0, 1, 2, \cdots \tag{20} $$

The stopping criterion for the iterative procedure was set as $\frac{A_{j+1}}{S_j} < tol$ where A_j and S_j were previously defined.

In case of $z \geq 80$, according to WatsonâĂŹs lemma, for $z \in \Re$ we can write:

$$ \mathbf{M}(z;a;b) \sim \frac{e^z \cdot z^{a-b}}{\Gamma(a)} \sum_{j=0}^{\infty} \frac{(b-a)_j \cdot (1-a)_j}{j! \cdot z^j} \tag{21} $$

and, consequently

$$ {}_1F_1(z;a;b) = \Gamma(b) \cdot \frac{e^z \cdot z^{a-b}}{\Gamma(a)} \sum_{j=0}^{\infty} \frac{(b-a)_j \cdot (1-a)_j}{j! \cdot z^j} \tag{22} $$

Firstly, we computed the sum from equation 21 taking the same strategy for the stopping criterion as in the previuos case (i.e. $\frac{A_{j+1}}{S_j} < tol$ where $A_j = \frac{(b-a)_j \cdot (1-a)_j}{j! \cdot z^j}, S_j = \sum_{j=0}^{N} A_j$) then we multiplied the sum with $\frac{e^z \cdot z^{(a-b)} \cdot \Gamma(b)}{\Gamma(a)}$.

An interested reader could consult the C++ implementation of the above formulas in `Kummer.cpp` file from the `src` directory of `pestim` source package. For efficiency reasons, considering that calling a C++ function from the R environment has an important overhead, to minimize the number of functions calls we pass three vectors z, a, and b to `Kummer()` function and, in turn, it returns the value of the confluent hypergeometric function for all the elements in the input parameters. The implementation will be further optimized in the next versions of the `pestim` package.

Going back to our example, we can observe how the more precise the prior value of $\lambda = N^{\text{Reg}}$ is, the more precise the final estimate around N^{Reg} will result. Notice how this final estimate inherits the original difference between $N^{(0)}$ and N^{Reg}, as expected.

3.2.2 $f_1 \simeq \mathbf{Unif}(u_m, u_M)$, $f_2 \simeq \mathbf{Unif}(N_m, N_M)$

For the intervals (u_m, u_M) we will choose as centres of the intervals the value $N^{\text{MNO}}/N^{\text{Reg}}$ and as radii, we will progressively shorten the intervals starting from $r_1 = min(N^{\text{MNO}}/N^{\text{Reg}}, 1 - N^{\text{MNO}}/N^{\text{Reg}})$ down to 0.005 and we will use a number of $nPar = 10$ points for u.

For the intervals (N_m, N_M) we will choose as centres of the intervals the natural value N^{Reg} and as radii, we will progressively shorten the intervals starting from $R_1 = \lfloor 0.25 \cdot N^{\text{Reg}} \rfloor$ down to 1 and we will also use the same number $nPar = 10$ of points.

In all cases we will use $\alpha = 1$ as a weakly informative choice. For each pair of interval lengths $(u_M - u_m, N_M - N_m)$ we will estimate the population and compute the relative bias $\frac{\hat{N} - N^{\text{Reg}}}{N^{\text{Reg}}} \cdot 100$ for the posterior mean, median and mode estimates. The following piece of code does this estimation (to keep the length of this document reasonable we don't reproduce here the actual numerical results). One can note that this code is similar with the previous example, the estimation being computed with a call to `postN0()` function, and only the prior distributions are different.

```
library(pestim)
library(data.table)
nReg <- 97
nMNO <- 19
nPar <- 10
radShares <- seq(from = nMNO / nReg, to = 0.005, length.out = nPar)
radPopSizes <- round(seq(from = 0.25 * nReg, to = 1, length.out = nPar))
alpha <- 1
flambda <- list('gamma', shape = alpha + 1, scale = nReg / alpha)
results.Mean <- matrix(NA, ncol = nPar, nrow = nPar)
results.Median <- matrix(NA, ncol = nPar, nrow = nPar)
results.Mode <- matrix(NA, ncol = nPar, nrow = nPar)
for (radShare.index in seq(along = radShares)) {
    for (radPopSize.index in seq(along = radPopSizes)) {
        um <- nMNO / nReg - radShares[radShare.index]
        uM <- nMNO / nReg + radShares[radShare.index]
        fu <- list('unif', xMin = um, xMax = uM)
        Nm <- nReg - radPopSizes[radPopSize.index]
        NM <- nReg + radPopSizes[radPopSize.index]
        fv <- list('unif', xMin = Nm, xMax = NM)
        auxResults <- postN0(nMNO, nReg, fu, fv, flambda)
        results.Mean[radShare.index, radPopSize.index] <- auxResults[['postMean']]
        results.Median[radShare.index, radPopSize.index] <- auxResults[['postMedian']]
        results.Mode[radShare.index, radPopSize.index] <- auxResults[['postMode']]
        }
}
rownames(results.Mean) <- round(2 * radShares, 2)
rownames(results.Median) <- round(2 * radShares, 2)
rownames(results.Mode) <- round(2 * radShares, 2)
colnames(results.Mean) <- 2 * radPopSizes
colnames(results.Median) <- 2 * radPopSizes
colnames(results.Mode) <- 2 * radPopSizes
relBias.Mean <- round((results.Mean - nReg) / nReg * 100, 1)
relBias.Median <- round((results.Median - nReg) / nReg * 100, 1)
relBias.Mode <- round((results.Mode - nReg) / nReg * 100, 1)
```

3.2.3 $f_1 \simeq \mathbf{Unif}(u_m, u_M),\ f_2 \simeq \mathbf{triang}(N_m, N_M, N^{\mathbf{Reg}})$

The same computations as in the preceding section is carried out using a triangular prior distribution f_2 for the actual population size. The limits N_m and N_M are chosen as in the preceding section and the mode as $N^* = N^{\mathrm{Reg}}$.

```
library(pestim)
library(data.table)
nReg <- 97
nMNO <- 19
nPar <- 10
radShares <- seq(from = nMNO / nReg, to = 0.005, length.out = nPar)
radPopSizes <- round(seq(from = 0.25 * nReg, to = 1, length.out = nPar))
alpha <- 1
flambda <- list('gamma', shape = alpha + 1, scale = nReg / alpha)
results.Mean <- matrix(NA, ncol = nPar, nrow = nPar)
results.Median <- matrix(NA, ncol = nPar, nrow = nPar)
results.Mode <- matrix(NA, ncol = nPar, nrow = nPar)
for (radShare.index in seq(along = radShares)) {
    for (radPopSize.index in seq(along = radPopSizes)) {
        um <- nMNO / nReg - radShares[radShare.index]
        uM <- nMNO / nReg + radShares[radShare.index]
        fu <- list('unif', xMin = um, xMax = uM)
        Nm <- nReg - radPopSizes[radPopSize.index]
        NM <- nReg + radPopSizes[radPopSize.index]
        fv <- list('triang', xMin = Nm, xMax = NM, xMode = nReg)
        auxResults <- postN0(nMNO, nReg, fu, fv, flambda)
        results.Mean[radShare.index, radPopSize.index] <- auxResults[['postMean']]
        results.Median[radShare.index, radPopSize.index] <- auxResults[['postMedian']]
        results.Mode[radShare.index, radPopSize.index] <- auxResults[['postMode']]
    }
}
rownames(results.Mean) <- round(2 * radShares, 2)
rownames(results.Median) <- round(2 * radShares, 2)
rownames(results.Mode) <- round(2 * radShares, 2)
colnames(results.Mean) <- 2 * radPopSizes
colnames(results.Median) <- 2 * radPopSizes
colnames(results.Mode) <- 2 * radPopSizes
relBias.Mean <- round((results.Mean - nReg) / nReg * 100, 1)
relBias.Median <- round((results.Median - nReg) / nReg * 100, 1)
relBias.Mode <- round((results.Mode - nReg) / nReg * 100, 1)
```

3.2.4 $f_1 \simeq \mathbf{Unif}(u_m, u_M),\ f_2 \simeq \Gamma(a+1, \frac{N^{\mathbf{Reg}}}{a})$

The same computation is exemplified now with $f_2 \simeq \Gamma(a+1, \frac{N^{\mathrm{Reg}}}{a})$ and $\log_{10}(a) = -3, -2, \ldots, 2, 3$.

```
library(pestim)
library(data.table)
nReg <- 97
```

```
nMNO <- 19
nPar <- 10
radShares <- seq(from = nMNO / nReg, to = 0.005, length.out = nPar)
aPopSizes <- 10^{seq(-3, 3, by = 1)}
alpha <- 1
flambda <- list('gamma', shape = alpha + 1, scale = nReg / alpha)
results.Mean <- matrix(NA, ncol = length(aPopSizes), nrow = length(radShares))
results.Median <- matrix(NA, ncol = length(aPopSizes), nrow = length(radShares))
results.Mode <- matrix(NA, ncol = length(aPopSizes), nrow = length(radShares))
for(radShare.index in seq(along = radShares)) {
    um <- nMNO / nReg - radShares[radShare.index]
    uM <- nMNO / nReg + radShares[radShare.index]
    fu <- list('unif', xMin = um, xMax = uM)
    for (aPopSize.index in seq(along = aPopSizes)) {
        fv <- list('gamma', shape = aPopSizes[aPopSize.index],
                 scale = nReg / aPopSizes[aPopSize.index])
        auxResults <- postN0(nMNO, nReg, fu, fv, flambda)
        results.Mean[radShare.index, aPopSize.index] <- auxResults[['postMean']]
        results.Median[radShare.index, aPopSize.index] <- auxResults[['postMedian']]
        results.Mode[radShare.index, aPopSize.index] <- auxResults[['postMode']]
    }
}
rownames(results.Mean) <- round(2 * radShares, 2)
rownames(results.Median) <- round(2 * radShares, 2)
rownames(results.Mode) <- round(2 * radShares, 2)
colnames(results.Mean) <- aPopSizes
colnames(results.Median) <- aPopSizes
colnames(results.Mode) <- aPopSizes
relBias.Mean <- round((results.Mean - nReg) / nReg * 100, 1)
relBias.Median <- round((results.Median - nReg) / nReg * 100, 1)
relBias.Mode <- round((results.Mode - nReg) / nReg * 100, 1)
```

3.2.5 $f_1 \simeq \mathbf{Triang}(u_m, u_M, u^*)$, $f_2 \simeq \mathbf{Unif}(N_m, N_M)$

The same example is shown now for $f_1 \simeq$ Triang and $f_2 \simeq$ Unif. The hyperparameters are chosen as before.

```
library(pestim)
library(data.table)
nReg <- 97
nMNO <- 19
nPar <- 10
radShares <- seq(from = nMNO / nReg, to = 0.005, length.out = nPar)
radPopSizes <- round(seq(from = 0.25 * nReg, to = 1, length.out = nPar))
alpha <- 1
flambda <- list('gamma', shape = alpha + 1, scale = nReg / alpha)
results.Mean <- matrix(NA, ncol = nPar, nrow = nPar)
results.Median <- matrix(NA, ncol = nPar, nrow = nPar)
results.Mode <- matrix(NA, ncol = nPar, nrow = nPar)
```

```
for(radShare.index in seq(along = radShares)) {
    um <- nMNO / nReg - radShares[radShare.index]
    uM <- nMNO / nReg + radShares[radShare.index]
    uMode <- nMNO / nReg
    fu <- list('triang', xMin = um, xMax = uM, xMode = uMode)
    for(radPopSize.index in seq(along = radPopSizes)) {
        Nm <- nReg - radPopSizes[radPopSize.index]
        NM <- nReg + radPopSizes[radPopSize.index]
        fv <- list('unif', xMin = Nm, xMax = NM)
        auxResults <- postN0(nMNO, nReg, fu, fv, flambda)
        results.Mean[radShare.index, radPopSize.index] <- auxResults[['postMean']]
        results.Median[radShare.index, radPopSize.index] <- auxResults[['postMedian']]
        results.Mode[radShare.index, radPopSize.index] <- auxResults[['postMode']]
    }
}
rownames(results.Mean) <- round(2 * radShares, 2)
rownames(results.Median) <- round(2 * radShares, 2)
rownames(results.Mode) <- round(2 * radShares, 2)
colnames(results.Mean) <- 2 * radPopSizes
colnames(results.Median) <- 2 * radPopSizes
colnames(results.Mode) <- 2 * radPopSizes
relBias.Mean <- round((results.Mean - nReg) / nReg * 100, 1)
relBias.Median <- round((results.Median - nReg) / nReg * 100, 1)
relBias.Mode <- round((results.Mode - nReg) / nReg * 100, 1)
```

3.2.6 $f_1 \simeq \mathbf{Triang}(u_m, u_M, u^*)$, $f_2 \simeq \mathbf{Triang}(N_m, N_M, N^{\mathbf{Reg}})$

Both prior distributions are considered too be traingular with the same choice for hyperparameters.

```
library(pestim)
library(data.table)
nReg <- 97
nMNO <- 19
nPar <- 10
radShares <- seq(from = nMNO / nReg, to = 0.005, length.out = nPar)
radPopSizes <- round(seq(from = 0.25 * nReg, to = 1, length.out = nPar))
alpha <- 1
flambda <- list('gamma', shape = alpha + 1, scale = nReg / alpha)
results.Mean <- matrix(NA, ncol = nPar, nrow = nPar)
results.Median <- matrix(NA, ncol = nPar, nrow = nPar)
results.Mode <- matrix(NA, ncol = nPar, nrow = nPar)
for(radShare.index in seq(along = radShares)) {
    um <- nMNO / nReg - radShares[radShare.index]
    uM <- nMNO / nReg + radShares[radShare.index]
    uMode <- nMNO / nReg
    fu <- list('triang', xMin = um, xMax = uM, xMode = uMode)
    for(radPopSize.index in seq(along = radPopSizes)) {
        Nm <- nReg - radPopSizes[radPopSize.index]
```

```
        NM <- nReg + radPopSizes[radPopSize.index]
        Nmode <- nReg
        fv <- list('triang', xMin = Nm, xMax = NM, xMode = nReg)
        auxResults <- postN0(nMNO, nReg, fu, fv, flambda)
        results.Mean[radShare.index, radPopSize.index] <- auxResults[['postMean']]
        results.Median[radShare.index, radPopSize.index] <- auxResults[['postMedian']]
        results.Mode[radShare.index, radPopSize.index] <- auxResults[['postMode']]
    }
}
rownames(results.Mean) <- round(2 * radShares, 2)
rownames(results.Median) <- round(2 * radShares, 2)
rownames(results.Mode) <- round(2 * radShares, 2)
colnames(results.Mean) <- 2 * radPopSizes
colnames(results.Median) <- 2 * radPopSizes
colnames(results.Mode) <- 2 * radPopSizes
relBias.Mean <- round((results.Mean - nReg) / nReg * 100, 1)
relBias.Median <- round((results.Median - nReg) / nReg * 100, 1)
relBias.Mode <- round((results.Mode - nReg) / nReg * 100, 1)
```

3.2.7 $f_1 \simeq \mathbf{Triang}(u_m, u_M, u^*)$, $f_2 \simeq \Gamma(a+1, \frac{N^{\mathbf{Reg}}}{a})$

In the last example we combined a triangular distribution for f_1 and a gamma distribution for f_2.

```
library(pestim)
library(data.table)
nReg <- 97
nMNO <- 19
nPar <- 10
radShares <- seq(from = nMNO / nReg, to = 0.005, length.out = nPar)
aPopSizes <- 10^{seq(-3, 3, by = 1)}
alpha <- 1
flambda <- list('gamma', shape = alpha + 1, scale = nReg / alpha)
results.Mean <- matrix(NA, ncol = length(aPopSizes), nrow = length(radShares))
results.Median <- matrix(NA, ncol = length(aPopSizes), nrow = length(radShares))
results.Mode <- matrix(NA, ncol = length(aPopSizes), nrow = length(radShares))
for(radShare.index in seq(along = radShares)) {
    um <- nMNO / nReg - radShares[radShare.index]
    uM <- nMNO / nReg + radShares[radShare.index]
    uMode <- nMNO / nReg
    fu <- list('triang', xMin = um, xMax = uM, xMode = uMode)
    for(aPopSize.index in seq(along = aPopSizes)) {
        fv <- list('gamma', shape = aPopSizes[aPopSize.index],
                       scale = nReg / aPopSizes[aPopSize.index])
        auxResults <- postN0(nMNO, nReg, fu, fv, flambda)
        results.Mean[radShare.index, aPopSize.index] <- auxResults[['postMean']]
        results.Median[radShare.index, aPopSize.index] <- auxResults[['postMedian']]
        results.Mode[radShare.index, aPopSize.index ] <- auxResults[['postMode']]
    }
```

```
}
rownames(results.Mean) <- round(2 * radShares, 2)
rownames(results.Median) <- round(2 * radShares, 2)
rownames(results.Mode) <- round(2 * radShares, 2)
colnames(results.Mean) <- aPopSizes
colnames(results.Median) <- aPopSizes
colnames(results.Mode) <- aPopSizes
relBias.Mean <- round((results.Mean - nReg) / nReg * 100, 1)
relBias.Median <- round((results.Median - nReg) / nReg * 100, 1)
relBias.Mode <- round((results.Mode - nReg) / nReg * 100, 1)
```

3.3 Estimation for several cells

Obtaining estimations for the target population for a grid of cells is similar to the process for a single cell since the estimation in each cell is independent of each other.

We can consider the ratios $\frac{N_i^{\text{MNO}}}{N_i^{\text{Reg}}}$ as an initial guess for the proportions of detected individuals u_i with a highly probable value whose uncertainty will depend both on the process to obtain N_i^{MNO} (preprocessing and aggregation stages of the mobile phone data) and on the process to compile the population register figures N_i^{Reg} (measurement errors, processing errors, coverage, etc.). Any of the three prior distributions (uniform, triangular, gamma) can be used to express this uncertainty around $\frac{N_i^{\text{MNO}}}{N_i^{\text{Reg}}}$.

For the prior distribution for the local cell size v_i we can make similar considerations around the value N_i^{Reg} for each cell i focusing on the process of construction of the population register.

The prior distribution for the parameter λ_i is very important. If we choose $\alpha_i \gg 1$, this means a high confidence on the population register as the true population. It is advisable to be conservative and choose low values so that we do not artificially "force" the final estimates to be close to N_i^{Reg}. In the choice of α_i we can make use of the grid construction and the distribution of N_i^{Reg} to propose some prior values.

The variance of $\Gamma(\alpha_i + 1, \frac{N_i^{\text{Reg}}}{\alpha_i})$ is $\frac{\alpha_i+1}{\alpha_i^2} \cdot N_i^{\text{Reg}}$ and under the assumption of having a regular grid over the population, we can equate $\frac{\alpha_i+1}{\alpha_i^2} \cdot N^{\text{Reg}} = \frac{1}{N_{cells}-1} \sum_{i=1}^{N_{cell}} \left(N_i^{\text{Reg}} - \bar{N}^{\text{Reg}} \right)^2$ to have a first value (upper bound) for $\alpha \leq \min_i \alpha_i$.

The estimation processes for the prior hyperparameters are very important if we want to obtain final estimates not based upon subjective beliefs.

In the following we will show an example how to estimate the population in $N_c = 50$ cells, and we will consider a range of values for the hyperparameters to observe the effects on the final estimate. For the intervals $(u_{m,i}, u_{M,i})$ we will choose as centres of the intervals the natural values $N_i^{\text{MNO}}/N_i^{\text{Reg}}$ and as radii, we will progressively shorten the intervals starting from $r_{1,i} = min(N_i^{\text{MNO}}/N_i^{\text{Reg}}, 1 - N_i^{\text{MNO}}/N_i^{\text{Reg}})$ down to 0.005. For the intervals $(N_{m,i}, N_{M,i})$ we will choose as centres of the intervals the natural values N_i^{Reg} and as radii, we will progressively shorten the intervals starting from $R_{1,i} = \lfloor 0.25 \cdot N_i^{\text{Reg}} \rfloor$ down to 1. We will generate a number of $nPar = 5$ values for each interval.

The following piece of code computes and displays the distribution of the relative bias $\frac{\hat{N}_i - N_i^{\text{Reg}}}{N^{\text{Reg}}} \cdot 100$ for the posterior mean, median and mode estimates, respectively, for all pairs of interval lengths $(u_{M,i} - u_{m,i}, N_{M,i} - N_{m,i})$ and all cells. In our example we will use synthetic data for initial population. The population given from the administrative register for each cell is generated using a Gaussian distribution with a mean of 71 and standard deviartion of 3 while

the population detected by the MNO is again generated using a Gaussian distribution with a mean of 19 and standard deviartion of 2. These values can be replaced with any real data. The values for posterior distribution of the population is again obtained by a call to `postN0()` function.

```
library(pestim)
library(data.table)
library(ggplot2)
nCell <- 50
nReg <- round(rnorm(nCell, 71, 3))
nMNO <- round(rnorm(nCell, 19, 2))
nPar <- 5
radShares <- lapply(1:nCell, function(i){
    seq(from = (nMNO / nReg)[i], to = 0.005, length.out = nPar)
})
radPopSizes <- lapply(1:nCell, function(i){
    round(seq(from = 0.25 * nReg[i], to = 1, length.out = nPar))
})
varnReg <- var(nReg)
alphaBound <- sapply(1:nCell, function(i){
    0.5 * (nReg[i] / varnReg + sqrt((nReg[i] / varnReg)^2 + 4 * nReg[i] / varnReg))
})
alpha <- min(alphaBound)
results.Mean <- lapply(1:nCell, function(i){matrix(NA, ncol = nPar, nrow = nPar)})
results.Median <- lapply(1:nCell, function(i){matrix(NA, ncol = nPar, nrow = nPar)})
results.Mode <- lapply(1:nCell, function(i){matrix(NA, ncol = nPar, nrow = nPar)})
relBias.Mean <- list()
relBias.Median <- list()
relBias.Mode <- list()
for(i in 1:nCell){
    for(radShare.index in seq(along = radShares[[i]])) {
        for (radPopSize.index in seq(along = radPopSizes[[i]])) {
            um <- nMNO[[i]] / nReg[[i]] - radShares[[i]][radShare.index]
            uM <- nMNO[[i]] / nReg[[i]] + radShares[[i]][radShare.index]
            fu <- list('unif', xMin = um, xMax = uM)
            Nm <- nReg[[i]] - radPopSizes[[i]][radPopSize.index]
            NM <- nReg[[i]] + radPopSizes[[i]][radPopSize.index]
            fv <- list('unif', xMin = Nm, xMax = NM)
            flambda <- list('gamma', shape = alpha + 1, scale = nReg[[i]] / alpha)
            auxResults <- postN0(nMNO[[i]], nReg[[i]], fu, fv, flambda)
            results.Mean[[i]][radShare.index, radPopSize.index] <-
                    auxResults[['postMean']]
            results.Median[[i]][radShare.index, radPopSize.index] <-
                    auxResults[['postMedian']]
            results.Mode[[i]][radShare.index, radPopSize.index] <-
                    auxResults[['postMode']]
        }
    }
    rownames(results.Mean[[i]]) <- round(2 * radShares[[i]], 2)
```

```
    rownames(results.Median[[i]]) <- round(2 * radShares[[i]], 2)
    rownames(results.Mode[[i]]) <- round(2 * radShares[[i]], 2)
    colnames(results.Mean[[i]]) <- 2 * radPopSizes[[i]]
    colnames(results.Median[[i]]) <- 2 * radPopSizes[[i]]
    colnames(results.Mode[[i]]) <- 2 * radPopSizes[[i]]
    relBias.Mean[[i]] <-
              round((results.Mean[[i]] - nReg[[i]]) / nReg[[i]] * 100, 1)
    relBias.Median[[i]] <-
              round((results.Median[[i]] - nReg[[i]]) / nReg[[i]] * 100, 1)
    relBias.Mode[[i]] <-
              round((results.Mode[[i]] - nReg[[i]]) / nReg[[i]] * 100, 1)
}
parNames <- expand.grid(paste0('u', 1:5), paste0('v', 1:5))
colnames(parNames) <- c('u', 'v')

relBias.Mean.df <-
           data.frame(u = character(0), v = character(0),
           cell = character(0), N = numeric(0))
relBias.Median.df <-
           data.frame(u = character(0), v = character(0),
           cell = character(0), N = numeric(0))
relBias.Mode.df <-
           data.frame(u = character(0), v = character(0),
           cell = character(0), N = numeric(0))
for(i in 1:nCell){
    aux <- cbind(parNames, cell = as.character(i), N = as.vector(relBias.Mean[[i]]))
    relBias.Mean.df <- rbind(relBias.Mean.df, aux)
    aux <- cbind(parNames, cell = as.character(i), N = as.vector(relBias.Median[[i]]))
    relBias.Median.df <- rbind(relBias.Median.df, aux)
    aux <- cbind(parNames, cell = as.character(i), N = as.vector(relBias.Mode[[i]]))
    relBias.Mode.df <- rbind(relBias.Mode.df, aux)
}
ggplot(relBias.Mean.df, aes(x = ' ', y = N)) + geom_boxplot() +
            facet_grid(factor(u) ~ factor(v)) +
            xlab('Diverse combinations of prior parameters') +
            ylab('Mean Estimates\n') +
            ggtitle(paste0('Relative bias (%) distributions of the ',
            nCell, ' cells\n')) +
            theme(plot.title = element_text(face = 'bold', size = 14, hjust = 0.5))

ggplot(relBias.Median.df, aes(x = ' ', y = N)) + geom_boxplot() +
           facet_grid(factor(u) ~ factor(v)) +
           xlab('Diverse combinations of prior parameters') +
           ylab('Median Estimates\n') +
           ggtitle(paste0('Relative bias (%) distributions of the ',
           nCell, ' cells\n')) +
           theme(plot.title = element_text(face = 'bold', size = 14, hjust = 0.5))
```

```
ggplot(relBias.Mode.df, aes(x = ' ', y = N)) + geom_boxplot() +
          facet_grid(factor(u) ~ factor(v)) +
          xlab('Diverse combinations of prior parameters') +
          ylab('Mode Estimates\n') +
          ggtitle(paste0('Relative bias (%) distributions of the ',
          nCell, ' cells\n')) +
          theme(plot.title = element_text(face = 'bold', size = 14, hjust = 0.5))
```

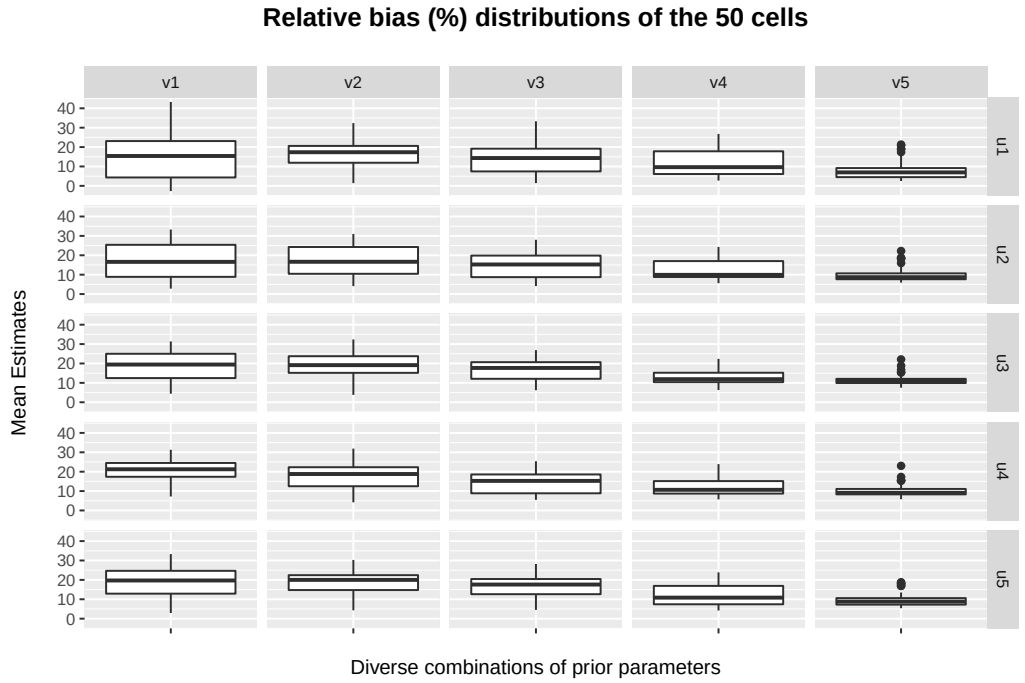

Figure 5: The distribution of the relative bias for the posterior mean.

The same combinations for distributions used for f_1 and f_2 can be used in the case of several cells like in the case of a single cell. Since this process is straightforward we will not show here each combination.

4 Estimates along a sequence of time

In this section we show how to extend the previous estimation process along a sequence of time instants. Figure 8 presents schematically this process.

As input data for the final inference stage we used the number of individuals $N_{ij}^{\mathrm{MNO}}(t_0, t_n)$ moving from territorial cell i to cell j in the time interval (t_0, t_n) according to the MNO. These data will be combined with official data and at the end we will provide the following outputs:

- the probability distribution of actual individuals in each territorial cell i at the initial time t_0;
- the probability distribution of actual individuals at the time instants t_n for $n = 1, 2, \ldots$.

We make two prior assumptions:

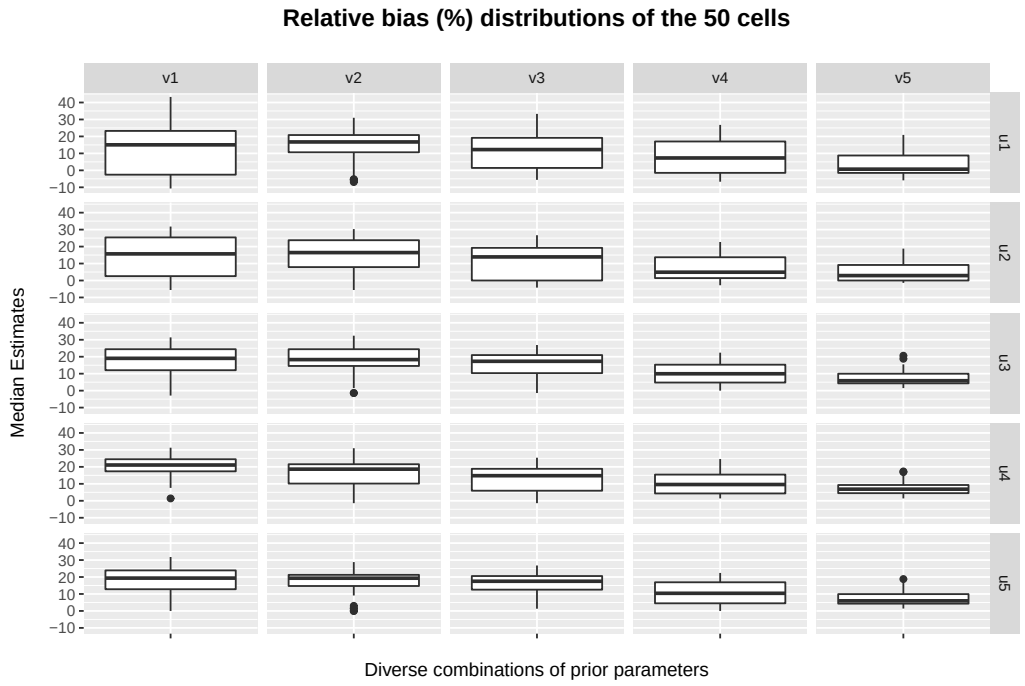

Figure 6: The distribution of the relative bias for the posterior median.

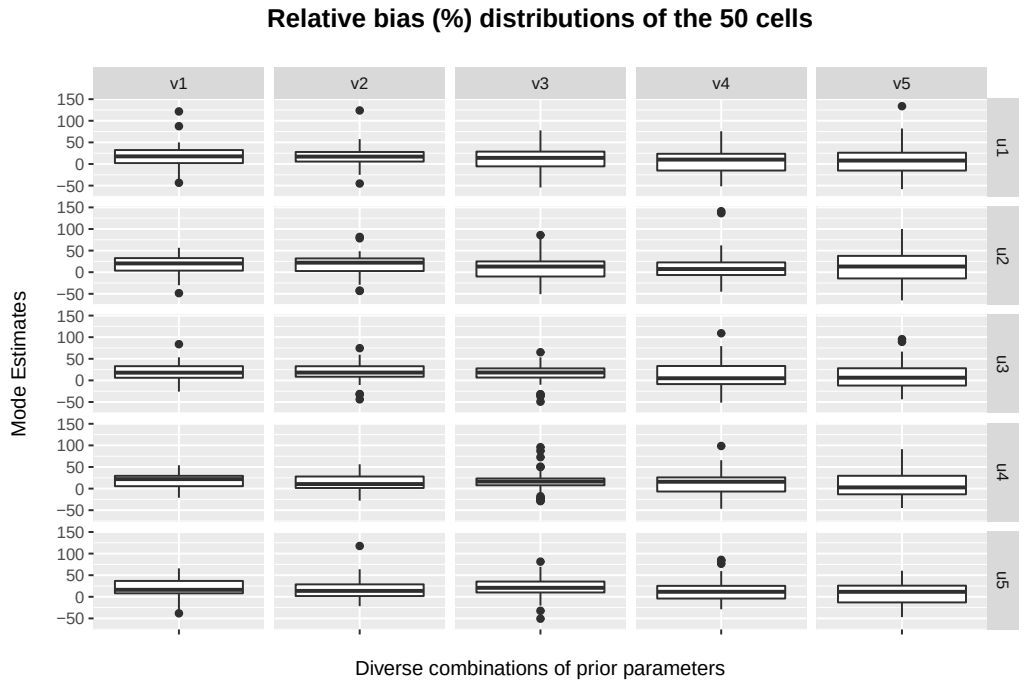

Figure 7: The distribution of the relative bias for the posterior mode.

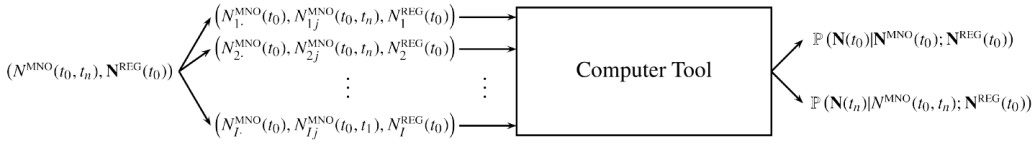

Figure 8: A diagram of the estimation process for the target population using mobile phone and official population data at a sequence of time instants.

1. To combine both aggregated mobile phone and official data we assume that a given time instant t_0 both population figures in each territorial cell can be equated to some extent. For simplicity we will take N_i^{Reg} as a fixed quantity without a prior distribution representing uncertainty in its knowledge. Therefore N_i^{Reg} will be fixed external parameters in the model.

2. The movements of individuals from one cell to another cell is assumed to be independent of being subscribers of a given MNO or another.

The following hierarchical model supports these hypotheses. Let $p_{ij}(t_0, t_n)$ denote the probability for an individual to move from cell i to cell j in the time interval (t_0, t_n). Let $N_{ij}^{\text{MNO}}(t_0, t_n)$ the number of individuals moving from cell i to cell j according to the network. We denote $N_{i\cdot}^{\text{MNO}}(t_0) = \sum_{j=1}^{I} N_{ij}^{\text{MNO}}(t_0, t_n)$.

$$N_i(t_n) = \left[N_i(t_0) + \sum_{\substack{j=1 \\ j\neq i}}^{I} p_{ji}(t_0, t_n) N_j(t_0) - \sum_{\substack{j=1 \\ j\neq i}}^{I} p_{ij}(t_0, t_n) N_i(t_0) \right], \quad i = 1, \ldots, I \tag{23a}$$

$$\mathbf{p}_{i\cdot}(t_0, t_n) \simeq \text{Dir}\left(\alpha_{i1}(t_0, t_n), \ldots, \alpha_{iI}(t_0, t_n)\right), \quad \mathbf{p}_{i\cdot}(t_0, t_n) \perp \mathbf{p}_{j\cdot}(t_0, t_n), \quad i \neq j = 1, \ldots, I \tag{23b}$$

$$\alpha_{ij}(t_0, t_n) \simeq f_{\alpha ij}\left(\alpha_{ij}; \frac{N_{ij}^{\text{MNO}}(t_0, t_n)}{N_{i\cdot}^{\text{MNO}}(t_0)}\right), \quad i = 1, \ldots, I \tag{23c}$$

$$N_i^{\text{MNO}}(t_0) \simeq \text{Bin}\left(N_i(t_0), p_i(t_0)\right), \qquad N_i^{\text{MNO}}(t_0) \perp N_j^{\text{MNO}}(t_0), \quad i \neq j = 1, \ldots, I \tag{23d}$$

$$N_i(t_0) \simeq \text{Po}\left(\lambda_i(t_0)\right), \qquad N_i(t_0) \perp N_j(t_0), \quad i \neq j = 1, \ldots, I \tag{23e}$$

$$p_i(t_0) \simeq \text{Beta}\left(\alpha_i(t_0), \beta_i(t_0)\right), \qquad p_i(t_0) \perp p_j(t_0) \quad i \neq j = 1, \ldots, I \tag{23f}$$

$$(\alpha_i(t_0), \beta_i(t_0)) \simeq \frac{f_{ui}\left(\frac{\alpha_i}{\alpha_i+\beta_i}; \frac{N_i^{\text{MNO}}(t_0)}{N_i^{\text{REG}}(t_0)}\right) \cdot f_{vi}\left(\alpha_i + \beta_i; N_i^{\text{REG}}(t_0)\right)}{\alpha_i + \beta_i},$$
$$(\alpha_i(t_0), \beta_i(t_0)) \perp (\alpha_j(t_0), \beta_j(t_0)), \quad i \neq j = 1, \ldots, I \tag{23g}$$

$$\lambda_i(t_0) \simeq f_{\lambda i}(\lambda_i; N_i^{\text{REG}}(t_0)) \quad (\lambda_i(t_0) > 0, \quad \lambda_i(t_0) \perp \lambda_j(t_0)), \quad i = 1, \ldots, I, \tag{23h}$$

where

- $[\cdot]$ denotes the nearest integer function;
- $f_{\alpha ij}$ is the probability density function of the parameters α_{ij}. The notation $f_{\alpha ij}\left(\alpha_{ij}; \frac{N_{ij}^{\text{MNO}}(t_0,t_n)}{N_{i\cdot}^{\text{MNO}}(t_0)}\right)$ is meant to indicate that $\frac{N_{ij}^{\text{MNO}}(t_0,t_n)}{N_{i\cdot}^{\text{MNO}}(t_0)}$ should be taken as the mode of the density function;
- f_{ui} is the probability density function of the parameter u in cell i with mode $\frac{N_i^{\text{MNO}}(t_0)}{N_i^{\text{REG}}(t_0)}$;
- f_{vi} isr the probability density function of the parameter v in cell i with mode $N_i^{\text{REG}}(t_0)$;
- $f_{\lambda i}$ is the probability density function of the parameter λ in cell i with mode $N_i^{\text{REG}}(t_0)$.

Equations (23d) to (23h) add time dependence to the model described in section 2. Equations (23a), (23b), (23c) take care of the time evolution of the estimates.

Equation (23a) states that the number of individuals in a cell i at time t_n equals the initial number of individuals plus those arriving from other cells in the given time interval minus those leaving for other cells in the same time interval. The number of individuals arriving and leaving are estimated using the transition probability among cells.

We modelled these transition probabilities for a given cell i as a multivariate random variable with a Dirichlet distribution with parameters $\alpha_{i1}, \ldots, \alpha_{iI}$ (in fact, Dirichlet distributions are commonly used as prior distributions in Bayesian statistics). These parameters are given unimodal prior distributions $f_{\alpha_{ij}}$ with mode in $\frac{N_{ij}^{\mathrm{MNO}}}{N_{i\cdot}^{\mathrm{MNO}}}$ according to our second working assumption.

The computation of the probability functions $\mathbb{P}\left(N_i(t_n)\middle|N^{\mathrm{MNO}}(t_0,t_1)\right)$ for each cell i will allow us to choose an estimator as, e.g., the posterior mean, posterior median or posterior mode. The computation is conducted empirically in three steps:

1. The initial population value $N_i(t_0)$ is generated for all cells $i = 1, \ldots, I$ according to the model using $N_{i\cdot}^{\mathrm{MNO}}(t_0)$ as input data and choosing weakly informative priors f_{ui}, f_{vi} and $f_{\lambda i}$.

2. A transition probability matrix $[p_{ij}(t_0, t_n)]$ is generated according to the model using $N^{\mathrm{MNO}}(t_0, t_n)$ as input data and choosing weakly informative priors $f_{\alpha ij}$.

3. These generated quantities are used in formula (23a) to generate $N_i(t_1)$ for all cells $i = 1, \ldots, I$.

Following these steps we can generate an empirical posterior distribution of values $N_i(t_n)$ for each cell i. Then we can use these distributions to provide a point estimate according to its mean, median, or mode.

5 An example how to estimate population at a sequence of time instants

In this example we will consider a territory divided into 12 cells. At 4 succesive time intervals the individuals follow their own trajectories so that the population in each cell evolves according to actual transition matrices $N_{ij}(t_{n-1}, t_n)$, $n = 1, 2, 3, 4$. We kept the number of cells and time intervals small to obtain a reasonable running time. At the initial time instant t_0 we also have the population register figures for each cell $N_i^{\mathrm{Reg}}(t_0)$ which can be equated to the initial real population $N_i(t_0)$, although they are not completely equal due to nonsampling errors.

We also generate the transition matrices for each time interval $n = 1, 2, 3, 4$ for the individuals $N_{ij}^{\mathrm{MNO}}(t_{n-1}, t_n)$ detected with the network.

`pestim` contains a dataset `MobPop` that provides population counts moving from each pair of cells at succesive time instants for a simulated true population, a corresponding official population in a register and a population detected with a mobile telecommunication network. The data are actually stored in a `data.table` with the following columns:

- ID_CELL_INI - identification code for each initial cell in the displacements;
- ID_CELL_END - identification code for each final cell in the displacements;
- ID_T - identification code of each time moment. It is very important to underline that the table collects always displacements between the initial time instant and the corresponding time instant specified by ID_T;

- N_REG - counts according to the population register. Note that these counts do not evolve in time;
- N_0 - counts of the simulated true population;
- N_MNO_1 - counts of individuals detected by the Mobile Network Operator

We will use these data in our examples. Firstly, we generate the estimates $N_i(t_0)$ for the initial time instant, as we did in the previous section.

```
library(pestim)
library(data.table)
library(ggplot2)

data(MobPop)
InitialPop <- MobPop[ID_T == 1 & ID_CELL_END == ID_CELL_INI]
Scale <- 1e3
nMNO_ini <- InitialPop[['N_MNO_1']] / Scale
nReg <- InitialPop[['N_REG']] / Scale
u0 <- InitialPop$N_MNO_1 / InitialPop$N_REG
fu <- lapply(u0, function(u){
    umin <- max(0, u - 0.10 * u)
    umax <- min(1, u + 0.10 * u)
    output <- list('triang', xMin = umin, xMax = umax, xMode = u)
    return(output)
})
v0 <- nReg
fv <- lapply(v0, function(u){
    umin <- max(0, u - 0.10 * u)
    umax <- u + 0.10 * u
    output <- list('triang', xMin = umin, xMax = umax, xMode = u)
    return(output)
})
alpha <- 1 / 0.1**2 - 1
flambda <- lapply(v0, function(v){list('gamma', shape = 1 + alpha,
                  scale = v / alpha)})
nSim <- 10000
N0cells <- rN0(nSim, nMNO_ini, nReg, fu, fv, flambda, scale = Scale)
ggplot(N0cells, aes(x = N0)) +
           geom_histogram(binwidth = 50) +
           geom_vline(aes(xintercept = nReg), color = 'red') +
           facet_grid(factor(cellID) ~ ., scales="free") +
           xlab('Posterior populations per cell') +
           ylab('Counts (different scales)\n') +
           theme(plot.title = element_text(face = 'bold', size = 14,
           hjust = 0.5), axis.text.y =
           element_blank())
```

The actual estimation is done using `rN0()` function which generates a series of random points according to the posterior probability distribution of the number of individuals . The results

Figure 9: Simulated posterior populations for each cell.

are presented in figure 9. This simulation was run under Linux and Mac OS X environments succesfully. Under Windows it may be possible to get a memory allocation error. In this case the value of the variable $nSim$ should be decreased.

Now we can provide two kind of time evolutions for the population in each cell. On the one hand, we can generate simulated populations conditioned upon their estimated initial size $\widehat{N}_i(t_0)$. On the other hand, we can provide these simulated populations unconditioned upon their estimated initial size but starting from the input data themselves (thus uncertainty in the initial population estimate is included).

In the first case, taking the mean of the preceding populations as estimates for the initial population of each cell we obtain the following evolving simulated populations in each cell (in dashed lines the assumed true values of the simulated population):

```
N0 <- N0cells[, list(Mean = round(mean(N0))), by = 'cellID'][['Mean']]
nT <- length(unique(MobPop$ID_T)) - 1
matList <- lapply(2:(nT + 1), function(i){
    output <- dcast(MobPop[ID_T == i],
                    ID_CELL_INI ~ ID_CELL_END, value.var = 'N_MNO_1')
    output[, ID_CELL_INI := NULL]
    return(output)
})

DistributNames <- rep('triang', 12)
Variation <- rep(list(list(cv = 0.10)), 12)
NtcondN0 <- rbindlist(lapply(seq(along = matList), function(i){
    output <- rNtcondN0(nSim, N0, as.matrix(matList[[i]]),
                        DistributNames, Variation)
    output <- as.data.table(output)
```

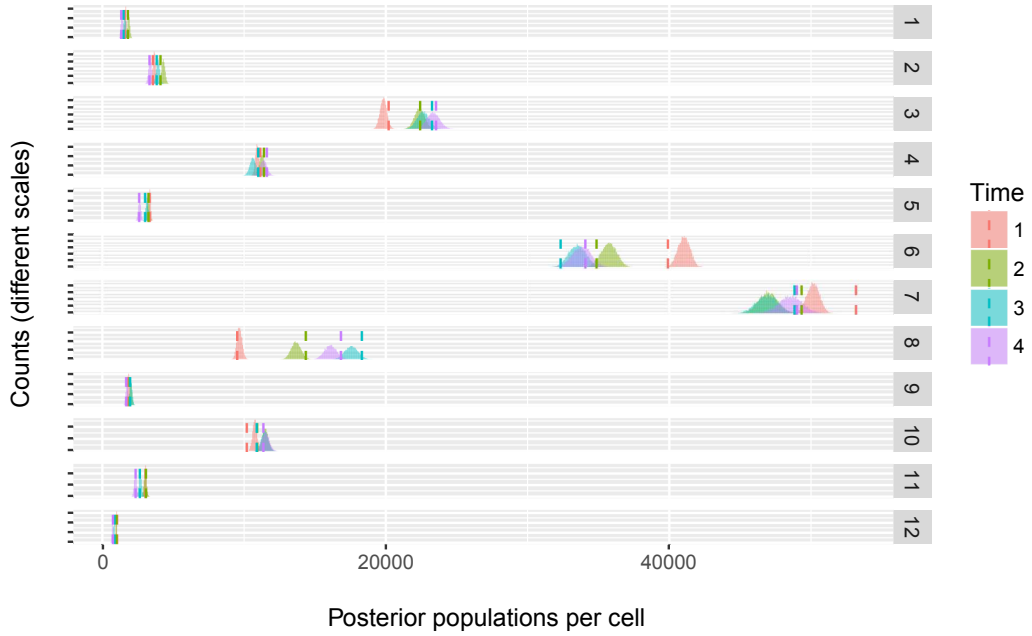

Figure 10: Simulated posterior populations for each cell conditioned on initial population.

```
    setnames(output, as.character(1:dim(output)[2]))
    output[, sim := 1:.N]
    output <- melt(output, id.vars = 'sim',  variable.name = 'Cell',
                   value.name = 'N0')
    output[, Time := factor(i)]
    return(output)
}))

N0True <- MobPop[Time != 1][, Time := factor(Time - 1)]
ggplot(NtcondN0, aes(x = N0, fill = Time)) +
          geom_histogram(binwidth = 50, alpha=.5, position="identity") +
          geom_vline(data = N0True, aes(xintercept = N0True, color = Time),
          linetype = 'dashed', size = 0.5) +
          facet_grid(factor(Cell) ~ ., scales="free") +
          xlab('\n Posterior populations per cell') +
          ylab('Counts (different scales)\n') +
          ggtitle(paste0('Simulated posterior populations for each
          cell\n (Conditioned on initial population)')) +
          theme(plot.title = element_text(face = 'bold', size = 14, hjust = 0.5),
          axis.text.y = element_blank())
```

In this code, `rNtcondN0()` function generates the random values for N_t (the posterior distribution) conditional on initial population count. This function uses `rmatProb()` to generate the transition probabilities according to a Dirichlet distribution with parameters generated by `alphaPrior()`. These parameters are generated with distributions whose names are taken from the input parameter `distNames` and construct the corresponding prior distribution for each cell j with mode at $u_j^* = N_j$. The rest of parameters of the distribution are computed according to the dispersion parameters specified in `variation`. The results are presented in figure 10

By comparing both figures we can detect how the uncertainty in the estimation of the initial

population size propagates along the evolving estimates of the population size of each cell. In cells 6 and 7 we can observe how the uncertainty in the simulated initial populations provide rather inaccurate estimates for later time periods.

Using the posterior mean as estimator of the population in each cell, we have the following time evolutions:

```
N0dt <- data.table(N = N0, Time = 0, Cell = seq(along=N0))
seriesNt <- NtcondN0[, list(N = round(mean(N0))), by = c('Time', 'Cell')]
setcolorder(N0dt, names(seriesNt))
seriesNt <- rbindlist(list(N0dt, seriesNt))
NTrue <- MobPop
NTrue[, Time := factor(Time - 1)]
setnames(NTrue, 'N0True', 'trueN')
seriesNt <- merge(seriesNt, NTrue)
seriesNt <- melt(seriesNt, id.vars = c('Time', 'Cell'), value.name =
                   'Population', variable.name = 'Series')
ggplot(seriesNt, aes(x = Time, y = Population, color = Series, group = Series)) +
           geom_line() +
           facet_grid(factor(Cell) ~ ., scales="free") +
           scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
           xlab('') +
           ylab('Population (different scales; watch out the origins)\n') +
           scale_color_manual(labels = c(expression(widehat(N)),
           expression(N^{(0)})), values = c('red', 'blue')) +
           theme(plot.title = element_text(face = 'bold', size = 14, hjust = 0.5),
           axis.text.y = element_text(size = 5))
```

The results of the time evolution are depicted in figure 11. In terms of the relative bias, this comparison is presented in figure 12.

```
 seriesNt.dcast <- dcast(seriesNt, Time + Cell ~ Series, value.var = 'Population')
 seriesNt.dcast[, relBias := round((N - trueN) / trueN * 100, 2)][,
                        (c('N', 'trueN')) := NULL]
 seriesNt.melt <-  melt(seriesNt.dcast, id = c('Time', 'Cell'))
 ggplot(seriesNt.melt, aes(x = Time, y = value, group = variable, color = variable)) +
             geom_line() +
             facet_grid(Cell ~ .) +
             geom_hline(yintercept = 0, size = 0.3, linetype = 'dotted') +
             xlab('\n Time Period') + ylab('Relative bias\n') +
             theme(plot.title = element_text(face = 'bold', size = 12, hjust = 0.5),
             axis.text.y = element_text(size = 5), legend.position="none")
```

We can also produce estimates unconditioned on the initial estimate of the population in each cell. The uncertainty in the estimation of the initial population of each cell is incorporated into the estimation process for later time periods. The results of the estimations are presented in figure 13.

```
nT <- length(unique(MobPop$ID_T)) - 1
```

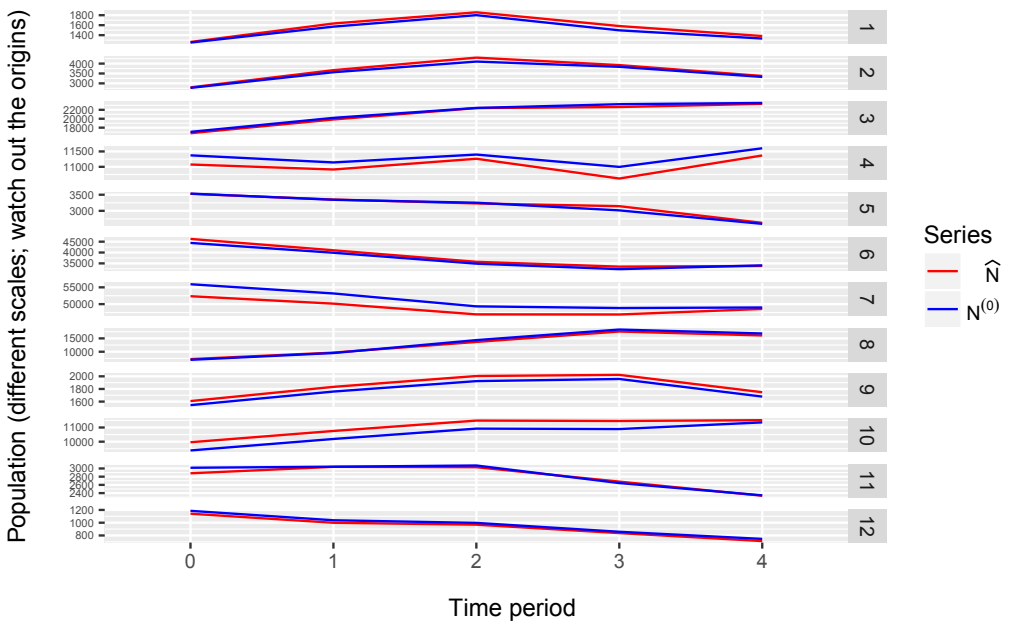

Figure 11: Estimated population for each cell conditioned on initial population for 4 time instants

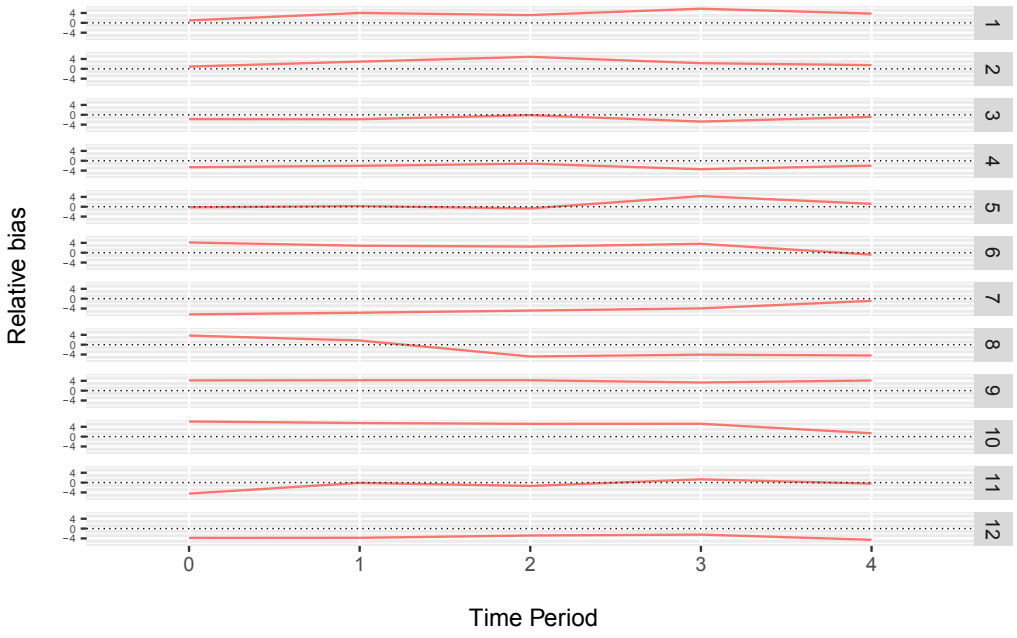

Figure 12: Relative bias of estimated population for each cell conditioned on initial population for 4 time instants

```
matList <- lapply(2:(nT + 1), function(i){
    output <- dcast(MobPop[ID_T == i], ID_CELL_INI ~ ID_CELL_END,
                            value.var = 'N_MNO_1')
    output[, ID_CELL_INI := NULL]
    output <- output / Scale
    return(output)
})

DistributNames <- rep('triang', 12)
Variation <- rep(list(list(cv = 0.10)), 12)
Nt <- rbindlist(lapply(seq(along = matList), function(i){
    output <- rNt(nSim, as.matrix(matList[[i]]),  nReg, fu, fv, flambda,
    DistributNames, Variation, scale = Scale)
    output[, Time := factor(i)]
    return(output)
}))

NTrue <- MobPop
NTrue[, Time := factor(Time - 1)]
setnames(Nt, c('sim', 'Cell', 'N', 'Time'))
seriesNt <- Nt[, list(N = round(mean(N))), by = c('Time', 'Cell')]
Nt0dt <- data.table(N = N0, Time = 0, Cell = seq(along = N0))
setcolorder(Nt0dt, c('Time', 'Cell', 'N'))
seriesNt <- rbindlist(list(Nt0dt, seriesNt))
seriesNt[, Cell := as.factor(Cell)]
seriesNt <- merge(seriesNt, NTrue)
setnames(seriesNt, 'N0True', 'NTrue')
seriesNt <- melt(seriesNt, id.vars = c('Time', 'Cell'),
             value.name = 'Population', variable.name = 'Series')
ggplot(seriesNt, aes(x = Time, y = Population, color = Series, group = Series)) +
           geom_line() +
           facet_grid(factor(Cell) ~ ., scales="free") +
           scale_y_continuous(breaks = scales::pretty_breaks(n = 3)) +
           xlab('\n Time Period') +
           ylab('Population (different scales; watch out the origins)\n') +
           scale_color_manual(labels = c(expression(widehat(N)),
           expression(N^{(0)})),
           values = c('red', 'blue')) +
           theme(plot.title = element_text(face = 'bold', size = 14, hjust = 0.5),
           axis.text.y = element_text(size = 5))
```

In terms of the relative bias, this comparison is depicted in figure 14

```
seriesNt.dcast <- dcast(seriesNt, Time + Cell ~ Series, value.var = 'Population')
seriesNt.dcast[, relBias := round((N - NTrue) /
                       NTrue * 100, 2)][, (c('N', 'NTrue')) := NULL]
seriesNt.melt <-  melt(seriesNt.dcast, id = c('Time', 'Cell'))
ggplot(seriesNt.melt, aes(x = Time, y = value, group = variable, color = variable)) +
          geom_line() +
```

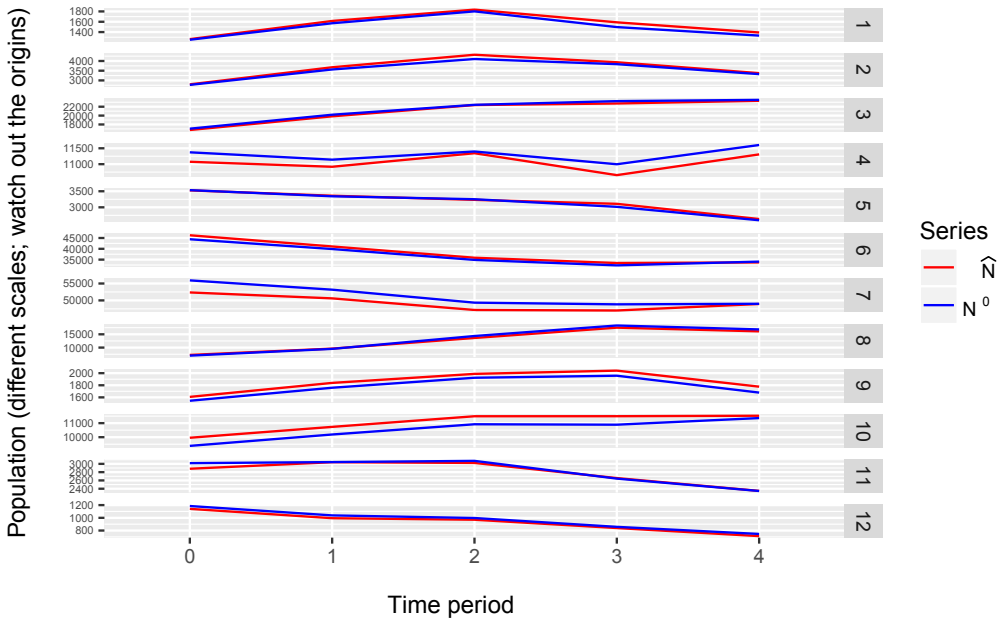

Figure 13: Relative bias of estimated population for each cell unconditioned on initial population for 4 time instants

```
facet_grid(Cell ~ .) +
geom_hline(yintercept = 0, size = 0.3, linetype = 'dotted') +
xlab('\n Time Period') + ylab('Relative bias\n') +
theme(plot.title = element_text(face = 'bold', size = 12, hjust = 0.5),
axis.text.y =
element_text(size = 5), legend.position="none")
```

6 Further developments

`pestim` package contains computationally intensive functions that needs to be optimized in the next versions to keep the running time in acceptable limits even for complex data. The optimizations will be done at two levels:

- each function will be profiled and then improved from the point of view of running time. Neverless, a balance will be kept between the readability of the code and it's performances;
- considering that nowadays almost all computers have multicore processors, we will introduce a level of parallelization in the internal code of the package. We will also show how parallelization can be introduced at the level of the user code (like the examples presented in this document).

The package will also be enhanced in the future with some visualisations capabilities such as maps, grids etc. Other future improvements will consider the underlaying theoretical model:modelling the uncertainty in population register, introducing some spatial correlation between cells and time correlation between succesive periods.

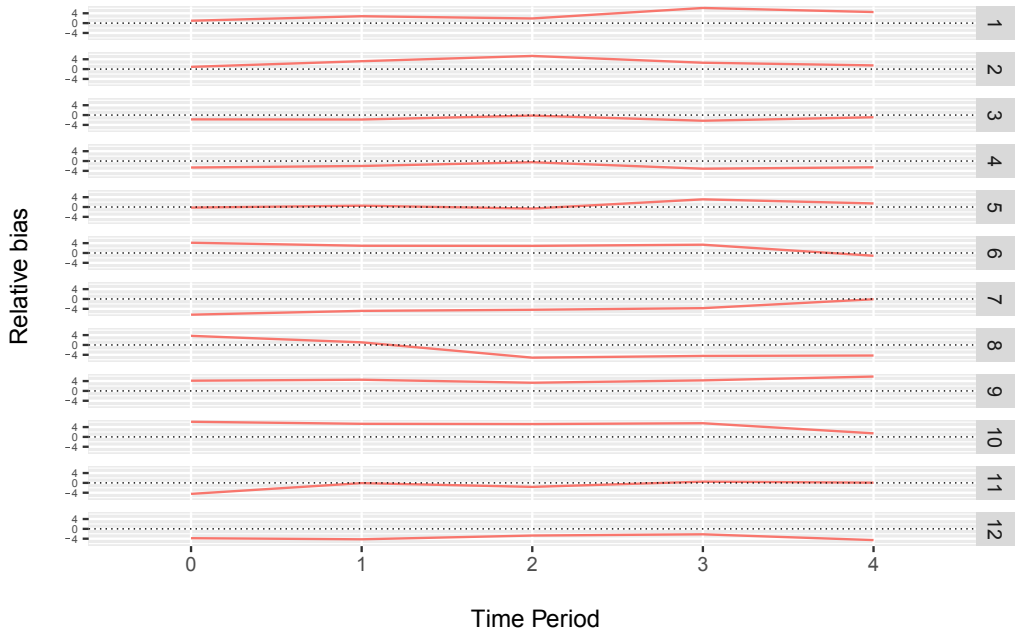

Figure 14: Relative bias of estimated population for each cell unconditioned on initial population for 4 time instants

References

[1] Department of Methodology and Development of Statistical Production, Statistics Spain (INE), A hierarchical model to estimate population counts from aggregated mobile phone data, January, 2018.

[2] Manly, B.F.J., Navarro Alberto, J.A., Introduction to ecological sampling, CRC Press, 2015.

[3] Royle, J.A., Dorazio, R.M., Hierarchical modeling and inference in ecology, Academic Press, 2008.

[4] Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A., Rubin, D.B., Bayesian Data Analysis (3rd ed), CRC Press, 2013

[5] Flajolet, P., Sedgewick, R., Mellin transforms and asymptotics; Finite differences and Rice's integrals, in Theoretical Computer Science 144, pp. 101-124, 1995.

[6] Graham, R.L., Knuth, D.E., Patashnik, O., Concrete Mathematics (2nd ed.), Addison-Wesley, 1996.

[7] Johnson, W.P., The curious history of FaÃă di Bruno's formula, in American Mathematical Monthly 109, pp. 217-234, March, 2002.

[8] Brown, J.W., Churchill, R.V., Complex variables and applications (7th ed.), 2003.

[9] Gradshteyn, I.S., Ryzhik, I.M., Tables of Integrals, Series, and Products (7th ed.), 2007.

[10] Devroye, L., Non-uniform random variable generation, Springer, 1986.

[11] Pearson, J.W., Olver, S., Porter, A.M., Numerical methods for the computation of the confluent and Gauss hypergeometric functions, Numerical Algorithms 74(3), August, 2016.

[12] De Meersman, F., Seynaeve, G., Debusschere, M., Lusyne, P., Dewitte, P., Baeyens, Y., Wirthmann, A., Demunter, C., Reis, F., Reuter, H.I., Assessing the Quality of Mobile Phone Data as a Source of Statistics, presented at Q2016 Conference, June, 2016.

[13] ESSnet on Big Data WP5, Current status of access to mobile phone data in the ESS, 2017.

[14] Oancea, B., Reference Manual, 2018, `https://github.com/MobilePhoneESSnetBigData/pestim/raw/master/doc/pestim_Reference_Manual.pdf`

[15] Watson, G.N., The Harmonic Functions Associated with the Parabolic Cylinder, Proceedings of the London Mathematical Society, 2, 1918, pp. 116âĂŞ148.