An R package to estimate the population using mobile phone data

INE Spain and INS Romania

Implementation of the population estimation methodology

Content of the presentation

Introduction

- We created a *github* account to disseminate the results of the WP5 – Mobile Phone data : ***https://github.com/MobilePhoneESSnetBigData***
- It is not meant to replace the *wiki* page of the project (***https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/index.php/ESSnet_Big_Data***), but mainly to make available the software tools developed during SGA2;
- Currently it contains 2 R packages:
  - ***pestim*** - provides population estimations;
  - ***mobloc*** - provides mobile location algorithms and tools;

Introduction

- The hierarchical methodology developed for population estimation was implemented in an R package: "***pestim***";
- Why R?
  - Freely available;
  - It seems to be the most used software inside the statistics community (at least in EU countries);
  - Portable: there are R distributions for all major operating systems currently in use in the official statistics community (the famous slogan "*wRite once Run anywhere*" is perfectly valid for the R environment too);
- ***pestim*** provides an implementation for the hierarchical model to combine aggregated mobile phone data and external official data to produce estimates of population counts in each cell of a division of a territory;

Introduction

- ***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;
- At this moment, we recommend to install it from sources (requires compilation)
  - For Windows users, Rtools should be installed;
  - For Linux and Mac OS X a proper C++ compiler should be available (gcc or LLVM);
- Installing the package is simple:

```
library(devtools)

install_github("MobilePhoneESSnetBigData/pe
stim", build_vignettes=TRUE)
```

Introduction

- We also provide binaries for Windows and Mac OS X (due to the ongoing development process, the binaries could lag behind the source distribution):
  - for Windows : **https://github.com/MobilePhoneESSnetBigData/Estimation_Population/blob/master/pestim_0.1.0.zip**
  - for Mac OS X : **https://github.com/MobilePhoneESSnetBigData/Estimation_Population/blob/master/pestim_0.1.0.tgz**
- The functions included in ***pestim*** package are computationally intensive and we recommend to be installed on a high performance workstation;
- Some *minimal* hardware requirements (to run some example simulations):
  - a computer with *at least 8GB of RAM memory and an Intel I7 processor with 4 physical cores* (although Intel I5 with 2 cores works, but with the corresponding increase in the running time);

Introduction

- Documentation of the package is available as:
  - A package vignette;
  - A Reference Manual, available at: pestim/doc/pestim_Reference_Manual.pdf
  - Usual R Help for each function included in this package callable from R console with:

➢ `?help(pestim)`

➢ `?function_name`

The underlaying model

- The model implemented in ***pestim*** package is based on two prior assumptions:
  - Given that mobile phone data and official data operate at different time scales, we assume that there is 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 underlaying model

- ***pestim*** package provides two types of estimations:
  - at the *initial time instant t_0*, it makes inferences for the actual population counts in each cell using mobile phone data and administrative data;
  - the *spatial and time evolution at later moments, $t_1, t_2, \ldots t_n$*, are produced using the transition matrices of individuals from cell to cell inferred from mobile phone microdata;
- The generation of simulated populations according to different probability distributions is at the core of the package;

The underlaying model

- The theoretical model implemented in ***pestim*** package can be summarized as:

$$
\begin{aligned}
&N_i^{MNO} \simeq Bin(N_i, p_i), \quad N_i^{MNO} \perp N_j^{MNO}, \quad i \neq j = 1,2,\dots,I \\
&N_i \simeq Po(\lambda_i), N_i \perp N_j, \quad i \neq j = 1,2,\dots,I \\
&p_i \simeq Beta(\alpha_i, \beta_i), \quad p_i \perp p_j, i \neq j = 1,2,\dots,I \\
&(\alpha_i, \beta_i) \simeq \frac{f_1\left(\frac{\alpha_i}{\alpha_i+\beta_i}; N^{REG}, z\right) * f_2\left(\alpha_i+\beta_i; N^{REG}, z\right)}{\alpha_i+\beta_i}, (\alpha_i, \beta_i) \perp (\alpha_j, \beta_j), i \neq \\
&j = 1,2,\dots,I \\
&\lambda_i \simeq f_3(\lambda_i; N^{REG}, z), \quad (\lambda_i > 0, \lambda_i \perp \lambda_j), i \neq j = 1,2,\dots,I
\end{aligned}
$$

- The prior information are incorporated in the probability distributions f_1, f_2 and f_3.

The underlaying model

- The posterior distribution **P**$(N|N^{MNO}; N^{REG})$ is given by:

$$P(N|N^{MNO}, N^{REG}) \propto \int_0^\infty d\lambda\, P(\lambda|\, N^{MNO}; N^{REG}) * Po(N;\lambda)$$

- The integral from the RHS is computed using a Monte Carlo technique using stratified importance sampling and relies on computing the product of a ratio of two Beta functions and the confluent hypergeometric function (${}_1F_1$);
- Computing ${}_1F_1$ is very demanding and it is implemented using a parallel algorithm in C++ and linked to the main package using Rcpp and RcppParallel;

The underlaying model

- The unnormalized posterior density $\mathbb{P}(\lambda | N^{MNO}; N^{REG})$ does not allow us to find easily the corresponding posterior distribution function to apply the inverse method to generate random variables;

- That’s why we’ve made use of the *acceptance-rejection* method.

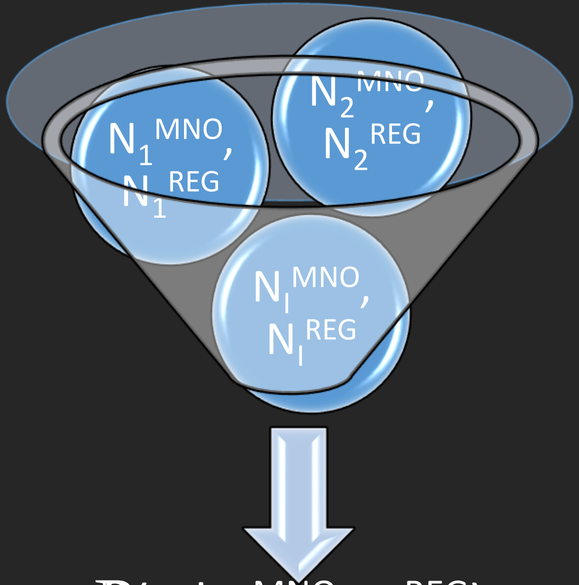
pestim structure

pestim software package contains basically three types of functions:

Auxiliary functions	Computations of mathematical functions, e.g: the confluent hypergeometric function, the ratio of two beta functions, an optimization routine for a concrete probability distribution, etc. Examples: *kummer, Phi, ratioBeta, modeLambda*;
Distribution-related functions	Generation of random values according to different probability distributions for priors, posteriors, and the generation of parameter specifications for these distributions. Examples: *d-p-q-rtriang, d-rlambda, rmatProb, rN0, rNt, rNtcondN0, alphaPrior, genAlpha, genUV.*
Estimation-related functions	Estimates based upon the populations generated with the preceding functions. Examples: ***postN0, postNt, postNtcondN0***

The process of computing population estimation

- The process diagram of computing population estimation using mobile phone and official population data is depicted below:



$\mathbb{P}(N|N^{MNO}; N^{REG})$

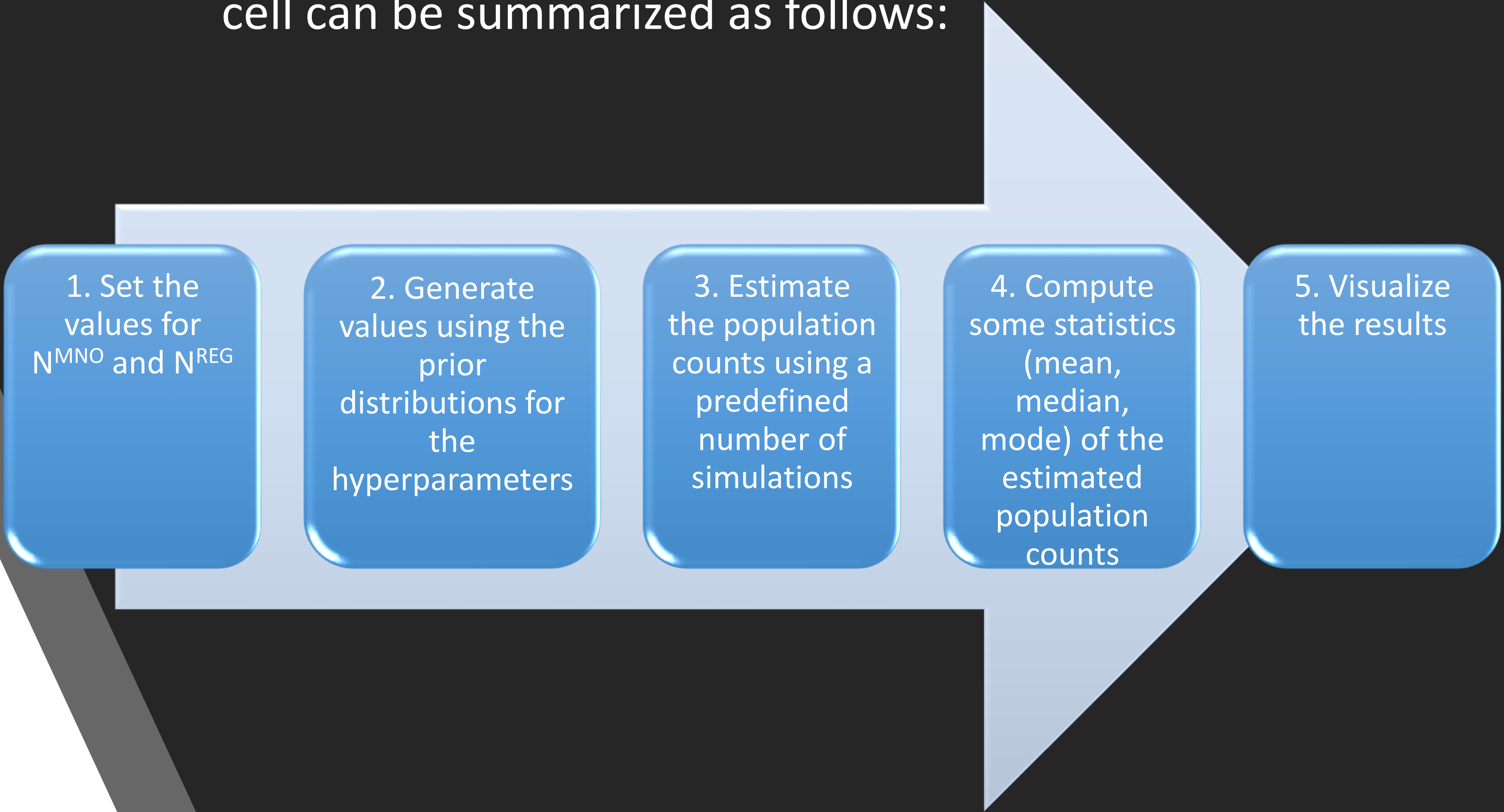
(N_1^{MNO}, N_1^{REG}) … (N_I^{MNO}, N_I^{REG}) are the population counts reported by the MNO in territorial cells and $\mathbb{P}(N|N^{MNO}; N^{REG})$ is the posterior probability distribution that can be used to assess the uncertainty in the output estimates;

Population estimation at a time instant

- In the following slides we will show how to use ***pestim*** package to compute population estimations;
- For the beginning we will show how ***pestim*** can be used to produce estimation at the *initial time instant* t_0 for a single cell;
- Then, we will generalize for several cells and for successive time instants;

Population estimation at a time instant

- The process of estimating the population counts for a single cell can be summarized as follows:



1. Set the values for N^{MNO} and N^{REG}
2. Generate values using the prior distributions for the hyperparameters
3. Estimate the population counts using a predefined number of simulations
4. Compute some statistics (mean, median, mode) of the estimated population counts
5. Visualize the results

Population estimation at a time instant

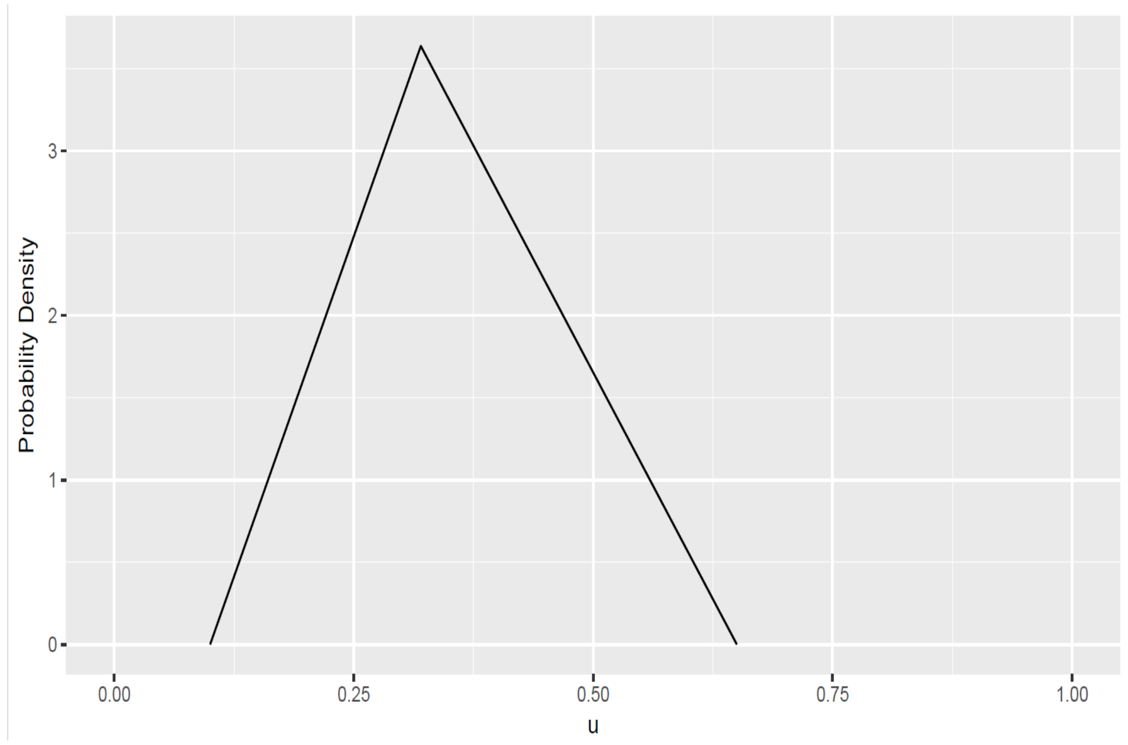
- In the following we will present an example code of estimating the population counts for one cell:
- Step 1: we'll assume:
  - A true population N_0=100;
  - The population count given by some administrative register N^{REG}=97 (a 3% error);
  - The population given by the MNO N^{MNO}=19 (around 20% proportion of detected population);

Population estimation at a time instant

- Step2: generate data for the prior distributions of the hyperparameters;
- Currently, ***pestim*** package supports the following distributions:
  - Uniform distribution;
  - Triangular distribution;
  - Gamma distribution;
  - Degenerate distribution;

Population estimation at a time instant

- For the *uniform* and *gamma* distributions we used the functions provided by the R base package;
- The *triangular* distribution is implemented using *rtriang*, *ptriang*, *dtriang* and *qtriang* functions;
- The triangular distribution can be used for modelling the local market shares u, the cell size v and the hyperparameter λ.
- An example of using the triangular distribution:



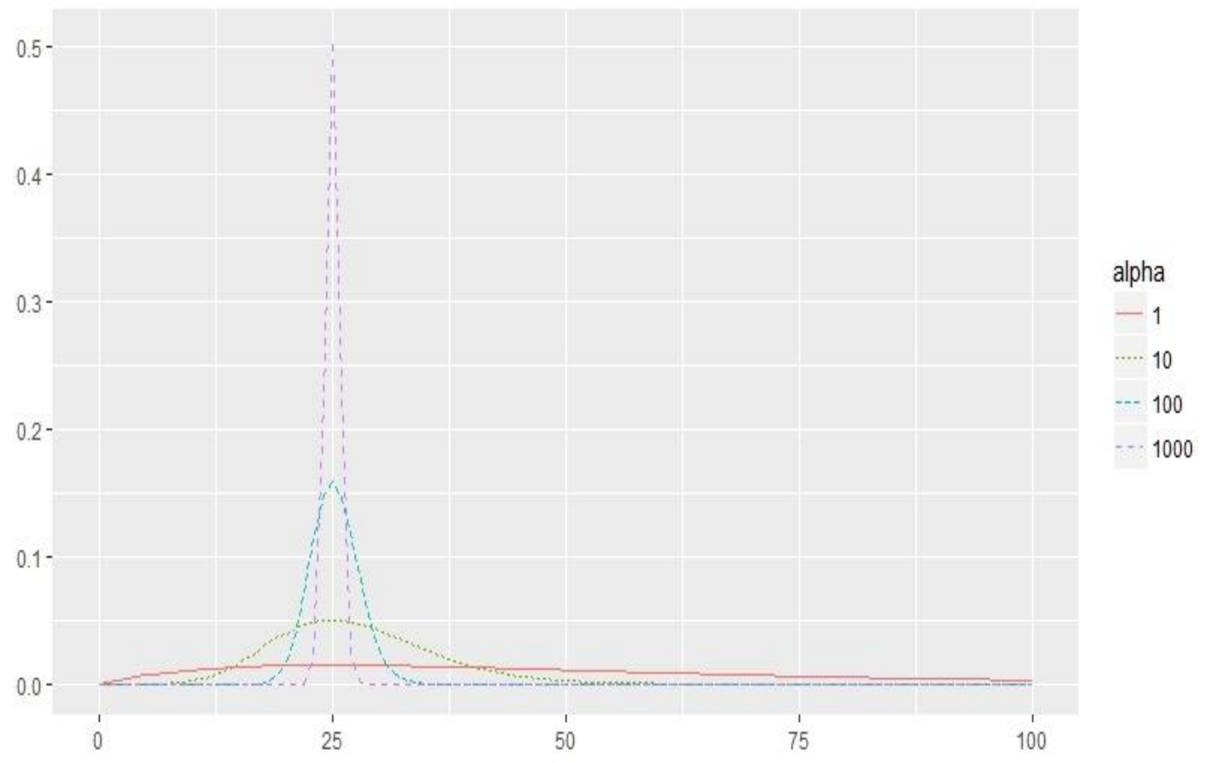
Population estimation at a time instant

```
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’)
```

Population estimation at a time instant

- The *gamma* distribution is another choice for modelling the cell size *v* and the hyperparameter *λ;*

```
alphas <- c(1, 10, 100, 1000)
mode <- 25
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('')
```



Population estimation at a time instant

Population estimation at a time instant

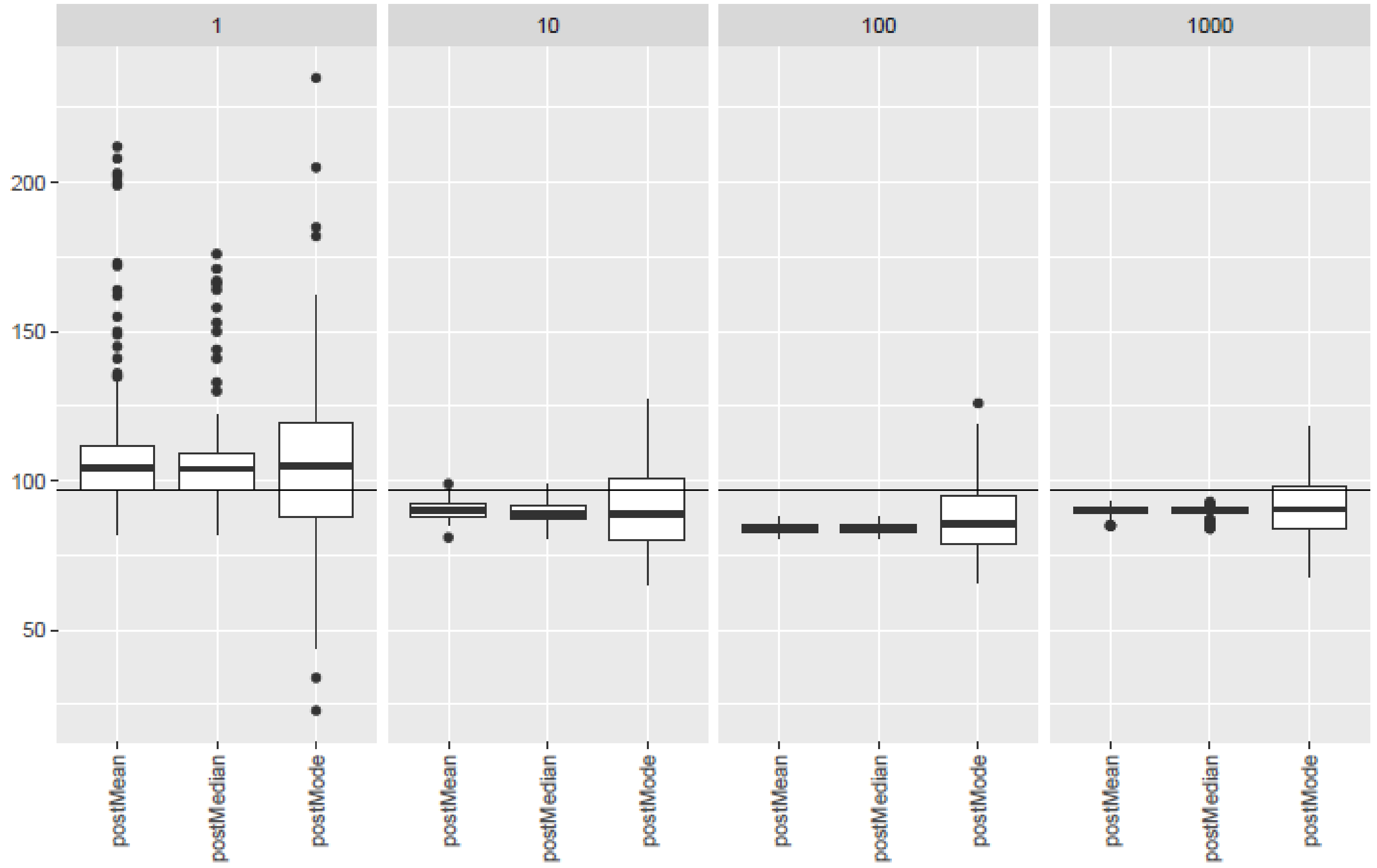
- Denote: $u = \frac{\alpha}{\alpha+\beta}$ (the proportion of detected individuals) and $v = \alpha + \beta$ (the cell size) and use:
  - a uniform distribution f_1 for the *u* parameter
    - $fu = \mathrm{Unif}(u_m; u_M)$ with $u_m = 0$ and $u_M = 0.50$
  - a triangular distribution for the *v* parameter
    - $fv = \mathrm{triang}(v_m, v_M, v_{mode})$ with $v_m = 87$, $v_M=107$, $v_{mode}=97$
  - a gamma distribution for the λ parameter
    - $f_3 \sim \Gamma(\alpha + 1; \frac{N^{REG}}{\alpha})$

Population estimation at a time instant

- Step 3: we’ll compute the population estimations for 4 values of α (1, 10, 100, 1000) and
- Step 4: observe the effect of the amount of uncertainty in the population size;
- Step 5: eventually, some visualization procedures can be used

- For an example code of steps 1 to 5 see the following slide;

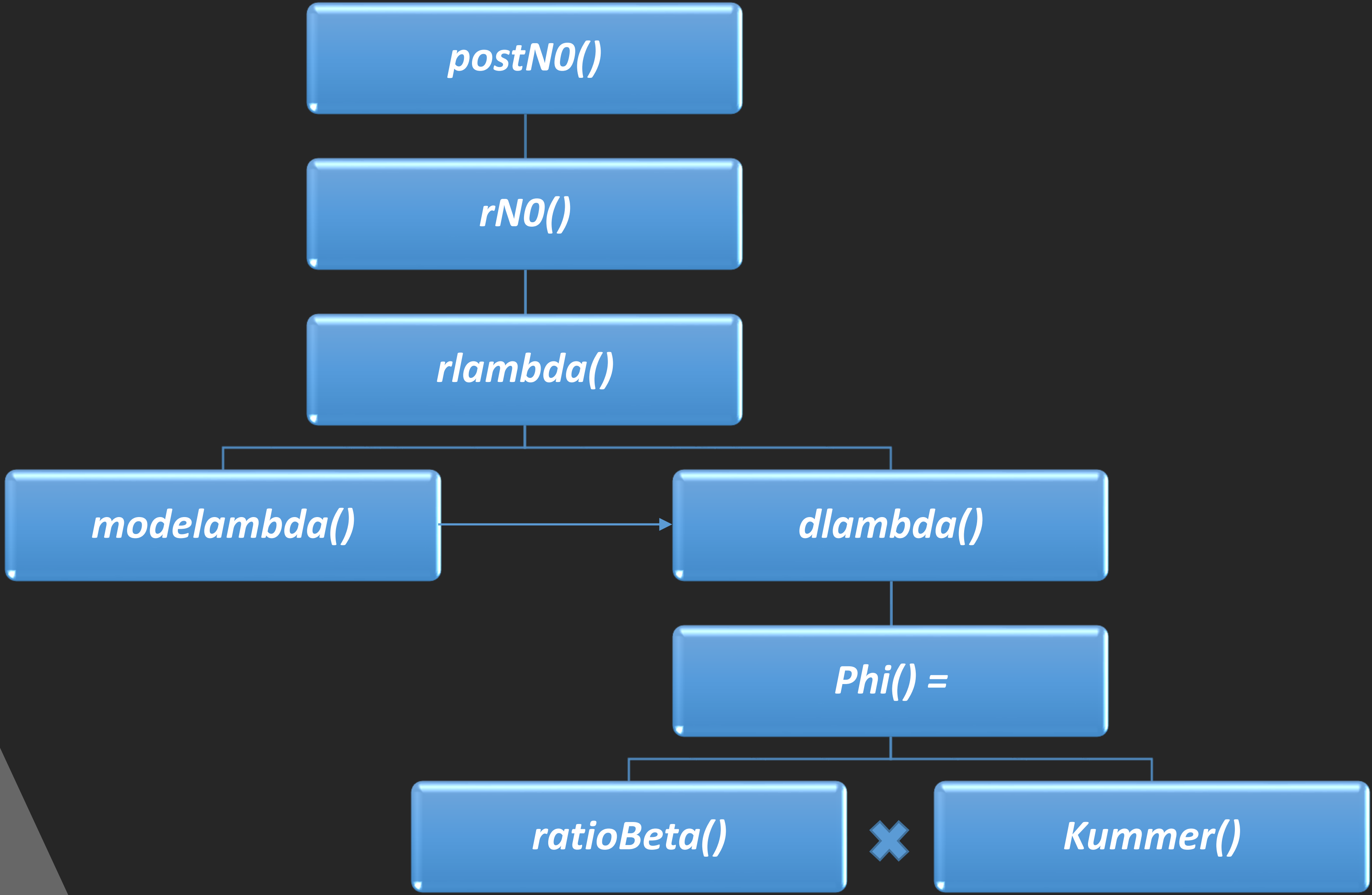
```
# load the required libraries
library(pestim)
library(data.table)
library(ggplot2)

# set de values for nREG, nMNO, and prior distributions
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)
}

#set the number of simulations
nSim <- 100

#generate nSim estimations for each value of alpha
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]
     output <- melt(output, id.vars = 'sim')
     output[, 'alpha' := alpha]
     return(output)
})
names(results) <- alphaSeq
results <- rbindlist(results)
# plot the 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))
```


1
10
100
1000
200
150
100
50
postMean
postMedian
postMode

An interlude: *pestim* internals

- The actual computation of the population estimation is done by the ***postN0()*** that takes the following parameters:
  - nMNO - the number of the individuals detected by the MNO
  - nREG - the number of individuals from the population register;
  - fu and fv - the prior marginal distributions (used for the Monte Carlo integration);
  - flambda - the prior distribution of the lambda parameter;
  - n - the number of points to generate in the posterior distribution for the computation (default is 1e3);
  - scale - a numeric vector with the scale to count the number of individuals (default is 1);
  - relTol - relative tolerance in the computation of the 1F1 (the default value is 1e-6);
  - nSim - number of two-dimensional points to generate to compute the integral (default 1e3)
  - nStrata - integer vector of length 2 with the number of strata in each dimension (default is c(1, 1e2))
  - nThreads – number of threads to be used for computations;

An interlude: *pestim* internals



An interlude: *pestim* internals

- ***postN0()*** generates ***n*** random values for population according to the posterior distribution by calling ***rN0()*** which in turn calls ***rlambda(),*** the posterior distribution being a Poisson distribution;
- ***rlambda()*** computes the mode for the posterior distribution of λ using ***modeLambda()*** and then apply the acception-rejection method to generate the random values;
- ***modeLambda()*** uses the posterior density function of the parameter λ from the hierarchical model, implemented by ***dlambda()***;
- ***dlambda()*** computes the unnormalized posterior density function of the λ parameter:

$$f(\lambda|N^{MNO};N^{REG}) \propto f(\lambda)*\mathrm{dPois}(N^{MNO};\lambda)*\mathrm{S}(\lambda;N^{MNO},N^{REG})$$

An interlude: ***pestim*** internals

where S is given by:

$$S(\lambda, N^{MNO}, N^{REG}) = \int_0^\infty dv f_2(v) \int_0^1 du\, f_1(u)\Phi(uv, (1-u)v; \lambda, N^{MNO}, N^{REG}) = \int_0^\infty dv f_2(v) \int_0^1 du\, f_1(u)\overline{\Phi}\,(u, v; \lambda, N^{MNO}, N^{REG})$$

and is computed using a Monte Carlo technique (with the points needed generated by ***genUV()***) and ϕ is computed as a ratio of two Beta functions (***ratioBeta()*** function) multiplied by the confluent hypergeometric function ${}_1F_1$ (***kummer()***);

- Since ***kummer()*** is one of the most computationally demanding functions in ***pestim*** package we implemented it using C++ language and Rcpp and RcppParallel packages;
- We provide few implementations details in the following slides;

An interlude: *pestim* internals

- We used Watson's lemma and computed the ${}_1F_1(z,a,b)$ function separate for z <80 and z >=80;
- First case: z <80

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

where $(a)_j$ is the Pochhammer symbol

- The implementation of the above formula:

```
A = 1, S = A
for(j=0, A / S< tol; j=j+1)
    $A = A * \frac{a+j}{b+j}\frac{z}{j+1}$
    $S = S + A$
endfor
```

An interlude: *pestim* internals

- Second case: z>=80
- In this case we have: $${}_1F_1(z;a;b) = \Gamma(b)\frac{e^z z^{a-b}}{\Gamma(a)}\sum_{j=0}^{\infty}\frac{(b-a)_j(1-a)_j}{j!\,z^j}$$
- Implementation:

```
A = 1, S = A
for(j=0; A / S > tol; j=j+1)
        $A = A * \frac{(1-a+j)}{j+1} * \frac{(b-a+j)}{z}$
        S = S + A
endfor
S = S * $\frac{e^z * z^{(a-b)} * \Gamma(b)}{\Gamma(a)}$
```

An interlude: *pestim* internals

- After the first tests we found that calling the C++ ***kummer(z, a, b)*** function from R for each tuple of parameters (z,a,b) is inefficient;
- We transformed the function to receive vectors as parameters, to reduce the number of function calls;
- The computation for a tuple (z_i, a_i, b_i) is independent from the computation for (z_j, a_j, b_j) so we can parallelize the computations:

```
1 divide vectors z, a, b in equal chunks
2 for(each chunk $z_c$, $a_c$, $b_c$) do in parallel
      kummer($z_c$, $a_c$, $b_c$)
```

- The parallel version of the algorithm was implemented using RccpParallel

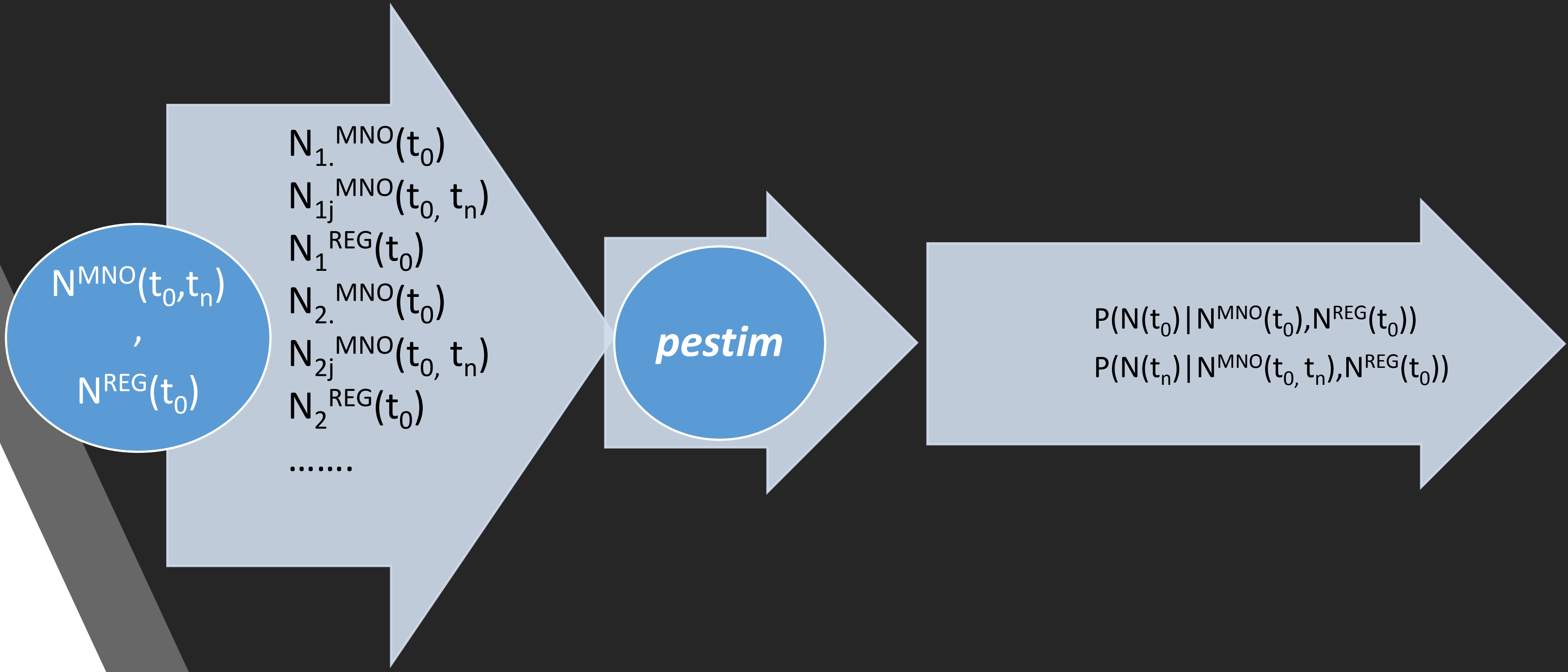
Population estimates for several cells

- Estimation for several cells is straightforward since the estimation in each cell is independent of each other;

- The same function ***postN0()*** should be used, sending it the corresponding parameters for each cell;

Population estimates along a sequence of time

- The process of generating population estimates along a sequence of time is depicted below


$N^{MNO}(t_0,t_n)$
,
$N^{REG}(t_0)$
$N_{1.}^{MNO}(t_0)$
$N_{1j}^{MNO}(t_{0,}\ t_n)$
$N_1^{REG}(t_0)$
$N_{2.}^{MNO}(t_0)$
$N_{2j}^{MNO}(t_{0,}\ t_n)$
$N_2^{REG}(t_0)$
…….
pestim
$P(N(t_0)\,|\,N^{MNO}(t_0),N^{REG}(t_0))$
$P(N(t_n)\,|\,N^{MNO}(t_{0,}\ t_n),N^{REG}(t_0))$

Population estimates along a sequence of time

- As the input data we used the number of individuals N^{MNO}_{ij} (t_0, t_n) moving from cell *i* to cell *j* in the time interval (t_0, t_n) according to the MNO data;
- These data will be combined with official data and to 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, …

Population estimates along a sequence of time

- The computations are based on the following assumptions:
  - we assume that at a given time instant t_0 both population figures in each territorial cell can be equated to some extent. We will take N_i^{Reg} as a fixed quantity without a prior distribution;
  - 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 number of individuals arriving and leaving a cell is estimated using the transition probability among cells.
- We modelled the transition probabilities for a given cell *i* as a multivariate random variable with a Dirichlet distribution with parameters $\alpha_{i1}, \alpha_{i2}, \ldots \alpha_{iI}$

Population estimates along a sequence of time

- The computation of the probability functions $P(N_i(t_n)|N^{MNO}(t_0, t_n))$ for each cell *i* will allow us to choose an estimator (the posterior mean, median or mode) and is conducted in three steps:
  - The initial population value $N_i(t_0)$ is generated for all cells i = 1, 2, …I, using $N_i^{MNO}(t_0)$ as input data and choosing weakly informative priors $f_{ui}, f_{vi}\,, and\ f_{\lambda i};$
  - A transition probability matrix $[p_{ij}(t_0, t_n)]$ is generated according to the model using $N^{MNO}(t_0, t_n)$ as input data and choosing weakly informative priors $f_{\alpha ij}$;
  - The population estimate $N_i(t_n)$ is computed as the sum between the initial population and the number of individuals coming to cell *i* minus the number of individuals leaving cell *i*;

Population estimates along a sequence of time

- ***pestim*** contains a dataset *MobPop* that provides:
  - population counts moving from each pair of cells at successive time instants for a simulated true population;
  - a corresponding official population in a register and a population detected with a MNO;
  - data for prior distributions for u, v and λ.
- In the following we will briefly show how to use ***pestim*** package to estimate population along a sequence of time;

Population estimates along a sequence of time

- 1. The first step is to generate values for $N_i(t_0)$

- set de initial values for nREG, nMNO, and prior distributions f_u, f_v, f_λ as in the previous example;
- generate the population counts at t_0 using the following function call:

```
N0cells <- rN0(n, nMNO_ini, nReg, fu, fv,
flambda, scale)
```

- ***rN0()*** will generate ***n*** random points according to the posterior probability distribution of the number of individuals in the hierarchical model

Population estimates along a sequence of time

- 2. Next, we can provide two types of time evolutions:

A. generate simulated populations conditioned upon their estimated initial size:

  - This is achieved with the following function call:

  `rNtcondN0(n, N0, nMNOmat, distNames, variation)`

  where ***n*** - the number of points to generate, ***N0*** – the mean of the previous population counts, N0cells, ***nMNOmat*** – a transition matrix with the number of individuals displaced from cell to cell detected by the MNO and ***distNames*** the prior distributions;

  - ***rNtcondN0*** function uses ***rmatProb***() to generate the transition probabilities according to a Dirichlet distribution with parameters generated by ***alphaPrior()***;

Population estimates along a sequence of time

B. 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

- A simple example how to produce these estimates is presented in the next slide using 3 consecutive time instants;

```
# The number of generated values
n <- 1e6

#The transition matrix of individuals detected by the MNO
nMNOmat <- rbind(c(9, 2, 5), c(4, 19, 9), c(2, 6, 15))

# Population at the initial time of each cell according to the
# population register
nReg <- c(125, 95, 121)

# List of priors for u and v and lambda
u0 <- rowSums(nMNOmat) / nReg
cv_u0 <- 0.1
fu <- lapply(u0, function(u) {
    umin <- max(0, u - cv_u0 * u)
    umax <- min(1, u + cv_u0 * u)
    output <- list('unif', xMin = umin, xMax = umax)
    return(output)
})

v0 <- nReg
cv_v0 <- 0.10
fv <- lapply(v0, function(u){
    umin <- max(0, u - cv_v0 * u)
    umax <- u + cv_v0 * u
    output <- list('unif', xMin = umin, xMax = umax)
    return(output)
})

cv_lambda <- 0.5
alpha <- 1 / cv_lambda^2 - 1
flambda <- lapply(v0, function(v){
        list('gamma', shape = 1 + alpha, scale = v / alpha)
})

# Names and parameters of priors for the transition
# probabilities
distNames <- rep('unif', 3)
variation <- rep(list(list(cv = 0.20)), 3)

# The population estimations
Nt <- rNt(n, nMNOmat, nReg, fu, fv, flambda, distNames,
variation)$N
```

Thank you!