pestim - an R package to compute population estimations using mobile phone data

Bogdan OANCEA, Ph.D.
(email:bogdan.oancea@faa.unibuc.ro)
University of Bucharest

David SALGADO, Ph.D.
(email: david.salgado.fernandez@ine.es)
Dept. Methodology and Development of Statistical Production, INE, Spain,

Antoniade Ciprian ALEXANDRU, PhD
(email: alexcipro@yahoo.com)
Ecological University of Bucharest and National Statistics Institute of Romania,

Luis SANGUIAO, Ph.D.
(luis.sanguiao.sande@ine.es)
Dept. Methodology and Development of Statistical Production, INE, Spain

ABSTRACT

Integration of the mobile phone data in the production of official statistics was one of the main goals of the ESSnet Big Data project. In this regard, we developed an R package to compute population estimates following a methodology inspired from the ecological sampling techniques. This methodology uses a Bayesian approach that is computationally intensive but allows a straightforward code parallelization. Since some functions of our package are very demanding from a computational point of view, we implemented them in C++ and integrated with the rest of the package using Rcpp. More, the C++ code is also parallelized, and we chose RcppParallel package for this purpose. The estimation procedure combines mobile phone data sets with another data source which can be a population register and produces estimates for each territorial division of a geographical area and along a sequence of time instants for which we have data from Mobile Network Operators. One of the hypotheses of the underlying mathematical model was the independence of the estimates between different cells that allowed us to use a parallel procedure to perform computations for each cell. Besides population estimates, it provides a set of accuracy indicators. pestim was developed with an eye on portability and it can be used on both Windows and Unix-like operating systems. We tested our package and showed that it has a good scalability that is an essential characteristic for very large data sets.

Keywords: *R, big data, mobile phone data*
JEL Codes: *C11, C15, C55, C63, C88.*

1. INTRODUCTION

Integration of the mobile phone data sets in the production of the official statistics was one of the main goals of the ESSnet Big Data project[1] and an important part of the work during this project was to develop an R package which we called *pestim* (Salgado et al., 2018b; Salgado et al., 2018c) to implement the methodology designed to estimate the population counts of different territorial cells. The methodology combines aggregated mobile phone data with other official data about population, for example a survey or administrative data, to estimate population counts. The theoretical model implemented by *pestim* package was based on a hierarchical model borrowed from the ecological sampling techniques (see e.g. Manly and Navarro-Alberto, 2014; Royle and Dorazio, 2014) which follows a Bayesian statistics approach and it allows estimating population counts both at a given time instant and along a sequence of successive periods. A complete description of the methodology can be found in (Salgado et al, 2018a).

Our model used to estimate the population counts is based on two prior assumptions:

- There is an initial time moment when we can equate population figures from mobile phone data sets and official data sets;
- The mobility patterns of individuals are independent on the mobile network operator that they are subscribed to.

The inferential model that we have developed is based on a two-step process:

- At the initial moment t_0 both data sources are used to compute the population counts in each territorial cell;
- At successive moments t_1, t_2, … we compute the transition probabilities from cell to cell using mobile phone data and then we use them to estimate the spatial and time evolution of the target population.

The package was released under the GPL3 and EUPL licenses and it can be found at the following address: *https://github.com/MobilePhoneESSnetBigData/pestim*.

In order to install and run it, R version 3.3.0 or newer is required. Although we provide binary distributions for Windows and Mac OS X operating systems, the most convenient way to install it is from sources, using *devtools* package:

1. https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/index.php/ESSnet_Big_Data

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

Some of the most computationally intensive functions in *pestim* package are written in C++, that is why a proper compiler is required: *Rtools* under Windows environment provides the necessary tools and an open source C++ compiler such as GNU C/C++ or LLVM under Linux or Mac OS X environments suffices. The package is accompanied by a documentation in the form of a vignette, a reference manual and usual R help for each function.

The rest of the paper is organized as follows: in section 2 we present how one can use this R package to estimate the actual population counts starting from aggregated mobile phone data, in section 3 we present some implementation details of the computations done by *pestim*, section 4 is dedicated to performance tests, while the last section concludes our paper.

2. POPULATION ESTIMATION USING *pestim*

In this section we will show how to use this package to estimate the population count, firstly at an initial time instant, then at successive time instants. We will denote by $N^{MNO}= (N_1^{MNO}, N_2^{MNO}, \dots N_n^{MNO})$ the population counts aggregated by a mobile phone operator for a set of territorial cells (1,2,... *n*). The second input needed to estimate the target population is the population count given by an official survey or register $N^{Reg} = \left(N_1^{Reg}, N_2^{Reg}, \dots N_n^{Reg}\right)$. *pestim* combines these two data sources to estimate the actual population counts $N= (N_1, N_2, \dots N_n)$ by computing the posterior probability distribution $P(N|N^{MNO}; N^{Reg})$. Having the posterior *pdf* the user can then compute an indicator such as the mean, mode or median and also some accuracy measures: the standard deviation, coefficient of variation etc. The mathematical model used to compute the posterior *pdf* is briefly presented below. An interested reader can find a detailed presentation and the rationale behind this model in (Salgado et. al, 2018a):

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

and the posterior *pdf* is given by:

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

pestim implements a series of distributions for the priors :

- Uniform distribution;
- Triangular distribution;
- Gamma distribution.

We will show how to estimate the population for a number of cells at the initial time instant considering that the estimations of all cells are independent of each other. For our purpose, we will use some simulated data sets for a set of N=24 territorial cells. We loaded the initial data sources into the dataframe *N_0* which have the columns *N_0$pop_official* storing the data for N^{Reg} and *N_0$pop_phone* for N^{MNO}. We assumed some weakly informative prior distributions for f_1 and f_2 using the uniform distribution for them and a gamma distribution for f_3.

The R code that estimate the actual population count is given below:

```
f1<-list()
f2<-list()
f3<-list()

sc = 100
cv_lambda <- 0.3
alpha <- 1 / cv_lambda**2 - 1
for(i in 1:24) {
  f1[[i]] <- list('unif', xMin = 0.3, xMax = 0.5)
  f2[[i]]<- list('unif', xMin = 1, xMax = N_0$pop_official[i]/sc+10)
  f3[[i]]<-list('gamma', shape = 1+alpha, scale = N_0$pop_
  official[i]/sc/alpha)

}

estim_t0 <- postN0(nMNO = N_0$pop_phone/sc, nReg = N_0$pop_official/sc,
                   fu = f1, fv = f2, flambda = f3, scale=sc,
nThreads = 8)
```

The actual computation is performed by the function *postN0*(). Note that we used the scaling parameter *sc=100* in order to avoid a numerical overflow during the computations and we also used 8 threads to make use of the full computation power of our computer. This function generates *n* random values according to the posterior distribution $\mathbb{P}(N|N^{\mathrm{MNO}};N^{\mathrm{REG}})$ that are

used to compute the mean/mode/median and accuracy indicators. Below is presented an output of the estimation procedure.

```
      Mean  StDev   CV Median Median_CI_LB Median_CI_UB MedianQuantileCV Mode Mode_CI_LB Mode_CI_UB ModeQuantileCV
[1]   6028 1580.19 26.21   5778     5687.675     5879.550          35.32 5852     3411     9356        34.87
[2]   8200 1773.30 21.63   8064     7949.725     8170.475          27.53 6815     5323    12143        32.58
[3]   2779  718.26 25.84   2710     2659         2744              34.85 2967     1449     4238        31.83
...
[23]  4401 1664.99 37.83   4160     4037.525     4242.900          54.43  3954    1546     7636        57.26
[24]  5127 1266.18 24.69   4996     4901.525     5100.475          32.66  5325    2835     7678        30.64
```

The default output of the *postN0*() shows the mean of the posterior distribution together with the standard deviation and the coefficient of variation, the median value for which we computed two accuracy indicators, namely the credible interval bounds, and the quantile coefficient of variation and the mode of the posterior distribution accompanied by the same two accuracy indicators.

Estimation for a sequence of time instants supposes to compute first the probability of transition from one cell to another using mobile phone data. Below we present a code snippet that implements the estimations for 40 successive time instants and the same 24 territorial cells. In the code presented below, `transitions_all` variable stores the number of people $N_{ij}^{MNO}(t_0, t_n)$ moving for a cell *i* to cell *j* in the time interval (t_0, t_n) as the mobile phone operator detects them. We used the same scaling factor as in the previous example, the uniform distribution for f_1 and f_2 and a gamma distribution for f_3. The reason for choosing these distributions is detailed in (Salgado et al, 2018a).

```
sc = 100
nReg = N_0$pop_official

# List of priors for f2
v0 <- nReg /sc
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)
})

# List of priors for f3 (flambda)
cv_lambda <- 0.6
alpha <- 1 / cv_lambda**2 - 1
f3 <- lapply(v0, function(v){list('gamma', shape = 1 + alpha,
scale =  v / alpha)})
```

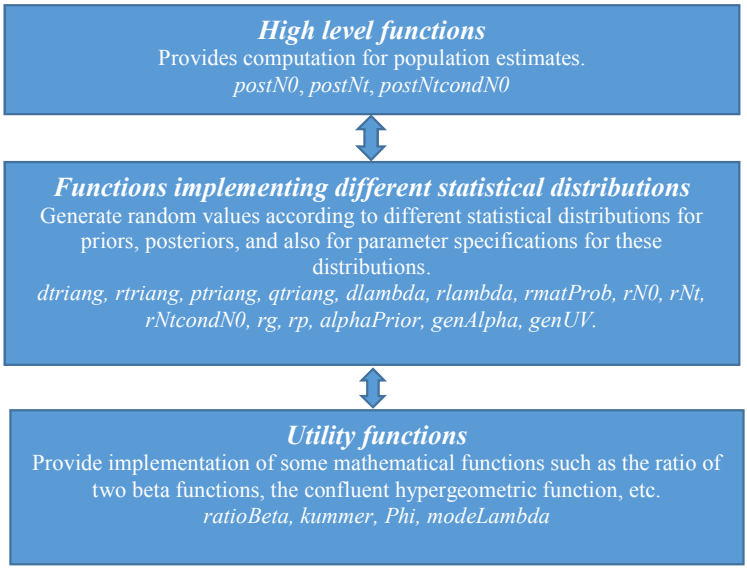
```
# Names and parameters of priors for the transition
probabilities
distNames <- rep('unif', 24)
variation <- rep(list(list(cv = 0.20)), 24)

for(i in 1:40) {
    nMNOmat = transitions_all[[i]]
    u0 <- rowSums(nMNOmat) /nReg
    cv_u0 <- 0.15
    f1 <- 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)
    })
postNt(nMNOmat/sc, nReg/sc, fu = f1, fv=f2, flambda=f3,
distNames, variation, scale = sc, nThreads = 8)
}
```

3. IMPLEMENTATION DETAILS

The *pestim* package has a layered structure that is shown in figure 1.

The structure of *pestim* package

Figure 1

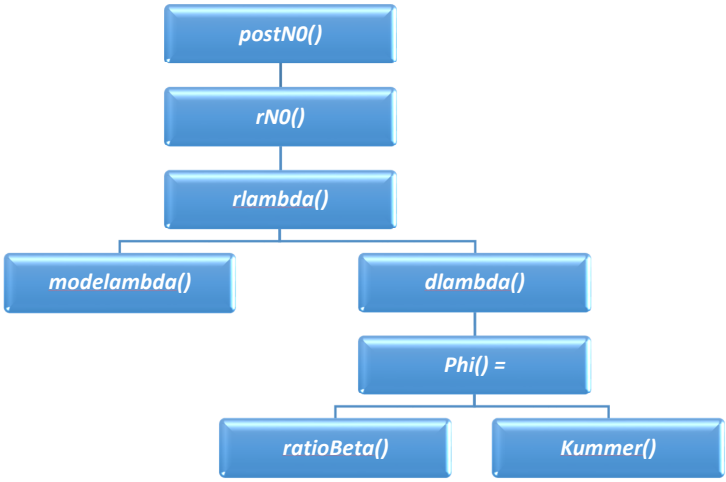


Only the high-level functions should be used to compute the actual population counts, but to pass the proper parameters to these functions the distribution related-functions from the intermediate level presented in figure 1 should also be used. Functions from the bottom level (utility functions) are only for internal computations.

In the following we will present some implementation details for the *postN0*() function that estimates the population counts for a set of territorial cells at the initial time. The complete calling tree of this function is depicted in figure 2.

The calling tree for *postN0()*

Figure 2



This function generates *n* random values according to the posterior distribution (which is a Poisson distribution, see the second row from eq. 1). In turn, *rN0()* calls *rlambda()* to generate these points. *rlambda()* generates these points according to an acception-rejection 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 l parameter. *rlambda()* computes the mode for the posterior distribution of l using *modeLambda*() and then applies the acception-rejection method. This latter function uses the posterior density function of the parameter l in the hierarchical model implemented by *dlambda*() function which computes the unnormalized posterior density function of the l parameter by multiplying a ratio of two beta functions and the confluent hypergeometric function which

is given by a call to *kummer()* function. The mathematical details of these computations are given in (Salgado et al, 2018a).

kummer() was implemented in C++ and called from *pestim* using *Rcpp* package (Eddelbuettel and Francois, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2017) 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 ${}_1F_1(z; a; b)$. The confluent hypergeometric function definition starts from:

$$\mathbf{M}(z;a;b) = \sum_{j=0}^{\infty} \frac{(a)_j}{\Gamma(b+j)} \times \frac{z^j}{j!} \tag{3}$$

with being the Pochhammer symbol:

$$a_0 = 1, (a)_j = a \times (a+1) \times \cdots (a+j-1), j = 1,2,\cdots \tag{4}$$

The sum in equation (3) always converge because **M** is analytic in the entire complex plane. We can define now:

$${}_1F_1(z;a;b) = \Gamma(b)\mathbf{M}(z;a;b) = \sum_{j=0}^{\infty} \frac{(a)_j}{(b)_j} \times \frac{z^j}{j!} \tag{5}$$

which is called the confluent hypergeometric function. A numerical computation procedure for this function is very complicated and is still a hot area of research. We first followed the work of Pearson et al (2016) which recommends splitting the computation according to the value of z: for $z < 80$ a Taylor series expansion is considered the best approach, while for $z \geq 80$ a computation that uses Watson's lemma is proposed (Watson, 1918). After a series of experiments, we concluded that the computation procedure according to Watson's lemma is not robust enough for the parameters regime in our case and we resorted only to the Taylor series expansion. The C++ code implementation is based on the following equations:

$$\begin{aligned} A_0 &= 1 \\ S_0 &= A_0 \\ A_{j+1} &= A_j \frac{a+j}{b+j} \frac{z}{j+1} \\ S_{j+1} &= S_j + A_{j+1}, \ j = 0,1,2,\cdots \end{aligned} \tag{6}$$

where the stopping criterion for the iterative procedure was set as $\frac{A_{j+1}}{S_j} < tol$.

The C++ implementation of the above formulas can be found 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.

We further optimized the execution time of this function starting from the observation that the computations for a tuple (z_i, a_i, b_i) are independent from the computations for (z_j, a_j, b_j) and we can naturally parallelize the computations according to the following schema:

```
 divide vectors z, a, b in equal chunks
 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* package (Allaire et al, 2018).

rN0() generates the random values according to the posterior distribution needed to compute the estimations for the territorial cells. Since the estimation for one cell is independent from the estimations for the other cells, this is the most natural point to introduce further parallelization of the code to reduce the execution time. We used *foreach* package (Microsoft and Weston, 2017a) to compute the estimations for each cell in parallel using *doParallel* package (Microsoft and Weston, 2017b) as a parallel backend with a *snow-like* PSOCK default cluster. The pseudo-code for parallelization is straightforward and is given below.

```
makecluster(nThreads)
for each(i = 1 to nCells) do in parallel {
    random_points[i] = rN0(...)
}
```

4. A PERFORMANCE STUDY OF THE *pestim*

Mobile phone data are usually very large data sets and, in this regard, an important feature of the *pestim* package should be the scalability. In this section we will present the results of our first performance and scalability study of *pestim*.

We used two different computers, one with Mac OS X and the other with Windows operating system. For Mac OS X we explored the performances

of *pestim* with both PSOCK cluster type which is available for all operating systems and FORK cluster type available only for Unix-like operating systems, including here Mac OS X and Linux. Thus, we had three experimental setups that are presented in tables 1 and 2. Both processors have 4 physical cores and supports 8 logical threads.

Experimental setup 1 and 2

Table 1

Processor	Intel I7 Skylake (6820HQ), 2.7Ghz, Turbo Boost 3.6Ghz, L1 32k x 32k per core, L2 256k per core, L3 8 MB
RAM	16 GB, 2133MHz, LPDDR3 SDRAM
OS	Mac OS X High Sierra 10.13.4
R	R version 3.4.1 64 bit
C++ compiler	C++ Compiler: Apple LLVM version 9.1.0
Cluster type	PSOCK for experimental setup 1 / FORK for experimental setup 2

Experimental setup 3

Table 2

Processor	Intel XEON E1246 v3, 3.5 Ghz, Turbo Boost 3.9 GHz, L1 32k x 32k per core, L2 256k per core, L3 8MB
RAM	16 GB 1600MHz DDR3
OS	Windows 8.1
R	R version 3.3.2 64 bit
C++ compiler	GNU C ver. 4.9.3 (min_gw)
Cluster type	PSOCK

We tested the performances of *postN0()* function using the code already presented in section 2, to estimate the population counts for a number of 24 territorial cells with the number of computing threads taking values from 1 to 8. Increasing the number of threads more than 8 makes no sense, since they will compete for the same cores of the processor. Here, the number of threads refers only to the number of computing threads used to evaluate *rN0()* in parallel for each territorial cell and does no include the number of threads used to compute the confluent hypergeometric function which is fixed. We chose to vary this number because it has the most important contribution to the volume of the floating-point operations performed by *pestim*. As we mentioned earlier, mobile phone data are regarded as high-volume data and it is very important for *pestim* to provide a high scalability with respect to the number of territorial cells. In table 3 we present the speedup obtained for each experimental setup and we represented these results in figure 3. The memory requirements for each experiment was approximately 450 MB per core.

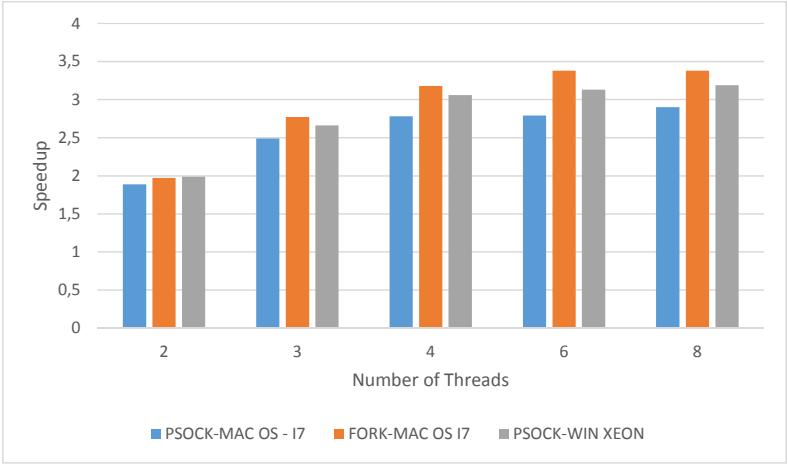
The speedup for postN0()

Table 3

No. threads	2	3	4	6	8
Experimental setup 1					
Speedup	1.89	2.49	2.78	2.79	2.90
Experimental setup 2					
Speedup	1.97	2.77	3.18	3.38	3.38
Experimental setup 3					
Speedup	1.99	2.66	3.06	3.13	3.19

Speedup for *postN0()*

Figure 3



These results show that a FORK based cluster under a Unix compatible operating system performs slightly better than a PSOCK based cluster which is normal if we consider the internal implementation of each type of cluster. The reader can also observe that the speedup become saturated after nThreads = 4 which can be intriguing at a first glance but there are two logical reasons for this behavior. Firstly, both processors have 4 physical cores and support 8 logical threads only by the means of the Hyper-threading technology. However, this technology does not promise a linear scalability, but only an increase in the performance of about 30% (Casey, 2011). Secondly, the parallelization of the computations done by *pestim* is split on two levels: the first level regards the numerical computation of the confluent hypergeometric function and here we kept a fixed number of threads, and the second one regards the generation of random deviates performed by *rN0()* for each territorial cell which should be directly correlated with the number of computing threads. Since some cores of the processor are always busy computing the confluent

hypergeometric function in parallel, this also explains the saturation of the speedup after reaching nThreads = 4.

We performed the same test for estimating the population counts for a number of 40 time instants using *postNt*() function and the results are presented in table 4 and figure 4.

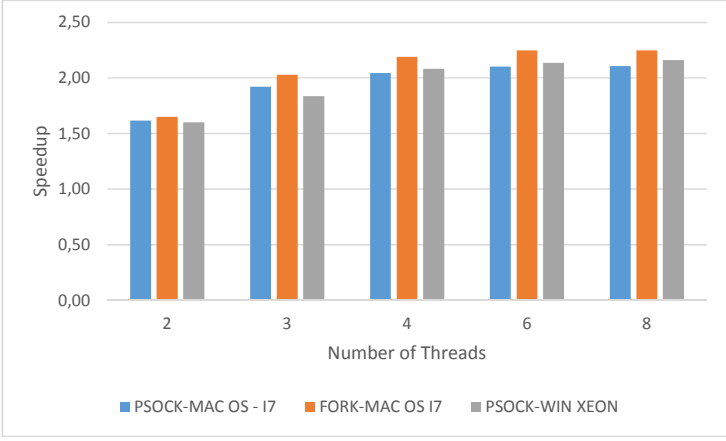
The speedup for *postNt()* for 40 time instants

Table 4

No. threads	2	3	4	6	8
Experimental setup 1					
Speedup	1.62	1.92	2.05	2.10	2.11
Experimental setup 2					
Speedup	1.65	2.03	2.19	2.25	2.25
Experimental setup 3					
Speedup	1.60	1.84	2.08	2.14	2.16

Speedup for *postNt()*

Figure 4



The results are similar to the pervious experiment, showing that a FORK cluster performs better than a PSOCK one. The absolute values for the speedup in this case are less than the ones for postN0() which is again normal considering that there is a code fragment that is run in a serial manner and which prepares the parameters for each territorial cell (the priors). Running this code in parallel will involve an overhead which will overcome the advantage of the parallelization.

5. CONCLUSIONS AND FUTURE DEVELOPMENTS

Mobile phone data is considered one of the most promising data source that can complement the data sources used now by official statistics for estimating population counts. In this regard, we implemented the methodology developed by Salgado et al (2018a) in an R package that combines aggregated mobile phone data with another official data source to estimate the population counts in a number of territorial cells both at an initial moment and at a sequence of successive time instants. We shortly described how this package can be used and then presented few implementations details. We also presented some performance tests since this package should present a good scalability with the number of territorial cells. The results obtained thus far are encouraging, *pestim* package showing a good scalability and also presenting and easy to use high level interface. For the future we intend to further improve the performance of the computations, considering more internal functions as possible candidates to be implemented in C++, and even to explore the possibility of using GPU computations (Oancea and Pospisil, 2018). We also intend to add a visualization layer and to improve the overall robustness of the package.

References

1. **Allaire, J.J., Francois, R., Ushey, K., Vandenbrouck, K., Geelnard, M.**, and **Intel**, 2018, *RcppParallel: Parallel Programming Tools for 'Rcpp'. R package version 4.4.0*. ***https://CRAN.R-project.org/package=RcppParallel***.
2. **Casey, S.**, 2011, *How to determine the effectiveness of Hyper-threading technology with an application*, Intel Technology Journal, vol. 6, issue 1.
3. **Eddelbuettel, D.**, and **Francois, R.**, 2011, *Rcpp: Seamless R and C++ Integration*. Journal of Statistical Software, 40(8), 1-18.
4. **Eddelbuettel, D.**, 2013, *Seamless R and C++ Integration with Rcpp*. Springer, New York. ISBN 978-1-4614-6867-7.
5. **Eddelbuettel, D.**, and **Balamuta, J. J.**, 2017, *Extending R with C++: A Brief Introduction to Rcpp*. PeerJ Preprints 5:e3188v1.
6. **Manly, B.** and **J. E., Navarro-Alberto**, 2014, *Introduction to ecological sampling*. CRC Press.
7. **Microsoft Corporation** and **Steve Weston**, 2017a, *foreach: Provides Foreach Looping Construct for R. R package version 1.4.4.* ***https://CRAN.R-project.org/package=foreach***.
8. **Microsoft Corporation** and **Steve Weston**, 2017b, *doParallel: Foreach Parallel Adaptor for the 'parallel' Package. R package version 1.0.11*. ***https://CRAN.R-project.org/package=doParallel***.
9. **Oancea, B., Pospisil, R.**, 2018, *The performances of R GPU implementations of the GMRES method*, Romanian Statistical Review, no. 1, 121-132.
10. **Pearson, J.W., Olver, S.**, and **Porter, A.M.**, 2016, *Numerical methods for the computation of the confluent and Gauss hypergeometric functions*, Numerical Algorithms 74(3), August.

11. **Royle, J., and R. Dorazio**, 2014, *Hierarchical modeling and inference in Ecology: The Analysis of Data from Populations*, Metapopulations and Communities. Academic Press.
12. **Salgado D., Debusschere, M., Nurmi, O., Piela, P., Coudin, E., Sakarovitch, B., Hadam, S., Zwick, M., Radini, R., Tuoto, T., Tennekes, M., Alexandru, C., Oancea, B., Esteban, E., Saldaña, S., Sanguiao, L.**, and **Williams, S.**, 2018a, *Proposed Elements for a Methodological Framework for the Production of Official Statistics with Mobile Phone Data*, ***https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/images/4/4d/WP5_Deliverable_5.3_Final.pdf***
13. **Salgado D., Debusschere, M., Nurmi, O., Piela, P., Coudin, E., Sakarovitch, B., Hadam, S., Zwick, M., Radini, R., Tuoto, T., Tennekes, M., Alexandru, C., Oancea, B., Esteban, E., Saldaña, S., Sanguiao, L.**, and **Williams, S.**, 2018b, *Some IT elements for the use of mobile phone data in the production of official statistics,* available at ***https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/images/4/4d/WP5_Deliverable_5.3_Final.pdf***
14. **Salgado, D., B. Oancea,** and **L. Sanguiao**, 2018c, *pestim - An R package for population estimation using mobile phone data.* ***https://www.github.com/MobilePhoneESSnetBigData/pestim.***
15. **Watson, G.N.**, 1918, *The Harmonic Functions Associated with the Parabolic Cylinder*, Proceedings of the London Mathematical Society, 2, pp. 116-148.