ESSnet Big Data

Specific Grant Agreement No 2 (SGA-2)

https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata
http://www.cros-portal.eu/

Framework Partnership Agreement Number **11104.2015.006-2015.720**

Specific Grant Agreement Number **11104.2016.010-2016.756**

Work Package 5

Mobile Phone Data

Deliverable 5.4

Some IT elements for the use of mobile phone data in the production of official statistics

Version 2018-05-31

Prepared by: David Salgado (INE, Spain)

Marc Debusschere (Statistics Belgium, Belgium)
Ossi Nurmi, Pasi Piela (Tilastokeskus, Finland)
Elise Coudin, Benjamin Sakarovitch (INSEE, France)
Sandra Hadam, Markus Zwick (DESTATIS, Germany)
Roberta Radini, Tiziana Tuoto (ISTAT, Italy)
Martijn Tennekes (CBS, Netherlands)
Ciprian Alexandru, Bogdan Oancea (INSSE, Romania)
Elisa Esteban, Soledad Saldaña, Luis Sanguiao (INE, Spain)
Susan Williams (ONS, UK)

ESSnet co-ordinator:

Peter Struijs (CBS, Netherlands)
p.struijs@cbs.nl
telephone : +31 45 570 7441
mobile phone : +31 6 5248 7775

Contents

1

Introduction

This deliverable introduces elements for the IT infrastructure necessary to access, store, and process mobile phone data for the production of official statistics. According to the integral approach of the whole ESSnet on Big Data, the forthcoming contents are based upon the concrete experience with actual mobile phone data compiled during the first part of the present project. Therefore, while offering a bottom-up approach, this is clearly conditioned on the success of accessing/using data for our purposes.

In this sense, it is convenient to have in mind an overall description of the whole production process with mobile phone data depicted in figure 1.1 (see WP5.3 (2018)). Following the code of colours, we see that we cannot directly use nor access raw telecommunication data. Statistical microdata can be used in highly limited conditions only in the form of CDRs by INSEE, Istat, and CBS, which can then be aggregated to produce aggregated data, which other partners receive with no choice to participate in their compilation.

It is immediate to conclude how the data access issue shows clear consequences on the needs for infrastructure. In no case in the current research process a partner NSI has needed to develop a specific IT infrastructure to store or process their mobile phone data, whatever their level of aggregation is. At most, in the case of the fruitful collaboration between Orange Labs and INSEE, the latter has been able to use an in-situ platform to access and process CDRs.

Accordingly, the scope of the present document is necessarily limited. The IT tools developed for the present project do not currently differ much from traditional software tools in an NSI because our compiled data are mostly aggregated. The document presents three main chapters. Firstly, taking advantage of the collaboration between Orange Labs and INSEE, we include in chapter 2 a brief description of the IT infrastructure to process mobile phone data for statistical purposes. Secondly, in chapter 3 we provide a

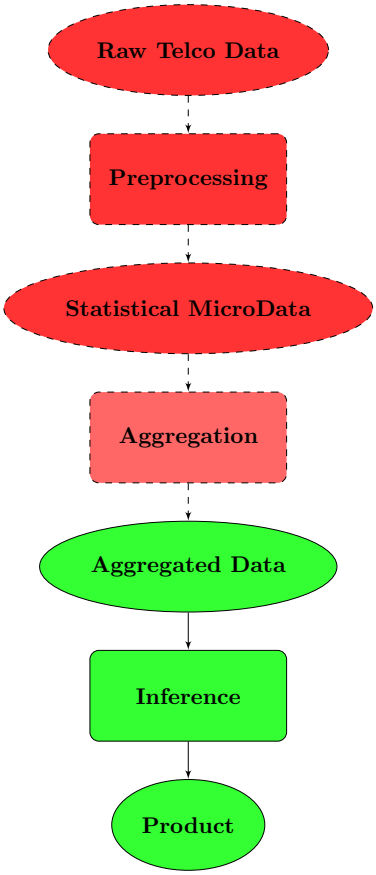

Figure 1.1 Schematic representation of the production process with mobile phone data.

description of the R package called `mobloc` for the location of mobile devices according to our proposals in WP5.3 (2018). Thirdly, in chapter 4 we also provide a description of the R package called `pestim` implementing the hierarchical model introduced in the same preceding deliverable.

2

Some IT infrastructures for analysing mobile phone data

2.1. At NSIs' premises

Although the access conditions have prevented us from fully processing microdata, in some cases we have been able to receive a set of Call Detail Records (CDRs) at the office. Here we exemplify an IT infrastructure to process these data with the example of ISTAT.

In order to manage such an amount of data, Istat acquired the Cloudera Enterprise 5.8 platform and completed the setup of the Big Data in early 2015. The cluster is composed of 8 nodes. Each node has 6 drives of 1.2Tb capacity, 128Gb of RAM, 32 or 16 CPU cores. The connection between nodes has a high performance and 20Gb/s speed.

The platform includes the standard Hadoop framework needed to store and process data, Spark as general engine for large-scale data processing, Impala that is a native analytic database, Hue as Analytics Workbench, Cloudera Manager and Security, which are Cloudera tools, respectively, for Administration and Advanced access control.

In Istat, other interesting uses of the Big Data platform are: the offloading of production databases (historical data storing), heavy processing of data (scanner data analysis), big data staging (agriculture census satellite images).

This new working paradigm requires acquiring and developing some new skills. In addition to more technical skills, it is necessary to consider algorithms and processing flows in a completely new way. This effort is necessary in order to open new approaches to data processing and answer to more challenging questions.

As far as CDRs are concerned, each record contains information about: user id, start and end cell of the caller, date and time of the call, the type of event (SMS or voice call). The number of CDRs in a given period of time can be very large: for example, one MNO collects about 280 million of records in 38 days for a single Italian region.

Under a legal agreement, the MNO transfers CDR data files using a secure channel to an external area shared with Istat. The files are then acquired by Istat systems and are deleted from the exposed area. At this stage, data are in the Big Data cluster but they are not loaded in the HDFS filesystem yet. CDRs are transformed and grouped in a convenient way and subsequently loaded in the HDFS filesystem. From this point of the workflow, it is possible to use Big Data tools environment to process data.

Tools like Impala or Hive make possible to manage files into structures that allow SQL-like querying. The first elaboration is the extraction of the needed columns and then the identification and deletion of duplicate records, this process is named "**Data Collection**". The subsequent step consists in associating a LAU to each record based on geo-localized information of the antenna sector present in the start cell field via BSA (this process is named "**Join CDR and BSA**"). The last step is the creation of some aggregates using Impala, according to the operations described by the pseudo-code in the previous paragraphs. These data are aggregated by LAU and time slots (e.g. daytime and nighttime population).

From now on, aggregated and original data are available to scientists for modeling and for further elaborations. These activities are often better elaborated using high level tools such as Python, R or SAS and the results archived in Hadoop.

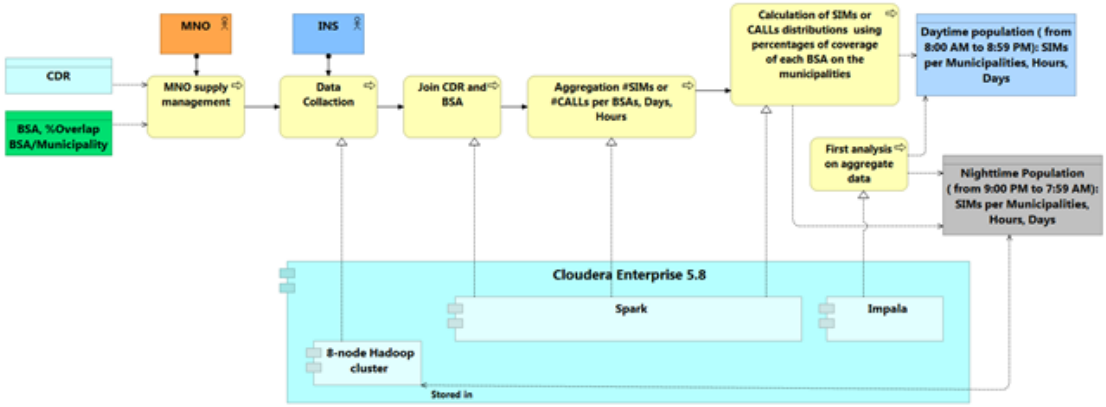

Figure 2.1 The workflow elaboration at Istat.

2.2. At MNOs' premises

2.2.1. Context: Working at Orange Labs premises

2.2.2. Orange Labs access to Orange data

Orange is the main MNO in France, with the largest market share and historically the largest coverage of the country. The R&D of the company is developed in in a large structure called Orange Labs.

Orange collects and stores many sorts of data. Yet Orange Labs does not have a direct access to the data generated on the mobile network. So the procedure is that Orange Labs teams make a request for a specific data flow to be opened, meaning a determined time frame of data collection for instance. Then the data is stored and processed on platform dedicated to R&D, that we describe below, and no production process runs on it.

The same type of data that is exploited for R&D is also the raw material for some commercial entities within the Orange group. FluxVision sells estimates of present population attending some events or estimation of flows between cities for some studies.

2.2.2.1. INSEE access to Orange Labs data

A convention between Eurostat, INSEE and Sense from Orange Labs, granted access to a specific dataset of CDR from 2007 for research purposes. This allowed the exploitation of this large record (more than 2To) of individual data by INSEE researchers. Yet all the work has only been conducted within Orange Labs premises, without any sort of remote access. This is why the infrastructure here described is not in place at INSEE but belongs to Orange Labs.

2.2.3. Description of the infrastructure

Orange Labs set up an Hadoop cluster for big data processing. This infrastructure aims at integrating as many technologies from the Big Data ecosystem as possible.

2.2.3.1. Architecture

The architecture is represented in figure 2.2.

2.2.3.2. Specifications

Hardware
The Hadoop ecosystem relies on a VMWARE cluster for SAN storage. According to the interview technicians this comes from the iterative process of building the infrastructure

more than clear choice of an adapted technology for large volumes within the Hadoop framework.

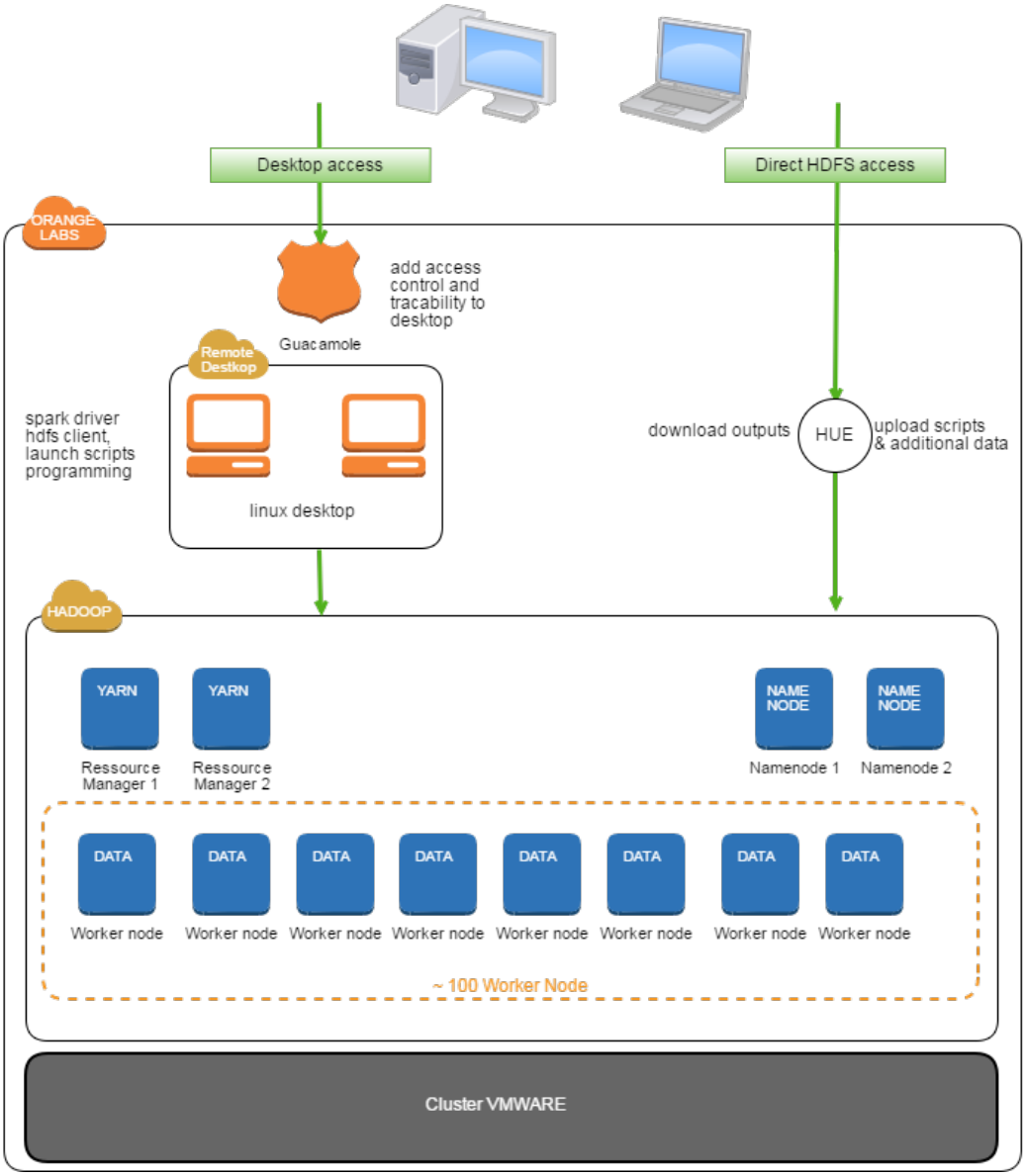

Figure 2.2 Architecture of Orange Labs big data platform

Hadoop
Orange Labs makes no use of a distribution like Cloudera or HortonWorks. Installations are run from sources or binaries of the different projects, as Hadoop Apache for instance. There are around a hundred nodes on the cluster with Yarn as a dependency manager.

The virtual machines have 24 Go for RAM on average. The operating system is a Centos distribution that is costly to maintain evolving so as to ensure the compatibility with every application of the Hadoop framework.

The total space in the HDFS cluster, the distributed file system, is around 3,5 Po. The access is secured through Kerberos authentication. The goal is to limit the access rights of each user (there are around 300 users) to the data according to the "need-to-know" principle. This is particularly important since the individual data that is stored is highly re-identifiable and many people from diverse teams (sociologists, ergonomists, communication experts, IT profiles...) have access to the HDFS.

Because so many projects are led on this platform, the choice was made to impose static quotas of RAM and CPU at the level of departments.

So as to offer a personal storage space on HDFS, HUE (Hadoop User Experience) has been installed. It allows to upload scripts or additional data (from official statistics for instance) and to download outputs like aggregates. There is no a priori control of the outputs.
To ensure both flexibility for the user and a fair level of traceability, Apache Guacamole gives access with SSH or RDP (remote desktop control) to a remote desktop recording the sessions and dealing with the authentication. It is through this interface that shells can be opened for programing and scripts that were uploaded in HDFS thanks to HUE can be launched.

Actually these office environments make use of the same distribution of the operating system. That allows for an easy execution of some solutions drivers like Spark. This environment also provides an HDFS client to give access to the data.

2.2.4. Feedback and experience from the user point of view

2.2.4.1. Ergonomy

Orange Labs infrastructure is very adapted to the manipulation of 5 months of CDR, for not too complicated operations computing times are quite short compared to the billions lines in the record. Yet one may need time to be familiar with the access to

the platform. The double access through HUE and Guacamole, each offering different features makes it a bit heavy for programming and debugging.

2.2.4.2. Keeping the software versions up-to-date

Another drawback of this solution is that maintaining the softwares up-to-date seems to require many efforts. Updating the versions for Spark or Python for instance has proven to take several months. In the context of the Big Data solutions evolving a very high pace one may find frustrating not to be able to enjoy the new features that are regularly released.

3

mobloc - an R package for mobile location algorithms and tools

3.1. Introduction

Data collected from the mobile phone network do typically not contain the exact geographic location of the logged events. Instead, only the id number of the site (which is the construction that contains one or more antennae, e.g. a cell tower) and the cell (antenna) are included.

The `mobloc` package contains a set of tools to approximate the location of mobile phone devices. For this approximation, the signal strength of the cells are modelled. Also, the fact than cells often overlap is taken into account.

Besides the mobile phone network data (Call Detail Records or signalling data), the *cellplan* is needed for the estimation of geographic locations. It contains the metadata of the cells. The number of variables that are included may vary. The more variables included, the better. The only required variables are the latitude and longitude of the cells. Other useful variables are: height, (horizontal) tilt, direction, horizontal beam width, vertical beam width, and the type of cell. These variables are used in 'mobloc' to approxiamate the location of mobile phone devices. There may be other, more advanced, variables that are useful to estimate the geographic location, such as Timing Advance. However, there are no methods implemented yet to use these variables.

The methods used in this package are described in Tennekes (2018), which is recommended to read first.

```
#library(mobloc)
library(devtools)
load_all()
```

3.2. Setup location model parameters

The first step to apply the approximate the geographic locations, is to determine the model parameters. The default parameters can be loaded with the function `location_model_parameters`. The result is a standard list:

```
param_default <- location_model_parameters()
str(param_default)
#> List of 13
#>  $ db0_tower            : num -45
#>  $ db0_small            : num -60
#>  $ azim_min3dB          : num 65
#>  $ azim_dB_back         : num -30
#>  $ elev_min3dB          : num 9
#>  $ elev_dB_back         : num -30
#>  $ db_mid               : num -92.5
#>  $ db_width             : num 5
#>  $ poly_shape           : chr "oval"
#>  $ max_range            : num 10000
#>  $ max_range_small      : num 100
#>  $ area_expension       : num 4
#>  $ max_overlapping_cells: num 20
#>  - attr(*, "class")= chr "location_model_parameters"
```

A short description of the parameters is provided in the table below. Some of these parameters are already discussed in Tennekes (2018). The first set of eight parameters are used to model the signal strength. The second set of five parameters are used to determine the coverage area. The parameters will be explained in more detail when needed.

Parameter	Description (related to signal strength)
db0_tower	Signal strength at 1 meter from cell (S_0 in Tennekes (2018)) for normal cells
db0_small	Signal strength at 1 meter from cell (S_0 in Tennekes (2018)) for small cells
azim_min3dB	Horizontal beam width γ_j in Tennekes (2018))
azim_dB_back	Signal strength at the back of the cell in the azimuth plane
elev_min3dB	Vertical beam width θ_j in Tennekes (2018))
elev_dB_back	Signal strength at the back of the cell in the elevation plane
db_mid	Midpoint of the logistic transformation (S^{mid} in Tennekes (2018))
db_width	Width of the logistic transformation (S^{width} in Tennekes (2018))

The `mobloc` package contains a tool in which the first set of eight parameters can be tuned. This tool is started as follows:

```
param_current <- cell_modelling_tool(param_default)
```

All parameter values that are set with the interactive tool are silently returned by `cell_modelling_tool`, and in this example assigned to `param_current`. The function `update_model_parameters` can also be used to change parameters.

Parameter	Description (related to coverage area)
poly_shape	Baisc shape of the cell coverage area, one of *pie*, *oval* , *Voronoi*
max_range	Maximum range of normal cells
max_range_small	Maximum range of small cells
area_expension	Expension factor of cell areas used to allow overlap
max_overlapping_cells	Maximum amount of cells that are overlapping each other

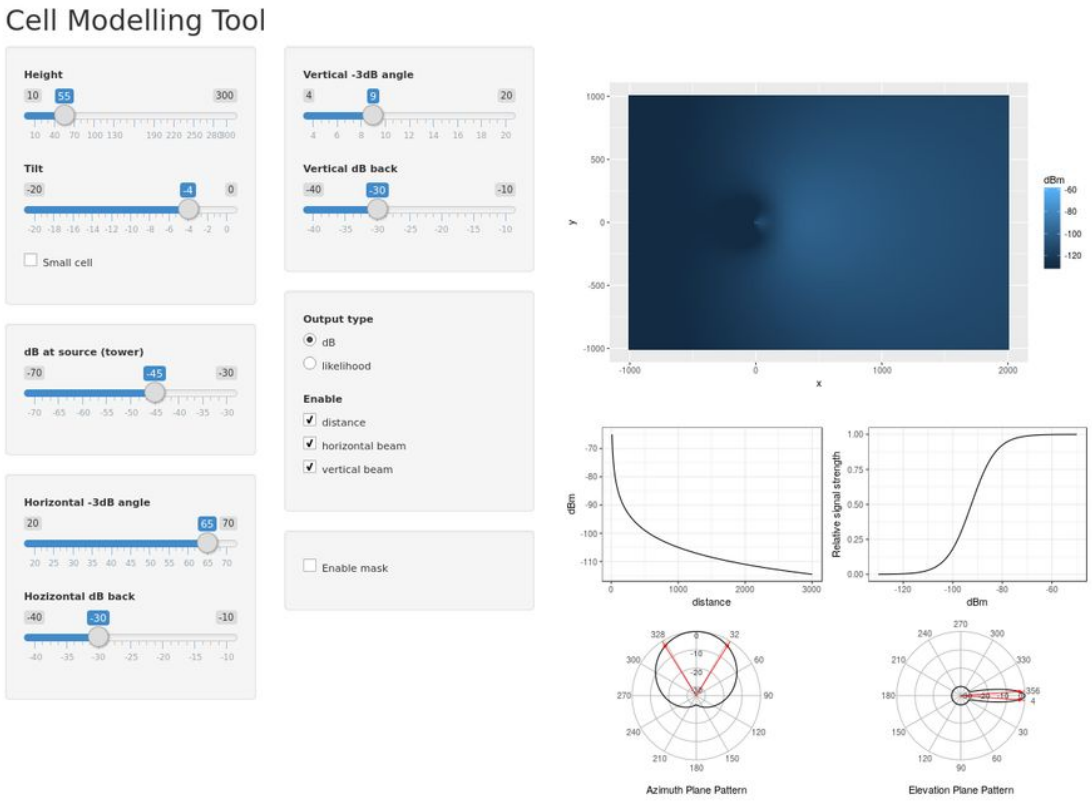

Figure 3.1 Cell modelling tool.

The top left box shows the settings of a cell which should be contained in the cellplan, namely, the height, the horizontal tilt (β_j in Tennekes (2018)), and whether it is a small cell (i.e. omnidirectional). If these variables are not available, not all tools from the `mobloc` package can be used.

The top right plot shows the top view of the signal strength of the cell. If *small cell* is unticked, than the azimuth direction (α_j in Tennekes (2018)) is east.

The next input boxes on the leftside of the screen configure the cells. These parameters are often contained in the cellplan. If not, the default parameters can be used. The first parameter is `db0_tower` or `do0_small` depending on whether the *small cell* checkbox is ticked. Notice that different default settings are used (-45dBm for normal cells, and -60dBm for small cells), since cells contained in cell towers and on rooftop are often much stronger than small cells. The top left plot below the heatmap shows the signal loss as a function of the distance (see equation (4) in Tennekes (2018)).

The next two input boxes contain the -3dB angles and dB back values for both the horizontal (azimuth) and vertical (elevation) planes. The radiation area of a cell can be seen as a three dimensional bulb. In the main direction (e.g. the direction in which the cell radiates) there is no loss in signal strength, irrespective of the distance. However, at a certain offset, there is signal loss. The -3dB angle is the angle at which the signal is halved. These angles (both for the horizontal and vertical plane) are usually contained in the cellplan. The dB back points are signal loss ratios between the front and the back of the cell. The two plots the the right bottom of the screen illustrate the radiation pattern in the horizontal/azimuth plane the the vertical/elevation plane. The black contour lines indicate the signal loss as a function of the offset angle. The red points are the -3dB points. These radiation plots can also be generated directly in R:

```
radiation_plot(beam_width = 65, db_back = -30)
radiation_plot(type = "e", db_back = -30, beam_width = 9)
```

The output type sets the type of radiation plot that is shown in the heatmap. dBm means the absolute signal strength value, and the likelihood the likelihood of connection, given the abolute signal strength values.

The transformation from absolute signal strength values to the likelood values is achieved by applying a logistic function, which is parameterized by `dm_mid` (S^{mid} in Tennekes (2018)) and `db_width` (S^{width} in Tennekes (2018)). The reason to apply such a transformation is that the probability of connection not only depends on the signal strength, but also on load balancing. For load balancing, the tails of the distribution are less important, e.g. whether a signal is very good (say -80dBm) or superb (say -70 dBm) is less important than the availability of that cell. The transformation function is plotted in the top right chart below the heatmap. The transformation is defined in equations (6) and (7) of Tennekes (2018).

3.3. Loading artificial cellplan data

When the parameters have been set, the model can be applied to cellplan data. To illustrate the model, we included artificial data to this package. This data can be loaded as follows:

```
data("ZL_cellplan", "ZL_land", "ZL_elevation")
```

It is artifical cellplan data from the NUTS3 region Zuid-Limburg, the most southern part of the Netherlands, which is roughly 30 by 30 kilometres large.

The object `ZL_cellplan` is an `sf` object (see packge `sf`) that contains all the geopgraphic locations of the cells and the metadata.

```
head(ZL_cellplan)
#> Simple feature collection with 6 features and 9 fields
#> geometry type:  POINT
#> dimension:      XY
#> bbox:           xmin: 176270.6 ymin: 317988.3 xmax: 189389.9 ymax: 335854.1
#> epsg (SRID):    28992
#> proj4string:    +proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889 +k=0.9999079 +x_0=155000
#>   Cell_name        x        y         z direction tilt beam_h beam_v small
#> 1     6d52a 176270.6 317988.3 104.23212        18    8     62   14.0 FALSE
#> 2     6d52b 176270.6 317988.3 104.23212       138    6     62   14.0 FALSE
#> 3     6d52c 176270.6 317988.3 104.23212       258    6     62   14.0 FALSE
#> 4     5e56a 189389.9 335854.1  78.89991       312    6     65    7.2 FALSE
#> 5     5e56b 189389.9 335854.1  78.89991       112    6     65    7.2 FALSE
#> 6     5e56c 189389.9 335854.1  78.89991       212    5     65    7.2 FALSE
#>                     geometry
#> 1 POINT (176270.6 317988.3)
#> 2 POINT (176270.6 317988.3)
#> 3 POINT (176270.6 317988.3)
#> 4 POINT (189389.9 335854.1)
#> 5 POINT (189389.9 335854.1)
#> 6 POINT (189389.9 335854.1)
```

The object `ZL_land` is a large multipolygon that defines the area. The object `ZL_elevation` is a `raster` object that contains the elevation heigths at 100 by 100 metre detail.

These example data can be plot with the `tmap` package:

```
library(tmap)
tmap_mode("view")
qtm(ZL_elevation) + qtm(ZL_land, fill=NULL) + qtm(ZL_cellplan)
```

The corresponding bounding box of Zuid-Limburg is created as follows.

```
library(sf)
#> Linking to GEOS 3.5.1, GDAL 2.2.1, proj.4 4.9.3
ZL_bbox <- st_bbox(c(xmin = 172700, ymin = 306800, xmax = 204800, ymax = 342700), crs = st_crs(28992))
```

3.4. Creating cell polygons

The first step is to create a polygon for each cell. Polygons are created with the function `create_cellplan_polygons`. We can create three types of polygons: `pie`, `oval` (pie with round edges), or `Voronoi`. The polygons defines the overall coverage area of a cell. However, the signal strength may vary within this coverage area. It is recommended to define the polygons are large as possible. However, the computation time increases a lot when the polygons are too large, especially when there is much overlap.

First, let us illustrate a Voronoi tessalation. For this we create a new parameter list, where we only adjust the polygon shape.

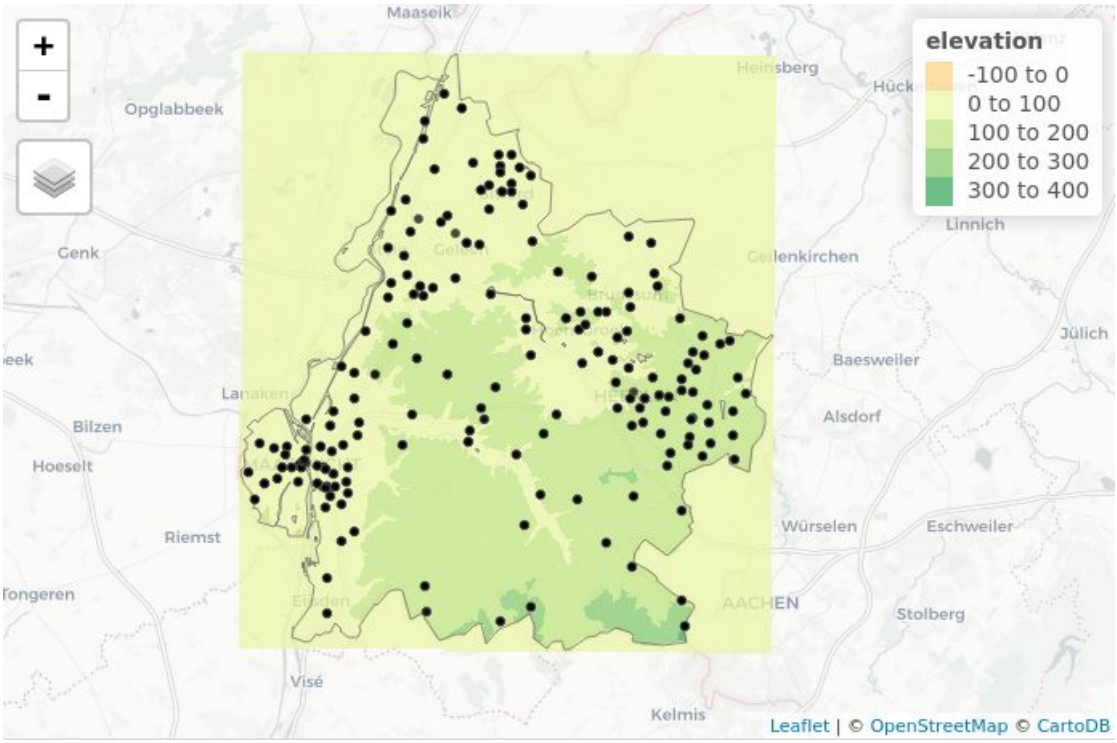

Figure 3.2 A large multipolygon.

```
param_voronoi <- update_model_parameters(param_current, poly_shape = "Voronoi")
ZL_voronoi <- create_cellplan_polygons(ZL_cellplan, ZL_land, ZL_bbox,
param = param_voronoi)
```

The `sf` object `ZL_voronoi` contains polygons for every cell. When cells are directional, the centroids for which the Voronoi polygons are generated, are shifted 100 meters in the direction of radiation. The major advantage of this adjustment to the Voronoi algorithm is achieved when there are multiple cells per site, typically 120 degrees apart. The result is that due to the shifted geographic locations, a Voronoi polygon is created for each cell.

The result can be visualized and inspected as follows.

```
qtm(ZL_voronoi) + qtm(ZL_cellplan)
```

As discussed in Tennekes (2018), there are a couple of downsides to the Voronoi tesselation. The most important one is that in reality, cells overlap in order to make to mobile phone network robust and dynamic. When a cell tower is out of order, or when the capacity is reached, other nearby cells can take over the connections.

In the following line of code, we create polygons that have a shape of an oval. The

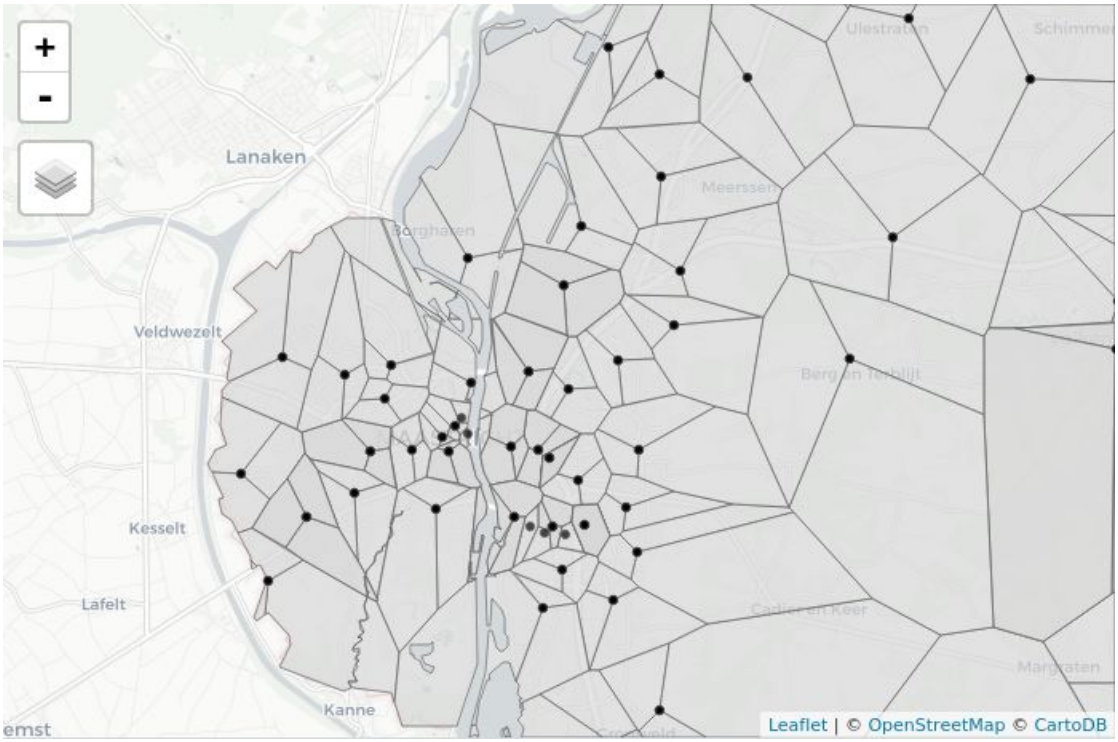

Figure 3.3 Voronoi polygons.

size is upper bounded by the parameters `max_range` and `max_range_small` for normal and small cells respectively. The sizes of the polygons are determined as follows. Under the hood, a Voronoi tesselation is generated, just like we did before. The sizes of those Voronoi polygons times the parameter `area_expension` determine the sizes of the polygons. The result is that cells in urban areas have smaller coverage areas, but still overlap each other.

```
ZL_poly <- create_cellplan_polygons(ZL_cellplan, ZL_land, ZL_bbox, param = param_current)
qtm(ZL_poly) + qtm(ZL_cellplan)
```

Recall that each of these polygons only determines the coverage area of the corresponding cell, but not the signal strength. At this stage, the shape of the polygons is not very important, as long as it covers the area for which the signal strength is expected to by high.

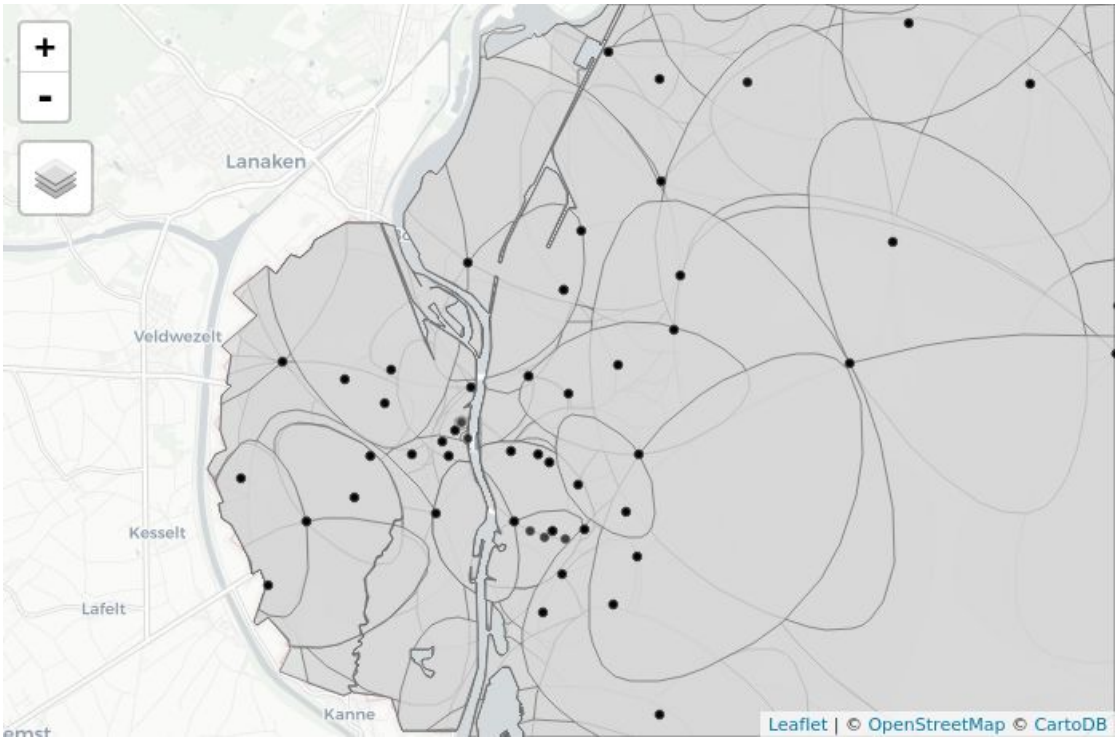

Figure 3.4 Ovals.

3.5. Calculating relative signal strength probabilites

In the final stage, the relative signal strength probabilities are calculated. See equation (1) in Tennekes (2018). These are the probabilities of presence at a certain grid cell (say 100 by 100 metres) given the id of the logged cell. Recall that these probabilities depend on the relative signal strengths (see $s(i, j)$ in equation (6) in Tennekes (2018)) of the logged cell as well the cells that also have coverage at the corresponding location.

The function to calculate these relative signal strength probabilities, called `rasterize_cellplan`, supports parallel computing. The `mobloc` package contains a couple of functions to manage the parallel backend:

```
current_cluster()
# No cluster defined.
start_cluster(4)
# Cluster with 4 nodes created.
current_cluster()
# Cluster with 4 nodes found.
stop_cluster()
# Cluster with 4 nodes stopped.
```

The following code chunk will calculate the relative signal strength probabilities. It returns a `data.frame` which consists of the following variables: `Cell_name`, `rid` (raster cell id), `p` (probabilities), `s` (relative signal strength), `dist` the distance between the cell and the raster cell, and `db` the signal strength in dBm.

```
ZL_raster <- create_raster(ZL_bbox)
ZL_prob <- rasterize_cellplan(cp = ZL_cellplan, cp_poly = ZL_poly, raster = ZL_raster,
elevation = ZL_elevation, param = param_current)
```

The result can be visualized using `cell_inspection_tool`.

```
cell_inspection_tool(ZL_cellplan, ZL_poly, ZL_raster, ZL_prob, param_current)
```

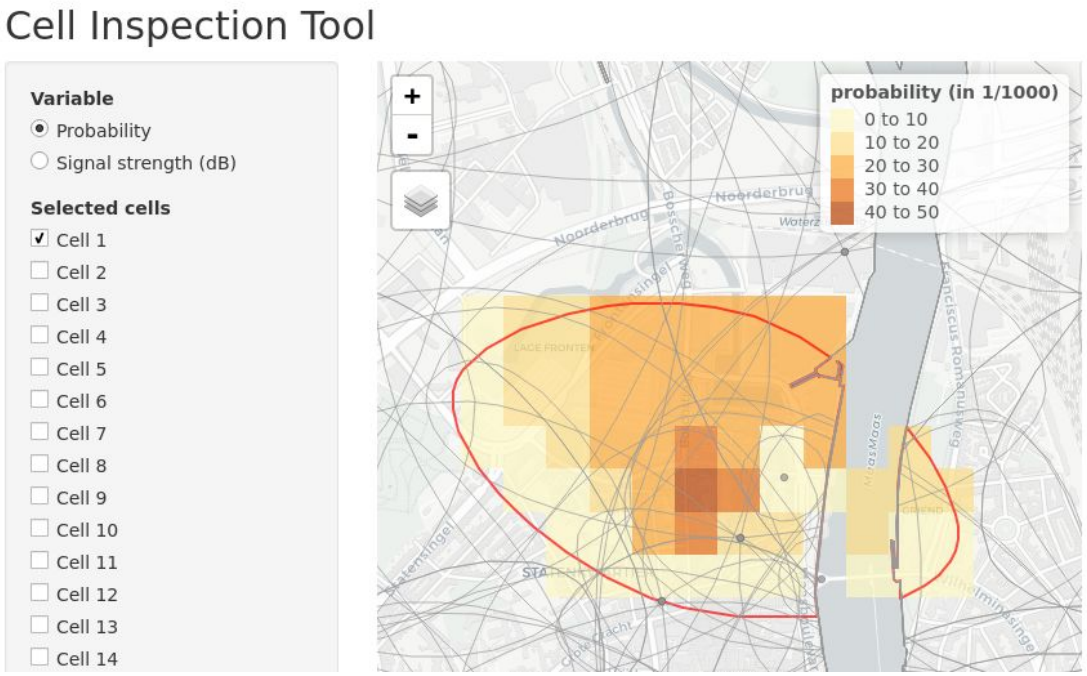

Figure 3.5 Cell inspection tool.

4

pestim - an R package to estimate population counts

4.1. Introduction

We have developed an R package called `pestim` (Salgado et al. (2018)) to implement the hierarchical model to estimate the population counts of different territorial cells combining the information from aggregated mobile phone data and official data (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` package follows the ecological sampling techniques to estimate population counts (see e.g. Manly and Navarro-Alberto (2014) and Royle and Dorazio (2014)). The complete methodology is described in WP5.3 (2018) and it follows a Bayesian approach to estimate population counts.

In a nutshell, 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, $t_1, t_2, ...$ transition probabilities from cell to cell are inferred from mobile phone data and then used to estimate the spatial and time evolution of the population.

Accordingly we will illustrate how this two-step process has been implemented in the package, which 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`.

The 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", buildVignettes = TRUE)
```

Since it contains `C++` code, the user needs to install `Rtools` under Windows environment 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:

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

These binary packages can be downloaded and installed with the standard command

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

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

The functions included in `pestim` package are computationally intensive and we recommend to be installed on a high performance workstation.

The documentation of the `pestim` package is available as:

- a package vignette.
- a reference Manual, available at `https://github.com/MobilePhoneESSnetBigData/pestim/blob/master/doc/pestim_Reference_Manual.pdf`.
- usual R help available for each function included in this package.
- a presentation of the package available at `https://github.com/MobilePhoneESSnetBigData/Estimation_Population/blob/master/pestim-presentation.pdf`.

4.2. Estimating the population at the initial time period

In this section we will give a short description of the theoretical model underlying the `pestim` R package for the 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`.

For the population count estimation at the initial time period, 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. Notice that both pieces of information are considered as data inputs into the estimation process here. Both counts refer to nonoverlapping territorial cells.

The package `pestim` implements a function to estimate the actual population counts $\mathbf{N} = (N_1, \ldots, N_I)^T$ combining both data sources by computing the posterior probability distribution $\mathbb{P}\left(\mathbf{N}|\mathbf{N}^{\text{MNO}}; \mathbf{N}^{\text{Reg}}\right)$ and finding the corresponding posterior mean, median, and mode as possible estimates to be chosen by the user. Notice that this posterior probability distribution can also be used to assess the uncertainty in the output estimates (this will be dealt with in the deliverable 5.5 on quality). This process can be represented schematically as shown in figure 4.1.

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 $\text{Bin}(N_i, p_i)$ whose parameter N_i is the target of the model (being assigned a weakly informative prior) and the detection probability p_i (also assigned a weakly informative prior based upon both data sources).

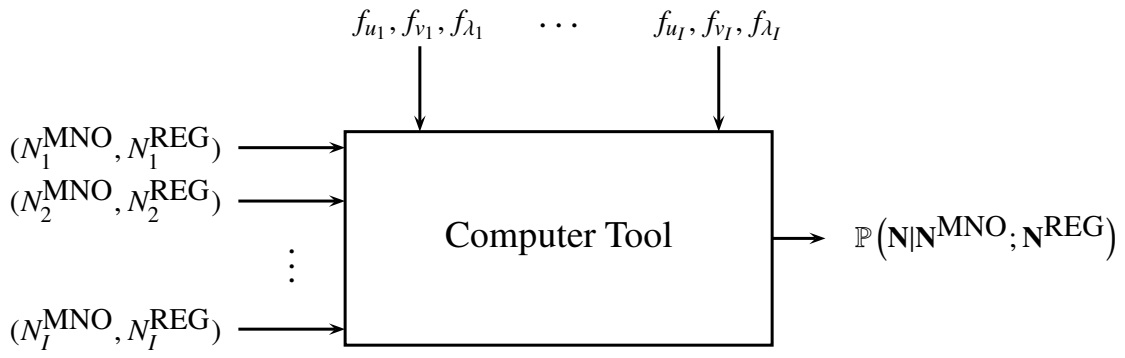

Figure 4.1 The diagram of the process for population estimation using mobile phone and official population data at the initial time period t_0.

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 network naturally following a binomial distribution.

The parameters N_i can be modeled as Poisson random variables with independent parameters λ_i, variables which are pairwise independent. In turn, the detection probabilities p_i are modeled as Beta-distributed independent random variables with parameters α_i and β_i 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 will depend on preceding phases of the process during the aggregation phase of microdata).

Now, the prior distribution of the hyperparameters 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 $u_i \equiv \frac{\alpha_i}{\alpha_i+\beta_i} \simeq f_u(\frac{\alpha_i}{\alpha_i+\beta_i}; \mathbf{N}^{\text{Reg}}, \mathbf{z})$ and $v_i \equiv \alpha_i + \beta_i \simeq f_v(\alpha_i + \beta_i; \mathbf{N}^{\text{Reg}}, \mathbf{z})$. Here f_u and f_v are weakly informative prior distributions for $u = \frac{\alpha}{\alpha+\beta}$ and $v = \alpha + \beta$ in each cell. They make use of the information from the population register ($\mathbf{N}^{\text{Reg}}$) and any other potential auxiliary information $\mathbf{z}$. The parameters $v_i = \alpha_i + \beta_i$ can be understood as the population size of each cell i (thus with support in $(0, \infty)$) upon which the detection is executed at that time instant and thus $u_i = \alpha_i/(\alpha_i + \beta_i)$ can be understood as an a priori proportion of individuals detected by the MNO in cell i.

With no prior information about the detection probability we may safely assume

$f_u = \text{Unif}[0,1]$. In turn, if we assume f_v 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.

The hyperparameters λ_i are modeled with another weakly information prior f_λ which may incorporate the information we have from the population register or similar sources. In our R package, the unnormalised posterior density function of λ (equation (A.3)) is implemented in the function `dlambda`. As explained in WP5.3 (2018), the accept-reject sampling method allows us to sample values from this unnormalized distribution. This has been implemented in the function `rlambda`. A short summary of the mathematical model used to compute estimates of the target population at initial time instant is given in Appendix A .

To generate random values N according to $\mathbb{P}\left(N|N^{\text{MNO}}; N^{\text{Reg}}\right)$ we generate values λ and then the corresponding values N according to the Poisson distribution with parameter λ. This has been implemented in the function `rN0`.

Once posterior populations have been generated the user can choose the mean, the median, the mode, or any other position indicator upon this distribution to provide a point estimate for the population count of each cell. In the package the three first choices (mean, median, mode) have been been implemented in the function `postN0`.

4.3. Examples

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

4.3.1. Prior distribution of the hyperparameters

Before illustrating the use of the package to estimate population counts, let us introduce how to manage and specify prior distributions. The package `pestim` implements functions for the following prior distributions:

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

The uniform distribution is well known and functions implementing the probability density function, distribution function, quantile function, and random variable genera-

tion are provided by the standard R distribution (dunif, punif, qunif, runif).

The triangular distribution can be used for modelling the a priori proportion of detected individuals u, the a priori cell size v, and the parameter λ. The corresponding functions for the density, distribution function, quantile function, and random variable generation are ptriang, dtriang, qtriang and rtriang, respectively. Below is an example of how to use the triangular distribution function (see figure 4.2).

```
# Load the libraries
library(ggplot2)
library(pestim)

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

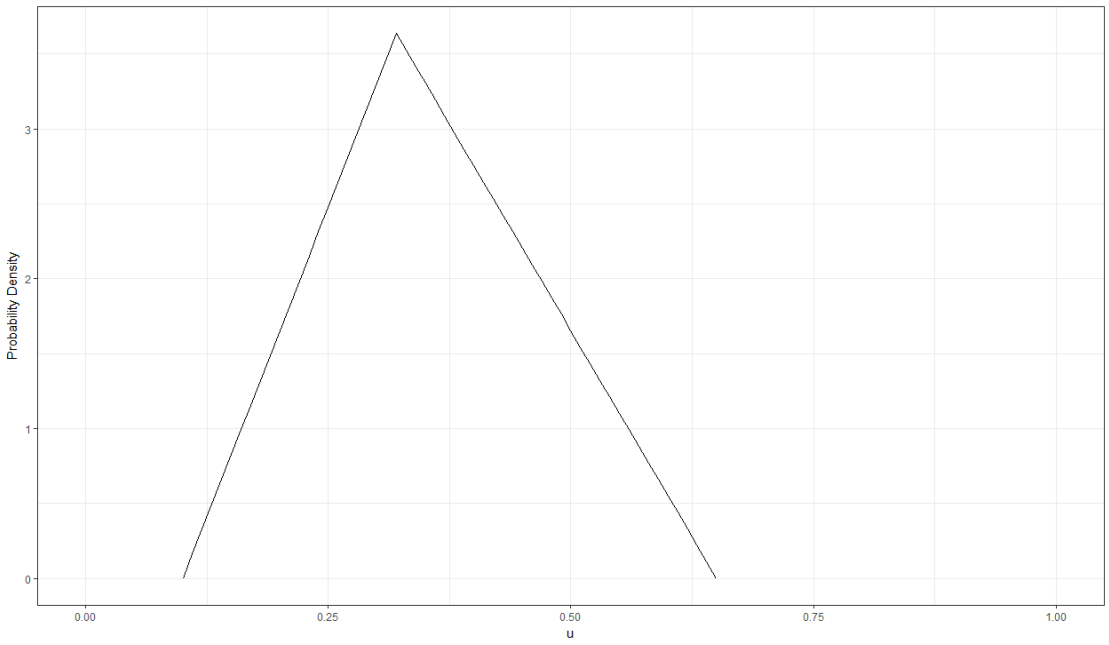

Figure 4.2 The triangular distribution.

The gamma distribution is another choice for modelling both the cell size v and the parameter λ. Functions for the probability density, the accumulative distribution, the quantile, and the random generation of values are included in the standard R

distribution (`dgamma`, `pgamma`, `qgamma`, `rgamma`). We can assume a parametrisation 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 4.3). In the following example we use 5 values for α (1, 5, 10, 100, and 1000). The following code shows how to use this 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), size = 1.1) +
          scale_x_continuous(limits = c(0, 100)) + xlab('') + ylab('') +
          theme_bw()
```

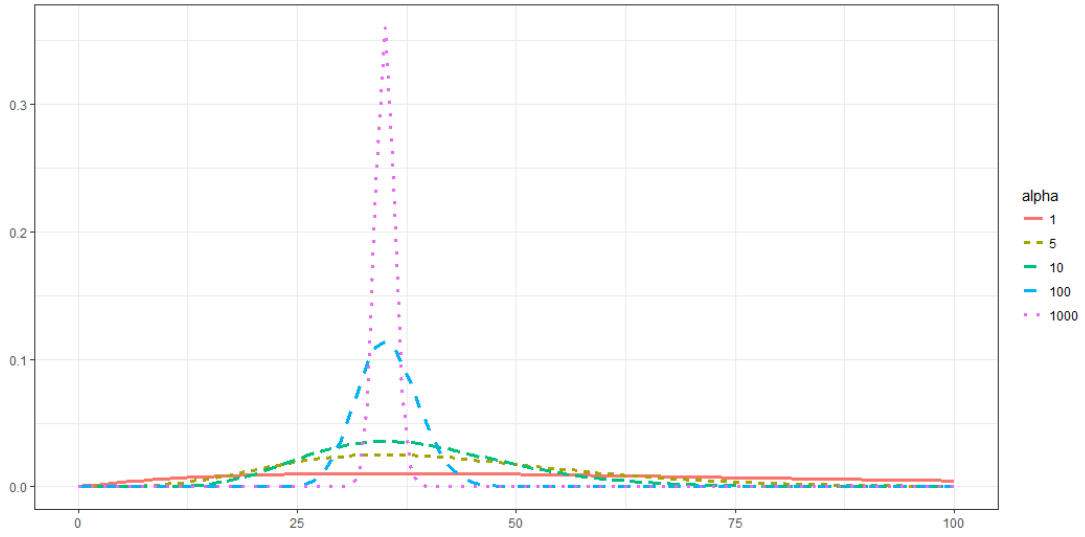

Figure 4.3 The gamma distribution with different degrees of concentration.

4.3.2. Estimation for a single cell

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 populations counts;
4. compute some statistics (mean, median, mode) of the estimated population counts;
5. visualize the results.

For the initial time period t_0 we shall assume (first working assumption) that there is a high correlation between the actual size and official population data. This assumption is supported by some preliminary studies with real mobile phone data (Meersman et al., 2016; Doyle et al., 2014). In our examples, we will use a simulated data set such that

- given the actual true value N_i^0 we will simulate a population register value $N_i^{\text{Reg}} \simeq \lfloor N(\mu = N_i^0, \sigma = 0.1 \cdot N_i^0) \rfloor$;
- 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 WP5.1 (2016) 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 $f_\lambda \simeq \text{Gamma}\left(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha}\right)$ for the parameter λ. Let us consider a true population size of $N^{(0)} = 100$ and an administrative population size given by $N^{\text{Reg}} = 97$ assuming an error of 3%. Let us also consider the number of individuals detected by the mobile network as $N^{\text{MNO}} = 19$ assuming a proportion of detected individuals of around 20%.

In this first example, 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.

A simple example with these parameters and $\alpha = 1$ (extremely weakly prior for λ) can be easily expressed in code:

```
# Load the libraries
library(pestim)
library(data.table)
# Set the input data
nReg <- 97
```

```
nMNO <- 19
# Set the priors
fu <- list('unif', xMin = 0, xMax = 0.50)
fv <- list('triang', xMin = 87, xMax = 107, xMode = 97)
alpha <- 1
flambda <- list('gamma', shape = 1 + alpha, scale = nReg / alpha)
#Compute the estimates accepting default values for other parameters
postN0(nMNO, nReg, fu, fv, flambda)
```

The result is:

```
  postMean postMedian   postMode
       110        109        128
```

The discrepancy between these estimates and the value N^{Reg} is explained right now in the following example.

As a second illustrative example, let us compute the estimates for values of $\alpha = 1, 10, 100, 1000$ and observe the effect of the amount of uncertainty in the prior for λ (see figure 4.4). Each estimate is computed 100 times for each value of α:

```
# Load the libraries
library(pestim)
library(data.table)
# Set the input data
nReg <- 97
nMNO <- 19
# Set the priors and compute the estimates for
# each replication and each value of alpha
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]
    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) +
```

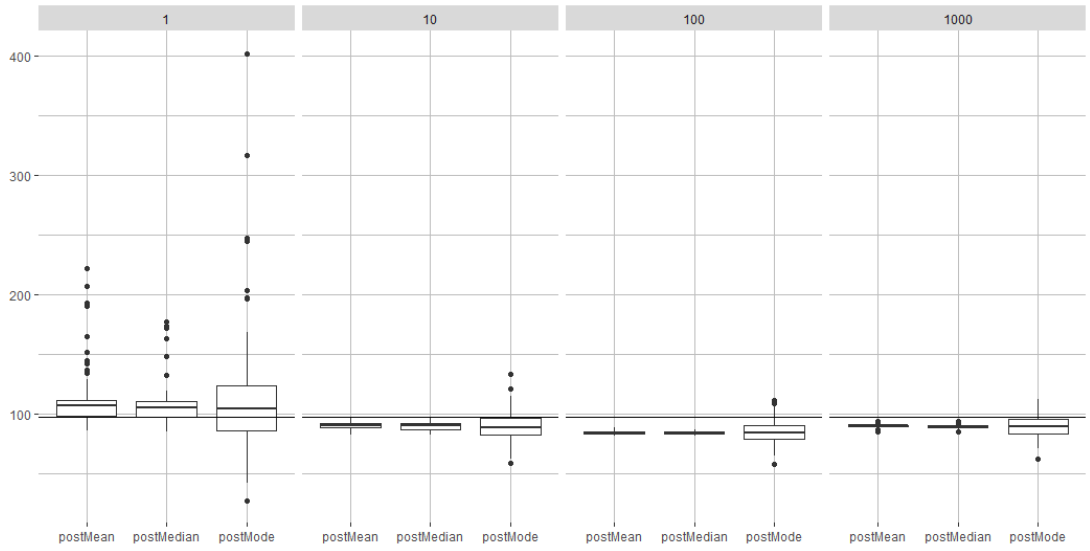

Figure 4.4 Distribution of estimated population counts for diverse degrees of prior uncertainty in the value of λ.

```
theme(axis.title.x = element_text(hjust = 1, vjust = .5),
      panel.background = element_blank(),
      panel.grid.major = element_line(color = 'grey', size = 0.2),
      panel.grid.minor = element_line(color = 'grey', size = 0.1))
```

The results appear in figure 4.4. We can observe how the more precise the prior value of λ around N^{Reg} is, the more precise the final estimate around this same value will result.

This prior uncertainty can also be expressed in terms of the prior coefficient of variation of λ. Since $\text{cv}(\lambda) = \frac{1}{\sqrt{1+\alpha}}$, we have for $\alpha = (1, 10, 100, 1000)$ the coefficients of variation given by $0.71, 0.30, 0.10, 0.03$, respectively. Notice how the final estimates inherit the original difference between $N^{(0)}$ and N^{Reg}, as expected.

Finally the posterior mode clearly appears as the worst estimator in comparison with the posterior mean and the posterior median. This will be observed in other examples.

In the appendix A, the reader can consult these computations for a variety of choices for the priors both in form and in parameters.

4.3.3. 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 high probability 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 now 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 a distribution 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 $\frac{\alpha_i+1}{\alpha_i^2} \cdot N_i^{\text{Reg}} = \frac{1}{N_{cells}-1} \sum_{i=1}^{N_{cell}} \left(N_i^{\text{Reg}} - \bar{N}^{\text{Reg}}\right)^2$ to obtain α_i and then propose a value (upper bound) for α as $\alpha \leq \min_i \alpha_i$.

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 piece of code below computes and displays the distribution of the relative bias

with respect to the administrative population (in percentage) $\frac{\hat{N}_i - N_i^{\text{Reg}}}{N_i^{\text{Reg}}} \cdot 100$ 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 the initial population. The population given from the administrative register for each cell is generated using a Gaussian distribution with mean 71 and standard deviation 3 while the population detected by the MNO is again generated using a Gaussian distribution with mean 19 and standard deviation 2. These values can be replaced with any real data.

The values for the posterior distribution of the population at the initial time instant is again obtained by a call to the function `postN0` for each set of parameters:

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

# Set the number of cells and the input data
nCell <- 50
nReg <- round(rnorm(nCell, 71, 3))
nMNO <- round(rnorm(nCell, 19, 2))

# Set the priors and compute the estimates and
# relative bias for each set of parameters and each cell
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){
  print(paste0('i=', i))
  for (radShare.index in seq(along = radShares[[i]])) {
    print(paste0('radShare.index=', radShare.index))
    for (radPopSize.index in seq(along = radPopSizes[[i]])) {
      print(paste0('radPopSize.index=', radPopSize.index))
      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)
}

# Draw the results
ggplot(relBias.Mean.df, aes(x = '␣', y = N)) +
  geom_boxplot() +
  facet_grid(factor(u) ~ factor(v)) +
  xlab('Prior␣parameters␣(u,␣v)') + ylab('Posterior␣Mean␣Estimate␣Relative␣Bias␣(%)\n') +
  #ggtitle(paste0('Relative bias (%) distributions of the ', nCell, ' cells\n')) +
  theme(axis.title.x = element_text(hjust = 0.5, vjust = .5),
        panel.background = element_blank(),
```

```
        panel.grid.major = element_line(color = 'grey', size = 0.2),
        panel.grid.minor = element_line(color = 'grey', size = 0.05))

ggplot(relBias.Median.df, aes(x = '␣', y = N)) +
  geom_boxplot() +
  facet_grid(factor(u) ~ factor(v)) +
  xlab('Prior␣parameters␣(u,␣v)') + ylab('Posterior␣Median␣Estimate␣Relative␣Bias␣(%)\n') +
  #ggtitle(paste0('Relative bias (%) distributions of the ', nCell, ' cells\n')) +
  theme(axis.title.x = element_text(hjust = 0.5, vjust = .5),
        panel.background = element_blank(),
        panel.grid.major = element_line(color = 'grey', size = 0.2),
        panel.grid.minor = element_line(color = 'grey', size = 0.05))

ggplot(relBias.Mode.df, aes(x = '␣', y = N)) +
  geom_boxplot() +
  facet_grid(factor(u) ~ factor(v)) +
  xlab('Prior␣parameters␣(u,␣v)') + ylab('Posterior␣Mode␣Estimate␣Relative␣Bias␣(%)\n') +
  #ggtitle(paste0('Relative bias (%) distributions of the ', nCell, ' cells\n')) +
  theme(axis.title.x = element_text(hjust = 0.5, vjust = .5),
        panel.background = element_blank(),
        panel.grid.major = element_line(color = 'grey', size = 0.2),
        panel.grid.minor = element_line(color = 'grey', size = 0.05))
```

Graphically, the results are represented in figures 4.5, 4.6, and 4.7. We observe how the overall precision (for the whole set of 50 cells) gets better as the intervals get shorter (within the precision limits achieved with the prior for λ).

The same combinations for distributions used for f_u and f_v can be used in the case of several cells as in the case of a single cell. Since this process is straightforward we will not show here each combination. Users can make choices of their own.

4.4. Estimates along a sequence of time periods

In this section we show how to extend the previous estimation process along a sequence of time instants. Figure 4.8 presents schematically this process.

As input data for the final inference stage we used the number of individuals $N_{ij}^{\text{MNO}}(t_0, t_n)$ moving from territorial cell i to cell j in the time interval (t_0, t_n) according to the network. 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:

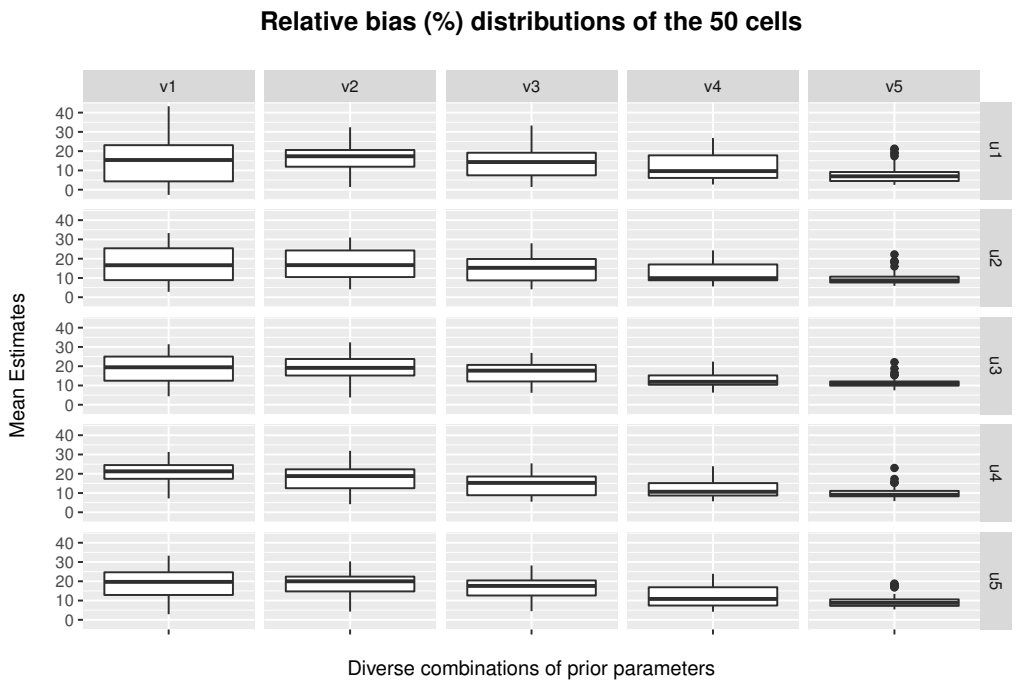

Figure 4.5 The distribution of the relative bias for the posterior mean for 50 cells.

1. As in preceding sections, to combine both aggregated mobile phone and official data we assume that at a given time instant t_0 both population figures in each territorial cell can be assimilated to some extent. Again, in the model we are taking $N_i^{\text{Reg}}(t_0)$ as fixed quantities without prior distributions (representing uncertainty in its knowledge).Therefore $N_i^{\text{Reg}}(t_0)$ will be fixed external parameters in the model.

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

Notice that the first hypothesis allows us to infer the actual population from the data at the initial time instant whereas the second hypothesis will now be used to reconstruct the time evolution of the actual population using only the mobile phone data. The hierarchical model used to estimate the target population in this case is shortly presented in A and a detailed presentation can be found in WP5.3 (2018).

The computation of the probability functions $\mathbb{P}\left(N_i(t_n)\middle|N^{\text{MNO}}(t_0,t_1)\right)$ for each cell i will allow us to choose as point estimator the posterior mean, posterior median, posterior

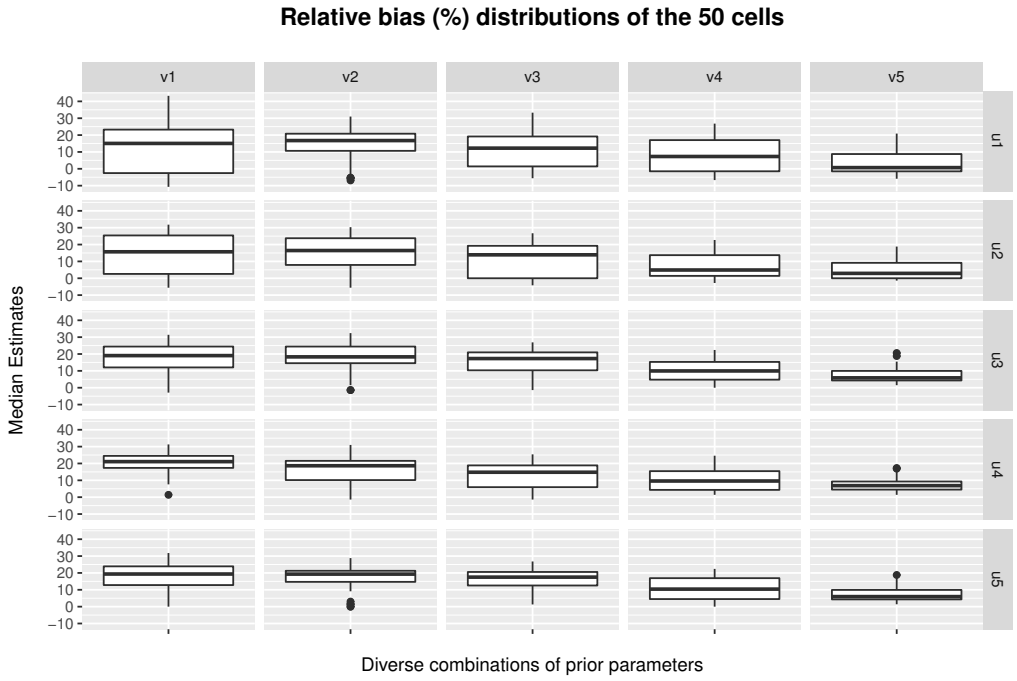

Figure 4.6 The distribution of the relative bias for the posterior median for 50 cells.

mode or any similar posterior indicator. 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 $N_{i\cdot}^{\text{MNO}}(t_0)$ 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 $N^{\text{MNO}}(t_0, t_n)$ as input data and choosing weakly informative priors $f_{\alpha ij}$.

3. These generated quantities are used in formula (A.6a) 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, mode, or any other distribution position statistics.

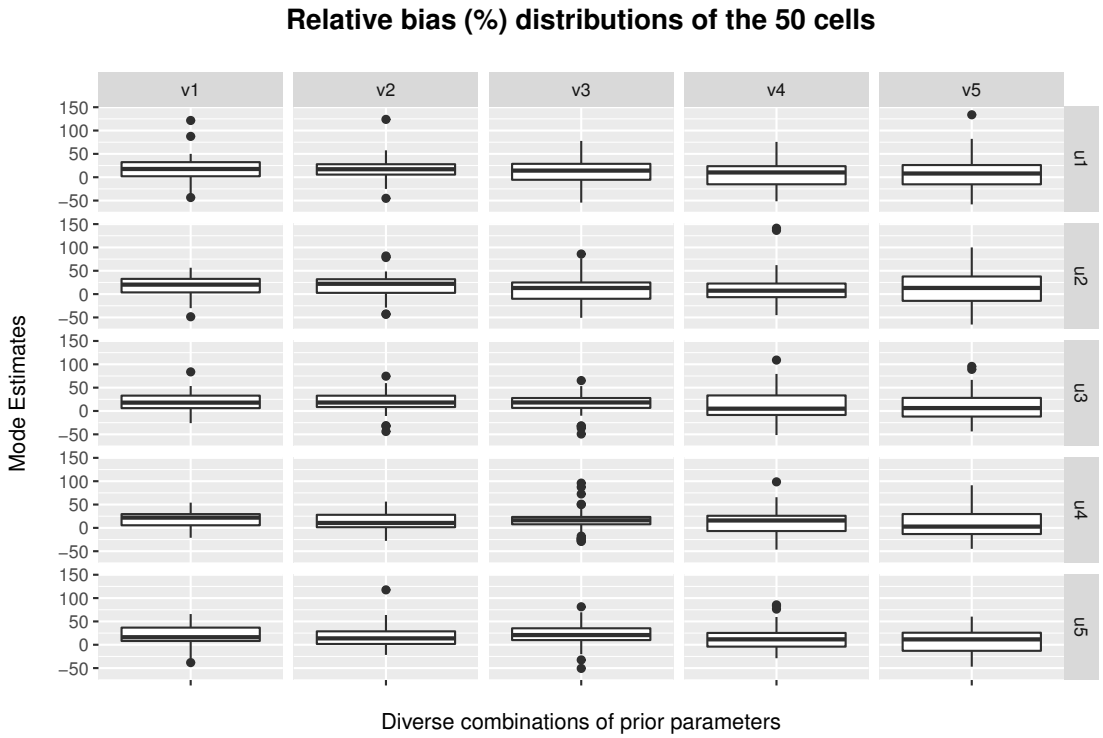

Figure 4.7 The distribution of the relative bias for the posterior mode for 50 cells.

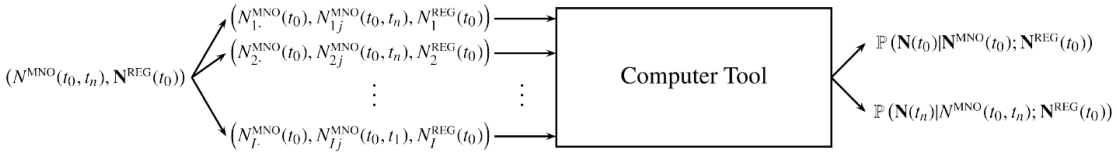

Figure 4.8 A diagram of the estimation process for the target population using mobile phone and official population data for a sequence of time instants.

4.5. Examples

Let us consider an extremely simple example of time evolution between an initial time period t_0 and a final time period t_1. We will consider 12 cells. The input data thus comprise the transition matrix of individuals from cell to cell in the time interval (t_0, t_1) according to the network together with the number of individuals according to the population register $N_i^{\mathrm{Reg}}(t_0)$ at each cell i at the initial time period t_0. In this sense, the package `pestim` contains a dataset called `MobPop` which provides population

counts moving from each pair of cells at each time interval (t_0, t_n) 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.

Additionally, we need to choose priors for u_i, v_i, λ_i, and α_{ij}, according to the hierarchical model:

- For u_i we will choose uniform distributions with interval ranges centered at $N_{i\cdot}^{\text{MNO}}/N_i^{\text{Reg}}(t_0)$ and moderately large radii (up to 10% of the centre values).
- For v_i we will also choose uniform distributions with interval ranges centered at $N_i^{\text{Reg}}(t_0)$ and moderately large radii (up to 10% of the centre values).
- For λ_i we will choose gamma distributions with shape parameters $\alpha_i + 1$ and scale parameters $N_i^{\text{Reg}}/\alpha_i$, with $\alpha_i = \frac{1}{0.1^2} - 1$ corresponding to coefficients of variations of 10%, too.
- For α_{ij} we will choose uniform distributions with interval ranges centered at $\frac{N_{ij}^{\text{MNO}}(t_0,t_1)}{N_{i\cdot}^{\text{MNO}}(t_0,t_1)}$ and coefficients of variation of 10% for all parameters.

The computation of the evolved population in the initial time interval (t_0, t_1) can thus be implemented with the following code:

```
# Load the libraries
library(pestim)
library(data.table)

# Set input data
data(MobPop)
Data <- MobPop[ID_T == 0]
Scale <- 1e3
NMNOmat <- dcast(Data, ID_CELL_INI ~ ID_CELL_END, value.var = 'N_MNO_1')
NMNOmat <- as.matrix(NMNOmat[, as.character(1:12), with = FALSE]) / Scale
InitialPop <- Data[ID_CELL_END == ID_CELL_INI]
nMNO_ini <- InitialPop[['N_MNO_1']] / Scale
nReg <- InitialPop[['N_REG']] / Scale
# Set priors
u0 <- nMNO_ini / nReg
fu <- lapply(u0, function(u){
    umin <- max(0, u - 0.10 * u)
    umax <- min(1, u + 0.10 * u)
    output <- list('unif', xMin = umin, xMax = umax)
    return(output)
})
v0 <- nReg
fv <- lapply(v0, function(u){
    umin <- max(0, u - 0.10 * u)
    umax <- u + 0.10 * u
    output <- list('unif', xMin = umin, xMax = umax)
    return(output)
})
alpha <- 1 / 0.1**2 - 1 # cv(lambda) = 0.1
flambda <- lapply(v0, function(v){list('gamma', shape = 1 + alpha,
                  scale = v / alpha)})

DistributNames <- rep('unif', 12)
Variation <- rep(list(list(cv = 0.10)), 12)
# Compute estimates and relative biases with respect to true population
Nt.mat <- postNt(NMNOmat, nReg, fu, fv, flambda, DistributNames, Variation, Scale)
N0_t1 <- Data[, sum(N_0), by = 'ID_CELL_END']$V1
relBias <- round((Nt.mat - N0_t1) / N0_t1 * 100, 2)
```

The resulting relative bias `relBias` for each cell $i = 1, \ldots, 12$ (in percentage) is given by:

```
  postMean postMedian   postMode
      15.2       -6.4        7.8
      10.4       -2.7      -43.9
       5.4        4.8       22.6
       2.3        1.2      -24.4
       8.4       -4.8      230.9
      -1.7       -1.8        1.3
       1.7        1.4       20.6
       7.9        2.5       22.5
       9.6      -12.5      -43.8
      12.9        7.6       29.8
       9.2       -3.0      -10.3
       5.7      -13.2      -20.9
```

Notice again how the posterior mean and median outperforms the posterior mode. For more time periods we can trivially repeat the same procedure as above. It is only a matter of computation time. In figure 4.9 we can find the same computation as above extended to the whole time span for the data set `MobPop`.

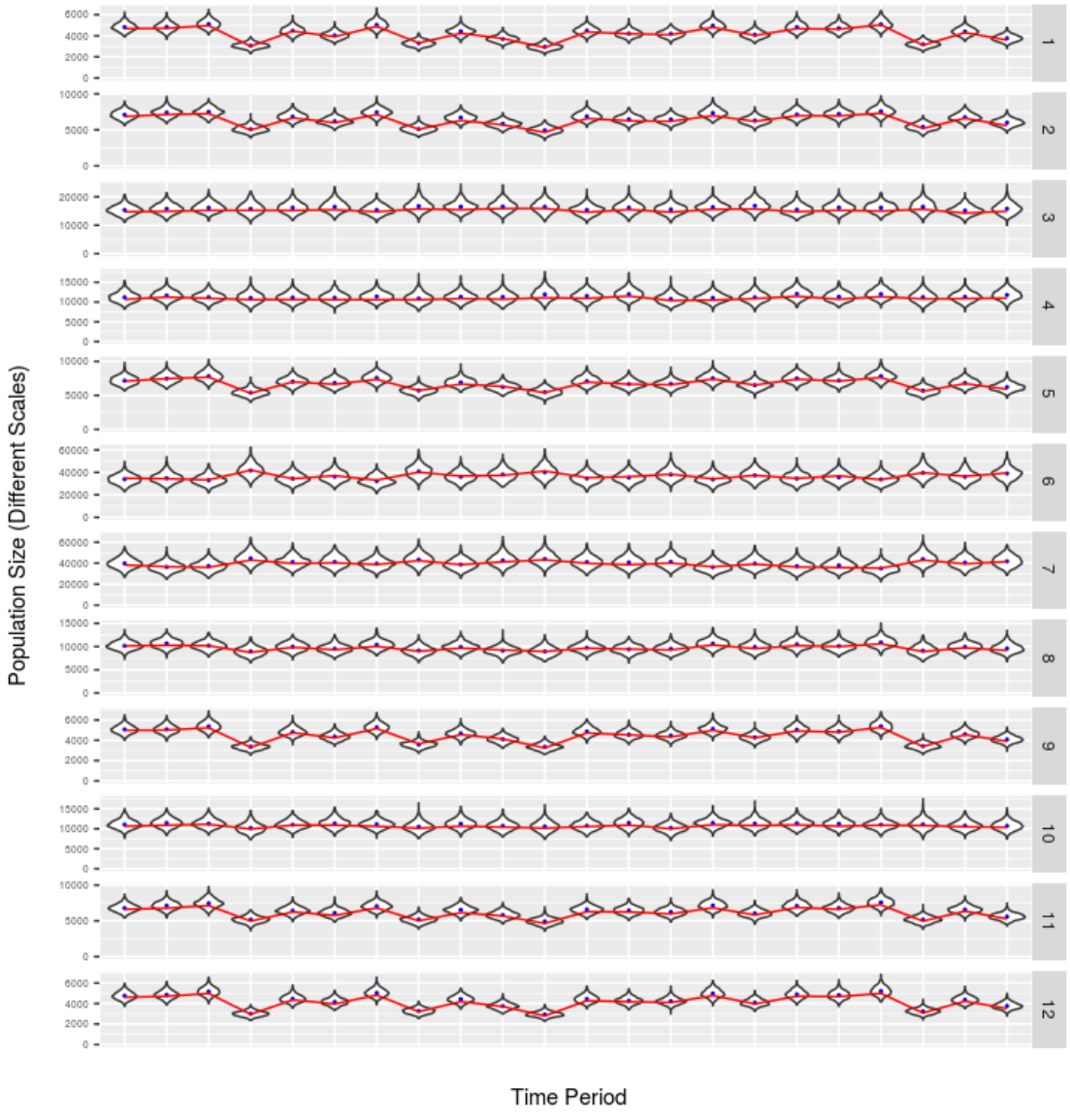

Figure 4.9 Time evolution of the population count estimates for 672 consecutive time periods.

4.6. Further developments

The `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 different levels:

- each function will be profiled and then improved from the point of view of running time and memory requirements. Nevertheless, a balance will be kept between the readability of the code and its performance;
- we will further extend the parallelization of the code (parallel computations are already used for the $kummer$ function and $rN0$). We will also show how parallelization can be introduced at the level of the user code (like the examples presented in this document);
- mathematical elements as the optimization algorithm to compute the mode of the posterior distribution of the parameter λ, the candidate distribution for the accept-reject method, ... can be further improved to gain performance;
- unit tests will be added in order to maintain the quality of the software throughout its life time;

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 successive periods.

Appendix A

Implementation details and examples of combinations of priors for pestim

A.1. The mathematical model used to estimate the target population at initial time

The model to estimate population counts at the initial time instant can be summarized as follows:

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

The quantity of interest here is the target population counts $\mathbf{N} = (N_1, \ldots, N_I)^T$ in each cell i. We followed a Bayesian approach to compute the posterior distribution of the target population. This approach allowed us to account for the inference and the assessment of the uncertainty, hence of the quality of estimations (to be dealt with in deliverable 5.5). We can leverage the prior information we have at our disposal by choosing the probability distribution f_u, f_v and f_λ. The posterior distribution $\mathbb{P}\left(\mathbf{N} \middle| \mathbf{N}^{\text{MNO}}; \mathbf{N}^{Reg}\right)$ is given by (we dropped the subscripts since each cell is treated independently of each other – see WP5.3 (2018) for details):

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

Since N is 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 $\mathbb{P}\left(\lambda \middle| \mathbf{N}^{\text{MNO}}; \mathbf{N}^{\mathbf{Reg}}\right)$ is given by WP5.3 (2018):

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

where we have defined

$$S(\lambda, N^{\text{MNO}}, N^{\text{Reg}}) = \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} I_{N^{\text{MNO}},n}(N^{\text{Reg}}), \tag{A.4}$$

$$I_{N^{\text{MNO}},n}(N^{\text{Reg}}) = \int_0^{\infty} \int_0^{\infty} d\alpha d\beta \, \frac{f_u(\frac{\alpha}{\alpha+\beta}; N^{\text{Reg}}) \cdot f_v(\alpha+\beta; N^{\text{Reg}}}{\alpha+\beta} \frac{B\left(\alpha + N^{\text{MNO}}, \beta + n - N^{\text{MNO}}\right)}{B\left(\alpha, \beta\right)}. \tag{A.5}$$

The mathematical details of the computation of these quantities can be consulted in the appendix of WP5.3 (2018).

A.2. The mathematical model used to estimate the target population for a sequence of time instants

The following hierarchical model was used to estimate the target population at successive time periods. 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 $N_{ij}^{\text{MNO}}(t_0, t_n)$ be the number of individuals moving from cell i to cell j according to the network. We denote $N_{i\cdot}^{\text{MNO}}(t_0) = \sum_{j=1}^{I} N_{ij}^{\text{MNO}}(t_0, t_n)$.

$$N_i(t_n) = \left[N_i(t_0) + \sum_{\substack{j=1 \\ j \neq i}}^{I} p_{ji}(t_0, t_n) N_j(t_0) - \sum_{\substack{j=1 \\ j \neq i}}^{I} p_{ij}(t_0, t_n) N_i(t_0) \right], \quad i = 1, \ldots, I \tag{A.6a}$$

$$\mathbf{p}_{i\cdot}(t_0, t_n) \simeq \text{Dir}\left(\alpha_{i1}(t_0, t_n), \ldots, \alpha_{iI}(t_0, t_n)\right), \quad \mathbf{p}_{i\cdot}(t_0, t_n) \perp \mathbf{p}_{j\cdot}(t_0, t_n), \quad i \neq j = 1, \ldots, I \tag{A.6b}$$

$$\alpha_{ij}(t_0, t_n) \simeq f_{\alpha ij}\left(\alpha_{ij}; \frac{N_{ij}^{\text{MNO}}(t_0, t_n)}{N_{i\cdot}^{\text{MNO}}(t_0)}\right), \quad i = 1, \ldots, I \tag{A.6c}$$

$$N_i^{\text{MNO}}(t_0) \simeq \text{Bin}\left(N_i(t_0), p_i(t_0)\right), \qquad N_i^{\text{MNO}}(t_0) \perp N_j^{\text{MNO}}(t_0), \quad i \neq j = 1, \ldots, I \tag{A.6d}$$

A.2 The mathematical model used to estimate the target population for a sequence of time instants

$$N_i(t_0) \simeq \text{Po}\left(\lambda_i(t_0)\right), \qquad N_i(t_0) \perp N_j(t_0), \quad i \neq j = 1, \ldots, I \tag{A.6e}$$

$$p_i(t_0) \simeq \text{Beta}\left(\alpha_i(t_0), \beta_i(t_0)\right), \qquad p_i(t_0) \perp p_j(t_0) \quad i \neq j = 1, \ldots, I \tag{A.6f}$$

$$(\alpha_i(t_0), \beta_i(t_0)) \simeq \frac{f_{ui}\left(\frac{\alpha_i}{\alpha_i+\beta_i}; \frac{N_i^{\text{MNO}}(t_0)}{N_i^{\text{Reg}}(t_0)}\right) \cdot f_{vi}\left(\alpha_i + \beta_i; N_i^{\text{Reg}}(t_0)\right)}{\alpha_i + \beta_i},$$

$$(\alpha_i(t_0), \beta_i(t_0)) \perp (\alpha_j(t_0), \beta_j(t_0)), \quad i \neq j = 1, \ldots, I \tag{A.6g}$$

$$\lambda_i(t_0) \simeq f_{\lambda i}(\lambda_i; N_i^{\text{Reg}}(t_0)) \quad (\lambda_i(t_0) > 0, \quad \lambda_i(t_0) \perp \lambda_j(t_0)), \quad i = 1, \ldots, I, \tag{A.6h}$$

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} is 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 (A.6d) to (A.6h) are indeed the same model described in section 4.2. We have explicitly added the time dependence. Equations (A.6a), (A.6b), and (A.6c) take care of the time evolution of the estimates.

Their meaning is straightforward. Equation (A.6a) 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 (see equation (A.6b)) 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 $\frac{N_{ij}^{\text{MNO}}}{N_{i\cdot}^{\text{MNO}}}$ (see equation (A.6c)) according to our second working assumption.

A.3. Technical comments on the functions

The actual computation of the population count estimates at the initial time instant is done by the function `postN0`, which takes the following input 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 population 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 confluent hypergeometric (Kummer) function. Default value is 1e-6;
- `nSim` - number of two-dimensional points to generate to compute the integral with Monte Carlo simulations. Default value is 1e4;
- `nStrata` - integer vector of length 2 with the number of strata in each dimension. Default values are 1 and 1e2, respectively;
- `nThreads` - the number of threads to be used for computing the value of the confluent hypergeometric function.

In the examples above we used the default values for the parameters `n, scale, relTol, nSim, nStrata, nThreads`.

The posterior distribution used to generate random numbers for the actual population counts $N_i(t_0)$ is a Poisson distribution (see the second row from model (A.1)). Thus, for the initial time instant t_0, the function `postN0` generates n random values of the posterior distribution of $N_i(t_0)$ by internally calling the function `rN0`. In turn, `rN0` executes internally `rlambda` to generate the corresponding values for λ_i.

`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. The function first computes the mode for the posterior distribution of λ using the function `modeLambda` and then applies the accept-rejection method.

The function `modeLambda` computes the mode of the posterior density function of the parameter λ in the hierarchical model. This unnormalized posterior density function is implemented in our package by the function `dlambda`, computed according to

$$f(\lambda \big| N^{\text{MNO}}; N^{\text{Nreg}}) \propto f(\lambda) \cdot \text{dpois}(N^{\text{MNO}}; \lambda) \cdot S(\lambda; N^{\text{MNO}}, N^{\text{Nreg}}),$$

where `dpois` is the probability density function of a Poisson distribution (implemented in the standard `R` distribution) and S is defined in equations (A.3) and (A.4).

$S(\cdot;\cdot;\cdot)$ is computed using the Monte Carlo method described in the appendix of the deliverable WP5.3 (2018). The points needed by this method are generated using the functions `genUV`, which makes use of the stratified importance sampling technique. These points are then used as inputs for the function `Phi`.

The function `Phi` multiplies a ratio of two Beta functions computed by `ratioBeta` function and the confluent hypergeometric function ${}_1F_1$, which is given by a call to the function `kummer`.

The function `ratioBeta` computes the ratio of 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.

The function `kummer` was implemented in C++ and called using the `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 the `pestim` package, we provide here few details about the implementation that we used in the current version of the package. The confluent hypergeometric function is defined as:

$$\mathbf{M}(z;a;b) = \sum_{j=0}^{\infty} \frac{(a)_j}{\Gamma(b+j)} \times \frac{z^j}{j!} \tag{A.7}$$

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

$$a_0 = 1, \quad (a)_j = a \times (a+1) \times \cdots (a+j-1), \qquad j = 1, 2, \cdots \tag{A.8}$$

The sum in equation A.7 always converge, function $\mathbf{M}$ being analytic throughout the complex plane $\mathbb{C}$. Next, we define:

$${}_1F_1(z;a;b) = \Gamma(b)\mathbf{M}(z;a;b) = \sum_{j-0}^{\infty} \frac{(a)_j}{(b)_j} \times \frac{z^j}{j!} \tag{A.9}$$

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

Although the work by PeaOlvPor (2017) recommend to divide the computation of ${}_1F_1(z;a;b)$ according to the value of z using two different approaches, one for $z < 80$ where the best method recommended is a Taylor series expansion, and another one for $z \geq 80$ where a computation procedure based on Watson's lemma (Watson, 1918) is recommended, our numerical experiments showed that the Taylor series approach is the method with the best results in terms of reliability for the numerical regime of the inputs in our case. Thus, we computed the confluent hypergeometric function as:

$${}_1F_1(z;a;b) \approx S_N = \sum_{j=0}^{N} \frac{(a)_j}{(b)_j} \frac{z^j}{j!} = \sum_{j=0}^{N} A_j \tag{A.10}$$

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

$$A_0 = 1 \tag{A.11}$$

$$S_0 = A_0 \tag{A.12}$$

$$A_{j+1} = A_j \frac{a+j}{b+j} \frac{z}{j+1} \tag{A.13}$$

$$S_{j+1} = S_j + A_{j+1} \quad j = 0, 1, 2, \cdots \tag{A.14}$$

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.

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. More, since the computation of the confluent hypergeometric function for (z_i, a_i, b_i) is independent form the computation for z_j, a_j, b_j we parallelized the computations as follows:

```
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 parallelization of the computation for confluent hypergeometric function was implemented using RcppParallel package.

A.4. $f_u \simeq \textbf{Unif}(u_m, u_M)$, $f_v \simeq \textbf{Unif}(N_m, N_M)$

Let us now illustrate the computation of estimates choosing uniform priors for u and v and investigating the effect of their interval amplitudes. 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. 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 $\alpha = 1$ (coefficient of variation of 71%) 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 with respect to the administrative population (in percentage) $\frac{\hat{N} - N^{\text{Reg}}}{N^{\text{Reg}}} \cdot 100$ 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 do not reproduce here the actual numerical results, only a graphical representation in figure [FIG]). One can note that this code is similar to the previous simple illustrative example, the estimation being computed with a call to the function `postN0`, and only the prior distributions being different in each call.

The code is:

```
# Load the libraries
library(pestim)
library(data.table)

# Set the input data
nReg <- 97
nMNO <- 19

# Set the priors and compute the estimates for each set of parameters
```

```
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)
```

The results are displayed in tables A.1. We do not see a strong effect of the reduction of the intervals (u_m, u_M) and (v_m, v_M) on the final estimates. The main reason is because these intervals are symmetric with respect to the prior modes and the point estimates are central position measures of the posterior distributions.

A.4 $f_u \simeq \text{Unif}(u_m, u_M)$, $f_v \simeq \text{Unif}(N_m, N_M)$

Mean	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	-9.3	47.4	-7.2	40.2	13.4	5.2	39.2	0.0	-13.4	56.7
0.35	9.3	-9.3	11.3	21.6	15.5	17.5	-13.4	-13.4	-10.3	-11.3
0.31	-5.2	59.8	-8.2	0.0	4.1	5.2	8.2	38.1	-14.4	-6.2
0.26	1.0	-2.1	23.7	4.1	13.4	-6.2	5.2	-9.3	61.9	-3.1
0.22	18.6	26.8	13.4	14.4	-7.2	16.5	-13.4	-10.3	6.2	-5.2
0.18	-8.2	12.4	-5.2	6.2	16.5	-11.3	-1.0	-5.2	56.7	-2.1
0.14	-12.4	-5.2	14.4	-9.3	17.5	4.1	-11.3	-1.0	11.3	-5.2
0.09	24.7	22.7	20.6	15.5	0.0	18.6	12.4	11.3	9.3	-6.2
0.05	-6.2	27.8	-2.1	16.5	0.0	8.2	-4.1	-11.3	21.6	-15.5
0.01	13.4	5.2	2.1	17.5	29.9	13.4	-5.2	2.1	7.2	-10.3
Median	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	-9.3	26.8	-7.2	22.7	11.3	4.1	19.6	0.0	-13.4	23.7
0.35	8.2	-10.3	10.3	20.6	13.4	16.5	-14.4	-13.4	-10.3	-11.3
0.31	-6.2	44.3	-8.2	0.0	3.1	4.1	8.2	16.5	-14.4	-6.2
0.26	1.0	-2.1	18.6	4.1	12.4	-7.2	4.1	-9.3	53.6	-3.1
0.22	17.5	24.7	11.3	13.4	-7.2	15.5	-13.4	-10.3	6.2	-5.2
0.18	-8.2	10.3	-5.2	5.2	15.5	-11.3	-1.0	-5.2	54.6	-3.1
0.14	-11.3	-5.2	12.4	-9.3	15.5	3.1	-11.3	-1.0	10.3	-5.2
0.09	22.7	20.6	19.6	13.4	-1.0	11.3	11.3	10.3	7.2	-5.2
0.05	-6.2	25.8	-2.1	16.5	0.0	8.2	-4.1	-11.3	19.6	-16.5
0.01	12.4	5.2	2.1	15.5	26.8	11.3	-5.2	1.0	7.2	-9.3
Mode	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	-16.5	24.7	-7.2	6.2	62.9	29.9	-14.4	10.3	-37.1	72.2
0.35	9.3	-3.1	42.3	-18.6	49.5	66.0	-19.6	-29.9	-25.8	-9.3
0.31	3.1	221.6	-23.7	-8.2	-2.1	8.2	-5.2	-26.8	-41.2	1.0
0.26	-7.2	-10.3	29.9	20.6	6.2	-2.1	12.4	-7.2	-33.0	9.3
0.22	43.3	17.5	-4.1	-6.2	-7.2	-30.9	-26.8	2.1	34.0	15.5
0.18	-7.2	-14.4	-33.0	42.3	-16.5	-2.1	-5.2	4.1	-30.9	14.4
0.14	-22.7	-11.3	62.9	3.1	-1.0	-2.1	-12.4	3.1	10.3	8.2
0.09	3.1	61.9	22.7	-7.2	3.1	60.8	48.5	-2.1	7.2	-14.4
0.05	-11.3	-1.0	-2.1	-2.1	13.4	-29.9	9.3	-19.6	13.4	-16.5
0.01	44.3	-5.2	2.1	-28.9	114.4	-16.5	-13.4	-15.5	-12.4	-5.2

Table A.1 Relative bias (in percentage) for the posterior mean, median, and mode estimates for priors $f_u \simeq f_v \simeq$ Unif and $f_\lambda \simeq \text{Gamma}(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha})$ with $\alpha = 1$ ($\text{cv}(\lambda) = 0.71$) .

A.5. $f_u \simeq \mathbf{Unif}(u_m, u_M)$, $f_v \simeq \mathbf{triang}(N_m, N_M, N^{\mathbf{Reg}})$

The same computations as in the preceding section can be carried out using a triangular prior distribution f_v for the a priori population size. The limits N_m and N_M are chosen as in the preceding section and the mode as $N^* = N^{\text{Reg}}$.

```
# Load the libraries
library(pestim)
library(data.table)

# Set the input data
nReg <- 97
nMNO <- 19

# Set the priors and compute the estimates for each set of parameters
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)
```

The results are displayed in tables A.2. We do not either see a strong effect of the reduction of the intervals (u_m, u_M) and (v_m, v_M) on the final estimates. Again the main reason is because these intervals are symmetric with respect to the prior modes and the point estimates are central position measures of the posterior distributions.

A.5 $f_u \simeq \text{Unif}(u_m, u_M)$, $f_v \simeq \text{triang}(N_m, N_M, N^{\text{Reg}})$

Mean	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	51.5	34.0	20.6	-8.2	-3.1	4.1	7.2	3.1	58.8	4.1
0.35	13.4	79.4	12.4	10.3	7.2	11.3	32.0	-6.2	1.0	-4.1
0.31	-6.2	64.9	-3.1	6.2	-15.5	4.1	9.3	11.3	4.1	0.0
0.26	-5.2	20.6	14.4	46.4	3.1	4.1	-12.4	-13.4	-5.2	0.0
0.22	48.5	-13.4	-7.2	-10.3	79.4	12.4	-10.3	-7.2	34.0	-3.1
0.18	-4.1	19.6	17.5	1.0	12.4	7.2	30.9	8.2	11.3	-3.1
0.14	12.4	21.6	11.3	-1.0	5.2	11.3	13.4	-5.2	10.3	-14.4
0.09	2.1	23.7	-9.3	11.3	-14.4	24.7	26.8	3.1	3.1	-2.1
0.05	22.7	23.7	8.2	15.5	16.5	-12.4	12.4	-8.2	-8.2	-13.4
0.01	1.0	14.4	19.6	-7.2	19.6	3.1	10.3	34.0	-1.0	-10.3
Median	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	26.8	23.7	18.6	-8.2	-3.1	5.2	6.2	3.1	34.0	3.1
0.35	12.4	62.9	11.3	10.3	6.2	10.3	14.4	-6.2	0.0	-5.2
0.31	-7.2	42.3	-4.1	5.2	-16.5	3.1	7.2	10.3	4.1	-1.0
0.26	-5.2	19.6	14.4	30.9	2.1	3.1	-12.4	-13.4	-6.2	0.0
0.22	38.1	-13.4	-8.2	-10.3	68.0	12.4	-10.3	-7.2	20.6	-3.1
0.18	-5.2	18.6	16.5	0.0	11.3	7.2	23.7	7.2	7.2	-4.1
0.14	13.4	20.6	11.3	-1.0	4.1	10.3	13.4	-5.2	10.3	-13.4
0.09	1.0	21.6	-9.3	10.3	-14.4	19.6	24.7	3.1	2.1	-2.1
0.05	21.6	21.6	7.2	13.4	13.4	-12.4	12.4	-8.2	-8.2	-13.4
0.01	0.0	13.4	17.5	-7.2	13.4	3.1	9.3	35.1	-2.1	-10.3
Mode	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	56.7	73.2	5.2	-1.0	-22.7	-13.4	-17.5	16.5	-29.9	20.6
0.35	-7.2	16.5	39.2	29.9	34.0	15.5	306.2	-15.5	-4.1	34.0
0.31	-14.4	246.4	-7.2	-2.1	-14.4	25.8	-21.6	6.2	-1.0	13.4
0.26	-16.5	20.6	10.3	-30.9	1.0	19.6	-11.3	-7.2	-1.0	4.1
0.22	15.5	-6.2	-12.4	-15.5	-19.6	10.3	8.2	-10.3	7.2	8.2
0.18	-4.1	20.6	5.2	-17.5	-1.0	-7.2	11.3	29.9	10.3	-8.2
0.14	36.1	10.3	36.1	-5.2	-2.1	18.6	19.6	-22.7	-12.4	-8.2
0.09	1.0	58.8	-9.3	9.3	-27.8	-13.4	-46.4	6.2	28.9	-1.0
0.05	48.5	4.1	-7.2	26.8	47.4	-6.2	17.5	-12.4	-19.6	-6.2
0.01	-28.9	52.6	-5.2	-13.4	30.9	22.7	-27.8	66.0	-16.5	-13.4

Table A.2 Relative bias (in percentage) for the posterior mean, median, and mode estimates for priors $f_u \simeq \text{Unif}$, $f_v \simeq \text{triang}$, and $f_\lambda \simeq \text{Gamma}(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha})$ with $\alpha = 1$ $(\text{cv}(\lambda) = 0.71)$.

A.6. $f_u \simeq$ **Unif**(u_m, u_M)**,** $f_v \simeq$ **Gamma**$(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha})$

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

```
# Load the libraries
library(pestim)
library(data.table)

# Set the input data
nReg <- 97
nMNO <- 19

# Set the priors and compute the estimates for each set of parameters
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)
```

The results are displayed in tables A.3.

A.6 $f_u \simeq \text{Unif}(u_m, u_M)$, $f_v \simeq \text{Gamma}(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha})$

Mean	γ						
Length **u**	0.001	0.01	0.1	1	10	100	1000
0.39	-13.4	-14.4	-14.4	-14.4	5.2	12.4	22.7
0.35	-13.4	-16.5	-14.4	-14.4	9.3	0.0	32.0
0.31	-12.4	-15.5	-15.5	-14.4	23.7	27.8	12.4
0.26	-10.3	-14.4	-15.5	9.3	18.6	24.7	-8.2
0.22	-16.5	-16.5	-15.5	-15.5	5.2	-4.1	-11.3
0.18	-15.5	-15.5	-15.5	-14.4	0.0	12.4	-11.3
0.14	-14.4	-15.5	-15.5	-15.5	5.2	6.2	20.6
0.09	-16.5	-15.5	-15.5	-15.5	6.2	12.4	11.3
0.05	-15.5	-16.5	-15.5	-15.5	15.5	-1.0	11.3
0.01	-14.4	-16.5	-16.5	-15.5	45.4	26.8	12.4
Median	γ						
Length **u**	0.001	0.01	0.1	1	10	100	1000
0.39	-13.4	-15.5	-14.4	-14.4	2.1	10.3	13.4
0.35	-13.4	-16.5	-15.5	-14.4	5.2	-1.0	15.5
0.31	-12.4	-15.5	-15.5	-14.4	20.6	24.7	11.3
0.26	-10.3	-14.4	-15.5	9.3	16.5	21.6	-8.2
0.22	-16.5	-16.5	-15.5	-15.5	2.1	-4.1	-11.3
0.18	-15.5	-15.5	-15.5	-14.4	-2.1	10.3	-11.3
0.14	-13.4	-15.5	-16.5	-15.5	2.1	5.2	15.5
0.09	-16.5	-15.5	-16.5	-14.4	3.1	12.4	9.3
0.05	-15.5	-16.5	-15.5	-15.5	11.3	-2.1	10.3
0.01	-14.4	-16.5	-16.5	-15.5	43.3	24.7	11.3
Mode	γ						
Length **u**	0.001	0.01	0.1	1	10	100	1000
0.39	-3.1	-22.7	-21.6	-9.3	11.3	7.2	20.6
0.35	-22.7	-32.0	-20.6	-20.6	46.4	-20.6	11.3
0.31	5.2	-22.7	-6.2	-7.2	33.0	-4.1	-32.0
0.26	-3.1	-9.3	-15.5	0.0	11.3	-2.1	-6.2
0.22	-26.8	-21.6	-13.4	-19.6	32.0	5.2	1.0
0.18	-21.6	-17.5	-12.4	-10.3	18.6	2.1	-14.4
0.14	-11.3	-5.2	-3.1	-18.6	24.7	6.2	-20.6
0.09	-20.6	-6.2	1.0	-19.6	-13.4	-25.8	33.0
0.05	-14.4	-17.5	-15.5	-32.0	51.5	3.1	13.4
0.01	-18.6	0.0	-7.2	-29.9	76.3	11.3	51.5

Table A.3 Relative bias (in percentage) for the posterior mean, median, and mode estimates for priors $f_u \simeq$ Unif, $f_v \simeq$ Gamma, and $f_\lambda \simeq \text{Gamma}(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha})$ with $\alpha = 1$ ($\text{cv}(\lambda) = 0.71$) .

A.7. $f_u \simeq$ **Triang**(u_m, u_M, u^*), $f_v \simeq$ **Unif**(N_m, N_M)

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

```
# Load the libraries
library(pestim)
library(data.table)

# Set the input data
nReg <- 97
nMNO <- 19

# Set the priors and compute the estimates for each set of parameters
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)
```

The results are displayed in tables A.4.

Mean	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	27.8	25.8	18.6	-5.2	-9.3	-8.2	-2.1	-7.2	9.3	-10.3
0.35	-3.1	-15.5	89.7	66.0	13.4	-2.1	24.7	11.3	-9.3	-12.4
0.31	2.1	79.4	0.0	-4.1	0.0	19.6	-6.2	3.1	32.0	0.0
0.26	-11.3	-11.3	15.5	0.0	22.7	3.1	-15.5	21.6	-7.2	6.2
0.22	27.8	27.8	17.5	15.5	-10.3	17.5	11.3	6.2	11.3	-8.2
0.18	1.0	26.8	23.7	2.1	18.6	-9.3	18.6	23.7	40.2	-9.3
0.14	-9.3	12.4	-3.1	-11.3	10.3	14.4	-5.2	24.7	19.6	2.1
0.09	26.8	-6.2	19.6	18.6	14.4	17.5	0.0	3.1	-5.2	3.1
0.05	-9.3	1.0	-16.5	17.5	17.5	-14.4	-9.3	-9.3	14.4	3.1
0.01	27.8	16.5	26.8	14.4	16.5	12.4	14.4	-5.2	6.2	-5.2
Median	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	25.8	24.7	18.6	-5.2	-10.3	-8.2	-3.1	-7.2	8.2	-10.3
0.35	-3.1	-15.5	84.5	61.9	13.4	-2.1	14.4	10.3	-9.3	-12.4
0.31	1.0	67.0	-1.0	-5.2	-1.0	16.5	-6.2	3.1	19.6	-1.0
0.26	-11.3	-11.3	14.4	0.0	16.5	2.1	-15.5	14.4	-7.2	3.1
0.22	23.7	23.7	16.5	14.4	-10.3	17.5	10.3	5.2	10.3	-8.2
0.18	1.0	23.7	23.7	3.1	16.5	-9.3	13.4	15.5	41.2	-10.3
0.14	-10.3	11.3	-3.1	-11.3	8.2	12.4	-6.2	16.5	13.4	1.0
0.09	26.8	-6.2	15.5	17.5	13.4	17.5	0.0	1.0	-5.2	2.1
0.05	-9.3	0.0	-16.5	16.5	15.5	-14.4	-10.3	-9.3	10.3	2.1
0.01	23.7	15.5	17.5	12.4	12.4	11.3	12.4	-6.2	4.1	-5.2
Mode	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	-17.5	-13.4	14.4	-16.5	-19.6	-6.2	20.6	2.1	18.6	-5.2
0.35	10.3	6.2	-32.0	24.7	10.3	-2.1	6.2	-10.3	-6.2	-10.3
0.31	29.9	172.2	1.0	3.1	3.1	2.1	-1.0	12.4	-23.7	6.2
0.26	-11.3	-11.3	26.8	-5.2	14.4	11.3	-14.4	-10.3	-8.2	19.6
0.22	175.3	37.1	-11.3	34.0	-8.2	21.6	27.8	19.6	-15.5	-13.4
0.18	-3.1	-35.1	10.3	-14.4	52.6	7.2	20.6	16.5	68.0	-32.0
0.14	-19.6	-9.3	4.1	-6.2	17.5	28.9	10.3	9.3	1.0	-6.2
0.09	-3.1	-8.2	0.0	81.4	4.1	19.6	-16.5	27.8	15.5	-29.9
0.05	-21.6	-19.6	-27.8	60.8	-25.8	-33.0	-1.0	-14.4	-17.5	-19.6
0.01	-7.2	10.3	48.5	30.9	28.9	-12.4	36.1	-4.1	8.2	1.0

Table A.4 Relative bias (in percentage) for the posterior mean, median, and mode estimates for priors $f_u \simeq$ triang, $f_v \simeq$ Unif, and $f_\lambda \simeq \text{Gamma}(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha})$ with $\alpha = 1$ $(\text{cv}(\lambda) = 0.71)$.

A.8. $f_u \simeq \textbf{Triang}(u_m, u_M, u^*)$, $f_v \simeq \textbf{Triang}(N_m, N_M, N^{\textbf{Reg}})$

Both prior distributions are considered to be triangular with the same choice for hyperparameters.

```
# Load the libraries
library(pestim)
library(data.table)

# Set the input data
nReg <- 97
nMNO <- 19

# Set the priors and compute the estimates for each set of parameters
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)
```

The results are displayed in tables A.5.

A.8 $f_u \simeq \text{Triang}(u_m, u_M, u^*)$, $f_v \simeq \text{Triang}(N_m, N_M, N^{\text{Reg}})$

Mean	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	2.1	120.6	1.0	-13.4	-10.3	15.5	2.1	23.7	1.0	-9.3
0.35	21.6	19.6	12.4	55.7	2.1	61.9	-8.2	0.0	4.1	51.5
0.31	82.5	39.2	-4.1	8.2	-14.4	6.2	-7.2	7.2	39.2	-9.3
0.26	-12.4	17.5	2.1	19.6	17.5	-10.3	14.4	7.2	4.1	21.6
0.22	-6.2	13.4	12.4	58.8	68.0	16.5	5.2	9.3	2.1	3.1
0.18	-1.0	-6.2	18.6	13.4	5.2	-4.1	10.3	6.2	30.9	-11.3
0.14	3.1	23.7	-3.1	10.3	8.2	-12.4	15.5	11.3	27.8	-4.1
0.09	20.6	21.6	5.2	0.0	-2.1	14.4	-14.4	-8.2	6.2	-14.4
0.05	20.6	-9.3	-3.1	-6.2	16.5	-8.2	11.3	36.1	-11.3	1.0
0.01	-4.1	15.5	19.6	1.0	11.3	7.2	14.4	-13.4	5.2	7.2
Median	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	2.1	89.7	1.0	-13.4	-10.3	13.4	2.1	12.4	-1.0	-9.3
0.35	19.6	19.6	11.3	51.5	1.0	56.7	-8.2	-1.0	4.1	42.3
0.31	72.2	22.7	-4.1	7.2	-14.4	6.2	-7.2	6.2	25.8	-9.3
0.26	-12.4	17.5	2.1	17.5	17.5	-10.3	13.4	5.2	2.1	11.3
0.22	-5.2	13.4	12.4	54.6	67.0	13.4	4.1	7.2	1.0	3.1
0.18	-2.1	-7.2	16.5	12.4	5.2	-5.2	9.3	5.2	23.7	-11.3
0.14	2.1	13.4	-4.1	10.3	8.2	-13.4	12.4	10.3	27.8	-5.2
0.09	20.6	20.6	4.1	0.0	-3.1	13.4	-14.4	-8.2	5.2	-14.4
0.05	21.6	-9.3	-3.1	-6.2	7.2	-9.3	9.3	35.1	-11.3	0.0
0.01	-4.1	14.4	18.6	0.0	10.3	6.2	11.3	-13.4	4.1	6.2
Mode	Length **v**									
Length **u**	48	44	38	32	28	22	18	12	8	2
0.39	-12.4	29.9	40.2	-15.5	-20.6	4.1	-3.1	53.6	-15.5	-29.9
0.35	-20.6	48.5	-14.4	-37.1	3.1	125.8	3.1	-18.6	11.3	35.1
0.31	-57.7	144.3	-5.2	-34.0	-16.5	-5.2	-8.2	2.1	-34.0	1.0
0.26	-17.5	38.1	33.0	0.0	-4.1	1.0	17.5	17.5	18.6	-13.4
0.22	-7.2	49.5	50.5	54.6	108.2	187.6	4.1	-18.6	3.1	-3.1
0.18	-2.1	-28.9	56.7	-18.6	-10.3	1.0	47.4	-5.2	133.0	-20.6
0.14	1.0	76.3	-1.0	40.2	14.4	-8.2	14.4	34.0	61.9	-13.4
0.09	-26.8	-13.4	-3.1	-10.3	-9.3	-21.6	-18.6	-22.7	-5.2	-18.6
0.05	15.5	-14.4	5.2	-16.5	-48.5	14.4	-5.2	55.7	1.0	17.5
0.01	1.0	20.6	27.8	-17.5	43.3	18.6	50.5	-23.7	-8.2	6.2

Table A.5 Relative bias (in percentage) for the posterior mean, median, and mode estimates for priors $f_u \simeq$ triang, $f_v \simeq$ triang, and $f_\lambda \simeq \text{Gamma}(\alpha + 1, \frac{N^{\text{Reg}}}{\alpha})$ with $\alpha = 1$ $(\text{cv}(\lambda) = 0.71)$.

A.9. $f_u \simeq$ **Triang**(u_m, u_M, u^*), $f_v \simeq$ **Gamma**$(a+1, \frac{N^{\mathbf{Reg}}}{a})$

In the last example we combined a triangular distribution for f_u and a gamma distribution for f_v.

```
# Load the libraries
library(pestim)
library(data.table)

# Set the input data
nReg <- 97
nMNO <- 19

# Set the priors and compute the estimates for each set of parameters
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)
```

The results are displayed in tables A.6.

A.9 $f_u \simeq \text{Triang}(u_m, u_M, u^*)$, $f_v \simeq \text{Gamma}(a+1, \frac{N^{\text{Reg}}}{a})$

Mean	γ						
Length **u**	0.001	0.01	0.1	1	10	100	1000
0.39	-12.4	-14.4	-14.4	-16.5	7.2	0.0	73.2
0.35	-10.3	-13.4	-14.4	-16.5	29.9	-12.4	-9.3
0.31	-13.4	-15.5	-15.5	-13.4	1.0	-11.3	-13.4
0.26	-9.3	-16.5	-15.5	-11.3	1.0	3.1	3.1
0.22	-14.4	-14.4	-15.5	-14.4	10.3	23.7	12.4
0.18	-15.5	-15.5	-15.5	-14.4	7.2	29.9	13.4
0.14	-15.5	-15.5	-16.5	-15.5	4.1	-12.4	-10.3
0.09	-15.5	-14.4	-15.5	-15.5	33.0	14.4	4.1
0.05	-15.5	-15.5	-16.5	-15.5	18.6	12.4	17.5
0.01	-13.4	-15.5	-16.5	-15.5	-5.2	22.7	-11.3
Median	γ						
Length **u**	0.001	0.01	0.1	1	10	100	1000
0.39	-12.4	-15.5	-14.4	-16.5	4.1	-2.1	58.8
0.35	-10.3	-13.4	-15.5	-16.5	26.8	-12.4	-9.3
0.31	-12.4	-15.5	-15.5	-13.4	-2.1	-11.3	-13.4
0.26	-10.3	-16.5	-15.5	-11.3	-1.0	2.1	3.1
0.22	-14.4	-14.4	-15.5	-14.4	6.2	18.6	10.3
0.18	-15.5	-15.5	-15.5	-14.4	5.2	27.8	12.4
0.14	-16.5	-15.5	-16.5	-16.5	1.0	-12.4	-11.3
0.09	-15.5	-15.5	-16.5	-15.5	25.8	14.4	3.1
0.05	-15.5	-16.5	-16.5	-15.5	15.5	11.3	11.3
0.01	-13.4	-16.5	-16.5	-15.5	-8.2	19.6	-12.4
Length **u**	0.001	0.01	0.1	1	10	100	1000
0.39	-15.5	-18.6	-20.6	-16.5	15.5	-7.2	143.3
0.35	9.3	-23.7	-13.4	-11.3	-17.5	-19.6	-10.3
0.31	-4.1	-28.9	-13.4	-9.3	-36.1	-24.7	-13.4
0.26	1.0	-23.7	1.0	-21.6	2.1	-2.1	3.1
0.22	-21.6	-12.4	-13.4	-6.2	11.3	0.0	-18.6
0.18	-10.3	-12.4	-28.9	-28.9	1.0	-26.8	4.1
0.14	-34.0	-37.1	-11.3	-15.5	6.2	-20.6	0.0
0.09	-8.2	-2.1	-9.3	-11.3	53.6	15.5	28.9
0.05	-9.3	-29.9	-18.6	-16.5	-22.7	-11.3	18.6
0.01	-7.2	-7.2	-10.3	-12.4	34.0	-1.0	-17.5

Table A.6 Relative bias (in percentage) for the posterior mean, median, and mode estimates for priors $f_u \simeq$ triang, $f_v \simeq$ Gamma, and $f_\lambda \simeq \text{Gamma}(\alpha+1, \frac{N^{\text{Reg}}}{\alpha})$ with $\alpha = 1$ $(\text{cv}(\lambda) = 0.71)$.

Appendix B

Documentation manual of package `pestim`

Package 'pestim'

March 16, 2018

Type Package

Title Population Estimations Using Mobile Phone Data

Version 0.1.0

Description This package contains functions that implement a simple 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.

License GPL-3 and EUPL

Encoding UTF-8

LazyData true

Depends R (>= 3.3.0)

Imports data.table (>= 1.10.4),
Rcpp (>= 0.12.12),
MCMCpack (>= 1.4-2)

LinkingTo Rcpp

RoxygenNote 5.0.1

Collate 'MobPop.R'
'ratioBeta.R'
'kummer.R'
'Phi.R'
'RcppExports.R'
'alphaPrior.R'
'dg.R'
'triang.R'
'dlambda.R'
'flambda.R'
'fu.R'
'fv.R'
'genAlpha.R'
'genUV.R'
'modeLambda.R'
'nMNO_ini.R'
'nReg.R'
'pestim.R'
'rlambda.R'
'rN0.R'
'postN0.R'

'rmatProb.R'
'rNtcondN0.R'
'rNt.R'
'postNt.R'
'postNtcondN0.R'
'rg.R'
'rp.R'

R topics documented:

`alphaPrior` *Generate prior distributions for parameters of the Dirichlet distribution.*

Description

Generate a list of prior distributions for the parameters of the Dirichlet distribution in the hierarchical model. Each component of the list corresponds to the prior distribution of the parameter $\alpha_{ij}(t_0, t_n)$ for each cell j. This function initial works over a fixed initial cell i. Each returned distribution is specified as a list with an identification name as first component and named components with the distribution parameters for the rest of components.

Usage

```
alphaPrior(nMNOfrom, names, variation)
```

Arguments

nMNOfrom	numeric vector with the number of individuals moving from the initial cell to the rest of cells (including those remaining)
names	character vector with the names of the prior distributions for each cell
variation	list of lists whose components are parameters providing a measure of variation of each prior distribution

Details

The function takes the number of cells from the input parameter nMNOfrom which specifies the number of individuals detected by the network moving from the initial cell to each of the cells (including those remaining in the same). The function executes the same construction for each final cell. It takes the name of prior distribution from the input parameter names and construct the corresponding prior distribution for each cell j with mode at $u_j^* = N_j$, where N_j is taken from nMNOfrom. Next the rest of parameters of the distribution are computed according to the dispersion parameters specified in variation.

As accepted distribution names, currently the user can specify unif, degen, triang, and gamma.

The dispersion parameters recognised so far are the coefficients of variation only (standard deviation divided by the mean of the distribution). These dispersion parameters must be specified by a named component cv with a numeric value in $[0, 1]$.

For each distribution the parameters are computed as follows:

- unif: This is the uniform distribution with parameters xMax and xMin. Both parameters are computed by $u_j^* \cdot (1 \pm \sqrt{3}\text{cv})$, respectively, in each cell j.
- degen: This is the degenerate distribution with parameter X0 taken as u_j^* in each cell j.
- triang: This is the triangular distribution dtriang with parameters xMax, xMin, and xMode. The latter is taken directly from nMNOfrom. The distribution is assumed to be symmetrical so that the two former parameters are computed by $u_j^* \cdot (1 \pm \sqrt{3}\text{cv})$, respectively, in each cell j.
- gamma: This is the gamma distribution with parameters shape and scale. The former is computed as $\frac{1}{\text{cv}^2}$ and the latter as $fracu_j^*\text{scale} - 1$.

Value

Return a list with a list in each component specifying the prior for each cell

Examples

```
# Three cells. Cell 1 under study. 10 individuals remain.
alphaPrior(c(10, 3, 4), c('unif', 'triang', 'gamma'),
           list(list(cv = 0.1), list(cv = 0.05), list(cv = 0.15)))
```

dg *Density function of a candidate distribution in the accept-reject method.*

Description

Generate values of a candidate distribution density function in the accept-reject method of generation of random variables applied to the distribution of the lambda parameter

Usage

```
dg(lambda, nMNO, nReg, fu, fv, flambda, relTol = 1e-06, nSim = 10000,
  nStrata = c(1, 100), verbose = FALSE)
```

Arguments

lambda	numeric vector with the lambda parameter values
nMNO,	nReg non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register
fu,	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
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)
verbose	logical (default FALSE) to report progress of the computation

Details

The candidate distribution is a gamma distribution with parameters shape = nMNO + 1 and scale = λ^* / nMNO, where λ^* stands for the mode of the posterior distribution of the lambda parameter.

It is important to know that currently this function accepts only parameters for a single cell at a time. In case of interest for the candidate density function values for a set of cells, the user should program his/her own routine to apply this function to every cell.

Value

dg generates length(lambda) values of the density probability function of the candidate distribution in the accept-reject method.

See Also

modeLambda, dlambda for related functions.

Examples

```
curve(dg(x, nMNO = 20, nReg = 115, fu = list('unif', xMin = 0.3, xMax = 0.5),
        fv = list('unif', xMin = 100, xMax = 120),
        flambda = list('gamma', shape = 11, scale = 12)), xlim = c(0, 150),
        main = '', ylab = 'density', xlab = 'lambda')
```

`dlambda` *Posterior density function of the lambda parameter.*

Description

Compute the unnormalized posterior density function of the parameter λ in the hierarchical model to estimate population counts given by

$$f(\lambda | N^{\text{MNO}}; N^{\text{Nreg}}) \propto f(\lambda) \cdot \text{dpois}(N^{\text{MNO}}; \lambda) \cdot S(\lambda; N^{\text{MNO}}, N^{\text{Nreg}}),$$

where `dpois` is the probability density function of a Poisson distribution and S is defined in the bibliographic reference.

Usage

```
dlambda(lambda, nMNO, nReg, fu, fv, flambda, relTol = 1e-06, nSim = 1000,
  nStrata = c(1, 100), verbose = FALSE)
```

Arguments

`lambda`	numeric vector
`nMNO,`	nReg non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register
`fu,`	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
`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 `1e3`
`nStrata`	integer vector of length 2 with the number of strata in each dimension. Default value is `c(1, 1e2)`
`verbose`	logical (default `FALSE`) to report progress of the computation

Details

The lengths of the input vectors `nMNO` and `nReg` must be both equal to 1 and independent of the length of the input vector `lambda`. The integral is computed using with Monte Carlo techniques using `nSim` points for each of the values `lambda` specified so that the final data.table has `length(lambda)` rows.

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: xMin, xMax for the minimum, maximum of the sampled interval.
- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

It is important to know that currently this function accepts only parameters for a single cell at a time. In case of interest for the density function values for a set of cells, the user should program his/her own routine to apply this function to every cell.

Value

dlambda returns a data.table with the values of the density function (column probLambda) for each value of lambda together with additional variables:

- The common length of nMNO and nReg identifies the number of territorial cells in which the number of individuals detected by the telecommunication network and official data. The column cellID identifies these territorial cells.
- The length of lambda identifies the number of parameters upon which the integral will be computed in each cell. The column parID identifies each of these input parameters.
- The inputs nMNO and nReg are also included in the output data.table in columns under the same name.
- The value on the integral times the Poisson density function ifalso included under the column integral

References

https://github.com/MobilePhoneESSnetBigData

See Also

genUV, Phi for related functions.

Examples

```
# This data.table must have 5x3= 15 rows
dlambda(seq(0, 1, length.out = 5),
        nMNO = c(20, 17, 25), nReg = c(115, 123, 119),
      fu = list('unif', xMin = 0.3, xMax = 0.5), fv = list('gamma', shape = 11, scale = 12),
        flambda = list('gamma', shape = 11, scale = 12))

# Easily, a function to draw conditioned on the parameters:
f <- function(x){
  dlambda(x, nMNO = 20, nReg = 115,
        fu = list('unif', xMin = 0.3, xMax = 0.5), fv = list('unif', xMin = 100, xMax = 120),
          flambda = list('gamma', shape = 11, scale = 12))$probLambda
}
curve(f, xlim = c(0, 150))
```

dtriang *The Triangular Distribution.*

Description

Density, distribution funtion, quantile function and random generation for the triangular distribution

Usage

```
dtriang(x, xMin, xMax, xMode)
```

Arguments

x,	q vector of quantiles
xMin	vector with the minimum values of the range of the random variable
xMax	vector with the maximum values of the range of the random variable
xMode	vector with the modes of the random variable
p	vector pf probabilities
n	number of observations

Value

dtriang gives the density, ptriang gives the distribution function, qtriang gives the quantile function, and rtriang generates random deviates.

The lengths of the input vectors (except n) must be all equal except when their length is 1. Otherwise NAs are produced.

See Also

Distributions for other distributions

Examples

```
curve(dtriang(x, 0, 3, 1), xlim = c(0, 3))
curve(ptriang(x, 0, 3, 1), xlim = c(0, 3))
curve(qtriang(x, 0, 3, 1), xlim = c(0, 1))
hist(rtriang(1e6, 0, 3, 1), breaks = seq(0, 3, by = 0.01))
```

flambda — *List of priors for the parameter lambda for the dataset* MobPop.

Description

This list contains the priors for each of the 12 cells of the simulated populated included in the data.table MobPop.

Usage

```
flambda
```

Format

A list with 12 components each of which is a list with three components:

name of the prior distribution (gamma in all cases in this example)

xMin shape parameter for the gamma prior distribution of each cell

xMax scale parameter for the gamma prior distribution of each cell

fu — *List of priors for the parameter u for the dataset* MobPop.

Description

This list contains the priors for each of the 12 cells of the simulated populated included in the data.table MobPop (see function genUV).

Usage

```
fu
```

Format

A list with 12 components each of which is a list with three components:

name of the prior distribution (unif in all cases in this example)

xMin minimum value of the range of values of the uniform prior distribution of each cell

xMax maximum value of the range of values of the uniform prior distribution of each cell

fv — *List of priors for the parameter v for the dataset* MobPop.

Description

This list contains the priors for each of the 12 cells of the simulated populated included in the data.table MobPop (see function genUV).

Usage

```
fv
```

Format

A list with 12 components each of which is a list with three components:

name of the prior distribution (unif in all cases in this example)

xMin minimum value of the range of values of the uniform prior distribution of each cell

xMax maximum value of the range of values of the uniform prior distribution of each cell

genAlpha — *Generate values for the parameters of the Dirichlet distribution.*

Description

Generate a matrix of values of the parameters $\alpha_{ij}(t_0, t_n)$ of the Dirichlet distribution in the hierarchical model. This function initial works over a fixed initial cell i under study.

Usage

```
genAlpha(nSim, flist)
```

Arguments

nSim	number of values to generate
flist	list with the prior distributions for each cell

Details

This function generates the nSim random values according to the prior of each cell specified in flist.

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest of components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: xMin, xMax for the minimum, maximum of the sampled interval.
- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

Value

Return a matrix with as many columns as cells and as many rows as number of generated values

Examples

```
priors <- alphaPrior(c(10, 3, 4), c('unif', 'triang', 'gamma'),
                     list(list(cv = 0.1), list(cv = 0.05), list(cv = 0.15)))
genAlpha(10, priors)
```

`genUV` *Generation of two-dimensional random deviates.*

Description

Generate two-dimensional random deviates for a Monte Carlo computation of the integral

$$\int_0^\infty dv f_2(v) \int_0^\infty f_1(u)\ \Phi(u \cdot v, (1-u) \cdot v; \lambda, N^{\text{MNO}}, N^{\text{Reg}}).$$

The Monte Carlo technique makes use of stratified importance sampling.

Usage

```
genUV(nSim, nStrata, f1, f2, lambda, nMNO, nReg)
```

Arguments

`nSim`	number of two-dimensional points to generate
`nStrata`	integer vector of length 2 with the number of strata in each dimension
`f1,`	f2 named lists with the prior marginal distributions of the two-dimensional points
`lambda`	numeric vector
`nMNO,`	nReg non-negative integer vectors

Details

The lengths of the input vectors `nMNO` and `nReg` must be equal and independent of the length of the input vector `lambda`. Notice that `nSim` points are generated for each of the `length(nMNO)`$\times$`length(lambda)` combinations so that the final data.table has `nSim`$\times$`length(nMNO)` $\times$`length(lambda)` rows.

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: `xMin`, `xMax` for the minimum, maximum of the sampled interval.
- degen: `x0` for the degenerate value of the random variable.
- triang: `xMin`, `xMax`, `xMode` for minimum, maximum and mode (see `qtriang`).
- gamma: `scale` and `shape` with the same meaning as in `rgamma`.

Value

genUV returns a data.table with the (u,v) coordinates of each point together with additional variables:

- The common length of nMNO and nReg identifies the number of territorial cells in which the number of individuals detected by the telecommunication network and official data. The column cellID identifies these territorial cells.
- The length of lambda identifies the number of parameters upon which the integral will be computed in each cell. The column parID identifies each of these input parameters.
- Stratum_u and Stratum_v jointly identify each stratum in which the region of integration has been divided with the stratification.

See Also

runif, qtriang, rgamma for related functions.

Examples

```
# This data.table must have 10x5x3= 150 rows and only one stratum
genUV(nSim = 10, nStrata = c(1, 1),
     f1 = list('unif', xMin = 0.3, xMax = 0.5), f2 = list('gamma', shape = 11, scale = 12),
      lambda = seq(0, 1, length.out = 5),
      nMNO = c(20, 17, 25), nReg = c(115, 123, 119))
```

---

kummer *Confluent hypergeometric or Kummer function*

Description

Partial implementation of the confluent hypergeometric function ${}_1F_1(x; a; b)$

Usage

```
kummer(x, a, b, relTol = 1e-06)
```

Arguments

x,	a, b numeric vectors of the same length
relTol	relative tolerance (default value 1e-6) understood as the ratio of each term in the series relative to the sum

Details

This function is implemented in C++. It is based on Pearson et al (2016). It only implements the Taylor series method together with an asymtoptic expansion based on Watson's lemma

Value

Return a numeric vector with the values of the function

Author(s)

Luis Sanguiao Bogdan Oancea

MobPop *Dataset with simulated data for population counts.*

Description

This dataset 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.

Usage

```
MobPop
```

Format

A data.table with 96768 rows and 6 variables:

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

modeLambda *Mode of the posterior density function of the lambda parameter.*

Description

Compute the mode of the unnormalized posterior density function of the parameter λ in the hierarchical model to estimate population counts.

Usage

```
modeLambda(nMNO, nReg, fu, fv, flambda, relTol = 1e-06, nSim = 10000,
  nStrata = c(1, 100), verbose = FALSE)
```

Arguments

`nMNO,`	nReg non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register
`fu,`	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
`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)
verbose	logical (default FALSE) to report progress of the computation

Details

The lengths of the input vectors nMNO and nReg must be equal. Currently the optimization algorithm is a simple direct algorithm taking into account the form of the density function.

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: xMin, xMax for the minimum, maximum of the sampled interval.
- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

Value

modeLambda returns a vector with the values of the mode of the density function (column probLambda) for each cell.

See Also

dlambda for the function to maximize.

Examples

```
# This data.table must have 5x3= 15 rows
modeLambda(nMNO = c(20, 17, 25), nReg = c(115, 123, 119),
        fu = list(list('unif', xMin = 0.3, xMax = 0.5),
                  list('unif', xMin = 0.35, xMax = 0.45),
                  list('unif', xMin = 0.25, xMax = 0.43)),
        fv = list(list('gamma', shape = 11, scale = 12),
                  list('gamma', shape = 12, scale = 12.3),
                  list('gamma', shape = 13, scale = 11.5)),
        flambda = list(list('gamma', shape = 11, scale = 12),
                       list('gamma', shape = 12, scale = 12.3),
                       list('gamma', shape = 13, scale = 12)))
```

nMNO_ini

Counts of individuals for the initial time period detected by the mobile network operator.

Description

This vector contains the counts of individuals for the initial time period detected by the mobile network operator in each of the 12 cells.

Usage

```
nMNO_ini
```

Format

A vector with 12 components.

nReg

Population counts for the initial time period according to the population register.

Description

This vector contains the population counts for the initial time period according to the population register.

Usage

```
nReg
```

Format

A vector with 12 components.

pestim

pestim: a hierarchical model to estimate population counts with aggregated mobile phone data.

Description

This package 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.

Context

This package has been developed in the context of a European research project within the European Statistical System called ESSnet on Big Data. More specifically this work corresponds to the work package on mobile phone data by which we assess the use of this data source in the production of official statistics. The goals of the project is many-fold. Firstly, the issue of accessing these data for the production of official statistics initially for research and then for standard production has been investigated. Secondly, in a hands-on bottom-up approach, we make some initial methodological proposals to produce concrete statistical output using those data sets compiled in the preceding phase. Thirdly, in parallel, IT tools, architecture and software development are assessed especially in contrast to traditional computer frameworks. Finally, quality is appraised especially in the context of the European Statistics Code of Practice and ESS Quality Assurance Framework. This package provides a first-step implementation of software routines to present a proof of concept about a methodological proposal (see below) to make inferences about a target population from a mobile phone dataset.

The hierarchical model in a nutshell

The methodological proposal giving rise to this package focuses on the inference exercise connecting aggregated mobile phone data with a target population under analysis. In concrete, the goal is to provide estimates of population counts in each cell in which we have divided the territory for which the telecommunication network provides count data. The estimation is assisted with official data at a larger time scale (either from a population register or from a survey).

The 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 model works in two stages. Firstly at the initial time instant, we use data from both sources to make the inference for the actual population counts in each cell. Secondly, the time evolution of these counts are produced using the transition matrices from cell to cell of individuals provided by the mobile network operator.

The essence 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 is also assigned a weakly informative prior based upon both data sources.

Computational paradigm

Computations are conducted following the Bayesian paradigm. In this sense the generation of simulated populations according to different probability distributions is at the core of the package. In this sense the package contains basically three types 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-relation 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-relation functions, providing computation of estimates based upon the populations generated with the preceding functions. Examples of these functions are postN0, postNt, postNtcondN0.

Phi *The product of ratioBeta and Kummer functions*

Description

Compute the product of ratioBeta and kummer functions with a specific set of arguments

Usage

```
Phi(alpha, beta, lambda, n, relTol = 1e-06)
```

Arguments

alpha,	beta non-negative numeric vectors
lambda	numeric vector
n	non-negative integer vector
relTol	relative tolerance in the computation of the kummer function. Default value is 1e-6

Value

Phi returns $\frac{B(alpha+m,beta+n)}{B(alpha,beta)} \cdot {}_1F_1(lambda; alpha; beta)$, where ${}_1F_1$ stands for the confluent hypergeometric function

The lengths of the input vectors must be all equal except when their length is 1, which are recycled. Otherwise NAs are produced.

See Also

ratioBeta, kummer for related functions.

Examples

```
Phi(1, 1, 0.5, 10)
Phi(1:10, 10:1, seq(0, 1, length.out = 10), 3)
Phi(1:4, 4:1, c(2, 3), c(4, 3, 1))
```

postN0 *Posterior mean, median, and mode for the number of individuals at the initial time.*

Description

Compute the posterior mean, median, and mode for the number of individuals generating posterior distribution according to the hierarchical model at the initial time instant

Usage

```
postN0(nMNO, nReg, fu, fv, flambda, n = 1000, scale = 1, relTol = 1e-08,
  nSim = 1000, nStrata = c(1, 100), verbose = FALSE)
```

Arguments

nMNO,	nReg non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register
fu,	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	number of points to generate in the posterior distribution for the computation. Default value is 1e3
scale	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)
verbose	logical (default FALSE) to report progress of the computation

Details

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest of components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: xMin, xMax for the minimum, maximum of the sampled interval.
- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

Value

postN0 computes the posterior mean, median, and mode of the posterior distribution for each cell. The function returns a matrix with the estimates in columns and the cells in rows.

See Also

rN0

Examples

```
# It takes a couple of minutes
postN0(nMNO = 20, nReg = 115, fu = list('unif', xMin = 0.3, xMax = 0.5),
        fv = list('unif', xMin = 100, xMax = 120),
        flambda = list('gamma', shape = 11, scale = 12))
```

---

postNt *Posterior mean, median, and mode for the number of individuals at an arbitrary time.*

Description

Compute the posterior mean, median, and mode for the number of individuals generating posterior distribution according to the hierarchical model.

Usage

```
postNt(nMNOmat, nReg, fu, fv, flambda, distNames, variation, scale = 1,
  n = 1000, relTol = 1e-06, nSim = 1000, nStrata = c(1, 100),
  verbose = FALSE)
```

Arguments

nMNOmat	transition matrix with the number of individuals displaced from cell to cell detected by the Mobile Network Operator
nReg	non-negative integer vector with the number of individuals detected in each cell according to the population register
fu,	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
distNames	character vector with the names of the prior distributions for each cell
variation	list of lists whose components are parameters providing a measure of variation of each prior distribution
scale	numeric vector with the scale to count the number of individuals. Default value is 1
n	number of points to generate in the posterior distribution for the computation. Default value is 1e3
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)
verbose	logical (default FALSE) to report progress of the computation

Details

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest of components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: `xMin`, `xMax` for the minimum, maximum of the sampled interval.
- degen: `x0` for the degenerate value of the random variable.
- triang: `xMin`, `xMax`, `xMode` for minimum, maximum and mode (see `qtriang`).
- gamma: `scale` and `shape` with the same meaning as in `rgamma`.

Value

`postNt` computes the posterior mean, median, and mode of the posterior distribution for each cell at an arbitrary time t. The function returns a matrix with the estimates in columns and the cells in rows.

See Also

`rNt`, `postN0`, `postNtcondN0`

Examples

```
## First, the inputs:

#The transition matrix of individuals detected by the MNO
nMNOmat <- rbind(c(10, 3, 4), c(5, 21, 3), c(3, 9, 18))

# Population at the initial time of each cell according to the population register
nReg <- c(90, 130, 101)

# List of priors for u
u0 <- rowSums(nMNOmat) / nReg
cv_u0 <- 0.15
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)
})

# List of priors for v
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)
})

# List of priors for lambda
cv_lambda <- 0.6
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)

# It takes a couple of minutes.
postNt(nMNOmat, nReg, fu, fv, flambda, distNames, variation)
```

---

postNtcondN0 *Posterior mean, median, and mode for the number of individuals at an arbitrary time conditioned upon the initial population.*

Description

Compute the posterior mean, median, and mode for the number of individuals generating posterior distribution according to the hierarchical model conditioned upon the initial population of each cell, which must be provided

Usage

```
postNtcondN0(N0, nMNOmat, distNames, variation, n = 1000)
```

Arguments

N0	initial population in each cell
nMNOmat	transition matrix with the number of individuals displaced from cell to cell detected by the Mobile Network Operator
distNames	character vector with the names of the prior distributions for each cell
variation	list of lists whose components are parameters providing a measure of variation of each prior distribution
n	number of points to generate in the posterior distribution for the computation. Default value is 1e3

Value

Return a matrix with three columns (mean, median, and mode estimates) and one row per cell

Examples

```
## First, the inputs:

# The initial population
N0 <- c(93, 123, 130)

#The transition matrix of individuals detected by the MNO
nMNOmat <- rbind(c(10, 3, 4), c(5, 21, 3), c(3, 9, 18))

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

```
# It takes a couple of minutes.
postNtcondN0(N0, nMNOmat, distNames, variation)
```

`ratioBeta` *The ratio of two beta functions.*

Description

Compute the ratio of two beta functions whose arguments differ by integer numbers

Usage

```
ratioBeta(alpha, beta, m, n)
```

Arguments

`alpha,`	beta non-negative numeric vectors
`m,`	n non-negative integer vectors

Value

`ratioBeta` gives $\frac{B(alpha+m,beta+n)}{B(alpha,beta)}$

The lengths of the input vectors must be all equal except when their length is 1, which are recycled. Otherwise `NA`s are produced.

See Also

`beta`, `lbeta` for related functions.

Examples

```
ratioBeta(10, 13, 2, 3)
ratioBeta(1:10, 10:1, 2, 3)
ratioBeta(1:3, 3:1, c(2, 3), 4)
```

`rg` *Generation of random deviates of the candidate distribution.*

Description

Generate random points according to the candidate probability distribution in the accept-reject method of generation of random variables applied to the distribution of the lambda parameter

Usage

```
rg(n, nMNO, nReg, fu, fv, flambda, relTol = 1e-06, nSim = 10000,
  nStrata = c(1, 100), verbose = FALSE)
```

Arguments

n	number of values to generate
nMNO,	nReg non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register
fu,	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
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)
verbose	logical (default FALSE) to report progress of the computation

Details

The candidate distribution is a gamma distribution with parameters shape = nMNO + 1 and scale = λ^* / nMNO, where λ^* stands for the mode of the posterior distribution of the lambda parameter.

It is important to know that currently this function accepts only parameters for a single cell at a time. In case of interest for the candidate density function values for a set of cells, the user should program his/her own routine to apply this function to every cell.

Value

rg generates n points according to the candidate distribution.

See Also

modeLambda, dlambda for related functions.

Examples

```
hist(rg(1e5, nMNO = 20, nReg = 115, fu = list('unif', xMin = 0.3, xMax = 0.5),
        fv = list('unif', xMin = 100, xMax = 120),
      flambda = list('gamma', shape = 11, scale = 12)), breaks = seq(1, 200, by = 2), main ='')
```

---

rlambda *Generation of random deviates of the posterior distribution of parameter lambda.*

Description

Generate random points according to the posterior probability distribution of the parameter lambda in the hierarchical model.

Usage

```
rlambda(n, nMNO, nReg, fu, fv, flambda, relTol = 1e-06, nSim = 10000,
  nStrata = c(1, 100), verbose = FALSE)
```

Arguments

n	number of values to generate
nMNO,	nReg non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register
fu,	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
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)
verbose	logical (default FALSE) to report progress of the computation

Details

The points are generated 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 lambda parameter.

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: xMin, xMax for the minimum, maximum of the sampled interval.
- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

Value

rlambda generates n points according to the posterior distribution of the parameter lambda. The function returns a vector with these points.

See Also

dlambda, rg for related functions.

Examples

```
# It takes a couple of minutes
hist(rN0(500, nMNO = 20, nReg = 115, fu = list('unif', xMin = 0.3, xMax = 0.5),
        fv = list('unif', xMin = 100, xMax = 120),
        flambda = list('gamma', shape = 11, scale = 12))$N0,
        breaks = seq(1, 200, by = 1), main ='', xlab = 'number of individuals')
```

rmatProb *Generate matrices of transition probabilities*

Description

Generate a list of matrices of transition probabilities computed with the transition matrices of individuals among pairs of cells detected by the network and specified probability input distributions per cell.

Usage

```
rmatProb(n, nMNOmat, distNames, variation)
```

Arguments

n	number of matrices to generate
nMNOmat	transition matrix with the number of individuals displaced from cell to cell detected by the Mobile Network Operator
distNames	character vector with the names of the prior distributions for each cell
variation	list of lists whose components are parameters providing a measure of variation of each prior distribution

Details

The function generates the 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 $u_j^* = N_j$, where N_j is taken from the sum of rows of nMNOmat. Next the rest of parameters of the distribution are computed according to the dispersion parameters specified in variation.

As accepted distribution names, currently the user can specify unif, degen, triang, and gamma.

The dispersion parameters recognised so far are the coefficients of variation only (standard deviation divided by the mean of the distribution). These dispersion parameters must be specified by a named component cv with a numeric value in $[0, 1]$.

For each distribution the parameters are computed as follows:

- unif: This is the uniform distribution with parameters xMax and xMin. Both parameters are computed by $u_j^* \cdot (1 \pm \sqrt{3}\text{cv})$, respectively, in each cell j.
- degen: This is the degenerate distribution with parameter X0 taken as u_j^* in each cell j.
- triang: This is the triangular distribution triang with parameters xMax, xMin, and xMode. The latter is taken directly from nMNOfrom. The distribution is assumed to be symmetrical so that the two former parameters are computed by $u_j^* \cdot (1 \pm \sqrt{3}\text{cv})$, respectively, in each cell j.
- gamma: This is the gamma distribution with parameters shape and scale. The former is computed as $\frac{1}{\text{CV}^2}$ and the latter as $fracu_j^*\text{scale} - 1$.

Value

A list of n matrices with transition probabilities

Examples

```
nMNOmat <- rbind(c(10, 3, 4), c(5, 21, 3), c(3, 9, 18))
distNames <- rep('unif', 3)
variation <- rep(list(list(cv = 0.20)), 3)
rmatProb(10, nMNOmat, distNames, variation)
```

`rN0` *Generation of random deviates of the posterior distribution of initial population counts.*

Description

Generate random points according to the posterior probability distribution of the number of individuals in the hierarchical model.

Usage

```
rN0(n, nMNO, nReg, fu, fv, flambda, scale = 1, relTol = 1e-06,
  nSim = 10000, nStrata = c(1, 100), verbose = FALSE)
```

Arguments

`n`	number of values to generate
`nMNO,`	nReg non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register
`fu,`	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
`scale`	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)`
`verbose`	logical (default `FALSE`) to report progress of the computation

Details

The posterior distribution is a Poisson distribution with parameter `lambda * scale`, where the values of `lambda` are generated with the function `rlambda`.

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: `xMin`, `xMax` for the minimum, maximum of the sampled interval.

- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

Value

rN0 generates n points according to the posterior distribution. The function returns a data.table with these points (under the column N0) together with the additional variables:

- The common length of nMNO and nReg identifies the number of territorial cells in which the number of individuals detected by the telecommunication network and official data. The column cellID identifies these territorial cells.
- The different values of the generated values of lambda are returned under the column lambda.
- The inputs nMNO and nReg are also included in the output data.table in columns under the same name.

See Also

rlambda, rg, rNt for related functions.

Examples

```
# It takes a couple of minutes
hist(rN0(500, nMNO = 20, nReg = 115, fu = list('unif', xMin = 0.3, xMax = 0.5),
        fv = list('unif', xMin = 100, xMax = 120),
        flambda = list('gamma', shape = 11, scale = 12))$N0,
        breaks = seq(1, 200, by = 1), main ='', xlab = 'number of individuals')
```

---

rNt *Generation of random deviates of the posterior distribution of population counts.*

Description

Generate random points according to the posterior probability distribution of the number of individuals in the hierarchical model at arbitrary time instants.

Usage

```
rNt(n, nMNOmat, nReg, fu, fv, flambda, distNames, variation, scale = 1,
  relTol = 1e-06, nSim = 1000, nStrata = c(1, 100), verbose = FALSE)
```

Arguments

n	number of values to generate
nMNOmat	transition matrix with the number of individuals displaced from cell to cell detected by the Mobile Network Operator
nReg	non-negative integer vectors with the number of individuals detected in each cell according to the network operator and the register

fu,	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
distNames	character vector with the names of the prior distributions for each cell
variation	list of lists whose components are parameters providing a measure of variation of each prior distribution
scale	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)
verbose	logical (default FALSE) to report progress of the computation

Details

The posterior distribution is a Poisson distribution with parameter lambda * scale, where the values of lambda are generated with the function rlambda.

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest of components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: xMin, xMax for the minimum, maximum of the sampled interval.
- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

Value

rNt generates n points according to the posterior distribution. The function returns a data.table with these points (under the column N0) together with the additional variables:

- The common length of nMNO and nReg identifies the number of territorial cells in which the number of individuals detected by the telecommunication network and official data. The column cellID identifies these territorial cells.
- The different values of the generated values of lambda are returned under the column lambda.
- The inputs nMNO and nReg are also included in the output data.table in columns under the same name.

See Also

rlambda, rg, rNt for related functions.

Examples

```
## First, the inputs:
# The number of generated values
n <- 1e3

#The transition matrix of individuals detected by the MNO
nMNOmat <- rbind(c(10, 3, 4), c(5, 21, 3), c(3, 9, 18))

# Population at the initial time of each cell according to the population register
nReg <- c(90, 130, 101)

# List of priors for u
u0 <- rowSums(nMNOmat) / nReg
cv_u0 <- 0.15
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)
})

# List of priors for v
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)
})

# List of priors for lambda
cv_lambda <- 0.6
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 output
Nt <- rNt(n, nMNOmat, nReg, fu, fv, flambda, distNames, variation)$N
hist(Nt, breaks = seq(1, max(Nt) + 10, by = 1), main ='', xlab = 'number of individuals')
```

`rNtcondN0` *Conditioned generation of random deviates of the posterior distribution of population counts.*

Description

Generate random deviates of the posterior distribution of the number of individuals at an arbitrary time instant conditioned upon the initial population.

Usage

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

Arguments

n	number of values to generate
N0	initial population in each cell
nMNOmat	transition matrix with the number of individuals displaced from cell to cell detected by the Mobile Network Operator
distNames	character vector with the names of the prior distributions for each cell
variation	list of lists whose components are parameters providing a measure of variation of each prior distribution

Details

The function generates the 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 $u_j^* = N_j$, where N_j is taken from the sum of rows of nMNOmat. Next the rest of parameters of the distribution are computed according to the dispersion parameters specified in variation.

As accepted distribution names, currently the user can specify unif, degen, triang, and gamma.

The dispersion parameters recognised so far are the coefficients of variation only (standard deviation divided by the mean of the distribution). These dispersion parameters must be specified by a named component cv with a numeric value in $[0, 1]$.

For each distribution the parameters are computed as follows:

- unif: This is the uniform distribution with parameters xMax and xMin. Both parameters are computed by $u_j^* \cdot (1 \pm \sqrt{3}\text{cv})$, respectively, in each cell j.
- degen: This is the degenerate distribution with parameter X0 taken as u_j^* in each cell j.
- triang: This is the triangular distribution triang with parameters xMax, xMin, and xMode. The latter is taken directly from nMNOfrom. The distribution is assumed to be symmetrical so that the two former parameters are computed by $u_j^* \cdot (1 \pm \sqrt{3}\text{cv})$, respectively, in each cell j.
- gamma: This is the gamma distribution with parameters shape and scale. The former is computed as $\frac{1}{\text{cv}^2}$ and the latter as $fracu_j^*\text{scale} - 1$.

Value

Return a matrix with as many columns as cells and n rows with the generated values

Examples

```
N0 <- c(93, 123, 130)
nMNOmat <- rbind(c(10, 3, 4), c(5, 21, 3), c(3, 9, 18))
distNames <- rep('unif', 3)
variation <- rep(list(list(cv = 0.20)), 3)
rNtcondN0(1e3, N0, nMNOmat, distNames, variation)
```

rp *Generate random vector deviates of transition probabilities.*

Description

Generate random vector deviates of the transition probabilities $p_{ij}(t_0, t_n)$ for a given cell i stacked into an $n\times$(number of cells) matrix

Usage

```
rp(n, flist)
```

Arguments

n	number of probability vectors to generate
flist	list with the prior distributions for each cell

Details

The prior distributions are specified as named lists where the first component of each list must be the name of distribution ('unif', 'triang', 'degen', 'gamma') and the rest components must be named according to the name of the parameters of the random generator of the corresponding distribution according to:

- unif: xMin, xMax for the minimum, maximum of the sampled interval.
- degen: x0 for the degenerate value of the random variable.
- triang: xMin, xMax, xMode for minimum, maximum and mode (see qtriang).
- gamma: scale and shape with the same meaning as in rgamma.

Value

Return a matrix with n rows and as many columns as cells taken from the length of flist. Each row is thus a probability vector

Examples

```
flist <- alphaPrior(c(10, 3, 4), c('unif', 'triang', 'gamma'),
       list(list(cv = 0.1), list(cv = 0.05), list(cv = 0.15)))
rp(10, flist)
```

Index

Bibliography

American Planning Association (2018). Land based classification standards. `https://www.planning.org/lbcs/`.

Banerjee, S., B. P. Carlin, and A. E. Gelfand (2015). *Hierarchical modelling and analysis of spatial data (2nd ed)*. CRC Press.

Basu, D. (1971). An essay on the logical foundations of survey sampling, Part 1 (with discussion), in V.P. Godambe and D.A. Sprott (eds.), *Foundations of Statistical Inference*, pp. 203–242. Holt, Reinhart and Winston.

Bethlehem, J. (2009). *Applied Survey Methods: A Statistical Perspective*. Wiley.

Brown, J. and R. Churchill (2004). *Complex variables and applications (8th ed.)*. McGraw-Hill.

Calabrese, F., L. Ferrari, and V. D. Blondel (2014). Urban sensing using mobile phone network data: A survey of research. *ACM Computing Surveys 47*, 25:1-25:20.

Casella, G. and R. Berger (2002). *Statistical Inference*. Duxbury Press.

Cassel, C.-M., C.-E. Särndal, and J. Wretman (1977). *Foundations of Inference in Survey Sampling*. Wiley.

Cochran, W. (1977). *Sampling Techniques* (3rd ed.). Wiley.

Deming, W. (1950). *Some theory of sampling*. Wiley.

Deville, P., C. Linard, S. Martin, M. Gilbert, F. Stevens, A. Gaughan, V. Blondel, and A. Tatem (2014). Dynamic population mapping using mobile phone data. *Proceedings of the National Academy of Sciences (USA) 111*, 15888– 15893.

Devroye, L. (1986). *Non-uniform random variable generation*. Springer.

Doyle, J., P. Hung, R. Farrell, and S. Mcloone (2014). Population mobility dynamics estimated from mobile telephony data. *Journal of Urban Technology 21*, 109–132.

EPSG (2018). epsg.io – Coordinate Systems Worldwide. https://epsg.io/28992.

ESS (2011). European Statistics Code of Practice. http://ec.europa.eu/eurostat/documents/3859598/5921861/KS-32-11-955-EN.PDF/5fa1ebc6-90bb-43fa-888f-dde032471e15.

Eurostat (2014). Methodological manual for tourism statistics (v3.1). http://ec.europa.eu/eurostat/documents/3859598/6454997/KS-GQ-14-013-EN-N.pdf.

Eurostat, NIT, University of Tartu, Statistics Estonia, Positium, IFSTTAT, and Statistics Finland (2014). Feasibility study on the use of mobile poistioning data for tourism statistics. http://ec.europa.eu/eurostat/web/tourism/methodology/projects-and-studies.

Gelman, A., B. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin (2013). *Bayesian data analysis*. CRC Press.

Graham, R., D. Knuth, and O. Patashnik (1996). *Concrete Mathematics (2nd ed.)*. Addison-Wesley.

Grimmet, G. and D. Stirzaker (2004). *Probability and random processes* (3rd ed.). Oxford Science Publications.

Groves, R. (1989). *Survey errors and survey costs*. Wiley.

Hájek, J. (1981). *Sampling from a finite population*. Marcel Dekker Inc.

Hansen, M. (1987). Some history and reminiscences on survey sampling. *Statistical Science 2*, 180–190.

Hansen, M., W. Hurwitz, and W. Madow (1966). *Sample survey: methods and theory* (7th ed.). Wiley.

Heckman, J. (1979). Sample selection bias as a specification error. *Econometrica 47*, 153–161.

Hedayat, A. and B. Sinha (1991). *Design and Inference in Finite Population Sampling*. Wiley.

ISO (2004). ISO 8601:2004. https://www.iso.org/standard/40874.html.

ISO (2007). ISO 19111:2007. https://www.iso.org/standard/41126.html.

Kruskal, W. and F. Mosteller (1979a). Representative sampling, i: Non-scientific literature. *International Statistical Review 47*, 13–24.

Kruskal, W. and F. Mosteller (1979b). Representative sampling, ii: scientific literature, excluding statistics. *International Statistical Review 47*, 111–127.

Kruskal, W. and F. Mosteller (1979c). Representative sampling, iii: the current statistical literature. *International Statistical Review 47*, 245–265.

Kruskal, W. and F. Mosteller (1980). Representative sampling, iv: The history of the concept in statistics. *International Statistical Review 48*, 169–195.

Lehtonen, R. and A. Veijanen (1998). Logistic generalized regression estimators. *Survey Methodology 24*, 51–55.

Lessler, J. and W. Kalsbeek (1992). *Nonsampling error in surveys*. Wiley.

Little, R. (2012). Calibrated bayes, an alternative inferential paradigm for official statistics. *Journal of Official Statistics 28*, 309–334.

Manly, B. and J. e. Navarro-Alberto (2014). *Introduction to ecological sampling*. CRC Press.

Meersman, F. D., G. Seynaeve, M. Debusschere, P. Lusyne, P. Dewitte, Y. Baeyens, A. Wirthmann, C. Demunter, F. Reis, and H. Reuter (2016). Assessing the quality of mobile phone data as a source of statistics. *Q2016 Conference paper, June 2016*.

Nemhauser, G. and L. Wolsey (1999). *Integer and combinatorial optimization*. Addison-Wesley.

NetMob (2017). Conference on the scientific analysis of mobile phone datasets. `http://netmob.org/`.

Neyman, J. (1934). On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society 97*, 558–625.

Okabe, A., B. Boots, K. Sugihara, and S.-N. Chiu (2000). *Spatial tessellations: concepts and applications of Voronoi diagrams (2nd ed)*. Wiley.

Positium (2016). Technical documentation for required raw data from mobile network operators for official statistics. Technical report, Positium.

Positium (2017). Common plan for methodology and data processing of mobile phone data from mobile network operators for official statistics. Technical report, Positium.

Positium (2018). `https://www.positium.com/`.

Rao, J. and I. Molina (2015). *Small area estimation (2nd ed)*. Wiley.

Robert, C. and G. Casella (2004). *Monte Carlo Statistical Methods (2nd ed)*. Springer.

Robert, C. and G. Casella (2010). *Introducing Monte Carlo Methods with R*. Springer.

Royle, J. and R. Dorazio (2014). *Hierarchical modeling and inference in Ecology: The Analysis of Data from Populations, Metapopulations and Communities*. Academic Press.

Salgado, D., B. Oancea, and L. Sanguiao (2018). pestim - An R package for population estimation using mobile phone data. `https://www.github.com/MobilePhoneESSnetBigData/pestim`.

Särndal, C.-E. (2007). The calibration approach in survey theory and practice. *Survey Methodology 33*, 99–119.

Särndal, C.-E., B. Swensson, and J. Wretman (1992). *Model assisted survey sampling*. Springer.

Seynaeve, G., C. Demunter, F. D. Meersman, Y. Baeyens, M. Debusschere, P. Dewitte, P. Lusyne, F. Reis, H. Reuter, and A. Wirthmann (2016). When mobile network operators and statistical offices meet - integrating mobile positioning data into the production process of tourism statistics. *14th Global Forum on Tourism Statistics (Venice, Italy, Nov. 2016)*.

Smith, T.M.F. (1976). The foundations of survey sampling: a review. *Journal of the Royal Statistical Society A 139*, 183–204.

Särndal, C.-E. and S. Lundström (2005). *Estimation in Surveys with Nonresponse*. Wiley.

Open Street Map Foundation. `https://www.openstreetmap.org`.

ESS (2017). ESSnet on Big Data. `https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/index.php`.

JAGS (2018). `http://mcmc-jags.sourceforge.net/`.

Pearson, J.W., S. Olver, A.M. Porter (2017). Numerical methods for the computation of the confluent and Gauss hypergeometric functions. *N*umerical Algorithms 74(3), 821–866.

Stan (2018). Stan. `http://mc-stan.org/`.

Tennekes, M. (2018). Geographic Location of Mobile Phone Events. *Vignette, Geographic Location Events.*

Thompson, S. (2012). *Sampling*. Wiley.

UNECE (2013). Generic Statistical Business Process Model v5.0. https://statswiki.unece.org/display/GSBPM/Generic+Statistical+Business+Process+Model.

UNECE (2016). Generic Statistical Data Editing Models. https://statswiki.unece.org/display/VSH/GSDEMs.

Valliant, R., A. Dorfmann, and R. Royall (2000). *Finite population sampling and inference. A prediction approach*. Wiley.

Vanhoof, M., F. Reis, T. Ploetz, and Z. Smoreda (2018). Assessing the quality of home detection from mobile phone data for Official Statistics. *Journal of Official Statistics*. In press.

Watson, G.N. (1918). The Harmonic Functions Associated with the Parabolic Cylinder. *P*roceedings of the London Mathematical Society, *2*, 116—148.

Wilysis (2018). Network Cell Info Lite app. https://play.google.com/store/apps/details?id=com.wilysis.cellinfolite&hl=en_419.

WP5 of ESSnet on Big Data (2016). Deliverable 5.1. https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/images/6/65/WP5_Deliverable_1.1.pdf.

WP5 of ESSnet on Big Data (2017). Deliverable 5.2. https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/images/6/65/WP5.Deliverable1.2.pdf.

WP5 of ESSnet on Big Data (2018). Deliverable 5.3. https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/images/6/65/WP5.Deliverable1.3.pdf.

Yates, F. (1965). *Sampling methods for censuses and surveys* (3rd ed.). Charles Griffins.

Zhang, L.-C. (2012). Topics of statistical theory for register-based statistics and data integration. *Statistica Neerlandica 66(1)*, 41–63.