# User's guide to R functions for PPS sampling *

## 1 Introduction

The `pps` package consists of several functions for selecting a sample from a finite population in such a way that the probability that a unit is selected is proportional to its size, hence the name `pps` (a more precise explanation of the main methods in this package is given at the end of this section). PPS is an acronym for *probability proportional to size*. The package also includes a function to select a stratified simple random sample (non-PPS) as well as a few utility functions. Section 2 of this document uses examples to illustrate the main functions.

The examples are illustrated using real data. The data set `calif` is based on data from the 2000 U.S. census, available at the U.S. Census Bureau's web site, http://www.census.gov. The file includes one record for each *place* in California. A place can be anything from a tiny town to Los Angeles. The file includes a county code, and a stratum code has been added. The four strata were derived according to *county* size: counties with fewer than 50,000 people are in stratum 1 (as are the places in those counties), with 50,000 to 300,000 people in stratum 2, with 300,000 to 1,000,000 people in stratum 3, and the remaining ones are in stratum 4. The other variables in the file are total population (variable name `population`), and the number of whites (`white`), American Indians (`amerind`) and Hispanics (hispanic). A second data set, `califcty`, is derived from `calif`. It is the county-level summary of the bigger file, i.e., each record in `califcty` corresponds to a county. Throughout this document, we will use the total population as the measure of size of a unit (i.e., of a place or a county), where appropriate. The other variables (white,

*Jack Gambino, Ottawa (Canada)

amerind, hispanic) play the role of the "variables of interest", but are not used in any of the illustrations below.

The rest of this document assumes that the pps package and data have been loaded in an R session. If you installed the pps package the "R way"[1], then the functions and data can be loaded into your R session as follows.

```
library(pps)
data(calif)
data(califcty)
```

Otherwise, load them directly as follows. If the `pps.tar.gz` file was extracted in your R working directory, then all files, including documentation will be is a new subdirectory called `pps`. The functions and data are loaded in an R session using the following commands.

```
source("pps/R/pps")
load("pps/data/calif.RData")
load("pps/data/califcty.RData")
```

The sampling methods in this package are described in standard textbooks in survey sampling theory. The classic text is Cochran (1977). To see the details of Sampford's method, especially the calculation of joint inclusion probabilities, the original paper, Sampford (1967), appears to be the only source.

Some of the functions in the package take stratification into account and others do not. The function names make this obvious: a function knows about strata if and only if the string *strat* is contained in its name.

The first function description, for `stratsrs`, is more detailed than the others, to give the reader a better feel for how these functions can be used. Subsequent descriptions are more terse.

## 1.1 Why are these functions useful?

Of course, these functions can be used to actually select samples from finite populations. However, the main reason for writing them was to have handy

[1]Under linux, the instructions in the R documentation to install a package work well. Under Windows, installing a local R package seems trickier. The following worked for me: Get the zip version of the package, say `pps.zip`, and put it in your R working directory. Then, in an R session, enter `install.packages("pps.zip",CRAN=NULL)`

tools for conducting simulation studies. This is useful for pedagogical reasons. It may also be useful for research. For example, one can compare commonly-used approximations (e.g., jackknife, bootstrap, Taylor) to the variance of an estimator, say the Horvitz-Thompson estimator of a total, to its exact variance as follows: Select a sample using Sampford's method. Estimate the variance using the jackknife, bootstrap, etc. Repeat this many times and use the results to compute an empirical MSE of the jackknife, bootstrap, etc. variance estimators. What makes the last step possible is the fact that the package includes a function, `sampfordpi`, that computes the joint inclusion probabilities of all units in the population, which can, in turn, be used to compute the *true* variance of the estimator.

## 1.2 A few words about the main methods

Stratified simple random sampling (`stratsrs`) and PPS sampling with replacement (`ppswr`) are straightforward.

PPS systematic sampling (`ppss` and `ppssstrat`) has the great advantage that it is easy to implement. It also has the property that the inclusion probability of a unit is proportional to its size. Thus it is a type of so-called $\pi$ps sampling, i.e., a unit's inclusion probability $\pi_i$ is proportional to its size. Like simple systematic sampling, the PPS version has the disadvantage that there is no variance estimator for it.

Sampford's method (`sampford`) has several desirable properties. It is a $\pi$ps scheme, it is relatively easy to implement, it selects units without replacement, and since joint inclusion probabilities can be computed explicitly (`sampfordpi`), variances and variance estimators are available. The version of Sampford's method implemented in this package is the one described briefly by Cochran (1977, page 262-263).

*Remark.* Note that in PPS sampling with replacement (`ppswr`), the probability of selecting a given unit on any given draw is proportional to its size, but the overall inclusion probability is not, i.e., it is not a $\pi$ps sampling scheme (unless the sample size is one).

# 2 Functions

## 2.1 stratsrs: Stratified simple random sampling

We begin with a function that performs stratified simple random sampling. The function `stratsrs` takes a vector of stratum indicators and a vector of sample sizes (one sample size per stratum) as input. The stratum indicators must be sorted, or at least grouped. For example, the vectors 1,1,1,1,2,2,2,3,3,3,3,3,8,8,8,8 and 2,2,2,1,1,1,1,3,3,3,3,3,8,8,8,8 are both acceptable. These vectors correspond to four strata. Stratum 1 has four units, stratum 2 has three units, and so on. Note that there is no stratum 4, 5, 6 or 7, i.e., the stratum indicators are just labels. Thus they need not be numbers. An example of a vector of sample sizes is 2,3,1,2, which corresponds to selecting 2 units from stratum 1, 3 from stratum 2, 1 from stratum 3 and 2 from stratum 8. An example of the use of `stratsrs` is the following.

```
stratum <- c(1,1,1,1,2,2,2,3,3,3,3,3,8,8,8,8)
nh <- c(2,1,3,2)
stratsrs(stratum,nh)
```

This returns the indices of the units in the sample. It is often convenient to keep all the population data, including stratification variables, in a data frame. If the population data are kept in the data frame `pop`, then a frame containing just the units in the sample is obtained using

```
samplesubframe <- pop[stratsrs(pop$strat,nh),]
```

where the data frame `pop` includes a stratum indicator variable called `strat`. This has the advantage that `samplesubframe` retains all the variables from the population data frame.

To illustrate a typical use of the stratified simple random sampling function with real data, here is an example using the California census data. We treat the data as a population of places, which have been placed in four strata, from which we would like to select a stratified random sample. For this example, we ignore the county indicator. First, let's select the sample.

```
table(calif$stratum)    # take a look at the size of each stratum
nh <- c(40,50,40,60)    # these are the sample sizes we want
calif<-calif[order(calif$stratum),] # sort by stratum
califsample <- calif[stratsrs(calif$stratum,nh),]
```

Now, let's compare the population mean and sample mean by stratum for the first variable, which gives the total population for each place.

```
aggregate(calif$popul,list(calif$stratum),mean)
aggregate(califsample$popul,list(califsample$stratum),mean)
```

Better still, let's get summaries for the four population variables for both the sample and the population.

```
by(califsample[,2:5],califsample$stratum,summary)
by(calif[,2:5],calif$stratum,summary)
```

## 2.2 ppswr

The function `ppswr` selects a sample of units *with* replacement, with the probability of selection of each unit proportional to its size. The function call is `ppswr(sizes,n)`, where `sizes` is a vector of the sizes of the population units and `n` is the sample size. We will use the California data to illustrate this function as well. For simplicity, we will select a sample of five units from the first county only.

```
county <- calif[calif$county==1,]   # get the units in county 1
countysample<-county[ppswr(county$population,5),] # select 5 units
```

## 2.3 ppss

The function `ppss` selects a sample of units using PPS systematic sampling. Under this approach, a random starting point is chosen, and it determines the units that fall in the sample. It has the property that if a unit in the population is very large, then it is selected with certainty and can even be selected more than once. The function `sizesok` can be used to check for this situation.

We will continue to use the single county used in the `ppswr` example to illustrate the `ppss` function.

```
county <- calif[calif$county==1,]   # get the units in county 1
countysample<-county[ppss(county$population,5),] # select 5 units
```

To check whether any units are so big that they will be selected with certainty, we do the following.

```
sizesok(county$popul,5) # if we select five units?
sizesok(county$popul,4) # or four units?
sizesok(county$popul,3) # or three units?
```

Note that the function returns the number of "bad" units, which may be zero.

## 2.4 ppssstrat

The function `ppssstrat` (note the three ses in a row) selects a PPS sample in each stratum by simply applying the `ppss` function to each stratum in turn. The input file must be sorted by stratum (or at least grouped by stratum). To illustrate this function, we will use the county-level aggregation of the California data as our population. The 57 counties are grouped into four size strata. We will select 5, 4, 3 and 2 units in the four strata, respectively, with 5 corresponding to the sample size in the stratum of very small counties, and so on.

```
califc<-califcty[order(califc$stratum),]   # sort by stratum
table(califc$stratum) # how big is each stratum?
csample <- califc[ppssstrat(califc$popul,califc$stratum,c(5,4,3,2)),]
```

As in the previous example, we could have also checked if any units are "too big". For example, for stratum 4, we would use:

```
str4<-califc[califc$stratum==4,]
sizesok(str4$popul,2) # is it ok to select two units?
sizesok(str4$popul,3) # how about three?
```

To perform *randomized* PPS systematic sampling, simply apply the function `permuteinstrata`, described below, to the population before selecting the sample.

## 2.5 sampford and sampfordpi

The function `sampford` select a PPS sample without replacement using Sampford's method. The function `sampfordpi` computes the corresponding inclusion $\pi_i$ and joint inclusion $\pi_{ij}$ probabilities. To illustrate these functions, we will use the data for the first county only, as we did for `ppswr`. Unlike the

previous functions, the Sampford ones incorporate a check for units that are too big and aborts if such units are found. First, we illustrate the use of `sampford` to select a sample.

```
county <- calif[calif$county==1,]
sampford(county$popul,5)  # what if n is too big?
samplesamp<-county[sampford(county$popul,3),]
```

The `sampford` function calls the `ppswr` and `pps1` functions, as well as the `sizesok` function.

The `sampfordpi` function returns a matrix with the inclusion probabilities $\pi_i$ along the diagonal and the joint inclusion probabilities $\pi_{ij}$ as off-diagonal elements. Of course, $\pi_{ij} = \pi_{ji}$.

```
piij<-sampfordpi(county$popul,3)
diag(piij)       # these are the inclusion probabilities
wt<-1/diag(piij) # these are the weights to use for estimation
```

## 2.6 Miscellaneous utility functions

We include several utility functions.

### 2.6.1 sizesok

The `sizesok` function was illustrated in the `ppss` and `ppssstrat` descriptions.

### 2.6.2 stratumsizes

Given a vector of stratum indicators, the `stratumsizes` function returns the number of units in each stratum. For example,

```
stratumsizes(c(1,1,1,1,2,2,2,3,3,3,3,3,8,8,8,8))
```

returns 4 3 5 4. As a second example, we apply the function to the county-level California data:

```
stratumsizes(califc$stratum)
```

returns the number of units in each stratum. `stratumsizes` simply converts the output of a call to `table` into a vector.

### 2.6.3 permuteinstrata

The function `permuteinstrata` is used to randomize the order of elements within each stratum. It takes the vector of stratum sizes as input and returns the permuted indices within each stratum. Thus, if there are three strata of size 9, 10 and 10 units respectively, then

```
permuteinstrata(c(9,10,10))
```

returns the permuted indices within each stratum. A sample output is

2 4 1 5 9 3 6 8 7 15 18 11 19 12 17 13 16 10 14 22 26 23 21 29 25 24 28 27 20

To illustrate further, let's continue the example from the `stratumsizes` function above:

```
permuteinstrata(stratumsizes(califc$stratum))
```

returns a vector of permuted indices for the California county-level data.

This function is useful for selecting a sample from a stratified population using *randomized* PPS systematic sampling.

### 2.6.4 pps1

The `pps1` function simply selects one unit from a population, with the selection probability proportional to the size of each unit. The input is the vector of unit sizes. `pps1` is used by the `sampford` function.

# References

Cochran, W.G. (1977). *Sampling Techniques.* John Wiley and Sons, New York.

Sampford, M.R. (1967). On sampling without replacement with unequal probabilities of selection. *Biometrika*, 54, 499-513.

200307