Introduction to the **mzR** package in R

Angelika Meraner, Statistics Austria

Revised: May 30, 2016

Contents

1 Introduction

The **mzR** package (mz is short for the German word Mikrozensus, the survey that incorporates the Austrian Labour Force Survey (LFS)) is an in-house R package at Statistics Austria (STAT) developed for computing estimates of sampling error, coefficient of variation and confidence limits for basically any estimator derived from LFS data. More specifically, it is also possible to estimate the sampling error of the net change of an indicator, e.g. year-to-year changes of annual averages.

2 Quick Introduction to Error Estimation with a Bootstrap Procedure

For estimating the error with the help of a bootstrap procedure, a certain number b of bootstrap samples are drawn from the original sample. Since the Austrian LFS is a rotating household sample without replacement drawn from a finite population, a "rescaled" bootstrap procedure as introduced by Rao and Wu 1988 is applied, leading to non-integer occurence frequencies. The bootstrap weights computed with these occurence frequencies are then calibrated using the same iterative proportional updating (IPU) method as for the original sampling weights. To account for

the rotational sample, only observations rotating into the sample are supplied with new bootstrap weights while the bootstrap weights pertaining to the remaining part of the sample, i.e. the overlap, are reutilized. To determine the standard error and the approximate 95% confidence interval of the population estimate of some population parameter we use the b bootstrap weights to compute said estimator b times. The standard deviation of these b computed estimates is then the estimated standard error while the 2.5% and 97.5% quantiles correspond to the lower and upper boundaries of the approximate 95% confidence interval. For a more detailed description please refer to Meraner et al. (accepted for publication).

3 Getting Started

We install the **mzR** package with:

```
> install.packages("...path.../mzR_...packageVersion....zip", repos = NULL,
+                  type="binary")
```

At the beginning of our R session we load the package:

```
> library("mzR")
```

Call up the help documentation for the **mzR** package

```
> help(package=mzR)
```

or for a specific function contained in the package, e.g. `ImportData()`:

```
> ?ImportData
```

4 Read Files

4.1 STAT users

To import the Austrian LFS data we are interested in as well as the corresponding bootstrap weights which are stored separately we use the following function which is based on STAT file management and only works in-house:

```
> ImportData(year = NULL, quarter = NULL, comp_diff_lag = NULL,
+   from = NULL, to = NULL, hh = FALSE, families = FALSE,
+   whichVar = NULL, nbw = NULL, weightDecimals = 2)
```

To find out more about this function and its parameters we can open the help page with `?ImportData`, as mentioned above.

Let's say we want to compute certain yearly estimates, e.g. for the year 2014. To do this we have to read in the data into first:

```
> dat <- ImportData(year=2014)
```

If we only want to import quarterly data, e.g. the 4th quarter 2014, we have to set the parameter `quarter` as well:

```
> dat <- ImportData(year=2014,quarter=4)
```

To compute net changes of certain indicators, e.g. year-to-year changes from 2013 to 2014 we have to set the parameter `comp_diff_lag` to the desired time lag (in years):

```
> dat <- ImportData(year=2014,comp_diff_lag=1)
```

If we are interested in changes between quarterly time points, e.g. changes from the 4th quarter 2013 to the 4th quarter 2014, we set the the parameter `comp_diff_lag` to the desired time lag (in quarters):

```
> dat <- ImportData(year=2014,quarter=4, comp_diff_lag=4)
```

If we are only interested in estimating household indicators we can restrict the imported data to the reference persons of the households (`bstell=0`) by setting the parameter `hh`:

```
> dat <- ImportData(year=2014,quarter=4, comp_diff_lag=1, hh=TRUE)
```

To reduce memory usage, which might be necessary for certain data and certain computers, and if we do not need all the variables contained in the data, we can set the parameter `whichVar` to the selection of variables we actually want to work with, e.g. age, gender and region:

```
> dat <- ImportData(year=2014,quarter=4, whichVar=c("balt","bsex","xnuts2"))
```

If we are only interested in estimating indicators concerning families, we can restrict the imported data to the reference persons of families (`xfstell=1`) with the parameter `families`:

```
> dat <- ImportData(year=2014,quarter=4, comp_diff_lag=1, families=TRUE)
```

Sometimes we would like things to go faster for testing purposes. In such cases we can reduce the number of imported bootstrap weights to e.g. 5:

```
> dat <- ImportData(year=2014,quarter=4, comp_diff_lag=1, nbw=5)
```

If we do not want to import a specific quarter or year but a time span, we use the parameters `from` and `to`:

```
> dat <- ImportData(from=c(2011,1),to=c(2014,4))
```

4.2 External users

For external users, i.e. users of LFS data who have no access to the STAT file management, the function `IndivImportData()` does the trick. At the moment, this function works best for LFS data but it can also be applied to other data sets with certain limitations.

```
> IndivImportData(curr_inFile, curr_inFile_bw, prev_inFile = NULL,
+   prev_inFile_bw = NULL, whichVar = NULL, mergeBy = "asbhh", nbw = NULL,
+   bwNames = NULL, weightName = "gew1", weightDecimals = 2)
```

LFS data provided by Statistics Austria to external users (so called dg8-files) can be read in by specifying the file paths.

For quarterly data we only have to specify two paths, the path of the LFS data (`curr_inFile`) and the path of the bootstrap weights (`curr_inFile_bw`):

```
> my_dg8_file <- c("...path.../dg8.mz2014Q1.sav")
> my_bootstrapweights <- c("...path.../mz2_2014q1_bootweights.csv.gz")

> dat <- IndivImportData(curr_inFile=my_dg8_file, curr_inFile_bw=my_bootstrapweights)
```

For yearly data, the parameters `curr_inFile` and `curr_inFile_bw` are character vectors containing all the paths of the quarterly files needed to construct the yearly data set:

```
> my_dg8_files <- c("...path.../dg8.mz2014Q1.sav",
+                   "...path.../dg8.mz2014Q2.sav",
+                   "...path.../dg8.mz2014Q3.sav",
+                   "...path.../dg8.mz2014Q4.sav")
> my_bootstrapweights <- c("...path.../mz2_2014q1_bootweights.csv.gz",
+                          "...path.../mz2_2014q2_bootweights.csv.gz",
+                          "...path.../mz2_2014q3_bootweights.csv.gz",
+                          "...path.../mz2_2014q4_bootweights.csv.gz")
```

```
> dat <- IndivImportData(curr_inFile=my_dg8_files, curr_inFile_bw=my_bootstrapweights)
```

If we are interested in estimating the changes of indicators for certain time points we have to specify `prev_inFile` and `prev_inFile_bw` as well. Furthermore, `IndivImportData()` also supports the parameters `whichVar` and `nbw` as introduced above, in Section 4.1. To import data other than LFS dg8 data we might have to specify the names of the weights (`weightName`) and the bootstrap weights (`bwNames`) as they will probably be different from the LFS naming conventions. Also the variable(s) used for merging the bootstrap weights to the survey data will probably need to be set in the parameter `mergeBy`.

4.3 Import Additional Data

If we want to read in additional data and merge it to data already imported with `ImportData()` or `IndivImportData()` we can use the following function:

```
> ImportAndMerge(x, curr_inFile, prev_inFile = NULL, mergeBy = "asbper",
+   whichVar = NULL)
```

As an example:

```
> my_additional_data <- "...path.../additional.data.mz2014q4.sav"
> dat <- ImportAndMerge(dat, curr_inFile=my_additional_data)
```

5 Variable Labels

The Austrian LFS data usually comes with variable labels. We can get retrieve these labels with the function `GetLabels()`, e.g. the labels for the regional variable `xnuts2` (Austrian federal states) of data set `dat` are queried via:

```
> GetLabels(dat,var="xnuts2")
```

```
$xnuts2
$xnuts2$label
                        xnuts2
"NUTS 2 Gebiete (Bundesländer)"
$xnuts2$value.labels
      Vorarlberg            Tirol         Salzburg   Oberösterreich
              34               33               32               31
      Steiermark          Kärnten             Wien Niederösterreich
              22               21               13               12
      Burgenland
              11
```

6 Estimates for Categorical Variables

Estimates (`est`) of group sizes and group rates for categorical variables as well as the corresponding estimated standard errors (`sd`), coefficients of variation (`cv`) and lower/upper confidence limits (`cil_2.5%`, `ciu_97.5%`) are computed with the functions `GroupSize()` and `GroupRate()`.

6.1 `GroupSize()`

The function contains the following parameters:

```
> GroupSize(x, TFstring = NULL, each = NULL, thousands_separator = TRUE, digits = 2)
```

Example: The number of unemployed people (`xerwstat==2`) aged 15-74 (`balt>=15 & balt<=74`) for the 4th quarter 2013 and the 4th quarter 2014 as well as the changes between these two time points is computed as:

```
> dat <- ImportData(year=2014, quarter=4, comp_diff_lag=4)

> GroupSize(dat,TFstring="xerwstat==2 & balt>=15 & balt<=74")

                       est        sd   cv   cil_2.5%  ciu_97.5%
2014q4          242,272.59  7,226.91 0.03 228,620.00 256,200.91
2013q4          231,594.08  6,801.72 0.03 217,854.01 245,024.14
Absolute change  10,678.51 10,109.26 0.95  -8,738.42  31,358.08
Relative change       4.61      4.49 0.97      -3.68      13.69
```

6.2 GroupRate()

The function contains the following parameters:

```
> GroupRate(x, TFstring, TFstring2 = NULL, each = NULL, byeach = TRUE,
+   thousands_separator = TRUE, digits = 2)
```

Example: The unemployment rate, i.e. the number of unemployed (`xerwstat==2`) by the number of unemployed and employed (`xerwstat%in%c(1,2)`) aged 15-74 for the 4th quarter 2013 and the 4th quarter 2014 as well as the changes between these two time points is computed as:

```
> dat <- ImportData(year=2014, quarter=4, comp_diff_lag=4)

> GroupRate(dat,TFstring="xerwstat==2 & balt>=15 & balt<=74",
+           TFstring2="xerwstat%in%c(1,2) & balt>=15 & balt<=74")

                 est   sd   cv cil_2.5% ciu_97.5%
2014q4          5.57 0.16 0.03     5.27      5.88
2013q4          5.36 0.15 0.03     5.05      5.66
Absolute change 0.21 0.22 1.09    -0.22      0.65
Relative change 3.85 4.29 1.11    -3.85     12.46
```

7 Estimates for Numerical Variables

Estimates (`est`) of totals and means for numerical variables as well as the corresponding estimated standard errors (`sd`), coefficients of variation (`cv`) and lower/upper confidence limits (`cil_2.5%`, `ciu_97.5%`) are computed with the functions `Total()` and `Mean()`.

The functions contain the following parameters

```
> Total(x, TFstring = NULL, each = NULL, var, negativeZero = TRUE,
+   thousands_separator = TRUE, digits = 2)
> Mean(x, TFstring = NULL, each = NULL, var, negativeZero = TRUE,
+   thousands_separator = TRUE, digits = 2)
```

7.1 Total()

Example: The number of hours worked (variables `estund` and `dtstd`) by people aged 15-74 for the 4th quarter 2013 and the 4th quarter 2014 as well as the changes between these two time points is computed as:

```
> dat <- ImportData(year=2014, quarter=4, comp_diff_lag=4)

> Total(dat, TFstring="xerwstat==1 & balt>=15 & balt<=74", var="estund*13 + dtstd*13")

                         est            sd   cv         cil_2.5%    ciu_97.5%
2014q4          1.743185e+09  7,938,445.15 0.00 1,728,431,889.89 1.758507e+09
2013q4          1.727835e+09  7,908,361.20 0.00 1,713,347,447.18 1.743993e+09
Absolute change 1.535035e+07 11,036,711.01 0.72    -5,811,137.99 3.870106e+07
Relative change 8.900000e-01          0.64 0.72            -0.33 2.250000e+00
```

7.2 Mean()

Example: The average number of hours worked (variables estund and dtstd) by gender (Variable bsex) of people aged 15-74 for the 4th quarter 2013 and the 4th quarter 2014 as well as the changes between these two time points is computed as:

```
> dat <- ImportData(year=2014, quarter=4, comp_diff_lag=4)
```

```
> Mean(dat, TFstring="xerwstat==1 & balt>=15 & balt<=74", var="estund + dtstd",
+      each="bsex")
```

```
$bsex_1
                  est   sd cv cil_2.5% ciu_97.5%
2014q4          37.27 0.18  0    36.94     37.64
2013q4          37.02 0.18  0    36.68     37.34
Absolute change  0.26 0.26  1    -0.20      0.75
Relative change  0.69 0.69  1    -0.54      2.03
$bsex_2
                  est   sd   cv cil_2.5% ciu_97.5%
2014q4          27.42 0.17 0.01    27.11     27.77
2013q4          27.39 0.17 0.01    27.08     27.72
Absolute change  0.03 0.24 7.31    -0.43      0.50
Relative change  0.12 0.88 7.31    -1.54      1.81
```

8 Export Results

If we want to save the results from Sections 6 and 7 in a CSV-file we can do so with the following function:

```
> export(x, outFilePath = getwd(), outFileName = NULL)
```

As an example, we export our result to the CSV-file "D:/myTestFile.csv":

```
> i_want_out <- Mean(dat,TFstring="xerwstat==1 & balt>=15 & balt<=74",
+                                  var="estund + dtstd", each="xnuts2")
> export(i_want_out, outFilePath="D:", outFileName="myTestFile")
```

If no path is specified, the CSV-file is generated in the working directory which can be determined with getwd(). If no file name is specified, a default file name depending on the underlying function from Sections 6 and 7 is used.

9 Memory

To make **mzR** run more smoothly when using large data sets, try to restrict the number of (large) objects you keep/load in/to your work environment (you can list the objects in your work environment with ls()) and maybe do some garbage collection (gc()). It might also help to reduce the number of variables in the imported data to the ones you needed for the analysis you are doing. This is accomplished e.g. with the parameter whichVar in the import functions ImportData() and IndivImportData(). To increase computation speed for test purposes you can also set a limit to the number of bootstrap weights, i.e. with the parameter nbw.

References

A. Meraner, D. Gumprecht, and A. Kowarik. Weighting procedure of the austrian microcensus using administrative data. *Austrian Journal of Statistics*.

J. N. Rao and C. Wu. Resampling inference with complex survey data. *Journal of the american statistical association*, 83(401):231–241, 1988.