Computer Workshop

Introduction to Small Area Estimation

This computer workshop presents data analyses using some of the methods we presented today. It covers direct estimation, model-based estimation of linear statistics using unit-level and area-level models and model-based estimation of non-linear statistics using the Empirical Best Predictor method. Variance and Mean Squared Error (MSE) estimation are also presented. No prior knowledge of R is assumed. The computer workshop follows the same structure as the lecture-based sessions.

Sections 1-3 include the steps for installing R or R-Studio and R packages on your personal computers. For the purposes of this workshop, these sections can be ignored and hence you can go directly to Section 4.

1 Introduction to R

R is a free, powerful object oriented software for performing statistical analyses. We have already written the code for you. You can use the code step by step in order to illustrate some aspects of small area estimation. Detailed documentation about the packages we use is available via R-Cran

2 How to download R and R-Studio

1 Go to **https://cloud.r-project.org/**

2 Click on operating system you are using Windows or Mac

3 Follow the installation process. R will be automatically installed and a shortcut will be created.

Alternatively you can install R-Studio, which offers a more integrated environment to work with. R-studio can be downloaded from the following webpage:
https://www.rstudio.com/products/rstudio/download/

3 How to open R and install R packages

The installation process automatically creates an R or an R-studio shortcut. Double click this icon to open the R environment.
A package can be installed in R as follows:

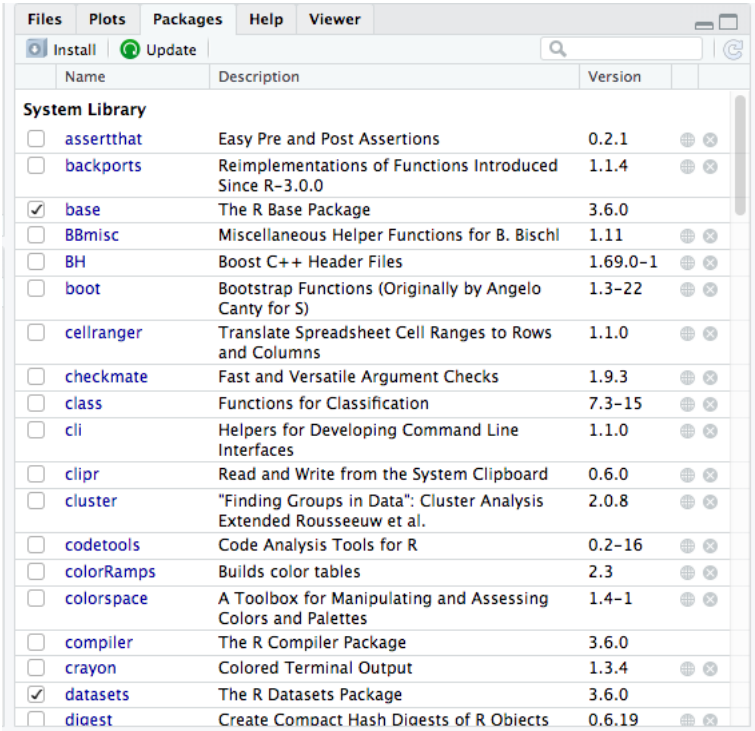
1 Open R

2 Go to Packages and then select Install Package(s)

3 Select a mirror from the list and click OK

4 Select the appropriate package from the list and click OK. This will install the package.

In R-studio a package can be installed by using the tabs on the bottom right panel and by clicking the packages tab, followed by install and by selecting the required package.



Once the package has been installed, you need to load it for carrying out the analyses. To do this type in the R console library(`name`) and press *enter*. The package has now been loaded and is ready for use.

4 Working with the files on your usb

Insert your course USB stick and navigate to the root directory. Double-click the R-Studio shortcut. When running R-Studio for the first time you should hold down the control key. This is done to prevent an already installed version of R on your machine being picked up.

You will then be presented with a dialogue asking you to choose an R installation, in which you need to click **Browse** and point to the version of R on the usb stick.

5 Small Area Estimation - R packages and Data

For the purposes of this computer workshop, the R source code and the packages have been saved on your usb. The R source code has been saved in file **Computer workshop_Bristol.R**. Launch R-Studio followed by Open File and select **Computer workshop_Bristol.R**.
First we need to set the working directory where the R packages and other files are saved. To do this we use the *setwd* command. Run the following command available from your R script file,

$$setwd(paste0(R.home(), \text{“/../}workshop\text{”}))$$

The following packages will be used: library(`laeken`), library(`emdi`), library(`colorRamps`), library(`gridExtra`), library(`simFrame`), library(`dplyr`), library(`sae`), library(`ggplot2`). You can load the necessary packages by running the following commands.

```
# Loading libraries and the data
library(laeken)
library(emdi)
library(colorRamps)
library(gridExtra)
library(simFrame)
library(dplyr)
library(sae)
library(ggplot2)
```

In this computer practical we will be working with EU-SILC data that are already part of the packages that you installed. Running the commands that appear in the figure below will load the datasets.

```
# Datasets

data("eusilc")
data("eusilcP")
data("eusilcA_pop")
data("eusilcA_smp")
```

5.1 SAE - Direct estimation

In this section we present approaches to direct estimation for small areas. To start with we use package *laeken* and we provide examples for different indicators. We start by computing the at-risk-of-poverty indicator at national level. This can be done by using command *arpr*, specifying the income variable, *eqIncome*, the survey weights, *rb050* and the data we are using, in this case the EU-SILC data. You will obtain the estimates by running the following command available from your R script file,

$$hcr\_national \leftarrow arpr(\text{“}eqIncome\text{”}, weights = \text{“}rb050\text{”}, data = eusilc)$$

Type the name of the object *hcr_national* and press *enter* to see the results. This will report the percentage of households below the poverty line (*value)* and the poverty line (*threshold*).

Next we will learn how to produce estimates of the at-risk-of -poverty indicator at subnational levels. This can be achieved by using the previous command by now using argument *breakdown* that specifies the geographical level we are interested in. For this example we will use what is known as the NUTS 2 geographical level. Run the following command available from your R script file,

$$hcr\_nuts2 \leftarrow arpr(\text{“}eqIncome\text{”}, weights = \text{“}rb050\text{”}, breakdown = \text{“}db040\text{”}, data = eusilc)$$

The results are saved in object *hcr_nuts2* and can be retrieved by typing the name of the object and pressing *enter*.

```
> hcr_nuts2
Value:
[1] 14.44422

Value by domain:
        stratum    value
1    Burgenland 19.53984
2     Carinthia 13.08627
3 Lower Austria 13.84362
4      Salzburg 13.78734
5        Styria 14.37464
6         Tyrol 15.30819
7 Upper Austria 10.88977
8        Vienna 17.23468
9    Vorarlberg 16.53731

Threshold:
[1] 10859.24
```

Estimates of the variance of the at-risk-of-poverty indicator for each NUTS 2 geography can be obtained by using command *variance*. The command requires the specification of the variable of interest for our analysis, in this case *eqIncome*, the survey weights (here *rb050*), the variable that specifies the geographical breakdown (here *db040*) and the variable(s) that define the survey design. In this case the design is assumed to be stratified by NUTS 2 and hence the *db040* variable is used. However, more complex designs are allowed (see the *laeken* package documentation). Finally, we need to specify the indicator for which we wish to estimate the variance (recall that in this case the arpr estimates are saved in object *hcr_nuts2*), the type of bootstrap variance estimation (in this case we use naive bootstrap but other bootstrap types are available), and the number of bootstrap replications (here $R = 500$). Run the following command available from your R script file,

$$hcr\_var \leftarrow variance(\text{“}eqIncome\text{”}, weights = \text{“}rb050\text{”}, breakdown = \text{“}db040\text{”}, design = \text{“}db040\text{”},$$

$$data = eusilc, indicator = hcr\_nuts2, bootType = \text{“}naive\text{”}, seed = 123, R = 500)$$

The results are saved in object *hcr_var* and can be retrieved by typing the name of the object and pressing *enter*. The output gives estimates of the variance and confidence intervals.

```
Value by domain:
        stratum    value
1    Burgenland 19.53984
2     Carinthia 13.08627
3 Lower Austria 13.84362
4      Salzburg 13.78734
5        Styria 14.37464
6         Tyrol 15.30819
7 Upper Austria 10.88977
8        Vienna 17.23468
9    Vorarlberg 16.53731

Variance by domain:
        stratum       var
1    Burgenland 2.8599386
2     Carinthia 1.3806124
3 Lower Austria 0.4508332
4      Salzburg 1.4093275
5        Styria 0.5591159
6         Tyrol 1.0308416
7 Upper Austria 0.3306110
8        Vienna 0.5629735
9    Vorarlberg 1.9744992

Confidence interval by domain:
        stratum     lower    upper
1    Burgenland 16.203852 22.82716
2     Carinthia 10.720872 15.37713
3 Lower Austria 12.501693 15.14681
4      Salzburg 11.589741 16.28962
5        Styria 12.982625 16.00579
6         Tyrol 13.487444 17.46962
7 Upper Austria  9.777863 12.01431
8        Vienna 15.728719 18.75112
9    Vorarlberg 13.854050 19.37823

Threshold:
[1] 10859.24
```

In some applications we may want to specify domains defined by the cross-classification of a geographic variable and a demographic variable. In the example below the breakdown variable is defined by the cross-classification of NUTS2 by gender. The new name for the domain variable is *genderregion* and can be created by running the following command available from your R script file,

$$eusilc\$genderregion \leftarrow interaction(eusilc\$db040, eusilc\$rb090)$$

The at-risk-of-poverty indicator for the newly defined variable can then be computed by running the following command available from your R script file,

$$hcr\_nuts2\_gender \leftarrow arpr(\text{“}eqIncome\text{”}, weights = \text{“}rb050\text{”}, data = eusilc, breakdown = \text{“}genderregion\text{”})$$

The results are saved in object *hcr_nuts2_gender* and can be retrieved by typing the name of the object and pressing *enter*.

```
> hcr_nuts2_gender
Value:
[1] 14.44422

Value by domain:
                stratum     value
1       Burgenland.male 17.414524
2        Carinthia.male 10.552149
3    Lower Austria.male 11.348283
4         Salzburg.male  9.156964
5           Styria.male 11.671247
6            Tyrol.male 12.857542
7    Upper Austria.male  9.074690
8           Vienna.male 15.590616
9       Vorarlberg.male 12.973259
10    Burgenland.female 21.432598
11     Carinthia.female 15.392924
12 Lower Austria.female 16.372949
13      Salzburg.female 17.939382
14        Styria.female 16.964539
15         Tyrol.female 17.604861
16 Upper Austria.female 12.574206
17        Vienna.female 18.778813
18    Vorarlberg.female 19.883637

Threshold:
[1] 10859.24
>
```

The *laeken* package can be used to compute other indicators for example, the quintile share ratio, which can be used to quantify income inequality. This can be done by using the *qsr* command. The commands below compute the quintile share ratio for each NUTS 2 area and plot the estimates on a map. Run the following commands from your R script file to produce the results.

```
# Produce estimates of Quintile Share Ratio and visualise on a Map


Karte <- readRDS("AUT_adm1.rds")
Karte@data$NAME_1 <- c("Burgenland","Carinthia","Lower Austria","Upper Austria","Upper Austria",
                                "Salzburg","Styria","Tyrol","Vorarlberg","Vorarlberg","Vienna")

# Produce the map
est_qsr<-qsr("eqIncome", weights = "rb050", data = eusilc,breakdown="db040")
ind <- match(Karte@data$NAME_1,est_qsr$strata) # Matching
Karte@data$QSR <- est_qsr$valueByStratum[ind,2] # Add estimator to map

# Define the color of the maps
ramp <- colorRamp(c("green","yellow", "red","black"))
# Plot
#pdf(file="Direct_austria_qsr.pdf", width=5, height=5, family="Times")
spplot(Karte,c("QSR"),names.attr=c("QSR"),main="Quintile Share Ratio",
              col.regions=c(rgb(ramp(seq(0, 1, length = 200)), max = 255)),par.strip.text=list(lines=1.3),
              at=c(seq(min(Karte@data$QSR),max(Karte@data$QSR),length.out=100),Inf),colorkey=TRUE)
#dev.off()
```

6 Model-based methods - Methods for estimating small area averages

In this section we will learn how to produce small area estimates of averages using two popular model-based methods, a) an area-level model, namely the Fay-Herriot model and b) a unit-level model, namely the Battese-Harter-Fuller model. We use synthetic EU-SILC data and the aim is to produce estimates of average income for NUTS 2 areas in Austria. Here we assume that unit-level data are available and we illustrate how unit-level data can be transformed to area-level for fitting the Fay-Herriot model.

6.1 Area-level analyses - The Fay-Herriot model

Before being able to estimate the Fay-Herriot model, we need to prepare aggregate (area-level) data. First, we use the unit-level synthetic EU-SILC population data to select a sample of households. This is achieved using command *sample*. Then the *sae* package is used for computing direct estimates of average income and corresponding estimates of the variance in each NUTS2 area in Austria. This is achieved using command *direct* in package *sae* and is an alternative to using package *laeken*. These are the basic inputs of the sampling part of the Fay-Herriot model. Run the following commands,

```
# we filter by the main income holder per household to get household data
eusilcP_HH <- eusilcP[eusilcP$main, ]

# draw random rows in order to receive a sample from the population data set
set.seed(2)
sample_id <- sample(1:25000, 400, replace = FALSE, prob = NULL)
# Receive sample on houehold level corresponding to population data
eusilcS_HH <- eusilcP[sample_id, ]

# How are the sample observations distributed in the regions?
summary(eusilcS_HH$region)
# Direct estimation of mean using sae-package
fit_direct <- direct(y = eqIncome,dom = region, data = eusilcS_HH, replace=T)
```

As mentioned before, here we assume that we have access to unit-level data. If area-level data are available, you can skip this part. The following commands transform the unit level data to area-level data. The area-level data are saved in object *data_frame*.

```
# Aggregation of the covariates on region level
eusilcP_HH_agg<-tbl_df(eusilcP_HH) %>%
  group_by(region) %>%
  summarise(hy090n = mean(hy090n)) %>%
  ungroup() %>%
  mutate(Domain = fit_direct$Domain)

data_frame <- left_join(eusilcP_HH_agg, fit_direct, by = "Domain") %>%
  mutate(var = SD^2)
```

We are now ready to estimate the Fay-Herriot model. This can be achieved by using the R package *sae* and function *mseFH*. This will produce both point estimates and Mean Squared Error (MSE) estimates using the Prasad-Rao analytic MSE estimator. The command requires the specification of the linking model (in this example we regress the direct estimates on one explanatory variable, namely *hy090n*), and the specification of the variance of the direct estimates. Run the following command from your R script.

$$fit\_FH \leftarrow mseFH(formula = Direct \sim hy090n, vardir = var, data = as.data.frame(data\_frame))$$

The following command computes estimates of coefficients of variation (CVs) of the small area estimates of average income under the Fay-Herriot model. Run the command, which is available from your R script file,

$$FH\_CV \leftarrow 100 * sqrt(fit\_FH\$mse)/fit\_FH\$est\$eblup$$

Finally, the results are saved in object *results* as follows. You can see the results by typing *results* and pressing *enter*.

$$results \leftarrow data.frame(fit\_direct, FH\_est = fit\_FH\$est\$eblup, FH\_CV)$$

6.2 Unit-level analyses - The Battese-Harter-Fuller model

We will now use the unit-level data to estimate the Battese-Harter-Fuller model using package *sae*. Before fitting the model, recall that when the aim is to produce estimates of small area averages, predictions can be formed by combining the estimates of the model parameters with area-averages of the explanatory variables. We will return to discuss this point in the next section of this computer workshop. The commands below create the new dataset with the area averages of the covariates saved in object *Xmean*. Run the following commands for preparing the data and check what is saved in object *Xmean* by typing the name of the object and pressing *enter*.

```
# Aggregation of the covariates on region level
eusilcP_HH_agg<-tbl_df((eusilcP_HH))%>%group_by((region))%>%summarise(py010n=mean(py010n),
                                                                      py050n=mean(py050n),
                                                                     hy090n=mean(hy090n),n=n())%>%
  ungroup()%>%mutate(Domain=fit_direct$Domain)
data_frame<-left_join(eusilcP_HH_agg,fit_direct,by="Domain")%>%mutate(var=SD^2)
data_frame<-as.data.frame(data_frame)

Xmean<-data.frame(region=data_frame[,1],hy090n=data_frame$hy090n,py010n=data_frame$py010n,py050n=data_frame$py050n)
Popsize<-data.frame(region=data_frame[,1],n=data_frame$n)
```

Having prepared the dataset, we will now estimate the Battese-Harter-Fuller model and produce small area estimates of average income for NUTS2 areas in Austria. This is achieved using command *eblupBHF*. MSE estimation is implemented using command *pbmseBHF*. The first part specifies the regression model (income modelled as a function of covariates), and the specification of the domains (used for the random effects specification). Note that both commands require the specification of the area-specific population averages of the covariates using argument *meanxpop*. Finally, function *pbmseBHF* uses a bootstrap MSE estimator. Run the following commands and type *results* to see the output from fitting the Battese-Harter-Fuller model.

```
# Estimation of the Unit-level model (Battese-Harter-Fuller)
fit_EBLUP<-eblupBHF(formula=as.numeric(eqIncome)~py010n + py050n+hy090n,dom=region,data=eusilcS_HH,meanxpop=Xmean,popnsize=Popsize)

# MSE estimation of the Unit-level model
MSE_EBLUP<-pbmseBHF(formula=as.numeric(eqIncome)~py010n + py050n+hy090n,dom=region,data=eusilcS_HH,meanxpop=Xmean,popnsize=Popsize)

# Comparison with direct estimator
EBLUP_CV<-100*sqrt(MSE_EBLUP$mse$mse)/fit_EBLUP$eblup$eblup
results<-data.frame(fit_EBLUP$eblup,EBLUP_CV)
```

7 Model-based methods - Unit-level models for estimating non-linear indicators for small areas

In this last section we will learn how to obtain estimates of non-linear indicators for small areas using model-based methods and unit-level models. In particular, we will illustrate estimation with the Empirical Best Prediction (EBP) method. This method is implemented in R using the package *emdi*. Note that since our interest is in estimating non-linear indicators, access to unit-level population covariate data is needed. This is in contrast to the estimation of small area averages we discussed

in the previous section. EBP estimation is implemented using function *ebp*. The structure of the command requires the specification of the model, the population and sample datasets, the domains and the poverty threshold (used for example in estimating the at-risk-of-poverty rate). In addition, L denotes the number of Monte-Carlo iterations used in EBP, the argument *MSE* allows for MSE estimation, B specifies the number of bootstrap samples, and the argument *transformation* specifies the possible use of data-driven transformations.

Run the following commands for producing the EBP results. Here we use a model with a Box-Cox transformation for income and the target is to estimate median income in small areas in Austria. Other estimators are also available and to see these type *?estimators* to see the help file. MSE estimation uses 50 bootstrap samples. The *summary* command provides a model summary, similar to what is produced by function *lme* used for fitting multilevel models in R. The *plot* command produces various residual diagnostic plots. Finally, the results are saved in object *results*.

```
# EBP estimation
set.seed(1)

#EBP with Box-Cox Transformation and bootstrap MSE

emdi_model_box_cox <- ebp(fixed = eqIncome ~ gender + eqsize + cash +
                      self_empl + unempl_ben + age_ben + surv_ben + sick_ben + dis_ben + rent +
                      fam_allow + house_allow + cap_inv + tax_adj, pop_data = eusilcA_pop,
                    pop_domains = "district", smp_data = eusilcA_smp, smp_domains = "district",
                    threshold = 0.6*median(eusilcA_smp$eqIncome), transformation = "box.cox",
                    L = 50, MSE = TRUE, B = 50,custom_indicator =
                      list(my_max = function(y, threshold){max(y)},
                           my_min = function(y, threshold){min(y)}), na.rm = TRUE, cpus = 1)
summary(emdi_model_box_cox)
plot(emdi_model_box_cox)
#Output of the results
results=estimators(emdi_model_box_cox, indicator = "Median", MSE = TRUE)
```

One advantage of the *emdi* package, compared to other small area estimation packages, is its data visualisation tools. In this final part of the workshop we illustrate how the command *map_plot* can be used for producing a map of the median estimates. Please note that for doing so you need to have access to appropriate shape file. The file shape for this particular application is part of the *emdi* package. Run the following commands for producing the map of the estimates.

```
load_shapeaustria()


# Create a suitable mapping table to use numerical identifiers of the shape
# file
# First find the right order
dom_ord <- match(shape_austria_dis@data$PB, emdi_model_box_cox$ind$Domain)
# Create the mapping table based on the order obtained above
map_tab <- data.frame(pop_data_id = emdi_model_box_cox$ind$Domain[dom_ord],
                      shape_id = shape_austria_dis@data$BKZ)
# Create map plot for mean indicator - point and CV estimates but no MSE
# using the numerical domain identifiers of the shape file
map_plot(object = emdi_model_box_cox, MSE = FALSE, CV = FALSE,
         map_obj = shape_austria_dis, indicator = c("Median"),
         map_dom_id = "BKZ", map_tab = map_tab)
```