FACTAGE

STATISTICS AUSTRIA
The Information Manager

Editing EU-SILC UDB Longitudinal Data for Differential Mortality Analyses

SAS code and documentation

FACTAGE – WP 4.3

Tobias Göllner and Johannes Klotz

Statistics Austria

May 2018

FACTAGE is a project coordinated in the Joint Programming Initiative: More Years, Better Lives. The Austrian contributions, such as this, are funded by the Federal Ministry of Education, Science and Research

BMBWF
FEDERAL MINISTRY OF EDUCATION, SCIENCE AND RESEARCH
www.bmbwf.gv.at

Preface

This SAS code is a deliverable of the Fairer Active Ageing for Europe (**FACT**AGE) project. **FACT**AGE is exploring emerging inequalities associated with longer working lives. It is a Joint Programming Initiative - More Years Better Lives (MYBL) - project. Details on the project can be found on www.factage.eu . Details on the Joint Programming Initiative can be found on www.jp-demographic.eu.

The SAS code is free to use at your own risk. It is available online at https://github.com/TobiasGold/FACTAGE-method_Mortality.
Suggested citation: Göllner, T., and Klotz, J.: Editing EU-SILC UDB Longitudinal Data for Differential Mortality Analyses. SAS code and documentation. FACTAGE project report, May 2018.

Abstract

This SAS code extracts data from EU-SILC User Database (UDB) longitudinal files and edits it such that a file is produced that can be further used for differential mortality analyses. Information from the original D, R, H and P files is merged per person and possibly pooled over several longitudinal data releases. Vital status information is extracted from target variables DB110 and RB110, and time at risk between the first interview and either death or censoring is estimated based on quarterly date information.

Apart from path specifications, the SAS code consists of several SAS macros. Two of them require parameter specification from the user. The other ones are just executed. The code was written in Base SAS, Version 9.4.

By default, the output file contains several variables which are necessary for differential mortality analyses, such as sex, age, country, year of first interview, and vital status information. In addition, the user may specify the analytical variables by which mortality risk should be compared later, for example educational level or occupational class. These analytical variables may be measured either at the first interview (the baseline) or at the last interview of a respondent. The output file is available in SAS format and by default also in csv format.

Contents

Getting Started

Code Execution Outline

Required Input

(1): the raw csv files of EU-SILC UDB longitudinal data **plus** the respective SAS codes to import them into SAS.

OR

(2): the sas7bdat files of EU-SILC UDB longitudinal data.

Steps

1. Run "0_Load-Macros.sas".
2. Open "1_Application.sas":
3. Edit the line `%let silcpath = C:\path\to-your\library-folder;` to your desired directory.
4. If you want your output file to contain only specific countries, edit the line `%let clist='AT', 'BE',..., 'UK';` to only include your desired countries.
5. Option (1): If you have the raw csv files of EU-SILC UDB longitudinal data and codes that import them into SAS, edit the following lines which point to your import SAS codes:

   ```
   %let dincl = C:\path\to-your\D-file-codes;
   %let hincl = C:\path\to-your\H-file-codes;
   %let pincl = C:\path\to-your\P-file-codes;
   %let rincl = C:\path\to-your\R-file-codes;
   ```

   Note that the import codes must be split according to the four EU-SILC file types (D, H, P and R). An example of a typical import code provided by Eurostat is found in the Appendix. Make sure that the codes themselves produce the correctly named datasets (e.g. dfile7, rfile10, pfile13, etc.) and point to the appropriate file directory.

   OR

   Option (2): If you have existing SAS files of the EU-SILC UDB longitudinal data, edit the following lines which point to your files:

   ```
   libname dfiles "C:\path\to-your\d-file-data";
   libname hfiles "C:\path\to-your\h-file-data";
   libname pfiles "C:\path\to-your\p-file-data";
   libname rfiles "C:\path\to-your\r-file-data";
   ```

Make sure that the files are named correctly (e.g. hfile6, dfile10, rfile11, etc.). You have to use different folders for the four different file types.

Note: You have to decide on one of the two options (1) or (2). You can either delete the other option or comment it out using `/* code */`.

6. Choose your desired parameters for the `%createdata` macro. The parameters are:

   `yr_from=` type in the **release** year (minus 2000) of the earliest longitudinal data you want to use. The default value is `6`. (Note: The release year is the final survey year of longitudinal data, so 6 means that observations are available from 2003, 2004, 2005 and 2006).
   `yr_to=` type in the release year (minus 2000) of the latest longitudinal data you want to use. The default value is `15`. If you do not want to pool data over several longitudinal data releases, but use just one release, then `yr_from` and `yr_to` are the same.
   `fileoption=` is either `1` or `2`. Default = `1`. Depends if you have the raw csv files or sas7bdat files (see Option (1) or (2) in the previous point).
   `countries=` is either `(&clist)` or `all`. The first one has to be in brackets. The default value is `all`.
   `vars=` is a list of analytical variables you want to be included in the output file. Simply type in all the variables separated by a space. If you do not specify analytical variables, the code will still execute, but the output file cannot be used for differential mortality analyses.

   For information on EU-SILC variables and UDB restrictions, see Klotz & Göllner (2017).

   Example: A typical macro call could be:
   ```
   %createdata(yr_from=6, yr_to=15, fileoption=1, countries=(&clist), vars=HY020 PH010)
   ```

7. Declare the point of measurement of your analytical variables (the ones specified by the `%createdata` macro in the `vars=` parameter) in the `%infos` macro. You can choose either measurement at the first interview (the baseline) or the last interview. Since some variables are not available when the respondent dies, one usually specifies the first interview.

   Example: A typical macro call would be:
   ```
   %infos(firstvars=PH010 HY020)
   ```

8. Save your modified code under a new name.

9. Run the modified code.

Remarks on the SAS Log

The error message "`ERROR: The variable r: in the DROP, KEEP or RENAME list has never been referenced.`" is given when your specified analytical variables do not contain any variable from the R file. The same applies for the other file types. Usually analytical variables are from the H and/or P file, so error messages for D and R are normal and do not affect the output in a negative way.

Other messages you might encounter are:

```
NOTE: Invalid data for PE040 in line 82357 83-85.
```

This can happen when importing the data from the raw csv files. Mostly this happens because an invalid or missing value is being imported. These data entries will be filled with missing values.

```
NOTE: Invalid argument to function YRDIF(-12008,.,'act/act') at line
17 column 138.
```

This happens in the calculation of time at risk (variable "Verweildauer") if one of the input date values is missing. There is no fix for this, since the data is simply missing. The cases will be deleted later.

Output File Structure

In the output file, each line corresponds to a distinct person who was between 16 and 79 years old at the time of the first interview. It contains the following variables (note: all date variables and their derivations have the midpoints of quarters assumed, since precise day and month information is usually not included in the UDB):

- **PS_ID** = Personal ID, a newly constructed personal identifying variable
- **Country** = The ISO 3166 code, meaning two letter abbreviation of a country
- **Year_Survey** = the year of the first interview
- **Sex** = the sex of the person
- **Dat_Survey** = the date of the first interview
- **Dat_birth** = the date of birth
- **Dat_Death** = the date of death, mostly missing
- **Dat_Cens** = the date of censoring, leaving the sample without dying
- **dt_exit** = date of exit, either Dat_Cens or Dat_Death
- **age** = the age at the time of first survey
- **Verweildauer** = the time at risk or follow-up time in years (i.e., the time span between Dat_Survey and dt_exit)
- **Died** = the vital status of a person (event indicator): 0 = alive, 1 = died
- Plus the **analytical variables** which were specified in the `%createdata` macro.

Note: In your output file, the order of variables may be different.

The output file is a sas7bdat file plus a csv file generated by the **`proc export`** command at the end of the code. If you want to save the output file in another format, you may either apply **`proc export`** in SAS or import the SAS or the csv file in your desired software.

Output File Screenshot

Four exemplary lines of a typical output file are given below. The analytical variable specified here was HY020, the total disposable household income in euros.

PS_ID	Country	Year_Survey	Sex	Dat_Survey	Dat_birth	Dat_Death
4627	AT	2004	1	15/05/2004	15/11/1954	.
4628	AT	2004	2	15/05/2004	15/02/1961	.
4629	AT	2004	1	15/05/2004	15/05/1961	.
4630	AT	2004	1	15/05/2004	15/05/1934	15/02/2005

Dat_cens	dt_exit	age	Verweildauer	Died	HY020
15/08/2007	15/08/2007	49	3.250	0	33891
15/08/2007	15/08/2007	43	3.250	0	33891
15/08/2006	15/08/2006	43	2.250	0	34948
.	15/02/2005	70	0.754	1	38123

Note: In your output file, the order of variables may be different. Depending on your SAS specifications, it may also happen that variables are labelled differently, or use a different decimal sign (e.g. comma instead of a point).

Such an output file can then be used for differential mortality analyses. A popular model class for such analyses is Proportional Hazards Regression. The files labelled "2_Analysis_in_*ABC.xyz*" contain exemplary codes to run a Cox regression in SAS, SPSS and R.

Details

Input data from EU-SILC UDB

Access to Required Input Data

The SAS code edits EU-SILC "User Database" (UDB) data, which are also called "Scientific Use Files" (SUF) data. This input data is not freely accessible, but you need to be approved by Eurostat to use UDB data. In the following it is assumed that your institution has already been approved and that you have access to UDB data. If not, you can apply at http://ec.europa.eu/eurostat/web/microdata/overview .
Note that application takes some time (4 + 8 weeks from Eurostat, plus your own time in completing all the required documents).

Input Data Structure

EU-SILC UDB files are split into cross-sectional and longitudinal data. For this SAS code you need longitudinal data. The UDB longitudinal data are organized by release years. The release year is the latest survey year of a four-year panel period. For example, longitudinal data with release year 2010 contains observations (for the same respondents) in 2007, 2008, 2009 and 2010.

Note that data may not be available for all respondents in all four years, since some respondents withdraw from the panel earlier. Also, in a few instances, interviews are deferred in single years and then continued later. Of course, data availability for a specific country depends on the implementation of EU-SILC in this country. To give an example, UDB longitudinal data for the release year 2006 can contain observations for 2003 only for those countries which have already started EU-SILC in 2003. For details on EU-SILC implementation by country, browse the UDB section on CIRCABC (see Further Information).

Notes

Compared to the original EU-SILC data structure regulated by European Regulation (EC) No 1177/2003 of the European Parliament and of the Council of 16 June 2003 concerning Community statistics on income and living conditions (EU-SILC) and to be transmitted to Eurostat by Member States, the UDB data comes with certain restrictions. These restrictions are implemented by Eurostat to guarantee non-identifiability of particular respondents in publicly accessible micro data. To give an example, in UDB data all respondents aged 80 and over are grouped into an open final age category. An overview of restrictions may be found in the document entitled "Differences Between Original Database (as Described in the Guidelines) and the Anonymised User Database", which again can be found on CIRCABC. Restrictions which are particularly important for differential mortality analyses are also summarized in Klotz & Göllner (2017).

EU-SILC UDB data are periodically updated by Eurostat when new information becomes available. Thus, different versions of UDB longitudinal data for the same release year may

exist. The 2016 release of longitudinal UDB data contained a major revision of all prior data and should be used for analysis.

The SAS code was written in 2017 and 2018 and reflects the state of the EU-SILC UDB longitudinal data structure at the time of its writing. Thus, the algorithm may not work for later UDB longitudinal data versions if there were changes in the data structure, such as newly implemented anonymization rules.

SAS Code Introduction

Aim of the SAS Code

This code is used to pool multiple releases of EU-SILC UDB longitudinal data and process it to obtain a file where each line of data corresponds to a distinct person. A time–at–risk variable (named by the German word "Verweildauer") and vital status information (survived or died) is calculated for every person. By default, the output file contains some variables required for differential mortality analyses, such as sex or age at baseline. Besides, you can specify analytical variables by which you want to analyse mortality differences, such as educational level or occupational class. The output file is ready to be used for differential mortality analyses. For example, Cox's Proportional Hazards regression models (Kleinbaum & Klein, 2005) may be estimated.

Preparation

Since this code is written in SAS there are two initial situations you can be in regarding the UDB input data:
(1): You have the raw longitudinal csv files per file type (D, H, R and P) per release year.
(2): You already have the sas7bdat files per file type (D, H, R and P) per release year.
Note that the files have to include all countries. If your files are split by country as well, then you have to merge them together first before applying this code successfully.

(1): If you have the raw csv files, normally Eurostat will provide you with SAS codes to load those files correctly. An example is given in the Appendix. Normally, the first two lines of code are something like this `data WORK.DFILE; infile "C:\some\path\d12.csv"`. You have to change this to `data WORK.DFILE12; infile "C:\your\path\d12.csv"`. The digit(s) you have to append to the file name are [release year minus 2000]. So if you load the 2012 release of the D file, then the generated SAS file has to be named DFILE12, and if you load the 2006 release of the R file, then the generated SAS file has to be named RFILE6.
Notes:

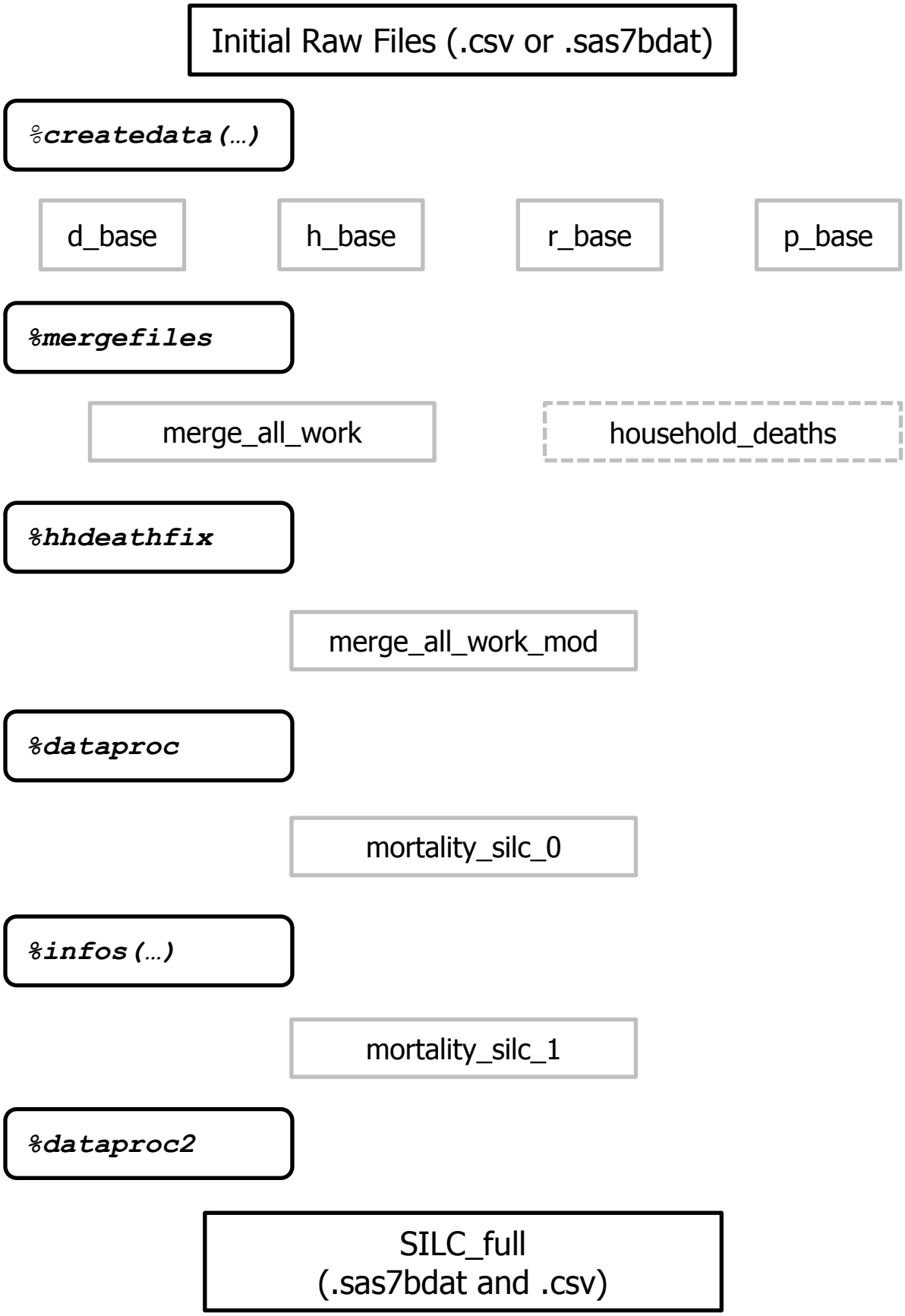
- You have to change **every** SAS file name and path to the csv file!
- SAS is case-insensitive, so RFILE6 is the same as rfile6, Rfile6, etc.
- The specification of the WORK library (working directory) may be omitted, for this is the default library in SAS.

(2): If you have the sas7bdat files already rename them like in option (1). The naming convention is [letter of the file type] + "file" + [release year minus 2000]. So if you have the D file of the 2012 release, then the file has to be named DFILE12.sas7bdat (again, uppercase or lowercase does not matter). Other examples would be: rfile6.sas7bdat, PFILE9.sas7bdat, Hfile10.sas7bdat, and so on. This has to be done for **every** file.

The folder structure is the same for the csv data loading codes (1) and the sas7bdat files (2). You have to have four different folders according to the file type. This means, one folder for all the D files, one for all the H files, one for all the R files and one for all the P files. Each folder contains either all the codes to load the csv files (1) or the sas7bdat files (2).

When you have prepared the files accordingly, then you can run the "0_Load-Macros.sas" code, which requires no input from you. Next you can open the "1_Application.sas" code and edit it to your needs. After you altered the "1_Application.sas" code to you needs and ran it successfully, the final output file "SILC_full" is created. This file can then be used for analytical purposes. To give you some ideas on possible analytical models see the section on Exemplary Analysis Code (2_Analysis_in_*ABC.xyz*). The code for a simple Cox proportional hazards regression model in SAS, SPSS and R is provided in the "2_Analysis_in_*ABC.xyz*" files.

Files Created by Macros



Macro Explanation (0_Load_Macros.sas)

In this section all macros that are loaded when running "0_Load-Macros.sas" are explained in detail and their code is given. This may help you to understand the process of data extraction and editing. For advanced users this allows for altering the macros, if needed.

The %drive Macro

This macro is based on http://support.sas.com/kb/45/805.html and was slightly modified so that it runs an `%include` statement. It is used in the `%`***createbase*** macro and runs all existing code in a folder that is specified earlier through a `%let` statement.

Depends on:
```
%let dincl = C:\path\to-your\D-file-codes;
%let hincl = C:\path\to-your\H-file-codes;
%let pincl = C:\path\to-your\P-file-codes;
%let rincl = C:\path\to-your\R-file-codes;
```

Used in: `%`***createbase*** (automatically)

Call with: `%`***drive***`(dir,ext)` where `dir` = directory and `ext` = file extension

Examples: `%`***drive***`(&hincl,sas)` as used in `%`***createbase***

`%`***drive***`("C:\path\to-a-folder\with-codes",sas)` normal use

Code:

```
%macro drive(dir,ext);
  %local filrf rc did memcnt name f;

  %let rc=%sysfunc(filename(filrf,&dir));
  %let did=%sysfunc(dopen(&filrf));

  %if &did eq 0 %then %do;
   %put Directory &dir cannot be open or does not exist;
   %return;
  %end;

   %do f = 1 %to %sysfunc(dnum(&did));

     %let name=%qsysfunc(dread(&did,&f));
      %if %qupcase(%qscan(&name,-1,.)) = %upcase(&ext) %then %do;
/*changed here*/  %include "&dir\&name";
      %end;
      %else %if %qscan(&name,2,.) = %then %do;
        %drive(&dir\%unquote(&name),&ext)
      %end;

   %end;

  %let rc=%sysfunc(dclose(&did));
```

```
  %let rc=%sysfunc(filename(filrf));

%mend drive;
```

In the commented line (`/*changed here*/`) the original code was changed from a `%put` statement to an `%include` statement. The details of this code are explained by the comments in the original code.

The %createdata Macro

This macro is the first one that you will run. It pools all the longitudinal data of the release years that you specify and stores them into the `eusilc` library. It then creates four new permanent files containing a limited set of variables, thus decreasing the file sizes. Overall, this macro requires specification of five parameters from you ("Options"). Depending on your system and the parameters you specify, it can be quite memory and time intensive.
If you work with the raw .csv-files of the longitudinal UDB data and the codes to import them into SAS, then you have to choose `fileoption=1`. Be aware that the created SAS files have to follow the naming convention mentioned in the Preparation section (e.g., `dfile9`, `rfile12`, etc.). If you have the longitudinal UDB data already stored as .sas7bdat-files, then you have to choose `fileoption=2`. In this case the files also have to follow the naming convention mentioned in the Preparation section (e.g., `dfile9`, `rfile12`, etc.). Either option requires you to specify locations where this data is stored and a different location has to be specified for each of the four file types.

Options:

`yr_from=` the release year (minus 2000) of the earliest longitudinal data you use. The default value is `6` (i.e., release year 2006)

`yr_to=` the release year (minus 2000) of the latest longitudinal data you use. The default value is `15`.

`fileoption=` either `1` or `2`. Default = `1`.

`countries=` either `(&clist)` or `all`. The first one has to be in brackets! The default value is `all`.

`vars=` a list of analytical variables which you want to include in your data extraction. Several analytical variables have to be separated by a blank; the order of variables is not important. Variables which are extracted by default by the code, such as sex and year of birth, should not be specified in here. (These default variables are: `DB010 DB020 DB030 DB110 HB010 HB020 HB030 HB050 HB060 PB010 PB020 PB030 PB100 PB110 RB010 RB020 RB030 RB040 RB070 RB080 RB090 RB110 RB140 RB150 RX010`.)

Depends on: If `fileoption=1` then:

```
%let dincl = C:\path\to-your\D-file-codes;
```

```
%let hincl = C:\path\to-your\H-file-codes;
%let pincl = C:\path\to-your\P-file-codes;
%let rincl = C:\path\to-your\R-file-codes;
```

If `fileoption=2` then:

```
libname dfiles "C:\path\to-your\d-file-data";
libname hfiles "C:\path\to-your\h-file-data";
libname pfiles "C:\path\to-your\p-file-data";
libname rfiles "C:\path\to-your\r-file-data";
```

If `countries=(&clist)` then:

`%let clist='AT', 'BE';` (an exemplary specification for Austria and Belgium; be aware that you need two codes 'EL' and 'GR' for Greece, which are later merged into "GR")

And the `%drive` macro.

Examples: `%createdata()` to use all default options.

`%createdata(yr_to=10, vars = HY020)` to use the longitudinal data from release year 2006 to 2010 and to include target variable HY020 in your data extraction.

`%createdata(yr_from=9, fileoption=2)` to use the longitudinal data from release year 2009 to 2015 for all countries and to use the already existing sas files.

`%createdata(yr_from=6, yr_to=10, fileoption=2, countries=(&clist))` to use the longitudinal data from release year 2009 to 2015, to use the already existing sas files, and to limit data extraction to a pre-specified list of countries.

Code:

```
%macro createdata(yr_from=5, yr_to=15, fileoption=1, countries=all,
vars=);

%local n i;

%global dvars hvars pvars rvars;

%global luyr;
%let luyr = %eval(&yr_to - &yr_from + 4);

%let n = %sysfunc(countw(&list));
%do i=1 %to &n;
     %let file = %scan(&list,&i);

filename zdat temp;
proc stream outfile=zdat prescol; begin
&vars
;;;;
```

```
data vars;
infile zdat;
input &vars;
run;

proc contents data=vars (keep=&file.:) out=&file.var (keep=name)
noprint;
run;

      data &file.var;
      set &file.var;
      name=upcase(name);
      run;

      proc sql noprint;
      select distinct name into :&file.vars separated by ' ' from
      &file.var order by name;
      quit;

/* if files have to be created */
      %if &fileoption=1 %then %do;

            %drive(&&&file.incl,sas)

            data &file.full;
            set %do num=&yr_from %to &yr_to;
                  &file.file&num (in=&file.&num.)
                  %end; ;
                        %do num=&yr_from %to &yr_to;
                        if &file.&num then &file._release = &num ;
                        %end;
            %if &countries = all %then ;
            %else where &file.b020 in &countries.; ;

            %if &file=d  %then %do;
      keep DB010 DB020 DB030 DB110 d_release &dvars;
            %end;
            %else %if &file=h %then %do;
      keep HB010 HB020 HB030 HB050 HB060 h_release &hvars;
            %end;
            %else %if &file=p %then %do;
      keep PB010 PB020 PB030 PB100 PB110 p_release &pvars;
            %end;
            %else %if &file=r %then %do;
      keep RB010 RB020 RB030 RB040 RB070 RB080 RB090 RB110 RB140
      RB150 RX010 r_release &rvars;
            %end;
            run;

            data &file.full_re;
            set &file.full;
            if &file.b020 = 'EL' then &file.b020 = 'GR';
            run;

            proc sql;
            create table &file.full_s as
```

```
            select *
            from &file.full_re
            group by &file.b020, &file.b030, &file.b010
            having &file._release = max(&file._release);
            quit;


            data eusilc.&file._base;
            set &file.full_s;

%if &file=d  %then %do;
            rename DB010=Year_Survey DB020=Country DB030=HH_ID ;
%end;
      %else %if &file=h %then %do;
            rename HB010=Year_Survey HB020=Country HB030=HH_ID ;
      %end;
      %else %if &file=p %then %do;
            rename PB010=Year_Survey PB020=Country PB030=PS_ID ;
      %end;
      %else %if &file=r %then %do;
            rename RB010=Year_Survey RB020=Country RB030=PS_ID
RB040=HH_ID ;
      %end;
            run;

            proc datasets lib=work nolist kill;
            quit;
            run;
      %end;

/*if files are already created */
      %else %if &fileoption=2 %then %do;

            data &file.full;
            set %do num=&yr_from %to &yr_to;
                  &file.files.&file.file&num (in=&file.&num.)
                  %end; ;
                        %do num=&yr_from %to &yr_to;
                        if &file.&num then &file._release = &num ;
                        %end;
            %if &countries = all %then ;
            %else where &file.b020 in &countries.; ;

      %if &file=d  %then %do;
            keep DB010 DB020 DB030 DB110 d_release &dvars;
      %end;
      %else %if &file=h %then %do;
            keep HB010 HB020 HB030 HB050 HB060 h_release &hvars;
      %end;
      %else %if &file=p %then %do;
            keep PB010 PB020 PB030 PB100 PB110 p_release &pvars;
      %end;
      %else %if &file=r %then %do;
            keep RB010 RB020 RB030 RB040 RB070 RB080 RB090 RB110 RB140
      RB150 RX010 r_release &rvars;
      %end;
```

```
            run;

            data &file.full_re;
            set &file.full;
            if &file.b020 = 'EL' then &file.b020 = 'GR';
            run;

            proc sql;
            create table &file.full_s as
            select *
            from &file.full_re
            group by &file.b020 &file.b030 &file.b010
            having &file._release = max(&file._release);
            quit;


            data eusilc.&file._base;
            set &file.full_s;

%if &file=d  %then %do;
            rename DB010=Year_Survey DB020=Country DB030=HH_ID ;
%end;
      %else %if &file=h %then %do;
            rename HB010=Year_Survey HB020=Country HB030=HH_ID ;
      %end;
      %else %if &file=p %then %do;
            rename PB010=Year_Survey PB020=Country PB030=PS_ID ;
      %end;
      %else %if &file=r %then %do;
            rename RB010=Year_Survey RB020=Country RB030=PS_ID
RB040=HH_ID ;
      %end;
            run;

            proc datasets lib=work nolist kill;
            quit;
            run;
      %end;

%end;

%mend createdata;
```

In the beginning you see the macro variable `&list` in use. From this macro variable the macro variable `&file` is created, which is in the first iteration `d`, then `h`, then `p` and finally `r`. This way all four file types are created one after the other. The SQL part makes sure that you use the most recent data release of every line.

Important: If you do not specify at least one analytical variable of each file type (d, h, p and r) in your `vars=` statement, then errors will be put in the log file when running the code. These errors are not fatal to the code. Read the “Remarks on the SAS Log” in the Code Execution Outline for more details.

Depending on which `fileoption` you chose either the top half or the lower half of the code is executed. Both codes end with a `proc datasets lib=work nolist kill; quit; run;` command, so you should close any other SAS session while running.

The %mergefiles Macro

This is the first macro that requires no input from you. It is a simple series of data and proc steps which creates the merged data file with information from all four file types.

Options: none

Depends on: all previous macros and statements.

Example: `%mergefiles`

Code:

```
%macro mergefiles();

proc sort data=eusilc.d_base;
by Country HH_ID Year_Survey;
run;

proc sort data=eusilc.h_base;
by Country HH_ID Year_Survey;
run;

data households;
merge eusilc.h_base eusilc.d_base (in=inD);
by Country HH_ID Year_Survey;
if inD;
run;

proc sort data=eusilc.r_base;
by Country PS_ID Year_Survey;
run;

proc sort data=eusilc.p_base;
by Country PS_ID Year_Survey;
run;

data persons;
merge eusilc.p_base eusilc.r_base (in=inR);
by Country PS_ID Year_Survey;
if inR;
run;

/* fix reassigned IDs */

      proc sort data=persons out=persons_sort
      (rename=(PS_ID=old_PS_ID));
      by Country PS_ID rb090 rb080 rb070 HH_ID Year_Survey;
      run;

      data persons_newID ;
```

```
      set persons_sort;
      by Country old_PS_ID rb090 rb080 rb070 HH_ID Year_Survey;
            if first.rb070 then PS_ID+1;
      run;

/* end */

proc sort data=households;
by Country HH_ID Year_Survey;
run;

      data eusilc.household_deaths;
      set households;
      where db110 = 5;
      keep country hh_id year_survey;
      run;

proc sort data=persons_newID;
by Country HH_ID Year_Survey;
run;

data eusilc.merge_all_work;
merge households persons_newID;
by Country HH_ID Year_Survey;
run;

%mend mergefiles;
```

First it merges the household files `h` and `d`, and then it merges the person files `p` and `r`. Before it merges those two files, it creates a new variable that contains a sequence number for all persons. This has to be done for two reasons. First, there are countries which reassign their personal IDs. Second, in the 2015 release of EU-SILC, there is a harmonization between the ID variables of the cross-sectional and the longitudinal component. The macro also creates a data file containing only the cases where the entire household died, which is used in the next macro. In the end it creates the fully merged file called `merge_all_work` in the `eusilc` library.

The %hhdeathfix Macro

This macro is needed to correctly assign the vital status to a person when the entire household died. It requires no input from the user.

Options: none

Depends on: all previous macros and statements.

Example: `%hhdeathfix`

Code:

```
%macro hhdeathfix();
```

```
/* first fix the deaths where an entire household dies */
     data household_deaths_IDs;
     set eusilc.household_deaths (rename = (year_survey =
     year_died));
     by country hh_id;
     year_survey = year_died - 1;
     if first.hh_id;
     run;

/* create a dataset for re-merging */
     proc sort data = eusilc.merge_all_work out =
     household_deaths_info;
     by country hh_id year_survey ps_id;
     run;

/* merge the two datasets */
/* adds the information of the year before */
     data hh_to_replace;
     merge household_deaths_info
          household_deaths_IDs (in = inD);
     by country hh_id year_survey;
     if inD;
     run;

/* select persons with ID and that are current household members */
/* this is from the year before the household dies */
     data hh_to_replace;
     set hh_to_replace;
     where ps_id ne . and rb110 in (1 2 3 4);
     year_survey + 1;
     db110 = 5;
     run;

/* sort */
     proc sort data = hh_to_replace;
     by Country HH_ID PS_ID Year_Survey;
     run;

     proc sort data = eusilc.merge_all_work;
     by Country HH_ID PS_ID Year_Survey;
     run;


/* now use the created data to replace the cases where the entire
household dies */
     data eusilc.merge_all_work_mod;
     merge eusilc.merge_all_work (where = (db110 ne 5))
          hh_to_replace;
     by Country HH_ID PS_ID Year_Survey;
     run;

%mend hhdeathfix;
```

This macro creates an intermediate dataset containing information from the year before the household died, and adds this information to the persons residing in the household.

The %dataproc Macro

This macro also requires no input from the user. This macro deals with the various restriction of the EU-SILC UDB (compared to the original data), creates date variables and assigns a conclusive vital status indicator ("Died") by merging the information of DB110 and RB110.

Options: none

Depends on: all previous macros and statements.

Example: %***dataproc***

Code:

```
%macro dataproc();

      /*Add Perso_ID variable and create date Variables */
      data merge_all_work_mod1;
      set eusilc.merge_all_work_mod;
      if Country ne 'BE' then do;

            /*survey month*/
            if pb100 = 1 then M_survey = 2;
                  else if pb100 = 2 then M_survey = 5;
                  else if pb100 = 3 then M_survey = 8;
                  else if pb100 = 4 then M_survey = 11;
                  else M_survey=.;

            /*deathORmove month*/
            if rb140 = 1 then M_deadORmove = 2;
                  else if rb140 = 2 then M_deadORmove = 5;
                  else if rb140 = 3 then M_deadORmove = 8;
                  else if rb140 = 4 then M_deadORmove = 11;
                  else M_deadORmove=.;

            /*add half a year to death or move. we assume the date of
            death or move is half a year after last interview */
            if db110 in (3 4 5) then do;
                  if hb050 = 1 then M_deadORmove=8;
                  else if hb050 = 2 then M_deadORmove=11;
                  else if hb050 = 3 then M_deadORmove=2;
                  else if hb050 = 4 then M_deadORmove=5;
                  else M_deadORmove=.;
                  end;

            /*birth month*/
            if rb070 = 1 then M_birth = 2;
                  else if rb070 = 2 then M_birth = 5;
                  else if rb070 = 3 then M_birth = 8;
                  else if rb070 = 4 then M_birth = 11;
                  else M_birth=.;

      end; /*BE only*/
```

```
      else if Country eq 'BE' then do;

            /*survey month*/
            if pb100 = 1 then M_survey = 2;
                  else if pb100 = 2 then M_survey = 5;
                  else if pb100 = 3 then M_survey = 8;
                  else if pb100 = 4 then M_survey = 11;
                  else M_survey=.;

            /*BE has full month data*/
            M_deadORmove=rb140;

            if db110 in (3 4 5) then do;
                  if hb050 = 1 then M_deadORmove=8;
                  else if hb050 = 2 then M_deadORmove=11;
                  else if hb050 = 3 then M_deadORmove=2;
                  else if hb050 = 4 then M_deadORmove=5;
                  else M_deadORmove=.;
                  end;

            /*birth month*/
            if rb070 = 1 then M_birth = 2;
                  else if rb070 = 2 then M_birth = 5;
                  else if rb070 = 3 then M_birth = 8;
                  else if rb070 = 4 then M_birth = 11;
                  else M_birth=.;
      end;
      run;



      /*Create Variables*/
      data merge_all_work_mod2;
      set merge_all_work_mod1;

      /*assume middle of quarter for survey date and birth date*/
      Dat_Survey = mdy(M_survey, 15, pb110);
      Dat_birth = mdy(M_birth, 15, rb080);

      /*assume middle of quarter for date of death or move; date
      between last interview and would be next interview*/
      if rb110 in (5 6) then Dat_deadORmove = mdy(M_deadORmove, 15,
rb150);
                  else if db110 in (3 4 5) AND hb050 in (3 4) then
Dat_deadORmove = mdy(M_deadORmove, 15, hb060+1);
                  else if db110 in (3 4 5) then Dat_deadORmove =
mdy(M_deadORmove, 15, hb060);
                  else Dat_deadORmove=.;
            RUN;


      /*Rename Variables*/
      /*IE, MT, NL, SI and UK all do no provide the Month of Birth
      (RB070) at all. */
```

```
/*We fix this by assuming the person lived half a year before
the interview */
/*Some other countries also have this problem but for far less
cases. These get fixed with this as well */
data merge_all_work_mod3;
set merge_all_work_mod2 (rename=(rb090=Sex rx010=Age_Survey));

if Dat_birth=. then do;

      if month(Dat_Survey) = 8 then do
            Dat_birth=mdy(2 ,15, (year(Dat_Survey)-Age_Survey));
            end;
      else if month(Dat_Survey) = 11 then do;
            Dat_birth=mdy(5, 15, (year(Dat_Survey)-Age_Survey));
            end;
      else if month(Dat_Survey) = 2 then do;
            Dat_birth=mdy(8, 15, (year(Dat_Survey)-Age_Survey-
            1));
            end;
      else if month(Dat_Survey) = 5 then do;
            Dat_birth=mdy(11, 15, (year(Dat_Survey)-Age_Survey-
            1));
            end;

end;
else do;
      calc_age = yrdif(Dat_birth, Dat_Survey, 'act/act');
end;

run;

/*Sort*/
proc sort data=merge_all_work_mod3 out=merge_all_work_mod4;
by PS_ID Year_Survey;
run;

/*Create Variables*/
data merge_all_work_mod5;
set merge_all_work_mod4;
by PS_ID;
Died=.;

if rb110=6 then Died = 1;  /* died */
if rb110=5 then Died = -1; /* moved out or last interview if
not contacted previous wave */
if rb110=1 then Died = 0;  /* current household member, was in
HH prev wave */
if rb110=2 then Died = 0;  /* moved into this HH from other
sample HH */
if rb110=3 then Died = 0;  /* moved into this HH from outside
sample */
if rb110=4 then Died = 0;  /* newly born into HH */
if rb110=7 then Died = 0;  /* lived in HH for at least 3 months
(inc ref per) not recorded in the register of this HH */

if last.PS_ID then do;
```

```
        if db110=5 then Died=1;
        else if db110 in (3 4) then Died=-1;
        end;
    RUN;

    /*Create Variables*/
    data merge_all_work_mod6;
    set merge_all_work_mod5;
    by PS_ID;
    Dat_Death=.;
    Dat_Cens=.;

    if last.PS_ID then do;
        if Died=1 then Dat_Death = Dat_deadORmove;
        if Died=0 then Dat_Cens = Dat_Survey;
        if Died=-1 then Dat_Cens = Dat_deadORmove;
        end;
    RUN;

    /* Sort & save file */
    proc sort data=merge_all_work_mod6
    out=eusilc.mortality_SILC_0;
    by PS_ID Year_Survey;
    Run;

%mend dataproc;
```

If you are having memory trouble with your computer a work library deletion `proc datasets lib=work nolist kill; quit; run;` is probably needed before or after this step.

The %infos Macro

This macro requires parameter specification from you. In detail, you have to partition all your analytical variables (which you specified in the ***%createdata*** macro in the `vars=` statement) according to the point of measurement: `firstvars=` for variables where the values should be measured at the beginning of the follow-up period (the first interview) and `lastvars=` for variables where the values should be measured at the end of the follow-up period (the last interview). Usually, one wants to measure covariates at the beginning of the follow-up period, but there may be reasons to measure (some) covariates at a later point. Note that some measurements may not be available at the last interview, for instance there is no household income when the entire household died.

Options:

`firstvars=` the analytical variables which should be measured at the first interview, delimited by blanks.

`lastvars=` the analytical variables which should be measured at the last interview, delimited by blanks.

Depends on: all previous macros and statements.

Examples: `%infos(firstvars=PH020, lastvars=HY020)`

`%infos(firstvars=PH020 HY020)`

Code:

```
%macro infos(firstvars=, lastvars=);

     /*create first entry for every person*/
     data first (keep=PS_ID old_PS_ID calc_age Age_Survey Sex
Dat_Survey Dat_birth Year_Survey Country &firstvars);
     set eusilc.mortality_SILC_0;
     by PS_ID;
     if first.PS_ID;
     RUN;

     /*create last entry for every person*/
     data last (keep= PS_ID Died Dat_Death Dat_Cens rb110 db110
&lastvars);
     set eusilc.mortality_SILC_0;
     by PS_ID;
     if last.PS_ID;
     Run;

     /* merge first and last */
     data eusilc.mortality_SILC_1;
     merge first last;
     by PS_ID;
     RUN;

%mend infos;
```

The %dataproc2 Macro

This is the last macro and requires no input from the user. It creates an output file which contains a time at risk variable (German = "`Verweildauer`"), which is essential for differential mortality analyses based on sample survey data. The file structure is given in the Output File Structure section.

Options: none

Examples: `%dataproc2`

Code:

```
%macro dataproc2();

     /*add variable Verweildauer */
     data mortality_SILC_2;
     set eusilc.mortality_SILC_1;
```

```
if Died=1 then Verweildauer = yrdif(Dat_Survey, Dat_Death,
'act/act');
else Verweildauer = yrdif(Dat_Survey, Dat_Cens, 'act/act');
if Verweildauer >0;
RUN;

/*compute new age variable*/
data mortality_SILC_agefix;
set mortality_SILC_2;
age=.;

      if Age_Survey=. then age=int(calc_age);
      else if Age_Survey=-1 then age=int(calc_age);
      else if Age_Survey=-2 then age=int(calc_age);
      else if Age_Survey=81 then age=int(calc_age);
            else age=Age_Survey;

run;

/* LU has permanent panel. change to appropriate value */
data mortality_SILC_vdfix;
set mortality_SILC_agefix;
where (age between 16 and 79);
if Country = "FR" then do;
      if Verweildauer < 9 then eligible = 1;
      else eligible = 0;
end;
else if Country = "LU" then do;
      if Verweildauer < &luyr then eligible = 1;
      else eligible = 0;
end;
else if  Country = "NO" then do;
      if Verweildauer < 8 then eligible = 1;
      else eligible = 0;
end;
else do;
      if Verweildauer < 4 then eligible = 1;
      else eligible = 0;
end;
if eligible = 1;
run;

data eusilc.SILC_full;
set mortality_SILC_vdfix (drop= db110 rb110 Age_Survey calc_age
eligible);
IF Died = -1 then Died = 0;
IF Dat_Death then dt_exit=Dat_Death;
IF Dat_Cens then dt_exit=Dat_Cens;
format dt_exit Dat_Death Dat_Cens Dat_Survey Dat_Birth
ddmmyy10.;
run;

%mend dataproc2;
```

The output file is stored in your library and called SILC_full.sas7bdat. Additionally, a csv file is created by the “1_Application.sas” code.

To successfully run the code, simply **run the whole "0_Load Macros.sas"** code and **then** proceed to **alter** the **"1_Application.sas"** code to your needs and **run it too**.

Macro Execution (1_Application.sas)

In this section all the commands of the "1_Application.sas" code are explained. After you ran the "0_Load Macros.sas" code you can open the "1_Application.sas" code and start making changes where they are needed. Following is a step by step explanation of each statement.

```
option compress=yes;
```

This option should be enabled to reduce computational time. It does not change the data itself in any way.

```
%let silcpath = C:\path\to-your\library-folder;
libname eusilc "&silcpath";
```

Here you **need** to specify the folder of your library. Do not change anything except the value of `silcpath` (written in italics). The library called `eusilc` is referred in all the macro codes when permanent saves are being made. The saves are usually at the end of a macro, this way you stay flexible in using the code. Be aware that the macros always save the files with the same name. If multiple runs of the macros are needed (for experimental purposes or else), different folders (eg. values of `silcpath`) are necessary. Alternatively, if only one macro will be re-run differently, one can simply rename this specific output file to prevent overwriting.

```
%let clist='AT', 'BE', 'BG', 'CY', 'CZ', 'EE', 'EL', 'ES', 'FI',
'FR', 'GR', 'HR', 'HU', 'IT', 'LT', 'LU', 'LV', 'MT', 'NL', 'NO',
'PL', 'PT', 'RO', 'SE', 'SI', 'SK', 'UK';
```

This macro variable contains the country list of all the countries part of EU-SILC UDB (at the time of writing this manual). If only a subset of those countries is needed, this list has to be changed accordingly. Note, if you are interested in Greece, be sure to have both "GR" and "EL" in your list! (After all macros ran, Greece will be labelled "GR".)

```
%let list=d h p r;
```

This is an essential macro variable which is needed to loop over all four file types. No change needed.

The next statements refer to the options of the first macro.

For `fileoption=1` use this:

```
%let dincl = C:\path\to-your\D-file-codes;
%let hincl = C:\path\to-your\H-file-codes;
%let pincl = C:\path\to-your\P-file-codes;
%let rincl = C:\path\to-your\R-file-codes;
```

If you have stored a code which will create the UDB files for you, you have to specify the folders here. You have to have a different folder for every file type, otherwise the code will fail!

For `fileoption=2` use this:

```
libname dfiles "C:\path\to-your\d-file-data";
libname hfiles "C:\path\to-your\h-file-data";
libname pfiles "C:\path\to-your\p-file-data";
libname rfiles "C:\path\to-your\r-file-data";
```

If you have the UDB data already ready to use in SAS, specify the paths to the files here. Again, you have to have different folders for different file types. The names of the files have to be in line with the following naming convention: [file type letter] + "file" + [year of the release – 2000]. This means files are named dfile8, hfile9, pfile10 or rfile11 for instance. No subfolders can be present!

Following are the macro executions with exemplary parameters.

```
%createdata(yr_from=6, yr_to=15, fileoption=1, countries=(&clist),
vars=hy020 PH020)
```

Here you have an example where all options are present. You would not need to specify the first three, since they are their default value. The options set the starting year (`yr_from`), the last year (`yr_to`), if your files are created from a code or already present as sas7bdat files (`fileoption`), which countries you want to include (`countries`) and which analytical variables you want to include (`vars`). The years are given by subtracting 2000 from their release year, e.g. `yr_from=8, yr_to=12` for the releases 2008, 2009, 2010, 2011, 2012. This in turn means your observed years will be from 2005 to 2012. The file option is either 1 or 2, nothing else. The countries should be selected either via the country list specified prior (`&clist`) or by using `all`. The analytical variables are simply separated by a blank delimiter and do not have to follow any specific order. Of course they must be real EU-SILC variables, found in the document "Methodological Guidelines and Description of EUSILC Target Variables. DocSILC065" (on CIRCABC).

`%`***mergefiles***
Requires no input and simply will merge the files from the previous macro. First it merges the household files `h` and `d` into `households`, then it merges the person files `p` and `r` into `persons`. It creates a new ID variable for all persons. Creates an intermediate dataset for the `%`***hhdeathfix*** macro, and then merges the `households` and `persons` files into the `eusilc.merge_all_work` file.

`%`***hhdeathfix***
Requires no input and will correctly fix the issues if a whole household dies where multiple persons lived.

`%`***dataproc***

Requires no input and will calculate the various date variables that you need to calculate the time at risk variable later on. Furthermore, it does some housekeeping on the data.

```
%infos(firstvars=PH020 HY020)
```

All of the analytical variables which you specified in the `%createdata` macro as `vars=` need to be referenced here. Usually, you simply copy all of them into the `firstvars=` variable. You can also use the `lastvars=` variable if you want them extracted at the end of the observational period, but be aware that some variables are not filled for deceased persons (eg. HY020, the household income).

`%dataproc2`
Requires no input. Creates the time at risk variable ("`Verweildauer`") and calculates the age of the respondent where it was not filled prior. Furthermore, housekeeping is done to arrive at our final dataset.

The final dataset is called "SILC_full.sas7bdat" and is a SAS data file. The last command will save a csv file of this dataset.

```
proc export data=eusilc.silc_full
  dbms=csv
  outfile="&silcpath.\silc_full.csv"
  replace;
run;
```

Note that the date variables will be saved exactly as they are displayed, e.g.: 22/02/1980. Keep this in mind when importing the csv file into a statistical program.

Results

Your final output contains information on the following variables:

- **PS_ID** = Personal ID, a newly constructed identifying variable
- **Country** = The ISO 3166 code, meaning two letter abbreviation of a country
- **Year_Survey** = the year of the first interview
- **Sex** = the sex of the person

All following date variables have the midpoints of quarters assumed, since precise day and month information is (usually) not included in the UDB.

- **Dat_Survey** = the date of the first interview
- **Dat_birth** = the date of birth
- **Dat_Death** = the date of death, mostly missing
- **Dat_Cens** = the date of censoring, leaving the sample without dying
- **dt_exit** = date of exit, either the Dat_Cens or Dat_Death value
- **age** = the age at the time of first survey
- **Verweildauer** = the "time at risk" or time inside the sample (German term for it)
- **Died** = the death status of a person (event indicator), 0 = alive / 1 = died
- Plus the **analytical variables** of your choosing.

The output file will be a sas7bdat file plus a csv file generated by the **`proc export`** command at the end of the code. Of course you can save the file in any other desired output format. The SAS commands to do this are quite straight forward. Simply read up on the **`proc export`** command in the SAS Help and Documentation or any online resource.

Exemplary Analysis Code (2_Analysis_in_*ABC.xyz*)

Here it is assumed that you have specified `HY020`, the annual total disposable household income, as the analytical variable in the ***createdata*** macro. It is furthermore assumed that the final output SAS file `SILC_full` is stored in a folder *"D:\path-to-silc".*

We estimate a simple Cox proportional hazards regression by comparing the mortality risk of individuals in three household income groups: (1) Less than 10,000 euros; (2) Between 10,000 and 30,000 euros; (3) More than 30,000 euros. Mortality hazard ratios of the two marginal income groups are compared with the reference middle income group. We restrict the sample to individuals aged 30-79 years and with known and positive household income at baseline. We control for differences in age distribution between the income groups.

Note: This is only a crude model to give you a brief overview of the required syntaxes. A social scientist would probably specify the model in a more sophisticated way, such as accounting for household size, purchasing power differences between countries, adjusting for inflation over time, including additional control variables, etc.

In the following, data preparation and model estimation code is given for SAS 9.4, SPSS 20, and R 3.4.2. Note that depending on your R installation, it may or may not be required to install additional packages.

SAS 9.4

```
/* Specify the working directory */
libname eusilc "D:\path-to\silc";

/* Data preparation */
data analysis;
     set eusilc.silc_full;
     where (age ge 30 and age le 79) and hy020 gt 0;
     if hy020 lt 10000 then income_group = 1;
          else if hy020 ge 10000 and hy020 le 30000
               then income_group = 2;
          else income_group = 3;
     run;

/* Model estimation */

proc phreg data = analysis;
     class income_group (param = ref ref = "2");
     model verweildauer * died (0) = age income_group;
     run;
```

SPSS 20

```
* Specify the working directory.
cd "D:\path-to\silc".

*Import the data from SAS and save the SPSS file.
set unicode on.
get sas data = "silc_full.sas7bdat".
save outfile "silc_full.sav".

* Data preparation.
select if (age >= 30 and age <= 79) and hy020 > 0.
compute income_group = 1.
   if (hy020 >= 10000 and hy020 <= 30000) income_group = 2.
   if (hy020 > 30000) income_group = 3.
save outfile "analysis.sav".

* Model estimation.
coxreg verweildauer
   /status = died (1)
   /contrast (income_group) = indicator (2)
   /method = enter age income_group
   /print = ci (95).
```

R 3.4.2

```
# Set the working directory
setwd ("D:/path-to/silc")

# (Install and) load special packages
install.packages (c("haven", "survival"))
library (haven)
library (survival)

# Import the sas file
silc_full <- read_sas ("silc_full.sas7bdat")

# Data preparation
# Attention: R is case-sensitive!

attach (silc_full)
analysis <- silc_full [ which(age >= 30 & age <= 79 & HY020 > 0),]
detach (silc_full)

attach (analysis)
analysis$income_group [HY020 < 10000] <- 1
analysis$income_group [HY020 >= 10000 & HY020 <= 30000] <- 2
analysis$income_group [HY020 > 30000] <- 3
detach (analysis)

# Model estimation
Model_I <- coxph (Surv(time=Verweildauer, event=Died) ~ age +
     factor(income_group, levels = c(2,1,3)), data = analysis)
summary(Model_I)
```

Contact

E-Mail: tobias.goellner@statistik.gv.at

Github: @TobiasGold and the repository https://github.com/TobiasGold/FACTAGE-method_Mortality

Further Information

https://www.factage.eu/ – for further information on FACTAGE

http://ec.europa.eu/research/era/joint-programming-initiatives_en.html – for more information on the various Joint Programming Initiatives

http://www.jp-demographic.eu/ – for information on the JPI – More Years, Better Lives

http://ec.europa.eu/eurostat/web/microdata/european-union-statistics-on-income-and-living-conditions – for information on EU-SILC

http://ec.europa.eu/eurostat/statistics-explained/index.php/Glossary:EU_statistics_on_income_and_living_conditions_(EU-SILC) – for brief information on EU-SILC

https://circabc.europa.eu/ – navigate "Browse categories" → "Eurostat" → "EU-SILC" → "Library". There you find nearly all available information on EU-SILC in general and specifically for the UDB. Under "02. Guidelines" you find the DocSILC065 with all variable information. Under "04. Data and Indicators Dissemination" → "4.1 User Database (UDB)" you find information regarding the UDB.

http://statistik.at/web_en/statistics/index.html – for more information on Statistics Austria and our products

References

Klotz, J. & Göllner, T. (2017). *Estimating Differential Mortality from EU-SILC Longitudinal Data. A Feasibility Study*. FACTAGE project report.

Kleinbaum, D. G., & Klein, M. (2005). *Survival Analysis: A self-learning text*. Springer.

Appendix

Exemplary Code for Loading of D File in CSV Format

Below you will find a typical import code for the D-file. Two important things: The data must follow the naming convention explained above: DFILE12, RFILE8, HFILE6, PFILE15, are all valid examples. And, the path stated after the `infile` must point to the correct csv file. An example path would be: `"D:\DATA\SILC_UDB\2016L\2016-1\2012\UDB_l12D.csv"` But this is of course up to your data structure. The rest of the code requires (usually) no change from the user.

```
data WORK.DFILE12;
     infile "C:\here\is-the\path\to-your\raw-files.csv"

delimiter = ',' MISSOVER DSD lrecl=32767 firstobs=2 ;
        informat DB010 best32. ;
        informat DB020 $2. ;
        informat DB030 best32. ;
        informat DB040 $4. ;
        informat DB040_F best32. ;
        informat DB060 best32. ;
        informat DB060_F best32. ;
        informat DB062 best32. ;
        informat DB062_F best32. ;
        informat DB070 best32. ;
        informat DB070_F best32. ;
        informat DB075 best32. ;
        informat DB075_F best32. ;
        informat DB090 best32. ;
        informat DB090_F best32. ;
        informat DB100 best32. ;
        informat DB100_F best32. ;
           informat DB110 best32. ;
        informat DB110_F best32. ;

        format DB010 best12. ;
        format DB020 $2. ;
        format DB030 best12. ;
        format DB040 $4. ;
        format DB040_F best12. ;
        format DB060 best12. ;
        format DB060_F best12. ;
        format DB062 best12. ;
        format DB062_F best12. ;
        format DB070 best12. ;
        format DB070_F best12. ;
        format DB075 best12. ;
        format DB075_F best12. ;
        format DB090 best12. ;
        format DB090_F best12. ;
        format DB100 best12. ;
        format DB100_F best12. ;
        format DB110 best12. ;
```

```
      format DB110_F best12. ;

   input
      DB010
      DB020 $
      DB030
      DB040 $
      DB040_F
      DB060
      DB060_F
      DB062
      DB062_F
      DB070 $
      DB070_F
      DB075
      DB075_F
      DB090
      DB090_F
      DB100
      DB100_F
      DB110
      DB110_F
      ;
         LABEL
         DB010 = "Year of the survey"
         DB020 = "Country"
         DB030 = "Household ID"
         DB040 = "Region"
         DB040_F = "DB040_F"
         DB060 = "PSU-1"
         DB060_F = "DB060_F"
         DB062 = "PSU-2"
         DB062_F = "DB062_F"
         DB070 = "Order of the selection of PSU"
         DB070_F = "DB070_F"
         DB075 = "Rotational group"
         DB075_F = "DB075_F"
         DB090 = "Household cross-sectional weight"
         DB090_F = "DB090_F"
         DB100 = "Degree of urbanisation"
         DB100_F = "DB100_F"
         DB110 = "Household status"
         DB110_F = "DB110_F"
;
run;
```