Swiss Household Panel (SHP)

Data management with SPSS

By Valérie-Anne Ryser and Florence Lebert

November 2025

How to cite this document:

Ryser, V.-A., & Lebert, F. (2025). *Swiss Household Panel (SHP). Data management with SPSS*. Lausanne: FORS.

Correspondence to:

Swiss Household Panel, FORS,

c/o University of Lausanne

Géopolis

CH-1015 Lausanne

swisspanel@fors.unil.ch

Schweizer Haushalt-Panel
Panel suisse de ménages
Swiss Household Panel

Table of contents

1. Introduction

The Swiss Household Panel (SHP) study is a rich data source that covers a wide range of topics, including health and well-being, employment, attitudes, social networks, and many more. The main advantages of the SHP are its longitudinal design and the ability to combine self-reported information from different household members. Starting in 1999 and interviewing all household members aged 14 years or older, the analytical potential is immense. However, the study has also become very complex over the years due to different aspects:

- The variety of data files available, including annual individual and household files, biographical data, social origin, and imputed income, among others.
- The different levels of data collection: individual, household, and proxy information.
- The large number of variables available.

Therefore, learning how to construct your own dataset to answer your specific research question is essential to working with the SHP data!

This document aims to guide SHP users through the essential data management steps to build their dataset of interest using the SPSS software. We will…

1) … give some tips on how to organise one's work and datasets with SPSS and how to select the variables of interest
2) … show how to merge different datasets, for instance, annual files or combining individual and household information
3) … discuss longitudinal datasets and show how to move from wide to long format
4) … demonstrate how to combine information from partners and show how to add information from mothers and fathers

This document provides users of the SHP data with step-by-step instructions and best practices, while also highlighting common pitfalls when using SPSS for SHP data management. By providing a structured, accessible, and reproducible framework for data analysis, this document will enable users to navigate some SPSS functionalities while efficiently minimising errors. This documentation will not be exhaustive, as there are often multiple ways to approach tasks, but it presents some of the methods we favour in our work.

2. The SPSS syntax and the benefits of using it

In its basic operations, the SPSS environment consists of several types of windows or screens. Each window corresponds to a specific task and a specific type of SPSS file. For this document, we will focus only on the three main standard windows: the *Data Editor,* where each column represents a variable and each row represents a participant or household; the *Output Viewer*, which displays a log of the actions taken and the associated output; and the *Syntax Editor,* which is a command language, used to perform data analysis, data management, or customization of outputs. We highly recommend using the latter as an alternative to the drop-down menus for data management and statistical analyses.

Before starting any data management or SHP analysis, it is important to consider how you want to organise your work. Start by creating a copy of your data: never work on your original data set! It is highly recommended to separate the raw or original data and the data sets you construct and analyse. You might also create a folder for your different data sets, a folder for the results, and one for your different syntaxes, including your data management and analysis syntaxes.

Many data manipulations are possible through buttons and SPSS dialogue boxes in the drop-down menus that allow the modification and analysis of the data. However, we will favour using the syntax for the management and analysis of the SHP data. The syntax is a command language that can be stored in a syntax file, which is a text file. The use of a syntax file has several main advantages:

1. Syntaxes provide a trace of the data manipulation and analyses performed and allow control of what has been done.
2. Syntaxes can easily be documented, making it easier to reproduce the files, restart the analysis, or undertake modifications.
3. A syntax can be reused for similar research questions or to add variables.
4. Using the syntax is quicker than the point-and-click interface if you do repetitive tasks.
5. The numerous SHP data files contain a huge number of variables. Writing the names of the variables of interest is easier and, most of the time, quicker than searching for them in the dialogue box's list.

Using the syntax saves time and organises your work. Organising the syntaxes is much more important than keeping data files.

SPSS uses colours and bolding to indicate the different functionalities of the terms in the syntax:

Grey	Comments must be introduced with an asterisk (*). It is essential to document the syntax: who wrote it, for which project, on which date, with which purpose, etc. The better the syntax documentation, the easier it will be to reuse or transmit it
Dark blue/purple	Procedure names; execution statements
Dark red/orange	Option keywords
Black	Variable names; other text
Green	Statements associated with the given procedure

It is highly recommended that you document the syntax commands. To do so, use the asterisk (*****) **as shown in the example below**. Any line starting with an asterisk will be recognised as text instead of a command and will be in light grey.

The syntax can also be used to open a data file, such as an individual annual file like SHP23_P_USER.sav, which contains information on all individuals of the SHP in the year 2023. To open a syntax file, go to the menu—FILE—and select—NEW—SYNTAX. A text file will open.

The SPSS command, strictly speaking, begins with the command name. In our example, it is **GET FILE**. The commands usually have additional specifications, such as subcommands and user specifications. The end of the command is always marked by a period/full stop. Note that SPSS is not case-sensitive, unlike other statistical software like Stata or R. However, we recommend keeping some logic and sticking to your personal choice.

The 'get file' SPSS command

```
***SHP GROUP.
***Teaching material.
***October 2025.

* Open a specific data file.
get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'.
exe.
***.
```

The command **GET FILE** followed by the equals sign **=**, the path where the file is located, and the file's name, allows opening a file. The path and the file name must be enclosed in quotes (single or double quotes can be used as long as the same type is used within a command), and for the data file, the extension **.sav** must be used[1]. The example above shows how to open a file stored on the **C**: drive in a folder called **Data_SPSS**[2]. To run one command, the cursor can be placed anywhere in the command. Then either click on the **right arrow** at the top of the syntax editor or use the keyboard shortcut **Ctrl + R**. It is also possible to run several commands together by selecting them.

An alternative way to use the SPSS syntax is by adapting the commands provided by SPSS. If you are unfamiliar with SPSS commands and syntax, you can start by creating your first command using the dialogue boxes. To do this, navigate to the menu and select the operation you wish to perform (e.g., TRANSFORM, ANALYZE). Then, click the "Paste" button in the dialogue box. SPSS will paste the corresponding syntax into the syntax editor, where it can be modified, executed, reused, and saved.

However, directly writing the syntax for tasks such as data management, variable selection, or constructing a partner file is often much faster than using dialogue boxes. In the upcoming chapters, we will therefore provide several examples to help you get started.

[1] Pay attention that for the paths, backslashes are used.
[2] All the examples presented in this document are based on an arborescence similar to the SHP data distribution structure.

It is possible to open multiple data sets simultaneously. This means that two or more different data files might be opened, such as several waves of SHP data. Accordingly, two or more *Data Editor windows* will be open as well. In this case, the syntax will be run on the "active" data set, i.e. the currently selected window. So, ensure the commands are executed on the right data set! Otherwise, you will receive error messages or wrong results.

3. Starting data management with the Swiss Household Panel

Before starting an analysis, we recommend focusing on the subset of variables relevant to the research question. The first step is to select the necessary variables while excluding unnecessary variables. As each wave of the SHP contains many variables, creating a smaller, more focused dataset will streamline the analytical work. Handling too many variables can be cumbersome, especially with complex data such as the SHP.

Construction of a reduced data file

Step 1: Select the variables of interest

To get a handle on the many variables, we recommend that you keep only the variables you are interested in. Two procedures are possible: you can either **KEEP** only the variables you need or **DELETE/DROP** the variables you do not need. However, the latter procedure could be cumbersome given the large number of variables in the SHP datasets.

The first example below shows you how to **KEEP** the variables of interest. In addition, the syntax presents the **SORT CASES** command. A sorted file is essential as the SHP is a longitudinal survey. The goal will be to have a longitudinal file; thus, sorted files are central to a successful merge of multiple data sets. Finally, we recommend saving the file using the command **SAVE OUTFILE**, sorted by the identifier IDPERS to facilitate the next steps.

Example of the SPSS “get file”, “keep”, “sort cases” and “save” command

```
***.
get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'
/keep IDPERS IDHOUS23 STATUS23 SEX23 AGE23 IDSPOU23 CIVSTA23 EDUCAT23 OCCUPA23
P23C44 P23C47 P23C48 P23C49 P23C50 P23C01 P23C02 I23PTOTN P23F01 P23F02 P23F03
P23F04.
sort cases by IDPERS (A) IDHOUS23 (A).
save outfile = 'C:\Data_SPSS\TEMP\SHP23_P_temp.sav'.
***.
```

The **GET FILE** command loads the dataset located at the specific path. The **KEEP** command specifies the variables to retain in the dataset; they will appear in the file in the same order as they are written. Variables not listed here will be dropped from the dataset. The **SORT** command sorts the data by IDPERS and IDHOUS23 in ascending order **(A)**. Sorting is often necessary for subsequent operations, such as merging datasets or performing analyses on ordered data. Finally, the **SAVE OUTFILE** command saves the processed dataset to the specified path under a new name, ensuring the original dataset remains unaltered.

If the variables you want to keep are side by side in the raw file "`SHP23_P_USER.sav`", for instance, by using the subcommand **TO,** you can save them in a new file, as in the fictive example below. SPSS will keep all the variables from var_1 (comprised) to var_13 (comprised). The rest of the procedure is identical to the previous example.

Generic example of the SPSS "get file", "keep", "sort cases" and "save" command if the variables of interest are side by side

```
***.
get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'
/keep IDPERS var1 to var13 var15 var16 var20 to var23.
sort cases by IDPERS (A) IDHOUS23 (A).
save outfile = 'C:\Data_SPSS\SHP23_P_temp.sav'.
***.
```

As shown in the next example, instead of keeping the variables you are interested in, you could also use the **DELETE** command to remove the variables you do not need. However, if you have a large number of variables, this may be more tedious. After having deleted the variables, it is recommended to save a sorted file as in the previous example.

Example of the SPSS "delete" command allowing the exclusion of variables

```
***.
get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'.
delete variables IDINT to P23MODES RELARP23 COHAST23 MAXCOP23 OWNKID23 EDCAT23 to
EDYEAR23 NAT_1_23 to P23D167 P23L01A to P23L95 P23C03 to P23I166 I23PTOTG to
I23OTHY P23F08 to X23I05.
sort cases by IDPERS (A) IDHOUS23 (A).
save outfile = 'C:\Data_SPSS\TEMP\SHP23_P_temp.sav'.
***.
```

The **GET FILE** command loads the dataset from the specified path. The **DELETE** command removes the specified variables from the dataset. The ranges (e.g., NAT_2_23 to P23D167) indicate that all variables between the start and end points in the dataset will be deleted. The **SORT CASES** command organises the data by IDPERS and IDHOUS23 in ascending order. Finally, the **SAVE OUTFILE** command stores the modified dataset under a new name at the given location, keeping the original file intact.

Another way to delete the variables you do not want to use in your analyses is to use the **DROP** subcommand. In the following example, the idea is to create a new data file where all the variables irrelevant to the research question are dropped as with the **DELETE** command.

In the example below, after opening the file "SHP23_P_USER.sav" with the **GET FILE** command, we present an alternative command, the **XSAVE OUTFILE** command. In this example, the **XSAVE OUTFILE** command is used to save a dataset to an external file. Unlike the **SAVE** command, which executes immediately, **XSAVE** is a transformation command. This means it does not execute until the data is read for the next procedure. This can help optimise processing time by consolidating multiple data passes into one.

Example of SPSS "drop" and "xsave" commands

```
***.
get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'.
sort cases by IDPERS (A) IDHOUS23 (A).

xsave outfile = 'C:\Data_SPSS\TEMP\SHP23_P_temp.sav'
 /drop = IDINT to P23MODES RELARP23 COHAST23 MAXCOP23 OWNKID23 EDCAT23 to EDYEAR23
NAT_1_23 to P23D167 P23L01A to P23L95 P23C03 to P23I166 I23PTOTG to I23OTHY P23F08
to X23I05.
exe.
***.
```

The **XSAVE** command saves the dataset to a new file (SHP23_P_temp.sav), dropping the specified variables in the process. This creates a new dataset (SHP23_P_temp.sav) with only the variables not included in the /drop list, preserving the original file (SHP23_P_USER.sav). SPSS executes commands in order. If you want to save the sorted data, you need to sort before saving

The **/DROP** command specifies the variables to be removed from the dataset before saving. Ranges such as `IDINT to P23MODES` mean all variables between `IDINT` and `P23MODES` (included) will be dropped.

Step 2: Select the observations with valid information

After selecting the variables of interest, it is possible to check the results based, for instance, on frequency tables. The **FREQUENCIES** command is used to create frequency tables, which display the counts and percentages of each unique value for a given variable. In addition, SPSS allows you to abbreviate commands to their first three letters. So, instead of writing **FREQUENCIES**, you can simply write **FRE**. In the example below, we created a frequency table for the STATUS23, `SEX23, and AGE23` variables.

Example of SPSS "frequencies" command

```
***.
fre STATUS23 SEX23 AGE23.
***.
```

Because the SHP uses several questionnaires (e.g., individual, proxy, household) and not every individual completes every questionnaire, the variable STATUS23 in this command will show you that there is only grid or proxy information for a certain number of SHP participants. Thus, the second step is to select only the observations with valid information. If the analytical unit is the individual, observations for whom only household or grid information is available should be deleted. The STATUS$$ variable ($$ represents the year of the interview) is central as it displays the number of observations with individual, proxy, and grid-only interviews. The syntax below opens the recently built individual file (SHP23_P_temp.sav) and selects only the individuals who responded to the individual questionnaire.

Example for selecting observations with valid information and creation of a frequency table

```
***.
get file = 'C:\Data_SPSS\TEMP\SHP23_P_temp.sav'.
fre STATUS23.
select if STATUS23 = 0.
exe.
fre STATUS23.

save outfile = 'C:\Data_SPSS\TEMP\SHP23_P_temp.sav'.
***.
```

The **SELECT IF** filters the dataset, retaining only rows where STATUS23 equals 0 (i.e., cases matching this specific status value). All other rows are excluded from the data file. The **FREQUENCIES** command creates the frequency tables to verify that only the information you are interested in are available in the dataset you want to work with.

4. Merging files

Having seen how to reduce datasets to the variables of interest and keep only the individual information, we will now show how to combine multiple datasets. This needs different procedures depending on the identifier variable.

4.1 Identifiers in the SHP

To combine two or more data sets, you need an identifier to be present in all files you want to merge. In the SHP, the identifiers listed in Table 1 are available:

Table 1: Identifier variables in the SHP

Type of file	Name of the data files	Identifier variables
Individual master file	SHP_MP.sav	idpers, idmoth$$[1], idfath$$, idhous$$
Household master file	SHP_MH.sav	idhous$$, refper$$
Individual annual files	SHP$$_P_USER.sav	idpers, idspou$$, idint[2], idhous$$
Household annual files	SHP$$_H_USER.sav	idhous$$, refper$$, idint, (gdenr[3])
Additional individual files (Social origin, last job, calendar)	SHP$$_SO.sav; SHP$$_LJ.sav SHP$$_CA.sav	idpers
Individual long file	SHPLONG_P_USER.sav	idpers, idspou, idint, idhous, year
Household long file	SHPLONG_P_USER.sav	idhous, refper, year
Interviewer data	SHP$$_V_USER.sav	idint
Biographic files 2013	SHPIII_CS_user.sav; SHPIII_FA_user.sav; SHPIII_HEA_user.sav; SHPIII_LA_user.sav; SHPIII_PM_user.sav; SHPIII_PROF_ACT_user.sav; SHPIII_RE_user.sav	idpers
Biographic files 2001-2002	SHP0_BH_USER.sav; SHP0_BVCS_USER.sav; SHP0_BVED_USER.sav; SHP0_BVFE_USER.sav; SHP0_BVLA_USER.sav; SHP0_BVLP_USER.sav; SHP0_BVRE_USER.sav; SHP0_BVSA_USER.sav; SHP0_BVWL_USER.sav; SHP0_MBI.sav	idpers
Imputed wealth	imputed_wealth_2012_2020.sav	idhous
Imputed individual income	imputed_income_pers_long_shp.sav; imputed_income_pers_wide_shp.sav	idpers, year idpers
Imputed household income	imputed_income_hh_long_shp.sav; imputed_income_hh_wide_shp.sav	idhous, year idhous
Cross-national equivalent (CNEF) files	shpequiv_$$$$ x11101ll	x11101ll (=idpers) x11102_1999 - x11102_2023 (=idhous$$)

Notes: [1]$$ represents the year of the interview; [2]The interviewer data can only be merged with other data files after decrypting idint. Please contact Oliver Lipps for more details (oliver.lipps@fors.unil.ch).[3] Municipality number of the Swiss Federal Statistical Office.

4.2 One-to-one merge

If you have two or more data files, you can combine them by merging them side by side, matching up observations based on an identifier.

For example, both the files **SHP22_P_USER.sav** and **SHP23_P_USER.sav** contain the identifier called **IDPERS**. We want to merge the files so that the **SHP22_P_USER.sav** observations are on the same line as the **SHP23_P_USER.sav** observations based on the key variable **IDPERS**. To do so, we will use the **MATCH FILES** command.

To merge two files, the next steps must be followed:

1. Use **SORT CASES**[3] to sort **IDPERS** in SHP22_P_USER.sav and save that file.
2. Use **SORT CASES** to sort **IDPERS** in SHP23_P_USER.sav and save that file.
3. Use **MATCH FILES** to merge SHP22_P_USER.sav and SHP23_P_USER.sav based on **IDPERS**.

Example of the steps to merge two annual data files

```
*get file and sort cases.

get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W24_2022\SHP22_P_USER.sav'.
sort cases by IDPERS (A).
save outfile = 'C:\Data_SPSS\TEMP\SHP22_P_temp.sav'.

get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'.
sort cases by IDPERS (A).
save outfile = 'C:\Data_SPSS\TEMP\SHP23_P_temp.sav'.

*the match file command.

match files
  /file = 'C:\Data_SPSS\TEMP\SHP22_P_temp.sav'
  /file = 'C:\Data_SPSS\TEMP\SHP23_P_temp.sav'
  /by IDPERS.
exe.
save outfile = 'C:\Data_SPSS\TEMP\SHP22_23_temp.sav'.
***.
```

After the datasets have been loaded and after being sure that each dataset is organized by IDPERS for compatibility during merging, the datasets are matched (using the **MATCH FILES** command) based on the common key variable IDPERS (declared with **/BY** IDPERS). This ensures that rows in both datasets corresponding to the same individual (identified by IDPERS) are aligned and merged correctly. The **EXE** command, which is the short form of the **EXECUTE** command, executes the merging operation. The resulting combined dataset is saved as **SHP22_23_temp.sav**, preserving the merged data while keeping the original files intact.

In the SHP, new household members join the survey annually. This means that observations for an extra individual arriving in 2025 will not match the data in the individual file in 2024. *Appendix 1* addresses this issue in a more detailed way.

[3] Note that SPSS sorts the cases in an ascending order (A). You can also sort them in descending order. Therefore, write “sort cases by IDPERS (D)”.

4.3 One-to-many merge

The next example considers a **one-to-many merge,** where one observation in a file may have multiple matching records in another file. For instance, household information concerns multiple individuals if there is more than one household member. In the same vein, interviewer information concerns multiple respondents if the interviewer has performed more than one interview.

The example below shows how to merge the household information of all household members. We call this a **one-to-many** merge since one household observation is combined with one or many individual observations because a household might comprise more than one individual. The household file **SHP23_H_USER.sav** contains one observation per household, whereas the **SHP23_P_USER.sav** file might have multiple observations within a household. The identifier to be used is **IDHOUS$$**.

IDPERS	IDHOUS23	AGE23
4101	41	37
4102	41	34
4103	41	11
5134	51	62
5135	51	69

IDHOUS23	HDATE23	HLINGU23	CANTON23
41	17.09.2023	1	8
51	19.11.2023	2	18

Figure 1: Illustration of an individual and a household file to be merged in the next step

As you see below, the steps for doing a **one-to-many** merge are quite similar to the **one-to-one** merge that we saw above.

1. Use **SORT CASES BY** to sort IDHOUS23 in the household file and save that file (with the same name or with another name)
2. Use **SORT CASES BY** to sort IDHOUS23 in the individual file and save that file
3. Use **MATCH FILES** to merge the household and individual files. However, since the SHP23_H_USER.sav file is the file with one observation, use **/TABLE** = 'SHP23_H_USER.sav' to specify the household file

The key difference between a **one-to-one merge** and a **one-to-many** merge is that you need to use **/TABLE** instead of **/FILE**. For the construction of a data file, two questions need to be asked:

Do I have a one-to-one merge?

If no:

Which file plays the role of one? For that file, use /TABLE= instead of /FILE=.

Example of the SPSS command:

```
***.
get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'.
sort cases by IDHOUS23 (A).
save outfile = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'.

get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_H_USER.sav'.
sort cases by IDHOUS23 (A).
save outfile = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_H_USER.sav'.

match files
  /file ='C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'
  /table ='C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_H_USER.sav'
  /by IDHOUS23.
execute.

save outfile = 'C:\Data_SPSS\TEMP\SHP_P_H_23.sav'.

***.
```

Before using the **MATCH FILES** command, make sure that each dataset is organised by **IDHOUS23** for compatibility during merging. The match files command is used to combine two datasets based on shared values in one or more key variables. It aligns rows across datasets where the key variable matches.

The active dataset (**SHP23_P_USER.sav**) is the one currently being worked on and will be matched with the second dataset. The second dataset (**SHP23_H_USER.sav**) will be used for matching. This file contains additional variables or cases related to the active dataset.

IDHOUS23 is the household identifier and the key variable used to match cases between the two datasets. Rows from both datasets with the same **IDHOUS23** value will be aligned during the merge.

This operation is typically performed to integrate household-level data (from **SHP23_H_USER.sav**) into a person-level dataset (**SHP23_P_USER.sav**) for comprehensive analysis. By merging these datasets, you can link individual data with household-specific information (see Figure 2 below).

IDPERS	IDHOUS23	AGE23	HDATE23	HLINGU23	CANTON23
4101	41	37	17.09.2023	1	8
4102	41	34	17.09.2023	1	8
4103	41	11	17.09.2023	1	8
5134	51	62	19.11.2023	2	18
5135	51	69	19.11.2023	2	18

Figure 2: Illustration of the output of a one-to-many merge of an individual and a household file based on Figure 1

5. Creating longitudinal data files

There are two ways to display panel data: the wide format and the long format. In the wide format, each respondent is represented by a single row containing distinct variables for each wave. This structure organises data such that multiple observations for an individual interviewed annually are consolidated into a single row. In other words, in the wide format, each observation stands for one respondent, and the variables are their characteristics at the respective time points.

In the long format, also called person-period file, each row/observation corresponds to a respondent at a specific point in time (wave). Therefore, each respondent has as many lines as the number of waves in which they participated. If a respondent or household participated in just one wave, there is only one row for them.

5.1 Wide format

A data set in the wide format is obtained by using the one-to-one merge described in section 4.2. Each variable of an additional year of observation that is merged onto the data set is displayed as an additional column, as illustrated in the Figure below. An identifying variable is needed in all data sets to merge them (here it is IDPERS). The identifier variable assures that one individual's values are on the same line. The same logic applies to the household files.

IDPERS	AGE21
4101	40
4102	37
4103	14
4104	12
4105	9

IDPERS	AGE22
4101	41
4102	38
4103	15
4104	13
4105	10

IDPERS	AGE23
4101	42
4102	39
4103	16
4104	14
4105	11

IDPERS	AGE21	AGE22	AGE23
4101	40	41	42
4102	37	38	39
4103	14	15	16
4104	12	13	14
4105	9	10	11

Figure 3: Illustration of a wide file

5.2 Long format

The second format for longitudinal data is the long format, i.e. the person-period file. In the person-period file, every unit has as many lines as there are time points available for the unit (see Figure 4 below). This long format is often used for panel analysis, such as multilevel models (MLM hierarchical, random effects, mixed effects models), fixed effects or first-difference models, and survival analysis.

IDPERS	YEAR	IDHOUS	SEX	AGE	PC44	PC47
5032	1999	712	1	34	9	8
5032	2000	712	1	35	8	7
5032	2001	712	1	36	8	9
5033	1999	712	2	31	7	6
5033	2000	712	2	32	8	8
5033	2001	712	2	33	7	7

Figure 4: Illustration of a long file

To build a person-period file in SPSS, you have to go through the construction of a wide format in a first step. In a second step, you can build your long file as shown in what follows.

The /**VARSTOCASES** command reshapes the data from wide to long format. However, first, the data can be organised, with the variables listed side by side as shown in the syntax below using the keep command.

Example of the 'keep' command to organise the variables side by side; - useful but not necessary-

```
***.
get file = 'C:\Data_SPSS\TEMP\SHP_P_W1_W25.sav'
/keep IDPERS BIRTHY
IDHOUS99 IDHOUS00 IDHOUS01 IDHOUS02 IDHOUS03 IDHOUS04 IDHOUS05 IDHOUS06 IDHOUS07
IDHOUS08 IDHOUS09 IDHOUS10 IDHOUS11 IDHOUS12 IDHOUS13 IDHOUS14 IDHOUS15 IDHOUS16
IDHOUS17 IDHOUS18 IDHOUS19 IDHOUS20 IDHOUS21 IDHOUS22 IDHOUS23
AGE99 AGE00 AGE01 AGE02 AGE03 AGE04 AGE05 AGE06 AGE07 AGE08 AGE09 AGE10 AGE11 AGE12
AGE13 AGE14 AGE15 AGE16 AGE17 AGE18 AGE19 AGE20 AGE21 AGE22 AGE23
STATUS99 STATUS00 STATUS01 STATUS02 STATUS03 STATUS04 STATUS05 STATUS06 STATUS07
STATUS08 STATUS09 STATUS10 STATUS11 STATUS12 STATUS13 STATUS14 STATUS15 STATUS16
STATUS17 STATUS18 STATUS19 STATUS20 STATUS21 STATUS22 STATUS23
P99C44 P00C44 P01C44 P02C44 P03C44 P04C44 P05C44 P06C44 P07C44 P08C44 P09C44 P10C44
P11C44 P12C44 P13C44 P14C44 P15C44 P16C44 P17C44 P18C44 P19C44 P20C44 P21C44 P22C44
P23C44.
sort cases by IDPERS (A).
save outfile = 'C:\Data_SPSS\TEMP\SHP_P_W1_W25.sav'.
***.
```

The **VARSTOCASES** command is used to reshape the data from the wide format to the long format. The **VARSTOCASES** command collapses all the variables into a new variable.

At the same time, a new variable, INDEX, is created to index the waves of observation for each subject.

The subcommand **/KEEP** means that variables that are constant within a subject (here IDPERS) are kept.

VARSTOCASES command: person-period file, individual and household information

```
***.
get file = 'C:\Data_SPSS\TEMP\SHP_P_W1_W25.sav'.

varstocases
 /make IDHOUS from IDHOUS99 IDHOUS00 IDHOUS01 IDHOUS02 IDHOUS03 IDHOUS04 IDHOUS05
IDHOUS06 IDHOUS07 IDHOUS08 IDHOUS09 IDHOUS10 IDHOUS11 IDHOUS12 IDHOUS13 IDHOUS14
IDHOUS15 IDHOUS16 IDHOUS17 IDHOUS18 IDHOUS19 IDHOUS20 IDHOUS21 IDHOUS22 IDHOUS23
 /make AGE from AGE99 AGE00 AGE01 AGE02 AGE03 AGE04 AGE05 AGE06 AGE07 AGE08 AGE09
AGE10 AGE11 AGE12 AGE13 AGE14 AGE15 AGE16 AGE17 AGE18 AGE19 AGE20 AGE21 AGE22 AGE23
 /make STATUS from STATUS99 STATUS00 STATUS01 STATUS02 STATUS03 STATUS04 STATUS05
STATUS06 STATUS07 STATUS08 STATUS09 STATUS10 STATUS11 STATUS12 STATUS13 STATUS14
STATUS15 STATUS16 STATUS17 STATUS18 STATUS19 STATUS20 STATUS21 STATUS22 STATUS23
 /make PC44 from P99C44 P00C44 P01C44 P02C44 P03C44 P04C44 P05C44 P06C44 P07C44
P08C44 P09C44 P10C44 P11C44 P12C44 P13C44 P14C44 P15C44 P16C44 P17C44 P18C44 P19C44
P20C44 P21C44 P22C44 P23C44
  /index = Index1(25)
 /keep = IDPERS BIRTHY
 /null = keep.
save outfile = 'C:\Data_SPSS\TEMP\SHP_P_longfile.sav'.
***.
```

The **GET FILE** command opens the dataset stored in the specified file path. The **VARSTOCASES** command reshapes the dataset from a wide format (where multiple variables represent observations over time) into a long format (where one variable represents repeated measures over time). The **/MAKE** subcommands create new variables (idhous, age, status, PC44) by combining variables that represent the same concept across different years (e.g., idhous99, idhous00, etc., for idhous). This allows SPSS to group the repeated measures for each variable into a single column, simplifying longitudinal analysis. The **/INDEX** assigns a numerical index (Index1) to distinguish repeated observations. The value 25 indicates that there are 25 time points (e.g., years or waves of data). The **/KEEP** command ensures that specific variables (IDPERS, BIRTHY) remain in the reshaped dataset. These might represent key individual characteristics or identifiers. The **/NULL** option defines how system-missing or blank values are handled. In this case, **KEEP** indicates that null values should remain in the dataset.

If the data is sorted by variable and wave, there is a shorter way to transform the data into a long file: Instead of listing all the variables, you can treat them as a series using **TO,** to tell SPSS which variables have to be accounted for.

Example of the VARSTOCASES command with data sorted by variable and wave:

```
***.
get file = 'C:\Data_SPSS\TEMP\SHP_P_W1_W25.sav'.

varstocases /id = id "wave"
 /make IDHOUS from IDHOUS99 TO IDHOUS23
 /make AGE from AGE99 TO AGE23
 /make STATUS from STATUS99 TO STATUS23
 /make PC44 from P99C44 TO P23C44
  /index = Index1(25)
 /keep = IDPERS BIRTHY
 /null = keep.
save outfile = 'C:\Data_SPSS\TEMP\SHP_P_longfile.sav'.

***.
```

In the examples above, the variable BIRTHY was imported from the SHP_MP.sav file, meaning that this information was merged during data integration.

The CASESTOVARS command

If you have a long data file and want to reshape it into a wide format, use the **CASESTOVARS** command. IDPERS is the identifier variable, and the variable to organise the time dimension is YEAR. The **/FIXED** subcommand specifies which variables do not change over time. To avoid any problems, we recommend sorting the dataset before running the **CASESTOVARS** command using **SORT CASES**.

Example of the CASESTOVARS command:

```
***.
get file = 'C:\Data_SPSS\TEMP\SHP_P_longfile.sav'.
sort cases by IDPERS (A) YEAR (A).

casestovars
  /id = IDPERS
  /fixed = BIRTHY.
save outfile = 'C:\Data_SPSS\TEMP\SHP_P_widefile.sav'.


***.
```

The **CASESTOVARS** command reshapes the dataset from a long format (where each row represents a time point for a unit) back to a wide format (where each unit has a single row, and variables are repeated across columns for different time points).

The **/ID** = IDPERS option specifies IDPERS as the unique identifier, grouping all rows for each individual into one single row.

The **/FIXED** = BRITHY option indicates that BRITHY is a fixed variable, meaning its values remain constant across all time points and do not require replication across new columns.

The long file is shaped back into the wide format:

IDPERS	SEX	AGE.1	AGE.2	AGE.3	AGE.4	AGE.5	AGE.6
4101	1	37	38	39	40	41	42
4102	2	34	35	36	37	38	39
4103	1	15	16	17	18	19	20

Figure 5: Illustration of the data after a CASESTOVARS transformation

5.3 Lag variable

Depending on your research question, a variable reporting the value at the previous wave can be helpful when you have a longitudinal file.

IDPERS	YEAR	SEX	EDUC	PC44	PC44_lag	PC47	PC47_lag
4101	1999	1	2	6	-	8	-
4101	2000	1	2	7	6	10	8
4101	2001	1	2	0	7	7	10
4101	2002	1	2	7	0	8	7
4102	1999	2	4	7	-	6	-
4102	2000	2	4	8	7	7	6
4102	2001	2	4	5	8	5	7

Figure 6: Illustration of an output that comprises two lag variables

The syntax below calculates lagged values for two variables (PC44 and PC47), but only for rows where IDPERS matches the value in the previous row. The resulting variables (PC44_lag and PC47_lag) can be used to analyse temporal relationships within groups of data (e.g., individuals in a longitudinal dataset).

Example of the lag SPSS command

```
***.
*Lag command.

do if IDPERS = lag(IDPERS).
       compute PC44_lag = lag(PC44).
       compute PC47_lag = lag(PC47).
end if.
execute.
***.
```

The **DO IF** command specifies a condition under which the following computations will be applied. IDPERS = lag(IDPERS) checks if the value of the variable IDPERS in the current row is equal to its value in the previous row. This condition ensures that the subsequent computations are performed only for cases where IDPERS is repeated (e.g., multiple rows representing the same individual in longitudinal or panel data).

The **COMPUTE** command creates new variables (PC44_lag and PC47_lag) that store the lagged values of PC44 and PC47, respectively. lag(PC44) and lag(PC47) take the values of PC44 and PC47 from the row immediately above the current one (if it is the same IDPERS) and assign them to the new variables.

These lagged variables are useful for analysing changes over time or relationships between the current and previous values within the same group (e.g., an individual in this case).

The **END IF** command marks the end of the conditional block. It ensures that the computations specified within the **DO IF** statement apply only to cases that meet the condition IDPERS = lag(IDPERS).

The **EXECUTE** command applies all pending transformations (defined by the conditional statement and computations) to the dataset.

5.4 The ready-to-use long file of the SHP data

In 2024, the SHP released two ready-to-use longitudinal datasets (individual and household files separately). These files contain all the annual files combined into a single longitudinal file. The variable names are the same as in the annual user files, but without the year identifier. A specific User Guide is available on the [SHP website] and on [SWISSUbase] where you download the SHP data. Please read this User Guide to be aware of the changes in the data over time.

The two main long files are the following:

> **SHPLONG_P_USER.sav:** Individual long file containing, with some exceptions, the same variables and cases as the annual individual files (SHP$$_P_USER.sav).
>
> **SHPLONG_H_USER.sav:** Household long file containing, with some exceptions, the same variables and cases as the annual household files (SHP$$_H_USER.sav).

These longitudinal data files are person/household-period files. Each unit has as many lines as the number of SHP waves it participated in. If a unit participated in only one wave, there is just one row for this unit. Each row corresponds to a unit at a specific point in time, with variables representing their observed characteristics. The variable IDPERS (IDHOUS) identifies the individual (household), the variable YEAR identifies the year of observation.

5.4.1 Reduce the variables of the long file

As the SHP long files comprise a large number of variables, we recommend selecting only the variables of interest, as we have shown in section 3. The procedures are the same for selecting variables in long files as in the annual files. Just make sure that you sort by both the unit identifier (e.g. IDPERS) and YEAR.

Example of 'KEEP', 'SORT CASES', and 'SAVE' command

```
***.
get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\SHPLONG_P_USER.sav'
/keep IDPERS YEAR IDHOUS STATUS SEX AGE IDSPOU CIVSTA EDUCAT OCCUPA PC44 PC47 PC48
PC49 PC50 PC01 PC02 IPTOTN PF01 PF02 PF03 PF04.
sort cases by IDPERS (A) YEAR (A).
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\SHPlong_p_reduced.sav'.
***.
```

The first line of the syntax opens the file '**SHPLONG_P_USER.SAV**', then retains only the listed variables with the **KEEP** command, removing unnecessary ones. Then it sorts the dataset by IDPERS and YEAR in ascending order and saves the cleaned and sorted dataset as a new reduced file, called **shplong_p_reduced.sav**.

5.4.2 One-to-one merge with the long file

The logic for a one-to-one merge with the long file is the same as for the annual files. However, in the individual long file, the individual is represented as many times as the number of times he or she has participated in the survey. Thus, the IDPERS identifier is insufficient; the correct data must be merged with the right year. Therefore, two identifiers are needed: IDPERS and YEAR. Both identifiers must be sorted in the same way before the merge.

Example of SPSS command for merging two long data files

```
***.
get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_reduced.sav'.
match files/file =*
  /file = 'C:\Data_SPSS\SHP-Data-Imputed-Income-Wealth-
SPSS\imputed_income_pers_long_shp.sav'
  /by IDPERS YEAR.
exe.
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_inc.sav'.
***.
```

In the SPSS syntax, the asterisk (*) refers to the data file that is already in memory—the active dataset. Therefore, the MATCH FILES/FILE=* command in the second part of the syntax tells SPSS to match the file currently loaded in memory with the external file specified in the **/file** subcommand.

In the example above, the syntax performs the following steps:

1. The **GET FILE** command reads the first data file, shplong_p_reduced.sav, into memory. This becomes the active dataset.
2. The **MATCH FILES** command then matches this active dataset (denoted by file = *) with the second file, 'imputed_income_pers_long_shp.sav', based on the specified key variables **/by IDPERS YEAR**.

3. The **exe.** command executes the operation.

This procedure is often used to merge datasets by matching cases on common identifier variables (e.g., **IDPERS** and **YEAR** in the example).

5.4.3 One-to-many merge with the long file

The logic for the one-to-many merge of a long file is similar to the merge of annual files. The only difference is again that two identifiers are needed: IDHOUS and YEAR. As each household and individual is in the file as many times as he/she participated in the survey, the year must be taken into consideration when merging individual and household information.

Example of the command to match different long files

```
***.
match files /file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_reduced.sav'
  /table = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_h_reduced.sav'
  /by IDHOUS YEAR.
exe.
save outfile = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\shplong_p_h_reduced.sav'.
***.
```

This example assumes that before the merge, the two datasets are sorted by IDHOUS and YEAR in ascending order.

6. Adding information from other household members

The SHP interviews all individuals 14 years of age or older who live in the same household. This opens up many analytical possibilities, such as looking at influences between partners or parents and children. Section 6.1 shows how to construct a partner file (first cross-sectional, then longitudinal), and Section 6.2 shows how to add information from mothers and fathers.

6.1 Creating a file with partner information

The cross-sectional partner file

Figure 7 illustrates what a partner file looks like. For each respondent in the individual file, partner information is added as additional variables if a spouse is present in the household.

Individual file

IDPERS	IDHOUS23	IDSPOU23	SEX23	EDUCAT23	P23F16
4102	41	4101	1	8	2
4101	41	4102	2	1	19

Output

IDPERS	IDHOUS23	IDSPOU23	SEX23	SEX23_P	EDUCAT23	EDUCAT23_P	P23F16	P23F16_P
4102	41	4101	1	2	8	1	2	19
4101	41	4102	2	1	1	8	19	2

Figure 7: Illustration of a partner file

To construct a partner file, the following steps are necessary:

1. Call the data file of interest (drop the IDSPOU$$ variable if it is included)
2. Create an identifier variable by renaming the IDPERS into IDSPOU$$
3. Rename the variables, adding the suffix _p (we are pretending that every individual is the partner of someone in the data)
4. Sort by IDSPOU$$ and save as a temporary partner file
5. Match files with the **/TABLE** command
6. Select only cases with a partner

```
Example syntax for the creation of a cross-sectional partner file:

*Call the data file of interest, rename IDPERS into IDSPOU23, add the suffix _p to
all the variables, sort by IDSPOU23 and save the file.

get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'
/keep IDPERS STATUS23 AGE23 SEX23 EDUCAT23 P23C44.
rename var IDPERS = IDSPOU23.
rename var STATUS23 = STATUS23_p
rename var AGE23 = AGE23_p.
rename var SEX23 = SEX23_p.
rename var EDUCAT23 = EDUCAT23_p.
rename var P23C44 = P23C44_p.
execute.
sort cases by IDSPOU23.
save outfile = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\2023temp_partner.sav'.

*Match the original file with the temporary partner file

get file = 'C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav'.
sort cases by IDSPOU23 (A).
exe.

match files
  /file = *
  /table = 'C:\DATA_SPSS\SHP-DATA-W1-W25-SPSS\W25_2023\2023temp_partner.sav'
  /by IDSPOU23.
exe.

*Select only cases with a partner.

select if not missing(IDSPOU23).
sort cases by IDPERS (A).
save outfile = 'C:\Data_SPSS\TEMP\shp23_partner.sav'.
***.
```

Step 1: Renaming variables to create partner information

The dataset **SHP23_P_USER.sav** is loaded, and the variables of interest are selected. Then all the variables are renamed with the suffix **_p**. The suffix **_p** clearly identifies the partner's information and avoids potential conflicts during subsequent steps. For example, the variable AGE23 is renamed as AGE23_p. To match the partner information (i.e. the variables with the suffix _p) with the original data, we have to create the identifier variable IDSPOU23 in the dataset containing the _p variables. As the IDSPOU23 corresponds to the IDPERS of the partner, we do this by renaming IDPERS into IDSPOU23. The execute command is used to apply these changes immediately.

Sort the new dataset by IDSPOU23 and save it as 2023temp_partner.sav. This ensures that the renamed dataset is preserved separately, leaving the original file intact for future use.

Step 2: Matching datasets

The original dataset, SHP23_P_USER.sav, is loaded again for a fresh operation. To prepare for merging, cases are sorted by the variable IDSPOU23 in ascending order. The MATCH FILES command is then used to merge the active dataset (*) with the file 2023temp_partner.sav. Cases are matched using the key variable IDSPOU23, resulting in a combined dataset containing variables from both sources.

Step 3: Select only cases with a partner

As we have pretended that every individual is the partner of someone in the data, in the last step, we filter the dataset using the SELECT IF command to retain only cases where IDSPOU23 is not missing. These filtered cases represent all individuals with a partner. However, be aware that some partners might not have participated in the individual interview (use the variable STATUS23_p to identify partners with valid information). The dataset is then sorted by IDPERS in ascending order to maintain an organised data structure.

Finally, the processed dataset is saved as shp23_partner.sav, ensuring that only relevant records with partner data are preserved for subsequent analysis.

The longitudinal partner file

The example below shows how the partner information is added to the individual long file. The logic for creating a partner file for the long file is similar to the one used to create a cross-sectional partner file.

```
Example syntax for a longitudinal partner file:
***.
get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_reduced.sav'.
delete variables IDSPOU.
rename var IDPERS = IDSPOU.
rename var AGE = AGE_p.
rename var SEX = SEX_p.
rename var EDUCAT= EDUCAT_p.
rename var PC44 = PC44_p.
exe.
sort cases by IDSPOU(A) YEAR(A).
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_partner0.sav'.

get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_reduced.sav'.
sort cases by IDSPOU(A) YEAR (A).
match files /file=*
  /table = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_partner0.sav'
  /by IDSPOU YEAR.
exe.
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_partner.sav'.
select if not missing(IDSPOU).   /* select only cases having a partner
sort cases by IDSPOU(A) YEAR (A).
exe.
```

This syntax prepares a dataset for analysis involving individuals and their partners by renaming, sorting, merging, and filtering data. It creates a final dataset comprising only individuals with valid partner information. The IDSPOU variable is the identifier used to match the files.

6.2 Adding information about the mothers and fathers of the respondents

Adding information about fathers and mothers is similar to constructing a partner file. One additional step is necessary because the fathers' and mothers' identifiers are not in the annual individual file/long file but in the individual master file and need to be added to the annual individual file or the long file first. Refer to Appendix 1 for detailed instructions on how to add the variables IDMOTH$$ and IDFATH$$ to the long file.

Example of the SPSS syntax for adding information about the respondent's mother

```
*Create a file containing the mother's variables.

get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_REDUCED.sav'.
delete variables IDMOTH$$.
rename var IDPERS = IDMOTH$$.
rename var AGE = AGE_mo.
rename var SEX = SEX_mo.
rename var EDUCAT = EDUCAT_mo.
rename var PC44 = PC44_mo.
execute.
sort cases by IDMOTH$$ (A) YEAR (A).
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_mother.sav'.
***.
```

As for the mother, a file with the father's information is created.

Example of the SPSS syntax for adding information about the respondent's father

```
***.
get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_REDUCED.sav'.
delete variables IDFATH$$.
rename var IDPERS = IDFATH$$.
rename var AGE = AGE_fa.
rename var SEX = SEX_fa.
rename var EDUCAT = EDUCAT_fa.
rename var PC44 = PC44_fa.
exe.
sort cases by IDFATH$$ (A) YEAR (A).
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_father.sav'.
***.
```

The last step consists of merging all these files.

Example of a SPSS syntax for merging individual information with mothers' and fathers' information

```
***.
get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_REDUCED.sav'.
sort cases by IDMOTH$$ (A) YEAR (A).
match files /file=*
  /table = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_mother.sav'
  /by IDMOTH$$ YEAR.
execute.
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_mother_tot.sav'.

get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_mother_tot.sav'.
sort cases by IDFATH$$ (A) YEAR (A).
match files /file=*
  /table='C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_father.sav'
  /by IDFATH$$ YEAR.
execute.

sort cases by IDPERS (A) YEAR (A).
exe.

save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_mother_father.sav'.

***.
```

Note that the respondents' and their parents' data can only be combined if they have lived in the same household at some point in time during the SHP survey. If this is not the case, information on the parents is available in the social origin file (SHP_SO.sav) for all individuals who were older than 20 years at the first interview. However, the information in the social origin file differs significantly from the information in the annual file in two respects.

1. First, the annual files include actual answers given by the parents themselves, whereas the social origin variables contain information about the parents given by the respondent.
2. Second, the annual file refers to the present situation during the interview. In contrast, the social origin variables are retrospective questions about the parents' situation when the respondent was about 15 years old.

7. Recommendations

- You should never work on the original data file! Just make a copy of it and work on the copy.
- Use the syntax instead of the point-and-click interface whenever possible.
- While you do data management, do not forget to sort your cases by the identifier. It can save you from unpleasant surprises.
- Be careful with the missing values (missing declaration, among others).

- Write the path without any mistakes, without forgetting dots, commas, and quotes.
- Regularly check your data when constructing your own data set to spot problems in your file construction.
- Comment your syntax for yourself and others – you might restart the analyses for a publication based on your supervisors' or reviewers' comments, or you might share your syntax later in your research process.

8. Further reading

More detailed information on the Swiss household panel

Tillmann, R., Voorpostel, M., Antal, E., Dasoki, N., Klaas, H., Kuhn, U., Lebert, F., Monsch, G.-A., & Ryser, V.-A. (2021). The Swiss Household Panel (SHP). *Jahrbücher für Nationalökonomie und Statistik, 242*(3), 403-420. https://doi.org/10.1515/jbnst-2021-0039

Voorpostel, M., Tillmann, R, Lebert, F., Kuhn, U., Lipps, O., Ryser, V.-A., Antal, E., Dasoki, N., Janssen, C., & Wernli, B. (2025). *Swiss Household Panel User Guide (1999-2023), Wave 25, April 2025.* Lausanne: FORS.

Online resources

In SPSS go to "Help" in the menu bar and choose "Topics", "Tutorial" or "Command Syntax Reference". Raynald Levesque. *Raynald's SPSS Tools*. http://www.spsstools.net/

Statistical Methods and Data Analytics. *SPSS Learning Modules*. UCLA. https://stats.oarc.ucla.edu/spss/modules/

University Libraries. *SPSS Tutorials*. Kent State University. https://libguides.library.kent.edu/SPSS/

Books

Field, A. (2024). Discovering Statistics Using IBM SPSS Statistics (6th ed.). Sage.

Pallant, J. (2020). *SPSS Survival Manual: A Step by Step Guide to Data Analysis Using IBM SPSS*. Routledge. https://doi.org/10.4324/9781003117452

9. Appendix

9.1 Appendix 1: Mismatching records in one-to-one merge

This Appendix shows you how to manage mismatching records in a one-to-one merge, as in the SHP, new household members join the survey annually. This means that observations for an extra individual arriving in 2025 will not match the data in the individual file in 2024.

To identify which record comes from the SHP99_P_USER.sav file or from the SHP23_P_USER.sav, the use of the subcommand **/IN=fromind_23** creates a 0/1 variable that indicates which records come from which file.

Example for identifying which records come from which file

```
*/IN command: example w1 1999 – w2 2023.
match files
/file ='C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W1_1999\SHP99_P_USER.sav' /IN
fromind_99
...
/file ='C:\Data_SPSS\SHP-Data-W1-W25-SPSS\W25_2023\SHP23_P_USER.sav' /IN
fromind_23
/by idpers.
list.

crosstabs
/tables = fromind_99 by fromind_23.
***.
```

The **/IN** subcommand creates a new indicator (binary) variable, in this case named fromind_$$. This variable is automatically assigned values:

- 1 for cases that exist in the specified file (SHP99_P_USER.sav).
- 0 for cases that do not exist in the file (if merged with other datasets later).

The purpose of this indicator variable is to help identify whether each case originated from the loaded file during subsequent operations, such as file merging.

The **LIST** and **CROSSTABS** show the mismatched records. For the SHP, considering the number of variables, the command **LIST** is interesting to use after a selection of variables. If you want to have individuals or households followed for a certain number of years, you can use the command **SELECT IF** to eliminate the non-matching records.

Example of the 'select if' command

```
***.
select if fromind_22 AND fromind_23.
list.
***.
```

9.2 Appendix 2: Adding mothers' and fathers' IDs in the long dataset

First, create a reduced version of the MP file using the **KEEP** command. Then, merge the information from the MP file into the long-format dataset using the variable **IDPERS** as the key. Since the MP file contains one piece of information per respondent—regardless of how many times they participated in the survey—it serves as a one-to-one reference file. For this file, use the **/TABLE** option.

Example of SPSS command to merge the information of the individual master file and the long personal file

```
***.
get file='C:\Data_SPSS\SHP-Data-WA-SPSS\SHP_MP.sav'
  /keep IDPERS IDMOTH$$ IDFATH$$ BIRTHM BIRTHY.
sort cases by IDPERS (A).
save outfile = 'C:\Data_SPSS\TEMP\MP_reduced.sav'.

get file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_reduced.sav'.
sort cases by IDPERS (A).
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_reduced.sav'.

***.

match files
  /file = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_reduced.sav'
  /table = 'C:\Data_SPSS\TEMP\MP_reduced.sav'
  /by IDPERS.
exe.
save outfile = 'C:\Data_SPSS\SHP-Data-Longfile-SPSS\shplong_p_h_reduced.sav'.
***.
```