# Import – Export app guide

## *Introduction*

The aim of this shiny app is to provide a rather simple way to convert (import and export) documents between the most common extensions (such as ".xlsx",".sav",…) without needing any software by using r functions that guarantee the least loss of the data information, giving special attention to the date variables and the labelled ones.

The app also offers multiple ways to check the database before finishing the process so it's easy to find out any potential issue that might have happened during the import section.

The user must follow 3 steps in order to complete all the process and receive the document in the desired extension, but before introducing each step we will see the structure of the app:

## *Structure*

Once the app is open we can see it's divided into 2 sections. In the left one we can see the 3 steps but their options are hidden , we need to click on each step to see its options. This options can be different depending on the extension we choose, and most of the extensions options also contain a button which displays more options on a pop-up.

In the right one we will see a set of panels with the "file" panel opened by default. We can switch between them by clicking on the name of each one, but at start they all will be empty. This whole section displays information about the database we are converting in different ways , which will be explained in the section of "Step2)".

## *Step1) Format detection + import*

This is the first step when we use this app, we need to select the file from our computer containing the database we want to convert. Once the file is selected, if we still have the "file" panel open in the right section we will see the format that has been automatically detected. Then, we can see a selected checkbox that assumes the format is correct, if that wasn't true we can unselect it and choose the right format.

When we have the selected file with his appropriate format we will see a button "Show import options" which will display a pop-up with necessary and optional options

required to perform the import process. The pop-ups may also contain a link to the description of the used R function.
Finally, we can close the pop-up, press the import button and go to the next step.

## *Step2) Database checkup*

Even though this step has few options in the scroll panel, the most important elements are at the right section of the app.

In the scroll panel we can see a checkbox to run format_corrector (a function to fix some format problems, detailed in the help link we can see above) and a list with the variables stored in the dataset. We can select which of these variables we want to keep.

In the right section of the app we can see 4 panels with relevant information of the dataset.

-File: Displays the status of the dataset we are working with. We can see the format used to read it, the selected variables and know if the data has been modified using format_corrector.

-Var_view: Displays the output of the var_view function, which provides a table containing relevant information about the variables such as their class or their labels.

-Datatable: Displays a shiny datatable where we can see observations of the dataset and use filters.

-Summary: Displays the output of the compareGroups function (from the compareGroups package) as a summary for the non-date variables. We can see a normal summary below for the date variables.

## *Step3) Export*

This is the last step and pretty similar to the first one. The first list allows us to select the format we want. Like in the first step, each format will have different options, and most of them will have an "export options" button which will display a pop-up with more options.

Most of the formats can be just downloaded using the download button, but there are special formats like Microsoft Office Access or SPSS which will be downloaded in a different way:

-Microsoft Office Access: The function to export data to Microsoft Office Access can not create new files, it can just add tables to an existing file so instead of downloading a file, the last step of the Access export will be selecting an existing file in our computer to add the data we have in the app.

-SPSS: The function to export data to SPSS don’t create the .sav file directly so it can’t be downloaded. The function needs an environment (a folder) to export the data and the spss code and then run the spss runsyntax to finally create the .sav file.

## *Import and export R functions*

### Import:

TXT and CSV = table_import() : Created function,calls read.table() and automatically detects the field separator.

XLS and XLSX= read.xlsx() : Function from the xlsx package. Can read both .xls and .xlsx. Requires java.

SPSS = spss_import() : Created function, calls spss.get() but automatically detects date variables.

Microsoft Office Access= access_import() : Created function, simplify the RODBC package functions to read from Access in a rather simple way.

Stata= read_dta() : Function from haven package.

SAS= read_sas() , read.xport() : Function from the foreign and haven package respectively, depending on the data extension we will need one or another.

### Export

TXT= write.table() : Function from R base

XLS and XLSX = excel_export() : Created function, calls wirte.xlsx() and can write to multiple sheets. Require java.

SPSS = spss_export() : Created function, it creates the spss file by running spss runsyntax with the data and the spss code. It conveniently stores date variables and the labels.

Microsoft Office Access= access_export() : Created function, simplify the RODBC package functions to write to Access in a rather simple way.

Stata= write_dta() : Function from haven package.

CSV= write.csv() : Function from R base

SAS= write.foreign() : Function from foreign package.