## Aims of exercise

The aim of this exercise is to practice implementing a basic set of analyses in R using R Markdown and ProjectTemplate. It assumes that you have done the online R training. The exercise is similar to an earlier exercise that we did in SPSS. So it's a good opportunity to think about how you do the same thing with different software.

* read about a data file, import data, and prepare a data file
* check missing data
* create scale scores
* perform reliability analysis
* run some additional analyses

## Main Instructions

1. Read description of bfi dataset in "bfi-data-description.rtf"
2. Setup a new project with ProjectTemplate and open in RStudio
3. Import the data "rcases.csv" and "meta.xls"
4. Add value labels for gender and education
5. Are all the values within the range of permissible values for all variables (e.g., all personality test items are on a 1 to 6 scale)?
6. To what extent is missing data a problem in the dataset? How much missing data is there? Which variables have more missing data? Why?
7. Create a variable that stores the number of missing responses a particular person has. Are there any cases with a huge amount of missing data? What threshold would you use for deleting the case? Why?
8. Compute scale scores for each big 5 factor using the scoreItems function in the psych package
9. Compute cronbach's alpha reliability for each of the big 5 factors (you get this from the scoreItems function

## Additional Exercises

1. Examine the distribution of each of the big 5?
2. Compare mean scores on the five factors of personality for the different genders. Are the results as you would expect?
3. Correlate the five factors of personality. Are they correlated? Is the pattern and general size of correlations what you would expect?
4. Examine plots of the relationship between age and each of the big 5?

# Answers

**2. Setup a new project with ProjectTemplate and open in RStudio**

As discussed in the online training, ProjectTemplate is a powerful tool for organising a data analysis project and making it consistent and reproducible. I also have a customised version that I find useful.

When starting a new project, you can download the customised version as a zip file from here:

<https://github.com/jeromyanglim/AnglimModifiedProjectTemplate>

Or directly from here:

<https://github.com/jeromyanglim/AnglimModifiedProjectTemplate/archive/master.zip>

* Unzip the file (note on Windows, you need to be careful that you actually unzip it; Windows has the annoying habit of letting you browse files within a zip file without unzipping the zip file)

This will generate a folder structure where different content goes in different folders.

* Rename the parent folder to something meaningful like "r-exercise" and rename the studio .rproj file to something meaningful like "r-exercise"
* Double click on the R project file to open in Rstudio

**3. Import the data "rcases.csv" and "meta.xls"**

There are various functions for importing data depending on the file format

<http://stackoverflow.com/documentation/r/481/importing-and-exporting-data>

However, a nice feature of ProjectTemplate is that it can automate data importation. It generally chooses the right function based on the file extension (e.g., csv, xls, and so on) and it will also assign a name to the data frame that corresponds to the dataset.

We'll do it this way:

* place the data file "rcases.csv" and the meta data "meta.xls" into the data folder. You may need to delete the existing meta data.

The data is now ready for importing.

In general, to initiate your project template project, you run the following command:

library(ProjectTemplate); load.project()

* This function loads the ProjectTemplate package, and then issue a command "load.project()" which will do a range of things including importing any data in the data folder.

This command is also written in the file "reports/explore.rmd". So you may want to open that and run that command.

If you are lucky, then you will see some messages about loading packages, and loading datasets, and you will see the data files listed in the enviornment (i.e., "rcases" and "meta.bfi"

However, more likely is that you will have several errors. In particular, you may need to install additional R packages.

In addition, the function for importing the Excel data requires you to have a Perl installation. If you don't have this (which is quite likely if you run Windows), then you can get Perl from here:

<https://www.perl.org/get.html>

Alternatively, if you don't want to install Perl, you could open up "meta.xls" in Excel and save as it as "meta.csv" and then delete the xls files from the data folder and just leave "meta.csv"

**4. Add value labels for gender and education**

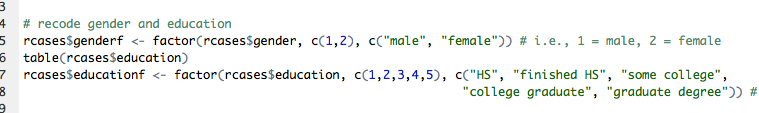
The logic of ProjectTemplate is that it organises analyses cleanly into a clear process: data import - data transformation - data analysis

This is an important workflow. The idea is to put all your data transformations in one location. In the ProjectTemplate system, these go in R files in the "munge" directory.

Adding value labels to a variable is an example of data manipulations, so it goes in that file in the "munge" directory.

* Open the existing file "munge/01-munge.R"

The following code will create new variables with value labels:

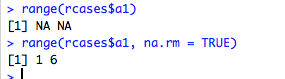


* rcases is the data frame (it's a convention that I use "r" for "raw" and "cases"). It's generally good to keep the names of data frames short, because you refer to them a lot.
* $ is an operator that allows you refer to variables in a data.frame
* genderf is the name of the new variable that is being created
* <- is the assignment operator
* factor is a function (see ? factor) in R for creating factors which are similar to numeric variables in SPSS that have value labels. It takes three main arguments: the data, the current values, and the textual labels. The second and third arguments need to be in corresponding order and of the same length.

**5. Are all the values within the range of permissible values for all variables (e.g., all personality test items are on a 1 to 6 scale)?**

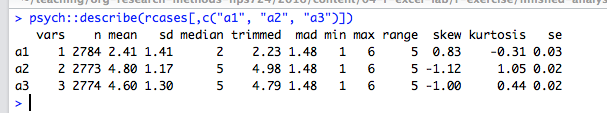
There are many ways to check this in R. There is the "range" function which gives min and max. There are specific "min" and "max" functions. There is also the "psych::describe" function which gives several descriptive statistics including min and max.

So we could go:



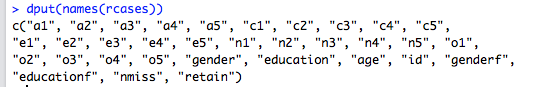
* when we first apply the function we get NA because there is missing data.
* but if we supply "na.rm = TRUE" is remove NA data, then we get the range for the data that is present.

Of course, it might be a bit tedious repeating this all 25 items. The psych::describe function is easier to apply to many numeric variables.



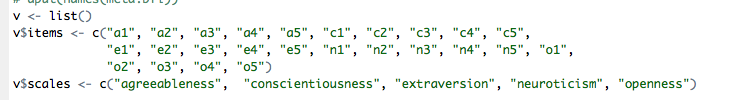
The function takes a data.frame and above I've supplied three variables "a1" to "a3" and we have the min and the max, which is all 1 to 6 which is as it should be.

But is there a quicker way of selecting all 25 variables? Here is my recommended approach.



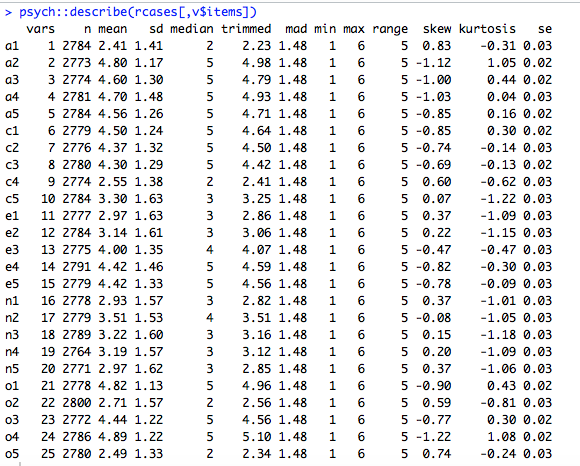
* dput(names(your.data.frame)) will return the names of the data.frame in a format that is easy to copy and paste
* So you can copy and paste a1 to o5

A second tip that I find extreme useful is to create a list variable that stores various variable sets that you may wish to access.



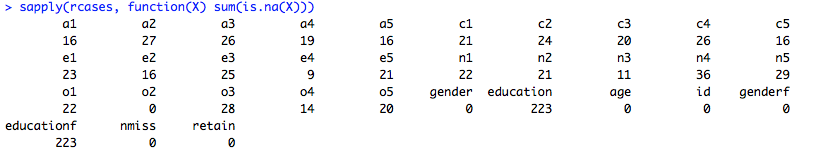
* I typically call this list "v" (v for variables)
* It's first declared as a list with the "list()" function
* then you can variable sets. The above shows one variable set called "items" i.e., v$items, where $ links the list with the specific element of the list.

Now we can use this variable list to get relevant information:



* This function is given descriptive statistics for all variables listed in v$items
* We've used the data.frame subsetting operation "[,]";

**6. To what extent is missing data a problem in the dataset? How much missing data is there? Which variables have more missing data? Why?**



This command tells how much missing data there is on each question.

* sapply is a function that repeats the specified function for each variable in the data.frame.
* The next argument is a function
* is.na(X) will say whether the given value is missing or not
* sum(is.na(X)) will say how many TRUE values there are (i.e., how many are missing)

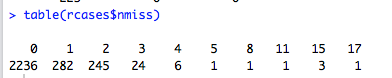
**7. Create a variable that stores the number of missing responses a particular person has. Are there any cases with a huge amount of missing data? What threshold would you use for deleting the case? Why?**

The following function will create a missing count for each case

****

* nmiss is arbitrary name of the new variable that is created
* apply is a function that can perform a function either over rows or columns of a data.frame
* the "1" indicates that the function is performed over rows (2 is for columns
* The third argument is the function. Thus, it takes a single person's data and works out the number of missing responses.

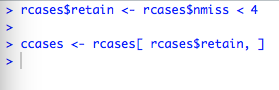
We can get a table showing the number of cases with a given number of missing data values as follows:

****

* table is a function that shows frequency counts

In terms of interpretation, this shows that there is one participant with 17 missing responses.

Perhaps we can delete all cases that have 4 or more missing values. We'll still have a lot of data, we're not losing much, and it'll make subsequent analyses a little easier to keep track of.

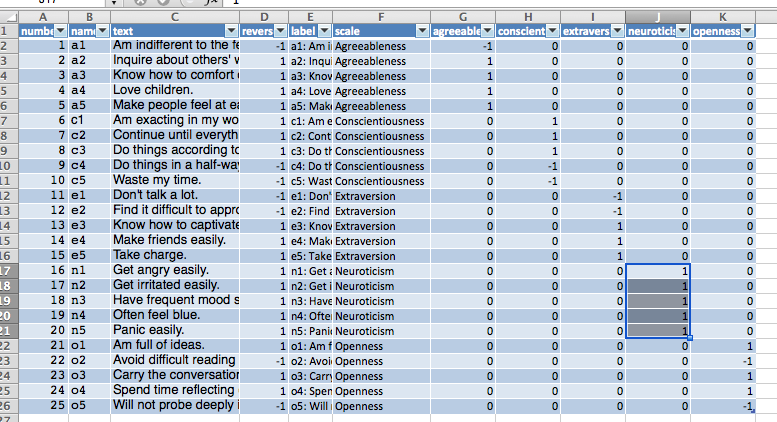


* The first line returns TRUE if the missing count is less than 4
* The second line selects only the cases in rcases that passed that missing data threshold. it uses the subsetting operation [,] where you put the rows you want before the first comma.
* This data frame is assigned to ccases (i.e., cleaned cases).

**8. Compute scale scores for each big 5 factor using the scoreItems function in the psych package**

The scoreItems function in the psych function is a great function for scoring typical likert-type scales in psychology. It also provides a range of reliability and item characteristic information.

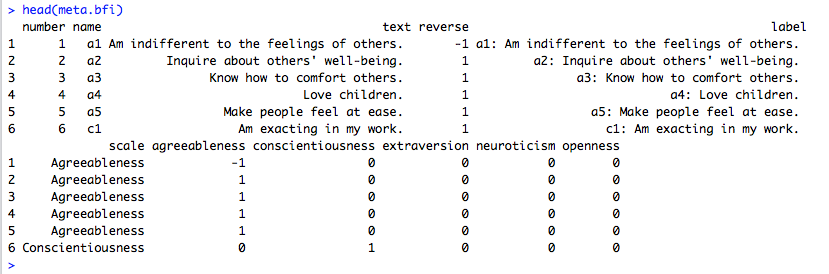
The only tricky bit is set up your data key correctly. I've set it up correctly in the meta.bfi data.frame.



* Essentially, each row is an item, and there are columns for each of the variables that you want to compute with the codes 1 (positively worded item) -1 (negatively worded item), and 0 (item not included in that scale).

This data.frame should be imported as meta.bfi

For example, here's a quick look at the first few rows in R:



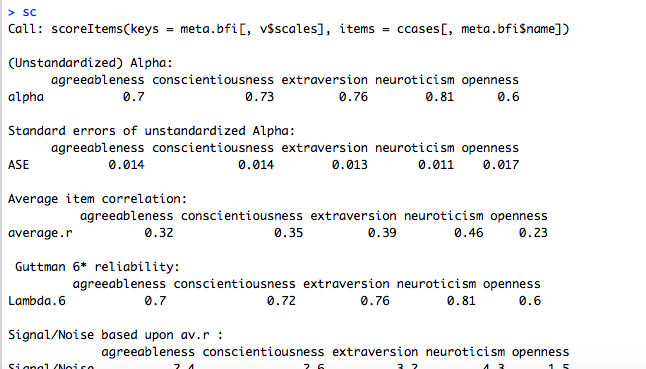
The following command will score the big 5 and add them to the data.frame:



* v$scales was created earlier and corresponds to the five column names in the meta data: one for each of the big 5
* meta.bfi$name is the item name which appears both in the actual data and in the meta data.
* An object is returned, which I've called "sc"
* One of the properties of this object is "scores" which have the test scores for each participant.
* These scores can be added to the data.frame as shown above

**9. Compute cronbach's alpha reliability for each of the big 5 factors (you get this from the scoreItems function**

If you don't assign scoreItems then you will get alpha and other output. Or if you assign, you can just type the variable name:



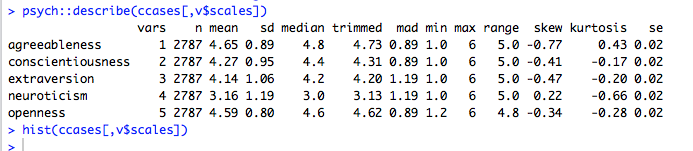
You get lots of output. But we can see the unstandardised alpha for each of the big 5.

If you want to get item statistics, you can run:

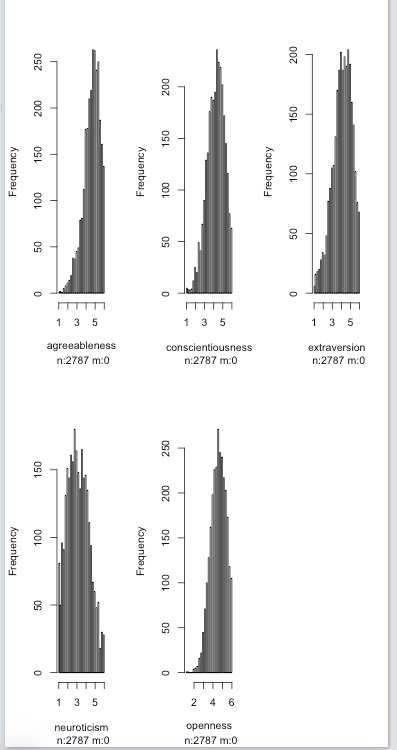


# Additional Exercises

**10. Examine the distribution of each of the big 5?**

****

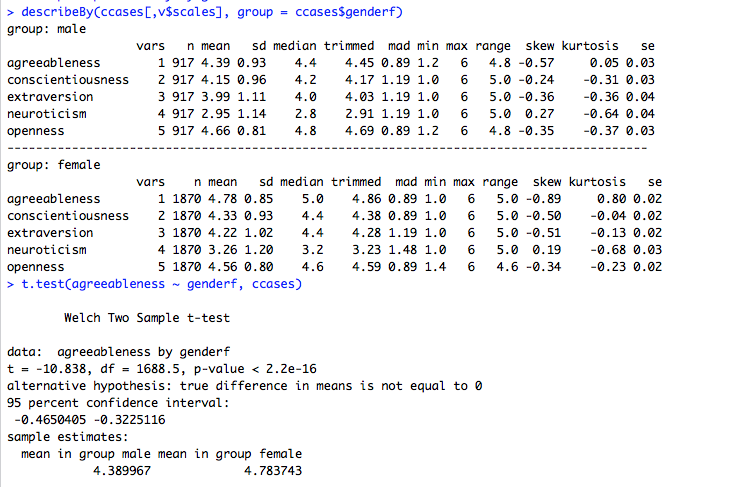
The describe function includes skewness and kurtosis information. So for example, the big 5, except for neuroticism has a bit of negative skew (particularly for agreeableness).

****

The hist function shows histograms which highlights this further.

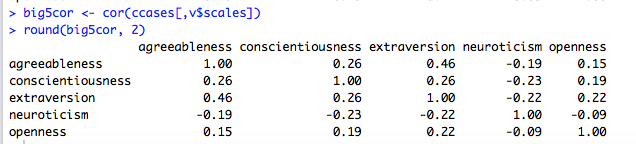
**11. Compare mean scores on the five factors of personality for the different genders. Are the results as you would expect?**

There are various approaches for comparing groups.



* describeBy takes a data.frame with numeric variables and a single grouping variable and presents full descriptive information for all variables.
* t.test can perform t tests for particular variables. Above shows an example of formula notation "dv ~ iv"

**12. Correlate the five factors of personality. Are they correlated? Is the pattern and general size of correlations what you would expect?**

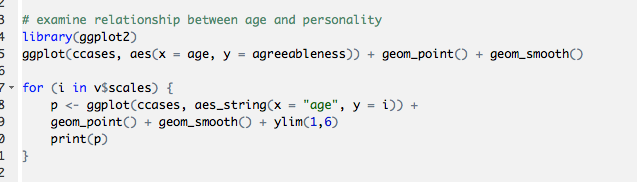
****

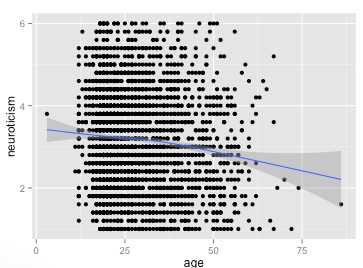
* cor will give the correlation between variables in a data.frame
* I used round to 2 digits to make it easier to read.

On a substantive level most correlations were low, with the exception of agreeableness and extraversion.

**13. Examine plots of the relationship between age and each of the big 5?**

The following shows an example of ggplot graphs



****

There's quite a bit going on here. In general, it's useful to start with an example from the website and then adapt: http://docs.ggplot2.org/current/geom\_point.html

* library(ggplot2) ensures the package is loaded
* ggplot(ccases, aes(x =, y...): this indicates that the plot will uses the ccases data.frame and it will asign age to x and agreeableness to y. aes stands for aesthetic attribute.
* geom\_point indicates that we want a scatter plot
* and geom\_smooth indicates we want to see a line of best fit

The next section indicates how you can use looping to generate a series of graphs (e.g., one for each of the big 5)

* for is the command for a for loop
* Thus, on each iteration i takes on one element of v$scales
* Otherwise the function is similar to above, except that the y variable is replaced with i
* we also needed to use "aes\_string" because the variable name was a string
* we needed to use print to ensure that the plot displayed

The last bit is quite advanced, but hints at the potential for R.