Notes for dapR1 Live R Session; Week 1, Semester1

2024-09-17

Summary of this week's session

In this week's session, we will have a look at some basic features of RStudio, and we will learn how to read in a data file. As a first step, please register for RStudio, following the instructions in the Week 1 folder on Learn. In the session, we will show the commands working in real time, on the data projector, but these notes are provided to help you to remember the main points.

Using R as a calculator

Although R is a programming language, it can also be used in interactive mode, and it is sometimes convenient to use it as a calculator. Once you have opened RStudio, the default view should show a panel on the left, with a tab laballed "Console" (you may need to click on this tab to select it). The console can be used to type single line commands. The commands are written after the '>' prompt, followed by pressing Return on your keyboard. In this way, you can use R as a calculator. For example, try typing the following at the console:

```
2*8
```

```
## [1] 16
```

The system should respond with the answer. Or, you could try something slightly more complex, for example:

```
2*(8+3)
```

```
## [1] 22
```

You can save the result of a calculation in a variable that can be accessed again later. For example, the following code uses the assignment operator "<-" to save the result of the above calculation in a variable called "mynumber":

```
mynumber <- 2*(8+3)
```

If you look in the environment panel, which is on the top right in RStudio by default, you will see that the variable mynumber has been listed, along with its value. We could then type the name of the variable in the console, and R will report its value:

```
mynumber
```

```
## [1] 22
```

We could also use the name of the variable in another calculation:

```
mynumber/2
```

```
## [1] 11
```

Loading packages and reading in data

When we use R, we often need to use special functions that are not included in the basic version, but need to be accessed by loading packages. If you look in the bottom right panel in R studio, you should see a tab

labelled "Packages". If you click that, you will see a list of all the packages available, with a check mark in the ones that are currently loaded. In this course, we will often need to use a package called "tidyverse". In order to load this package you can type the following at the console:

```
library(tidyverse)
```

If you scroll down in the Packages panel, you should see a tick mark showing that "tidyverse" has been loaded. Now, let's read in some data. We will read in a simple dataset that was used in this week's lecture. The dataset is currently stored on a website associated with this course, and as long as the tidyverse package has been loaded, you should be able to read in the data with the following command:

```
mydata <- read_csv("https://uoepsy.github.io/data/lecture1_data.csv")
```

Notice that we use the assignment operator again, storing the data in a variable named "mydata". The remaining part of the command uses the function "read_csv()", because the data file is stored in a csv file. The argument of the function, written inside the parentheses, is the url where the csv file is located, enclosed in double quotes.

Looking at the data

Now that we have read in the data, there are several different ways that we can look at it. Perhaps the simplest thing we could do is just to type the name of the variable:

```
mydata
```

```
## # A tibble: 5 x 9
##   ID    Hair_colour Likert_item      Likert_values Degree ReactionTime Height_cm
##   <chr> <chr>       <chr>                   <dbl> <chr>         <dbl>     <dbl>
## 1 ID101 Brown       Strongly Agree              5 No              1.2      191.
## 2 ID102 Brown       Agree                       4 No              0.9      181.
## 3 ID103 Blonde      Agree                       4 Yes             3.2      165.
## 4 ID104 Blonde      Disagree                    2 Yes            55.5      177.
## 5 ID105 Black       Strongly Disagr~            1 Yes             2.1      201
## # i 2 more variables: Weight_kg <dbl>, IQ <dbl>
```

The output shows the format of the dataset, which is called a "tibble", and its dimensions are given, showing that there are 5 rows and 9 columns. Then, each column has a variable name in the first row, with the type of each variable given directly below. For example some columns have the type of "chr", which stands for "character", meaning that the column contains information given as sequences of characters. Other columns have the type of "dbl", which is a way of describing numeric data.

An alternative way to examine the data is to use the "str()" command, which stands for structure:

```
str(mydata)
```

```
## spc_tbl_ [5 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID           : chr [1:5] "ID101" "ID102" "ID103" "ID104" ...
##  $ Hair_colour  : chr [1:5] "Brown" "Brown" "Blonde" "Blonde" ...
##  $ Likert_item  : chr [1:5] "Strongly Agree" "Agree" "Agree" "Disagree" ...
##  $ Likert_values: num [1:5] 5 4 4 2 1
##  $ Degree       : chr [1:5] "No" "No" "Yes" "Yes" ...
##  $ ReactionTime : num [1:5] 1.2 0.9 3.2 55.5 2.1
##  $ Height_cm    : num [1:5] 191 181 165 177 201
##  $ Weight_kg    : num [1:5] 88.9 76.6 52 81.5 105.8
##  $ IQ           : num [1:5] 100 105 99 120 131
##  - attr(*, "spec")=
##   .. cols(
```

```
##   ..   ID = col_character(),
##   ..   Hair_colour = col_character(),
##   ..   Likert_item = col_character(),
##   ..   Likert_values = col_double(),
##   ..   Degree = col_character(),
##   ..   ReactionTime = col_double(),
##   ..   Height_cm = col_double(),
##   ..   Weight_kg = col_double(),
##   ..   IQ = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

In this case, the output has a different format, with each variable appearing as a separate row, showing "chr" for character variables and "num" for numeric variables.

Another way of examining the data is to use the "glimpse()" command, which gives a very similar output to the previous command:

```
glimpse(mydata)
```

```
## Rows: 5
## Columns: 9
## $ ID            <chr> "ID101", "ID102", "ID103", "ID104", "ID105"
## $ Hair_colour   <chr> "Brown", "Brown", "Blonde", "Blonde", "Black"
## $ Likert_item   <chr> "Strongly Agree", "Agree", "Agree", "Disagree", "Strongl~
## $ Likert_values <dbl> 5, 4, 4, 2, 1
## $ Degree        <chr> "No", "No", "Yes", "Yes", "Yes"
## $ ReactionTime  <dbl> 1.2, 0.9, 3.2, 55.5, 2.1
## $ Height_cm     <dbl> 191.2, 180.8, 165.3, 177.1, 201.0
## $ Weight_kg     <dbl> 88.9, 76.6, 52.0, 81.5, 105.8
## $ IQ            <dbl> 100, 105, 99, 120, 131
```

You will notice that, as mydata has been defined as a variable, it is listed in the "Environment" panel on the top right. If you click on the name of the variable, it will show the dataset in a tabular format.

How about if we only want to look at one column of the data. For example, we want to examine the contents of the "Hair_colour" column. Then we can use the dollar sign symbol as follows:

```
mydata$Hair_colour
```

```
## [1] "Brown"  "Brown"  "Blonde" "Blonde" "Black"
```

The above command gives the list of values in the "Hair_colour" column of mydata. We could also do the same with another variable, for example:

```
mydata$ReactionTime
```

```
## [1]  1.2  0.9  3.2 55.5  2.1
```

How about if you want to look at the data in a specific cell? The following command will show the data in the cell occupying the third row and the fourth column:

```
mydata[3,4]
```

```
## # A tibble: 1 x 1
##   Likert_values
##           <dbl>
## 1             4
```

If I want the whole of the third row, I could remove the column figure, like this (notice the comma remains):

```
mydata[3,]
```

```
## # A tibble: 1 x 9
##   ID    Hair_colour Likert_item Likert_values Degree ReactionTime Height_cm
##   <chr> <chr>       <chr>               <dbl> <chr>         <dbl>     <dbl>
## 1 ID103 Blonde      Agree                   4 Yes             3.2      165.
## # i 2 more variables: Weight_kg <dbl>, IQ <dbl>
```

If I wanted the whole of the fourth column, I could delete the row figure, like this:

```
mydata[,4]
```

```
## # A tibble: 5 x 1
##   Likert_values
##           <dbl>
## 1             5
## 2             4
## 3             4
## 4             2
## 5             1
```

There are also commands that report how many rows or how many columns are in the data set. First, the number of rows:

```
nrow(mydata)
```

```
## [1] 5
```

Now the number of columns:

```
ncol(mydata)
```

```
## [1] 9
```

Then, if we want to find the overall dimensions (number of rows and number of columns), we can use the "dim()" command:

```
dim(mydata)
```

```
## [1] 5 9
```

It is also possible to add a row or a column. Here, we will add a new column, showing the favourite animal of each person in the dataset. To do this, we will again use the dollar-sign operator, and the assignment operator. We will assign a list of five strings of characters as the value of the new column. The strings are enclosed in quotes:

```
mydata$fav_animal <- c("cat","dog","horse","rabbit","donkey")
```

We should be able to see the new column if we look at the data set:

```
mydata
```

```
## # A tibble: 5 x 10
##   ID    Hair_colour Likert_item      Likert_values Degree ReactionTime Height_cm
##   <chr> <chr>       <chr>                    <dbl> <chr>         <dbl>     <dbl>
## 1 ID101 Brown       Strongly Agree               5 No              1.2      191.
## 2 ID102 Brown       Agree                        4 No              0.9      181.
## 3 ID103 Blonde      Agree                        4 Yes             3.2      165.
## 4 ID104 Blonde      Disagree                     2 Yes            55.5      177.
## 5 ID105 Black       Strongly Disagr~             1 Yes             2.1      201
## # i 3 more variables: Weight_kg <dbl>, IQ <dbl>, fav_animal <chr>
```

Notice, in the above, we use the "c()" expression to write the list: c("cat","dog","horse","rabbit","donkey"). This is used very often to express a list in R.

Declaring a factor

For some statistical analyses that you will study in this course, it is necessary to declare a variable to be a *factor*. Briefly, a factor is a variable that is interpreted as a set of categories, which are called *levels*. For example, in our dataset, the "Degree" variable could be interpreted as a factor with two levels, namely "Yes" and "No". We can declare a column to be a factor using the following command:

```
mydata$Degree <- factor(mydata$Degree)
```

Now, if you run the "str()" command, you can see that "mydata$Degree" is treated as a factor with two levels:

```
str(mydata)
```

```
## spc_tbl_ [5 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ID           : chr [1:5] "ID101" "ID102" "ID103" "ID104" ...
##  $ Hair_colour  : chr [1:5] "Brown" "Brown" "Blonde" "Blonde" ...
##  $ Likert_item  : chr [1:5] "Strongly Agree" "Agree" "Agree" "Disagree" ...
##  $ Likert_values: num [1:5] 5 4 4 2 1
##  $ Degree       : Factor w/ 2 levels "No","Yes": 1 1 2 2 2
##  $ ReactionTime : num [1:5] 1.2 0.9 3.2 55.5 2.1
##  $ Height_cm    : num [1:5] 191 181 165 177 201
##  $ Weight_kg    : num [1:5] 88.9 76.6 52 81.5 105.8
##  $ IQ           : num [1:5] 100 105 99 120 131
##  $ fav_animal   : chr [1:5] "cat" "dog" "horse" "rabbit" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ID = col_character(),
##   ..   Hair_colour = col_character(),
##   ..   Likert_item = col_character(),
##   ..   Likert_values = col_double(),
##   ..   Degree = col_character(),
##   ..   ReactionTime = col_double(),
##   ..   Height_cm = col_double(),
##   ..   Weight_kg = col_double(),
##   ..   IQ = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Uploading, creating and exporting files

If you want to upload a file from your local computer onto the rstudio server, you can do it by clicking the "Upload" button on the Files panel. You choose the file on your disk, and also choose the destination folder on the Rstudio server.

You can also save a dataset to the RStudio server. In this case, we already have the dataset stored in the variable "mydata". What if we want to save this dataset as a csv file on the RStudio server? The following command will save it in a csv file called "mydata.csv".

```
write_csv(mydata,file="mydata.csv")
```

If you execute this command, the file "mydata.csv" should appear in the "files" tab of the bottom right panel in your RStudio session.

How about if you want to export this file to your local disk on your computer. To do this, select the check-box next to the file in the "Files" panel, and click "More" at the top of this panel, then select "Export".

How to create and use an R Markdown document

In RStudio it is often convenient to use a special type of file format called "R Markdown". An R Markdown file can be converted into a pdf document, and can also include what we call "code chunks", which are pieces of R code that can be run inside the document. This is useful because you can save the document, and you will keep all your work, unlike in the console, where you lose the code that you have written if you quit R Studio. The R Markdown format will be used extensivly in the lab sessions, and you will get plenty of practice. In the Live R lecture session on Tuesday, we will give a very brief introduction, to show how to create a new R Markdown document, and how to create and run code chunks.

Very briefly, you can create an R Markdown document via the File menu in R Studio, like this:

File > New File > R Markdown

This will create a new panel in R Studio with some example contents of an R Markdown file. You can edit these contents to make your own document.

We will show more details in the Live R session.