Instructions for ICW

Maria Kjellsson, Uppsala University

08 November, 2017

Abstract

This document provides an introduction to the In-Class Workshop or `ICW`, which are a part of the course *Introduction to Programming in Python and R for Bioscience*. The In-Class Workshops are related to the modules in *https://www.scalable-learning.com*. It is four modules: Basic R, Reproducible Research, Data Management and Graphics. The extent of each module varies with the 1st and 3rd being the most extensive. If you do not have time to finish all exercises related to a module in an ICW, you can revisit these exercises at another ICW if time permits.

Contents

1) Basic R

In this module, we go through the most basic aspects of R and programming in R.

1.1) Execution of commands

There are several ways to execute commands in R studio.

The easiest way is to type the command directly in the console and press `enter`. This will execute the command. This is an excellent way of testing how to write the commands for a certain operation. The main drawback with this way of working is that the commands are not saved anywhere else than in the workspace. It will look something like this:

```
1+1
```

```
## [1] 2
```

If you use scripts, that you write in the source editor (recommended), you could execute the code either line-by-line, in chunks by highlighting a section of your code or a subset of your code by highlighting the subsection. The action of execution is done by either using the **Run** button in the right corner of the source editor (see figure 1) or the short-command `ctrl-enter` (Windows)/`cmd-enter` (Mac).

In the tutorial on basic commands by Mike, he used the source editor with short-commands and executed either line by line or subset of source.

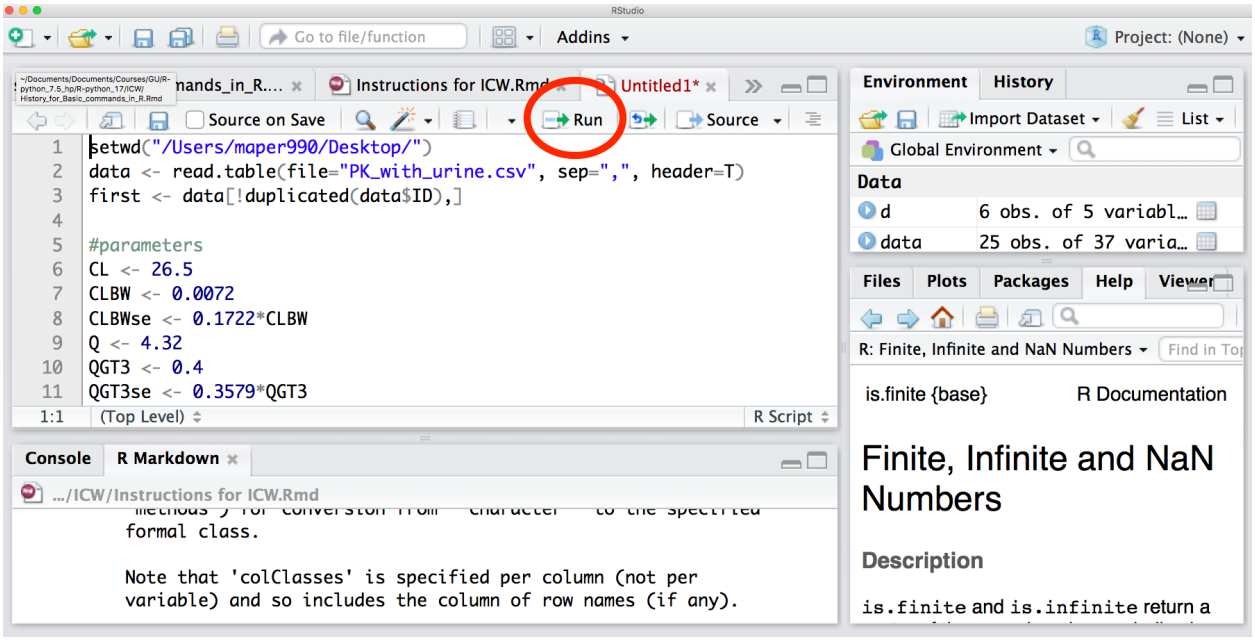


Figure 1: *Run button in R studio*

1.2) Work directory

A work directory is a place where R work and the only files and folder visible for R are those in the work directory. In this section, you will work with work directories.

Exercises

Exercise A)

In the console, type the following command:

```
getwd()
```

What is the output? What do you think the function 'getwd() does? Discuss with your classmates.

Exercise B)

In the console, type the following command, place the cursor between the quotes and use **tab** for autocompletion.

```
setwd("")
```

What options are you given? What does the function setwd() do? Discuss with your classmates.

Exercise C)

In the console, type the following command, place the cursor between the quotes and use **tab** for autocompletion.

```
read.table("")
```

How are the options different from exercise 1b? Why do you get other alternatives? Discuss with your classmates.

Exercise D)

In the console, type the following command:

```
setwd("..")
getwd()
```

What happend? What does . . do? Discuss with your classmates.

Test 1.2

1. Create a new folder in your home directory, called *R_course* with a subdirectory called *ICW_BasicR*. *NB! No spaces in names. These can cause problems later.
2. Set the working directory to the newly created *ICW_BasicR*.
3. Show your code for a tutor signature before moving on.

Tutor signature:

1.3) Help and tutorials in R

There are several ways to get help for R; there is internal help and online help. Apart from all help functions and code-snippets available for R, you can also find tutorials for self-education. One such platform is Quick-R, on http://statmethods.net. In this section, you will work with various ways of getting help in R.

Exercises

Exercise A)

The following commands do not give the same output:

```
?read.table()
??read.table()
```

What are the differences between these two help functions? Discuss with a classmate.

Exercise B)

Try the following commands:

```
?read.table
help(read.table)
```

What are the differences between these two help functions? Discuss with a classmate.

Exercise C)

Use the menu at the top and select **Help**. Get the cheatsheet for R studio IDE. Find the short-command for **Search command history**. Use it.

What happens?

Exercise D)

Use the menu at the top and select **Help**, then **R help**.

What resources do R studio suggest for getting help? Discuss with a classmate.

Exercise E)

Go to http://statmethods.net/about/learningcurve.html. Read the text at this page.

Exercise F)

Enter the free interactive course at the end of the homepage http://statmethods.net/about/learningcurve.html. Select `Intro to basics`. Read the instructions and do all exercises belonging to `Intro to basics`.

Question to discuss with the classmates: How do you see modulo %% being used?

Test 1.3

1. Download the file *data_for_ICW.csv* from the student portal.
2. Place it in the newly created folder *ICW_BasicR*.
3. Check that *ICW_BasicR* is your working directory.
4. Use any of the sources of help to guide you in how to get these data into R.
5. Import the data and show a tutor for signature.

Tutor signature:

1.4) Assigning and indexing objects

To be able to work with the vectors, datasets etc., you need to assign objects containing your vectors, datasets etc and then to work with parts of your object you need to be able to index an object. In this section, you will work with both assigning objects (which you've already done during the online Intro to R) and indexing.

Exercises

Exercise A)

Create an object called `int1` containing integers from 5 to 15 with steps of 1.

How many different ways can you do this? Discuss with a classmate.

Exercise B)

Look at the object `int1`.

How many different ways can you look at the vector? What class is it? Discuss with a classmate.

Exercise C)

Take the sum of all even elements of the object `int1` and substract the sum of all odd elements of the object `int1`. Can you come up with different ways to do this?

Exercise D)

Do Test 1.3 again, but this time, assign an object called `data` with the output from the function.

Use the following command to look at your object `data`:

```
str(data)
```

How big is the dataset in terms of columns (`variables`) and rows (`obs`)? What is the name of the first `variable`?

Test 1.4

1. Do Test1.3 again, but this time, assign an object called `data` with the output from the function.
2. Get a signature from a tutor.

Tutor signature:

1.5) Class of object

What class an object is determines how the functions applied on the object will work. In this section, we will focus on classes of objects.

Exercise

Exercise A)

Create five vectors with the following commands:

```
int  <- as.integer(1:6)
num  <- as.numeric(7:13)
char <- paste0("hello", int)
fac  <- factor(rep(c("hello","bye"), 3))
log  <- rep(c(TRUE,FALSE), 3)
```

How would you investigate with class these vectors are? Discuss with a classmate.

Exercise B)

Add the vectors together two by two, like this:

```
fac_log  <- c(fac,log)
```

Check the class of the combined vectors. Can you identify a pattern? Discuss with a classmate.

Exercise C)

Add all vectors together, like this:

```
new  <- c(fac, log, char, num, int)
```

Can you guess in advance what class the vector `new`will be? Discuss with a classmate.

Exercise D)

Create a numerical vector, containing 1, 1, 2, 2, 3, 3. Change the class to factor. Tips: Use the function factor. How many levels are created? Discuss with a classmate.

Test 1.5

1. Create an integer vector, with 1, 0, 1.
2. Create a logical vector, with TRUE, TRUE, FALSE.
3. Create a factor vector, using the vectors created in 1) and 2)
4. Show a tutor your code for signature.

Tutor signature:

---

2) Reproducible Research

In this module, we have emphasis the importance of performing research in a reproducible way and although the experiments may be difficult to sure reproducibility, the way we perform data management should be have an audit trail. One way to ensure this is to use markdown to produce reports with imbedded data management, graphical analyses and statistical calculations. Knitr is a package for R that allows reproducible research.

This document was created using Rmarkdown (`.Rmd`).

2.1) YAML title

The first part of an R-markdown document is the YAML with the information about how the document should look.

Exercises

Exercise A)

Create an R-markdown document. Go to *File*, choose *New File*, then *R Markdown*. Choose a title for the document and select html-output. Save the document and knit it. Go to http://rmarkdown.rstudio.com/html_document_format.html and read about how to adapt the YAML.

Exercise B)

Change so that your html-document includes a table of content. What happens when you specify `toc_float`?

Exercise C)

Try the various themes available. Again, look at http://rmarkdown.rstudio.com/html_document_format.html. Which theme do you prefer?

Test 2.1

Create the YAML for a document you would like to create. Show a tutor for signature.

Tutor signature:

2.2) Code Chunk

The code chunk is the part where the R-code is written. In specifying these there are also options for appearance.

Execises

Exercise A)

Use the document you created in section 2.1) and add the following chunk:

````
```{r first chunk, echo=TRUE, eval=TRUE}
int  <- as.integer(1:6)
num  <- as.numeric(7:13)
int
num

```
````

Figure 2: *Chunk*

Knit your document.

Now change echo from `TRUE` to `FALSE` and knit again. What happens? Discuss with a classmate.

Exercise B)

Change echo back to `TRUE`. Now change eval from `TRUE` to `FALSE`. What happens? Discuss with a classmate.

Exercise C)

Change eval back to `TRUE`. Now add the option `results = "hold"`. What happens? Discuss with a classmate.

Exercise D)

Copy the previous chunk and add underneath. Change `int  <- as.integer(1:6)` to `log <- c(FALSE,TRUE)` and you get an error. Why? Discuss with a classmate.

Test 2.2

Create a new chunk and adapt the settings as you like. Show a tutor to get a signature.

Tutor signature:

2.3) Text

There are also options related to how to format the text.

Exercises

Exercise A)

Add a new chunk to the document created in section 2.1).

```
```{r average chunk, echo=TRUE, eval=TRUE}
mnum  <- mean(num)`
```
```

Figure 3: *Chunk*

Then type underneath (outside) the chunk:

`The average of the num object is mnum.`

What happens? Discuss with a classmate.

Exercise B)

Now instead type an r inside the quotation marks before mnum.

What happens? Discuss with a classmate.

Test 2.3

Type the following text in your document.

- With various symbols, it is **possible** to format the text *quite* much.

- You can even make ~~lists~~.

Show a tutor to get a signature.

Tutor signature:

---

3) Data Management

To become proficient in data management in R you need to play with the functions. The below exercises gives you a few exercises but please feel free to test other things to understand how to do data management in R.

3.1) Import/export

You have already imported a table as the object `data`, see Test 1.3. At the end of this section you will export the data, but we first want to do some data management.

Test 3.1

1. Import the data `example_data.csv` and assign it `wide_data`.
2. Show a tutor and get signature.

Tutor signature:

3.2) Subsetting/merging

Exercises

Exercise A)

Subset the object `data` by keeping only the first 3 columns and the first 150 rows. How many ways can you do this using only base R? Discuss with a classmate.

Exercise B)

Subset the object `data` by keeping only the column `ID` with rows of column `GT5` being `2`.

Exercise C)

Create a vector containing 100:105. Investigate if weights between 100 and 105 are represented in `data$WT` by using match.

Check if this is accurate by using unique() on `data$WT`. Could you have performed the matching in a different way? Discuss with a classmate.

Exercise D)

Subset data with only one line per individual and call the new object `first`.

If you instead wanted to keep one line per unique weight, how would you write? Discuss with a classmate.

Exercise E)

Create a new column in the object `first` called `TEST`. This column should be a combination of GT1 and GT2 separated by `_`.

Can you think of another way to do the same? Did you do the same as your classmates? Discuss.

Exercise F)

Now split `first$TEST` in two columns. Add these two "new" columns to the data as `NGT1` and `NGT2`.

Can you think of another way to do the same? Did you do the same as your classmates? Discuss.

Test 3.2

1. Check that the newly created NGT1 and NGT2 are the same as GT1 and GT2.
2. Show a tutor for signature.

Tutor signature:

3.3) Loops/ifelse

Exercises

Exercise A)

Create a new column in data called `charGT4` which is `wildtype` if `GT4` is `0` and `variant` if not.

Exercise B)

Replace the variable `charGT4` so that it is `wildtype` if `GT4` is `0`, `heterozygot` if `GT4` is `1` and `homozygot` if `GT4` is `2`.

Are there more than one way of doing this? Discuss with a classmate.

Exercise C)

Create a new column in data called `num` which starts at `2183` at the first row and decrease with 1 per row. Use a for-loop.

How could you have done this without a for-loop? Discuss with a classmate.

Test 3.3

Now, do the same exercise as in C) but only as long as `TRT` is 1. Use a while-loop.

Show your code to a tutor and get a signature.

Tutor signature:

3.4) Functions

Exercises

Exercise A)

Create a function called my_func, with one argument `x`, being the name of a csv-file that the function imports. In addition to importing, the function should subset the imported object by keeping only the first four colums, the rows of column `GT5` that are `2` and create a new column called BMI, where $BMI = WT^2/HT$.

Compare your code to your classmates. Did you write the same function?

Test 3.4

Define a function where with 2 arguments: data.frame and the position of a variable that should be removed from the input file. The function should also subset the data.frame for just the first observation per individual. Variable individual is named `ID`. The created data.frame should be exported with a generic name using the imput data.frame name and a time stamp with the suffix `.csv`. Show your code to a tutor and get a signature.

Tutor signature:

3.5) Installing, loading and starting packages

One of the strengths of R are the many available packages. A package is a set of function written by R-users. These have been checked by the R-core team but are maintained by the R-users who submitted the package.

Exercises

Exercise A)

There are two ways in R-studio to install packages. Choose one to install the package: `dplyr` What are the two different ways? Discuss with your classmates.

Exercise B)

Have a look at the internal data.frame `cars`. How many columns and rows does it have?

`select()` is a function belonging to `dplyr`, not basic R. Use the function `select()` to select the column starting with "sp".

What arguments should be used? How do you get the arguments? Does the help work? Discuss with your classmates.

Exercise C)

Load the package `plyr` using the function `library` and then again: use the function `select()` to select the column starting with "sp".

What happend? Does the help work? Why? Discuss with your classmates.

Exercise D)

Some packages have menu systems which you will work with in this exercise. Download the file *xptab1* from the student portal and put into the folder you are using as your working directory. Install and load the package `xpose4`. Active the menu system by typing `xpose4()`. The console should show the message in figure 2.

Enter `1` in the console.

Select `4: Goodness of fit plots ->`

Select `2: Basic goodness of fit plots ->`

What happend? Discuss with a classmate.

Test 3.5

1. Install and load the packages `tidyr` and `dplyr`.
2. Give the name of at least one function that belongs to the package `dplyr`. Get a signature from a tutor.

Tutor signature:

3.6) dplyr/tidyr

Exercises

Exercise A)

Import *wide.csv* as `untidy` and use the ´gather`function intidyr‘ to change the format from untidy to tidy.

Exercise B)

Subset the object `tidy` by keeping only the first 150 rows, using `dplyr`.

```
> xpose4()

               Welcome to Xpose!

       Xpose is a population analysis model
       building aid for NONMEM developed by:

       Andrew C. Hooker, Justin J. Wilkins,
       Mats O. Karlsson and E. Niclas Jonsson

       Pharmacometrics research group,
       Department of Pharmaceutical Biosciences,
       Uppsala University, Sweden.

Version: Xpose 4.6.0, NA.

http://xpose.sourceforge.net

Please report bugs to Andrew C. Hooker <andrew.hooker@farmbio.uu.se>!

Xpose, Copyright (C) 2009-2017 Andrew C. Hooker, E. Niclas Jonsson and Mats O.
Karlsson. 2005-2008 Andrew C. Hooker, Justin J. Wilkins, Mats
O. Karlsson and E. Niclas Jonsson. 1998-2004 E. Niclas Jonsson
and Mats Karlsson.

Xpose is free software and comes with ABSOLUTELY NO WARRANTY.
Xpose is made available under the terms of the GNU Lesser General
Public License (LGPL), version 3 or later. You are welcome to redistribute
it under the conditions described therein.  http://www.gnu.org/licenses/

Please enter the run number you want to process (0 = exit):
```

Figure 4: *Fig 2. The package xpose4 menu system*

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Exercise C)

Create an object called `WT` containing 100 elements, all being 70. Add these as a new column to `tidy`.

Test 3.6

Export the object `tidy` with the file name *exported_tidy_data.csv* and add a date stamp to the file name. Show a tutor and get a signature.

Tutor signature:

---

4) Graphics

In this last module, you will perform exercise related to visualising data and results. In this course, we teach how to perform the scripting for plots but which plots that are the most suitable for visualising your data/results is a completely different thing.

4.1) Working with basic graphics

For the exercises in basic graphics you will use a package called ´swirl´. ´swirl´ has the slogan **Learn R, in R**, and that is indeed how to use the package.

Exercises

Exercise A)

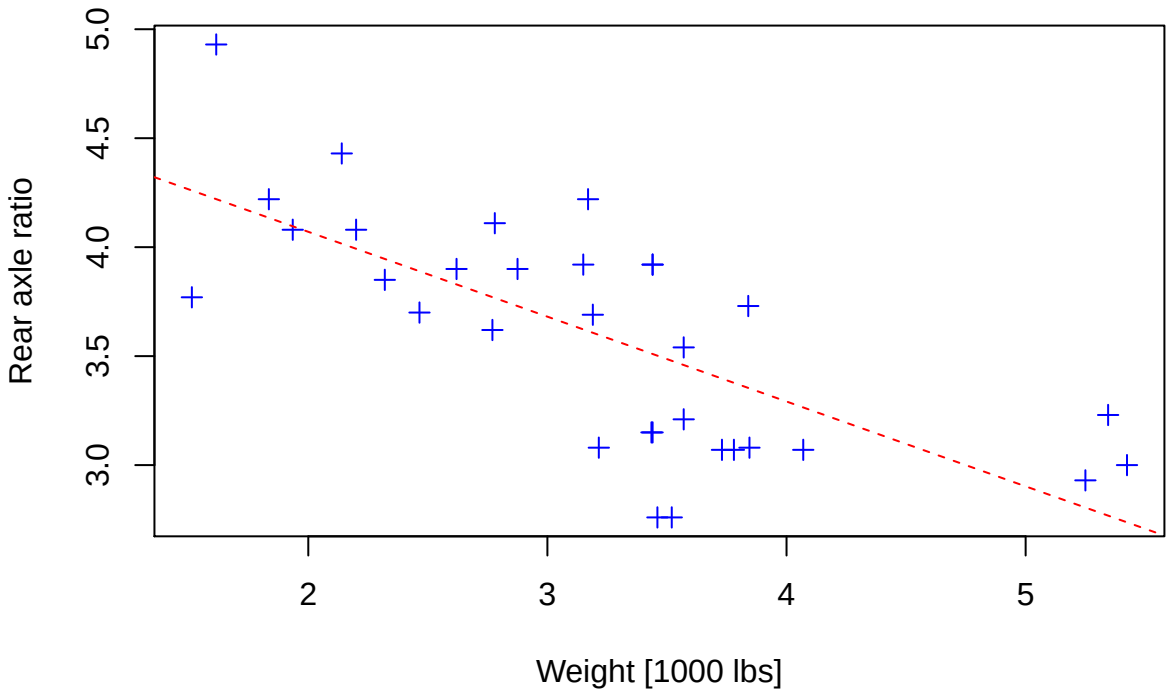
Install the package `swirl`. *Remember how a package was installed? Check above.*

Load `swirl`. *Remember how to load a library? Check above.*

Start by typing `swirl()`. Follow the instructions in the console to take the mini-course *Base Graphics*.

Test 4.1

Create a plot like the below using basic graphics and the dataframe mtcars.



NB! The line is a linear model fit (lm).

Show a tutor and get a signature.

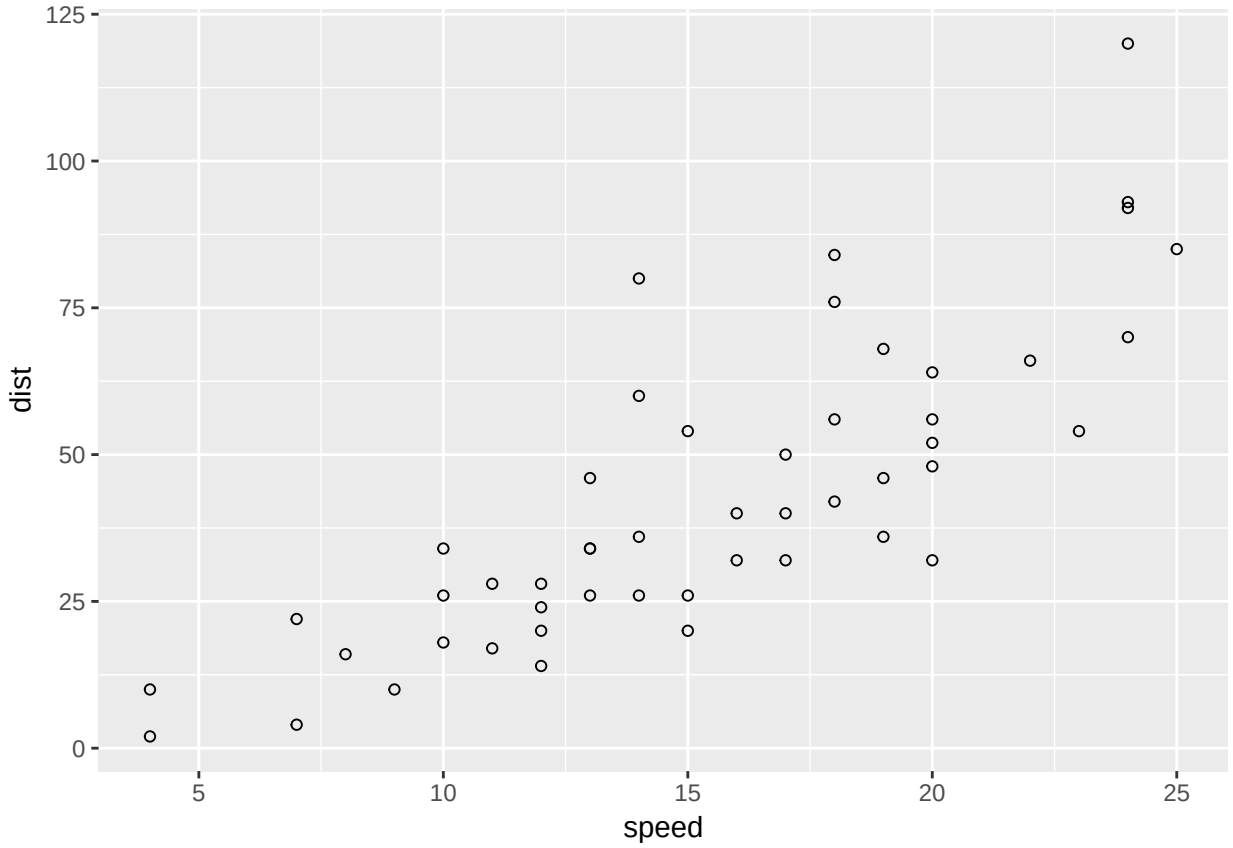
Tutor signature:

4.2) Working with ggplot

Exercises

Exercise A)

Create a graph like the below using the dataframe cars and ggplot.



Exercise B)

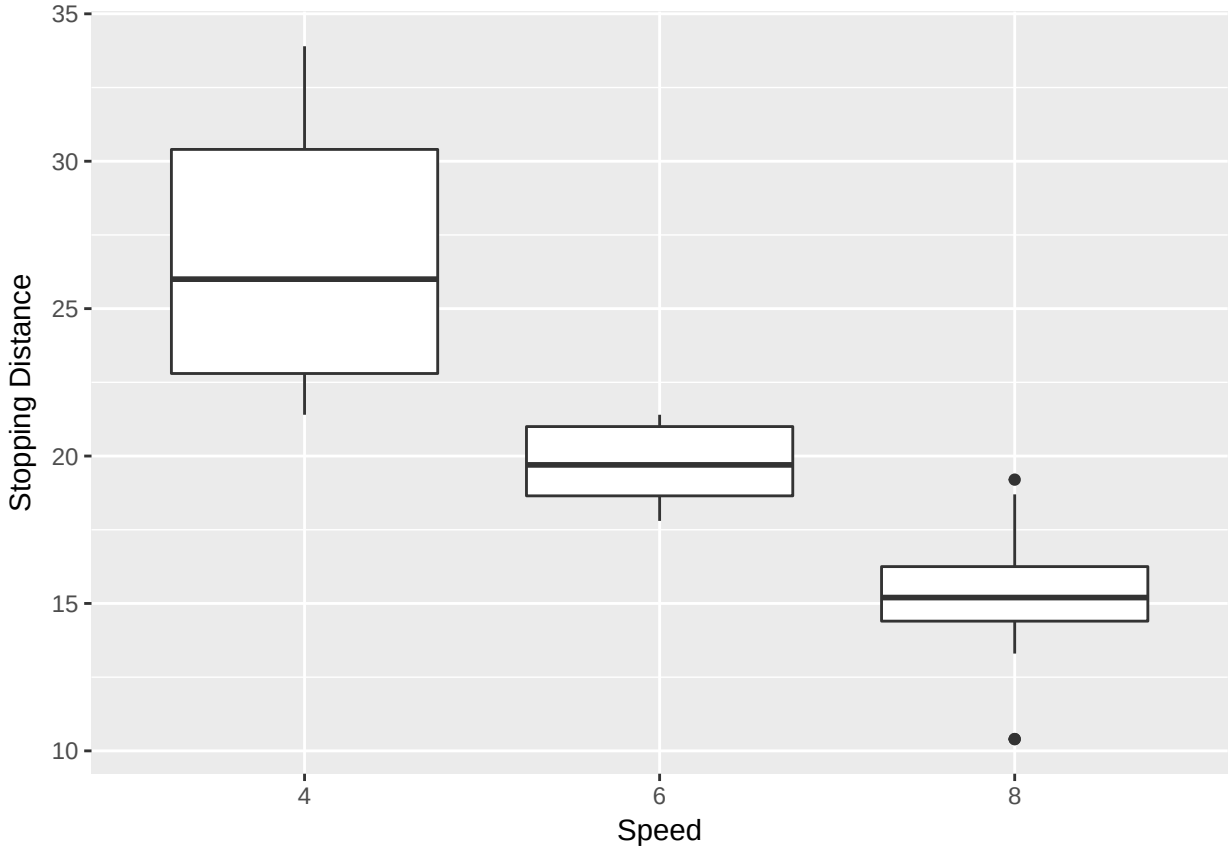
Define in the previous graph the lables for x- and y-axis as `Speed` and `Stopping Distance`.

Exercise C)

Define in the previous graph the color of the plotting symbol to red.

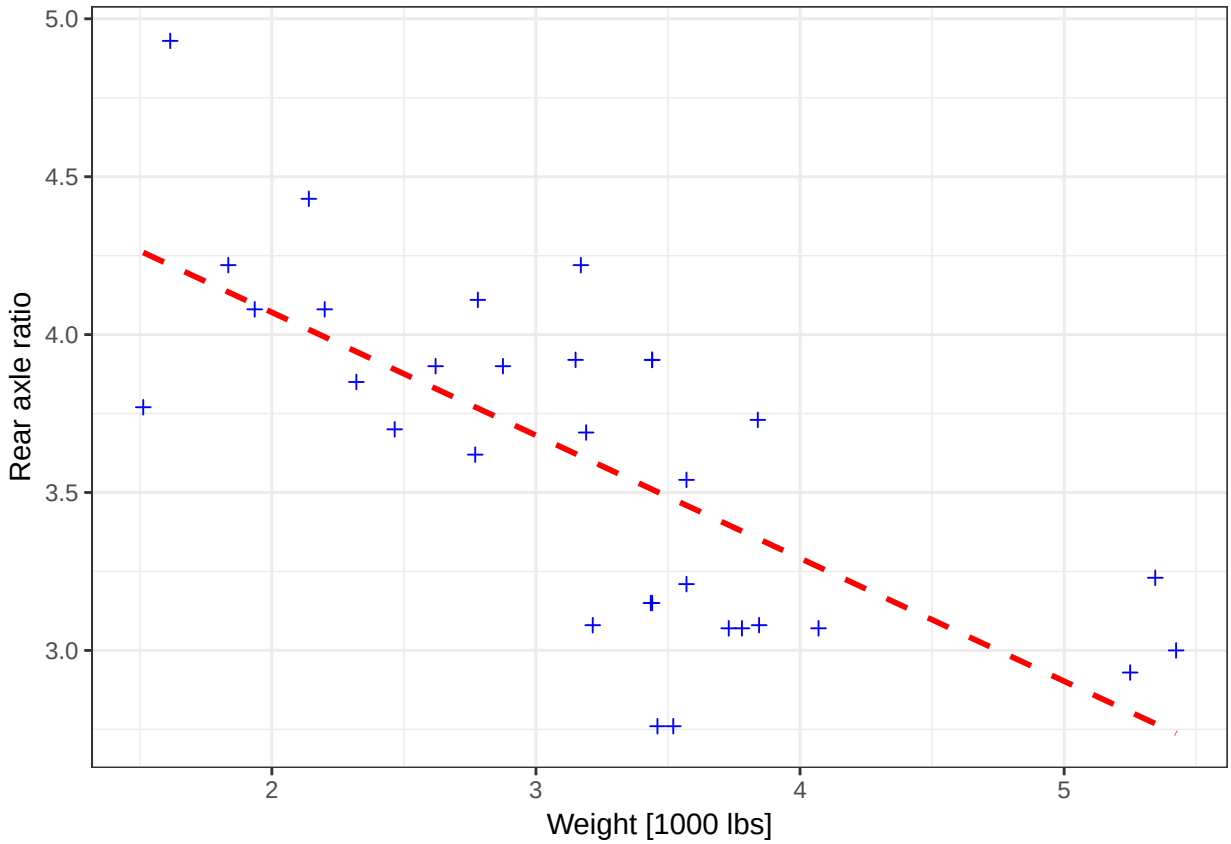
Exercise D)

Create a boxplot of mpg versus cyl using ggplot and mtcars. See below plot for reference.



Test 4.2

Create a plot like the below using gglot2 and the data.frame mtcars.



NB! The line is a linear model fit (lm).

Show a tutor and get a signature.

Tutor signature:

You are now ready for project 2! Well done!!