國立臺灣師範大學
National Taiwan Normal University

Basics in R

21 Aug 2024

Long count: 13.0.11.15.1 (1 lmox)

Sia Ming Yean

siayean@ntnu.edu.tw
@MYSia05381231

Many thanks to…

My PhD supervisor,

Prof. Julien Mayor

My R hero,

Dr. Roger Mundry

Workshop overview

Part 1: Getting to know R

- Basic operations in R

Part 2: Hands-on practice

- Work on sample data
- Plot pretty graphs

Part 3: Learning to be independent

- Seek help from the web

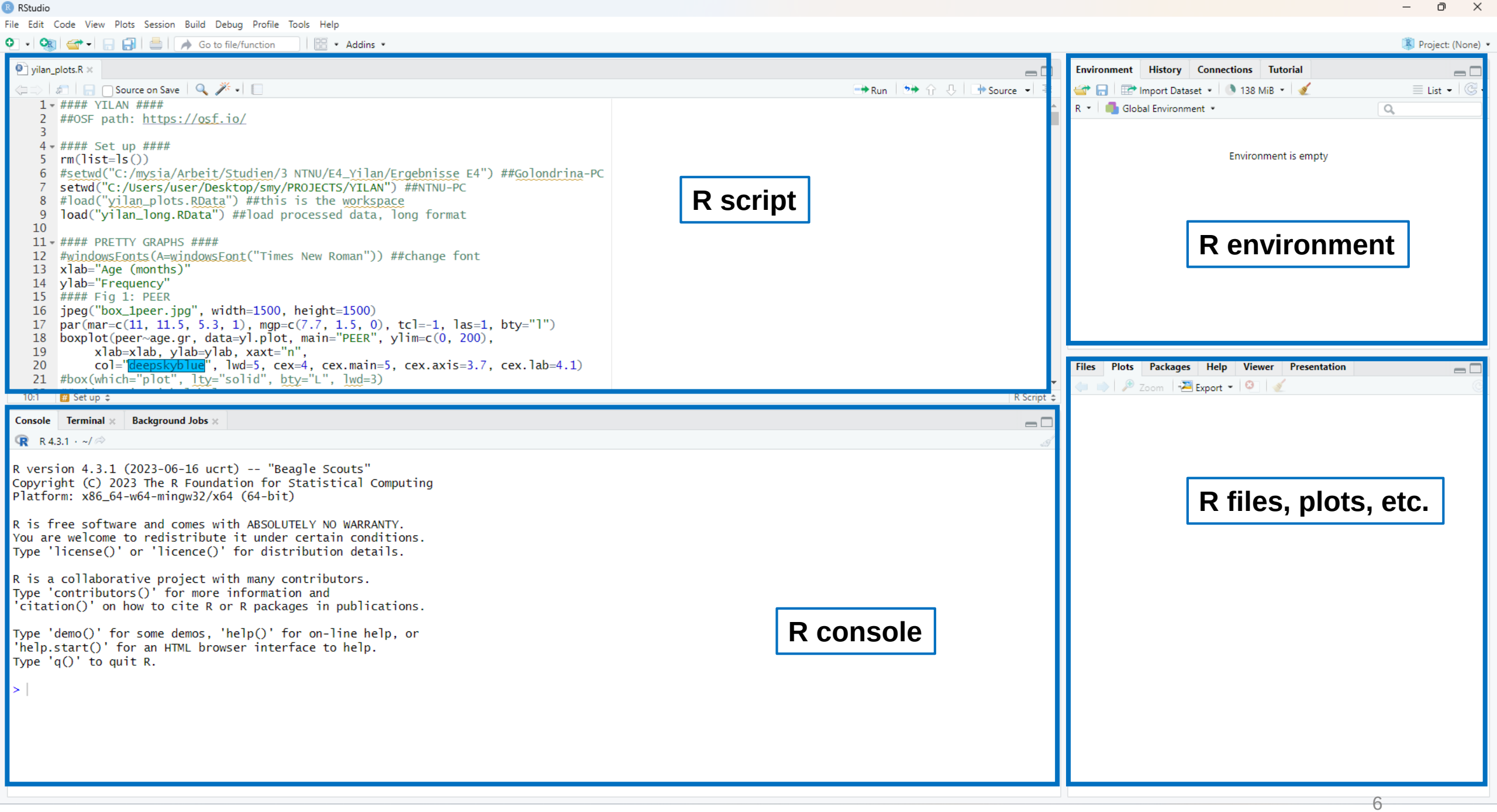
Part 1
Getting to know R

R and RStudio

- R
  - Statistical programming language
  - https://cran.r-project.org/bin/windows/base/

- RStudio (Posit)
  - A clean interface to work and interact with R
  - https://posit.co/downloads/

- I use kate text editor
  - Supports many other programming languages
  - E.g., python, MATLAB, javascript, html, etc.
  - Colour codes the script accordingly


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
yilan_plots.R
Source on Save
Run
Source
YILAN
##OSF path: https://osf.io/
Set up
rm(list=ls())
#setwd("C:/mysia/Arbeit/Studien/3 NTNU/E4_Yilan/Ergebnisse E4") ##Golondrina-PC
setwd("C:/Users/user/Desktop/smy/PROJECTS/YILAN") ##NTNU-PC
#load("yilan_plots.RData") ##this is the workspace
load("yilan_long.RData") ##load processed data, long format
PRETTY GRAPHS
#windowsFonts(A=windowsFont("Times New Roman")) ##change font
xlab="Age (months)"
ylab="Frequency"
Fig 1: PEER
jpeg("box_1peer.jpg", width=1500, height=1500)
par(mar=c(11, 11.5, 5.3, 1), mgp=c(7.7, 1.5, 0), tcl=-1, las=1, bty="l")
boxplot(peer~age.gr, data=yl.plot, main="PEER", ylim=c(0, 200),
xlab=xlab, ylab=ylab, xaxt="n",
col="deepskyblue", lwd=5, cex=4, cex.main=5, cex.axis=3.7, cex.lab=4.1)
#box(which="plot", lty="solid", bty="L", lwd=3)
10:1
Set up
R Script
R script
Environment History Connections Tutorial
Import Dataset
138 MiB
List
R
Global Environment
Environment is empty
R environment
Files Plots Packages Help Viewer Presentation
Zoom
Export
R files, plots, etc.
Console Terminal Background Jobs
R 4.3.1 · ~/
R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
R console

Getting started

- Packages

```
install.packages("<name of the package>")
library("<name of the package>")
```

- Working directories

```
getwd() ##get the current working directory
setwd("<the path>") ##use forward slash
```

- Workspace (R environment)

```
ls()
rm(x) ##removes the variable x
rm(list=ls()) ##clear workspace
```

09:30

Simple calculations

Mathematical operators	
Addition	+
Subtraction	-
Multiplication	*
Division	/
Integer division	%/%
Modulus	%%

Examples

- 10 / 3 = 3.33333
- 10 %/% 3 = 3
- 10 %% 3 = 1

Comparison operators	
Equal to	==
Not equal to	!=
Greater than	>
Less than	<

Basic mathematical functions	
Absolute	`abs()`
Mean	`mean()`
Standard deviation	`sd()`
Squareroot	`sqrt()`
Round *x* to *n* d.p.	`round(x, digits=0)`

Storing variables

- Assigning values
  - Integers: `a=3` is the same as `a <- 3`
  - Characters: `a="ID01"` is the same as `a <- "ID01"`
- Vectors

```
id=c(1, 2, 3, 4, 5, 6, 7, 8, 9)
str(id)##check the structure
id=as.character(id) ##converts the numbers into characters
id[2] ##gives the second element of the vector id
```

- Factors vs characters

```
sex=c("f", "f", "m", "f", "m", "m", "m", "f", "m")
sex=as.factor(sex)
lvl=factor(c("diff", "easy", "diff", "diff", "easy", "diff", "easy",
    "easy", "diff"), levels=c("easy", "diff"))
```

Data frames

- A data structure, like a table
- The vectors must be of the same length
- Can contain any kind of variables (characters, numbers, etc.)
- Variables are treated like a data set
- Create a data frame using the existing vectors:

```
d=data.frame(id, sex, lvl) ##create a data frame called d
str(d) ##check structure of data
d$score=c(30, 65, 51, 48, 73, 33, 80, 75, 60) ##create a new variable
summary(d) ##get summary of data
table(d$sex, d$lvl)
```

- Something cool that we can do with indexing:

```
d$id[d$score==80] ##which id scored 80?
```

10:00

Part 2a
Hands-on practice

Batch processing data

The scenario

- Experiment: 150 elementary school children were asked to read a book on a topic that they either like or not. The children's brain activation were measured using fNIRS while they were reading. They also completed a computerised flanker task to measure their attentional network. Their parents filled in an online questionnaire afterwards.

- RQ: Children activate their brain more when they are reading a book that they like.

- Data files:
  - One excel file with all data entered by different research assistants
  - 150 csv files of the flanker task

Your task as a researcher

1. Compile the 150 data files into one
2. Check for errors
3. Combine both files into one

Task 1

- File name: mock_data.xlsx
  - Sheet1: data entered by a few research assistants
  - Sheet2: data downloaded from the online questionnaire

- What we need to do:
  1. Read the excel file
     - Need to load a package
     - There are various packages (you can Google them), I use "`readxl`"
  2. There are two sheets, which means:
     - We need to specify them (default is the first sheet)
     - Then check for errors/mistakes
     - Finally combine data from both sheets into one data frame

Task 1

Let's get started:

```
setwd("<enter your path here>")
library("readxl")
d1=read_excel("mock_data.xlsx", sheet=1) ##could also specify the name
```

Check the structure of the data:

```
str(d1)
```

- It says **`tibble [150 × 14]`**. Let's convert it into a data frame

```
d1=as.data.frame(d1)
```

- Here, I overwrite the previous object. This may not always be a good idea. If you want to keep the original object and create a new one, give it a new name.

Task 1

Now, let's check the structure again:

```
> str(d1)
'data.frame':   150 obs. of  14 variables:
 $ id         : num  1 2 3 4 5 6 7 8 9 10 ...
 $ sex        : chr  "m" "f" "f" "m" ...
 $ age        : num  8 8.9 8.1 7.8 8.1 8.2 8.4 7 8.7 8.6 ...
 $ grp        : chr  "interest" "interest" "interest" "interest" ...
 $ ma.edu     : chr  "uni" "grad.sch" "grad.sch" "uni" ...
 $ fam.income: num  14218 12230 11313 15856 12675 ...
 $ lang.his   : chr  "mono" "mono" "mono" "mono" ...
 $ book       : chr  "vehicle" "animal" "country" "occupation" ...
 $ read.dur   : num  2.5 6.8 8.3 4.7 6.7 4.5 3.9 7.8 4.7 8.7 ...
 $ ch1.hbo    : chr  "1.8259000000000001" "0.95720000000000005" "0.29420000000000002" "0
 $ ch2.hbo    : chr  "1.8133999999999999" "1.7865" "2.1261999999999999" "1.1233" ...
 $ ch3.hbo    : chr  "0.42709999999999998" "0.2392" "0.60799999999999998" "1.15419999999
 $ ch4.hbo    : chr  "0.83579999999999999" "1.3149" "0.68" "3.2199999999999999E-2" ...
 $ ch5.hbo    : chr  "-0.44850000000000001" "-0.53120000000000001" "-2.2414999999999998"
```

What does *150 obs. of 14 variables* mean?

There are many variables that are characters. Is that a problem?

Task 1

Cleaning the data structure:

- I would prefer *ID* to be characters instead of numbers.
- *Sex, group, mother's education, language history* and *book* should be factors.
- *HbO levels of channels 1 – 5* should be numbers.

The script:

```
d1$id=as.character(d1$id)
d1$sex=as.factor(d1$sex)
```

Let's check the outcome:

```
> str(d1$sex)
 Factor w/ 4 levels "f","F","m","M": 3 1 1 3 1 1 3
```

Factor with four levels! Some of the data were entered as capital and some as small letters.
How would you rectify this?

Task 1

Continue cleaning the data:

```
d1$sex=as.factor(tolower(d1$sex)) ##convert all letters to lower case
d1$grp=as.factor(d1$grp)
d1$ma.edu=as.factor(d1$ma.edu)
```

Let's check the outcome:

```
> unique(d1$ma.edu)
[1] uni      grad.sch mid.sch
Levels: grad.sch mid.sch uni
```

Logically, I'd put *middle school* first, followed by *graduate school* and *university*:

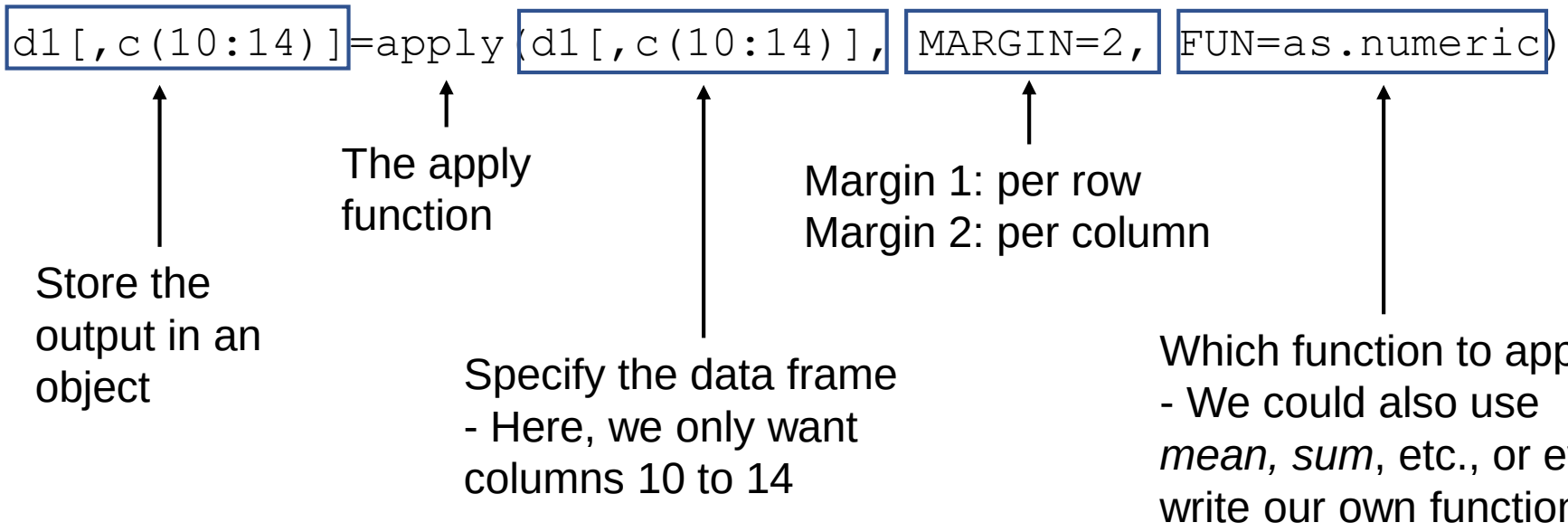
```
d1$ma.edu=factor(d1$ma.edu, level=c("mid.sch", "grad.sch", "uni"))
```

And we do the same thing for the other variables

```
d1$lang.his=factor(d1$lang.his, level=c("mono", "bi"))
d1$book=as.factor(d1$book) ##the order doesnt really matter
```

Task 1

Next, we convert the characters to numbers. There are five variables to change. Instead of writing five lines of code, we can use the `apply` function.



Task 1


```
> d1[,c(10:14)]=apply(d1[,c(10:14)], MARGIN=2, FUN=as.numeric)
Warning messages:
1: In apply(d1[, c(10:14)], MARGIN = 2, FUN = as.numeric) :
  NAs introduced by coercion
2: In apply(d1[, c(10:14)], MARGIN = 2, FUN = as.numeric) :
  NAs introduced by coercion
3: In apply(d1[, c(10:14)], MARGIN = 2, FUN = as.numeric) :
  NAs introduced by coercion
```

We received a warning message. What happened?

More about NAs:

```
any(is.na(d1$ch1.hbo)) ##check whether there is any NA in ch1.hbo
mean(d1$ch1.hbo)
mean(d1$ch1.hbo, na.rm=T) ##ignore NA and calculate mean
```

Task 1

Data from sheet 1 looks good now.

```
> str(d1)
'data.frame':   150 obs. of  14 variables:
 $ id         : chr  "1" "2" "3" "4" ...
 $ sex        : Factor w/ 2 levels "f","m": 2 1 1 2 1
 $ age        : num  8 8.9 8.1 7.8 8.1 8.2 8.4 7 8.7
 $ grp        : Factor w/ 2 levels "interest","not":
 $ ma.edu     : Factor w/ 3 levels "mid.sch","grad.sc
 $ fam.income: num  14218 12230 11313 15856 12675 ..
 $ lang.his   : Factor w/ 2 levels "mono","bi": 1 1 1
 $ book       : Factor w/ 5 levels "animal","country"
 $ read.dur   : num  2.5 6.8 8.3 4.7 6.7 4.5 3.9 7.8
 $ ch1.hbo    : num  1.826 0.957 0.294 0.606 1.294 ..
 $ ch2.hbo    : num  1.81 1.79 2.13 1.12 1.29 ...
 $ ch3.hbo    : num  0.427 0.239 0.608 1.154 -0.343 .
 $ ch4.hbo    : num  0.8358 1.3149 0.68 0.0322 1.0422
 $ ch5.hbo    : num  -0.449 -0.531 -2.241 -0.544 -0.5
```

We can move on to the data in the second sheet. How will you start?

Task 1


```
d2=read_excel("mock_data.xlsx", sheet=2)

str(d2)

d2=as.data.frame(d2)
```

```
> str(d2)
'data.frame':   150 obs. of  6 variables:
 $ id       : num  1 2 3 4 5 6 7 8 9 10 ...
 $ fb.freq  : num  2 2 3 2 2 2 3 4 3 2 ...
 $ ig.freq  : num  4 4 4 4 4 4 4 4 4 4 ...
 $ tv.hr    : num  3 5 4 4 5 4 4 4 4 5 ...
 $ read.freq: num  4 2 4 2 3 4 2 5 4 4 ...
 $ sleep.hr : num  6 7 5 6 8 5 7 6 6 6 ...
```

```
d2$id=as.character(d2$id)
```

Data from sheet 2 also looks ok. Let's combine the two data. How would you do it?

Task 1

Combining data from sheet 1 and sheet 2:

```
d.all=merge(d1, d2, by="id", all=T)
```

The merge function

Store the output in an object

Specify the two data frames that we want to merge

Merge by which variable? If the variable names are different, use `by.x` and `by.y`

Do we want to keep all data points (even if it does not exist in one of the data frames)? We can also use `all.x` and `all.y`

```
> str(d.all)
'data.frame':   150 obs. of  19 variables:
 $ id         : chr  "1" "10" "100" "101" ...
 $ sex        : Factor w/ 2 levels "f","m": 2 2 1 1 1 2 2 2 1 1 .
 $ age        : num  8 8.6 7.4 8.8 7.6 8.8 8.3 8.3 9 8.8 ...
 $ grp        : Factor w/ 2 levels "interest","not": 1 1 2 1 1 1
 $ ma.edu     : Factor w/ 3 levels "mid.sch","grad.sch",..: 3 2 2
 $ fam.income: num  14218 16578 11670 14745 12166 ...
 $ lang.his   : Factor w/ 2 levels "mono","bi": 1 1 1 1 2 1 1 1 1
 $ book       : Factor w/ 5 levels "animal","country",..: 5 1 2 4
 $ read.dur   : num  2.5 8.7 3.4 5.8 7 7.9 5.1 6.2 3.5 5.6 ...
 $ ch1.hbo    : num  1.826 1.116 1.192 0.637 0.911 ...
 $ ch2.hbo    : num  1.81 1.45 1.63 1.88 1.6 ...
 $ ch3.hbo    : num  0.427 0.591 0.636 1.167 1.198 ...
 $ ch4.hbo    : num  0.836 0.448 0.873 0.24 0.246 ...
 $ ch5.hbo    : num  -0.449 -0.772 -1.318 0.313 -0.335 ...
 $ fb.freq    : num  2 2 3 3 3 2 4 4 4 4 ...
 $ ig.freq    : num  4 4 4 4 3 3 3 4 4 4 ...
 $ tv.hr      : num  3 5 4 4 4 4 3 3 4 4 ...
 $ read.freq : num  4 4 4 3 3 5 4 3 3 2 ...
 $ sleep.hr   : num  6 6 8 5 5 8 7 5 7 6 ...
```

Before we end task 1, remember to save your workspace:

```
save.image("bir.RData")
```

11:00

Task 2

- The children also participated in a flanker task
- Conditions: neutral, congruent, incongruent
- Blocks: practice (3 trials), test (12 trials)
- Each csv file represents the data of a child

- What we need to do:
  - Loop through all the csv files
  - Identify children who did not pass the practice block (pass refers to getting all practice trials correct)
  - Calculate a total score for each condition
  - Store this information in a data frame

Task 2

- Before we use a `for` loop, we need to make sure that our script is working properly with only one data.

```
ef=read.csv("./mock_ef/mock_ef_1.csv")
```

```
> str(ef)
'data.frame':   15 obs. of  9 variables:
 $ id       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ block    : chr  "prac" "prac" "prac" "test" ...
 $ cond     : chr  "neut" "cong" "inco" "cong" ...
 $ prac.nr  : int  1 2 3 NA NA NA NA NA NA NA ...
 $ prac.corr: chr  "L" "R" "L" "" ...
 $ prac.resp: chr  "L" "R" "L" "" ...
 $ test.nr  : int  NA NA NA 1 2 3 4 5 6 7 ...
 $ test.corr: chr  "" "" "" "R" ...
 $ test.resp: chr  "" "" "" "R" ...
```

- What do you understand from the data structure?
- Is there anything that we should change?

Task 2

- I would convert *ID* into a character, *condition* into a factor, and empty cells into NAs

```
ef$id=as.character(ef$id)
ef$cond=factor(ef$cond, level=c("neut", "cong", "inco"))
ef[ef==""]=NA ##convert empty cells into NAs
```

- We can then check whether the child passed the practice block

```
is.prac.corr=ifelse(ef$prac.resp==ef$prac.corr, 1, 0)
pass=ifelse(sum(is.prac.corr, na.rm=T)==3, "pass", "no")
```

Store the output in an object

The ifelse function

Set the condition

The value if the condition is met

The value if the condition is not met

Task 2

- Next, we can calculate the total score for each condition of the test block.
- I would first calculate the score of each trial (1 for correct, 0 for incorrect), then sum up the scores for each condition

```
ef$is.test.corr=ifelse(ef$test.resp==ef$test.corr, 1, 0)
```

```
test.score=aggregate(is.test.corr~cond, data=ef, FUN=sum, na.rm=T)
```

Store the output in an object

The aggregate function

The response variable

The grouping variable

The dataset

The function to be used

Task 2

- You can double-check these numbers by hand.

```
> test.score
  cond is.test.corr
1 neut             3
2 cong             4
3 inco             0
```

- We have all the information that we want. Now, we need to store them in a dataframe so that each row represents a child.

- What variables do we want to store?

Task 2

- I would go for ID, whether the child passed the practice block, and the total scores of each condition.

```
xx=c() ##create an empty holder
xx$id=ef$id[1]
xx$pass=pass
xx$neut=test.score$is.test.corr[1]
xx$cong=test.score$is.test.corr[2]
xx$inco=test.score$is.test.corr[3]
```

```
test.score$is.test.corr[test.score$cond=="neut"]
test.score$is.test.corr[test.score$cond=="cong"]
test.score$is.test.corr[test.score$cond=="inco"]
```

- Let's try adding the first data to an empty data frame

```
d.ef=data.frame(matrix(NA, nrow=0, ncol=5))
d.ef=rbind(d.ef, xx)
```

```
> d.ef
  id pass neut cong inco
1  1 pass    3    4    0
```

Task 2

- Now, we are ready to loop through all the csv files.
- This is how a `for` loop works in general:

```
for (i in 1:n){
    ...
}
```

You can read this as "for *i* in 1 to *n*", where
- *n* is the total number of items
- *i* is the index of each of these items

What we want to do with the items

- First, we specify the list of files to loop:

```
eflist=list.files(path="./mock_ef/", pattern="*.csv", ignore.case=T,
    all.files=T)
```

```
> str(eflist)
 chr [1:146] "mock_ef_1.csv" "mock_ef_10.csv"
```

- Notice that we have 146 files instead of 150, i.e., 4 missing data.

Task 2

- We want to loop through all the csv files in the list, so we start with:

```
for (i in 1:length(eflist)){}
```

- Then we add our previous codes into the curly brackets:

```
for (i in 1:length(eflist)){
ef=read.csv(paste0(path="./mock_ef/", eflist[i])) ##use this instead of the file's name
ef$id=as.character(ef$id)
ef$cond=factor(ef$cond, level=c("neut", "cong", "inco"))
ef[ef==""]=NA
##calculate scores
is.prac.corr=ifelse(ef$prac.resp==ef$prac.corr, 1, 0)
pass=ifelse(sum(is.prac.corr, na.rm=T)==3, "pass", "no")
ef$is.test.corr=ifelse(ef$test.resp==ef$test.corr, 1, 0)
test.score=aggregate(is.test.corr~cond, data=ef, FUN=sum, na.rm=T)
##extract data
xx=c() ##create an empty holder
xx$id=ef$id[1]
xx$pass=pass
xx$neut=test.score$is.test.corr[1]
xx$cong=test.score$is.test.corr[2]
xx$inco=test.score$is.test.corr[3]
##add data to an empty dataframe
d.ef=rbind(d.ef, xx)
}
```

* Remember to indent the codes

* Change 1

* Change 2: empty data frame should be placed before the for loop

Task 3

- Finally, we can merge both data frames together to create a complete data set.
- Do you remember how to do it?

```
##combine all data together
xdata=merge(d.all, d.ef, by="id", all=T)
str(xdata) ##always check the data after merging!!
```

- And remember to save the workspace.

```
save.image("bir.RData")
```


Part 2b
Hands-on practice

Plotting data

13:30

Pretty graphs

My PI in Göttingen (Germany),

Prof. Nivi Mani

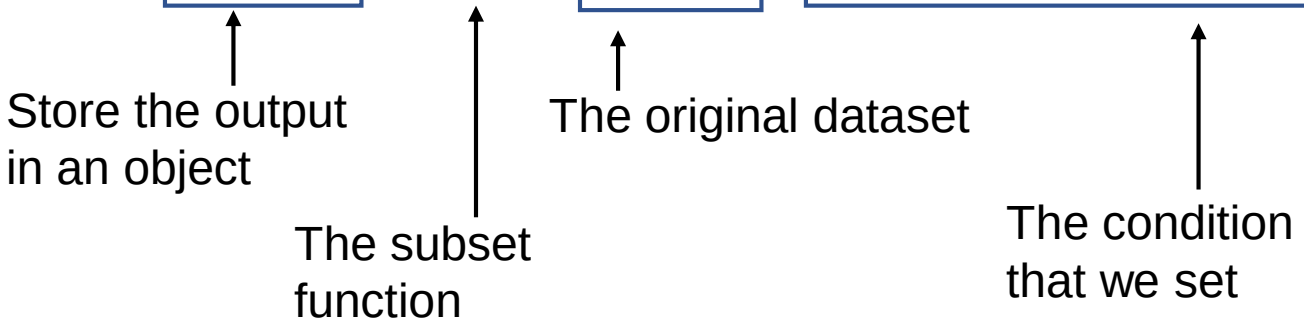
Roger’s rules

- If ggplot can do it, base R can do it better
- You want it, you get it (in base R)
- Colour for presentations, greyscale for manuscripts

Let's start plotting!

- Let's say we want only data from monolingual children who passed the flanker task.

```
x.mono=subset(xdata, xdata$lang.his=="mono")
x.pass=subset(x.mono, x.mono$pass=="pass")
```



- Notice that I created a new object instead of overwriting the original data.
- Since we have many variables, let's try several different plots
- We'll use base R to plot instead of *ggplot2*

Plot 1: Scatterplot

- Let's say we want to visualise whether older children read books longer
- We start with the basic plot:

```
plot(x=x.pass$age, y=x.pass$read.dur)
```

- What do you think about the plot?
  - The points look random and that there is no apparent relationship between age and reading duration.
  - That is because this is a randomly generated dataset.
- Nevertheless, we can beautify the plot. Let's do the following:
  1. Change the axes labels
  2. Change the points (type, size, colour)
  3. Add a line of best fit
  4. Change font size and the space around the plot

Plot 1: Scatterplot

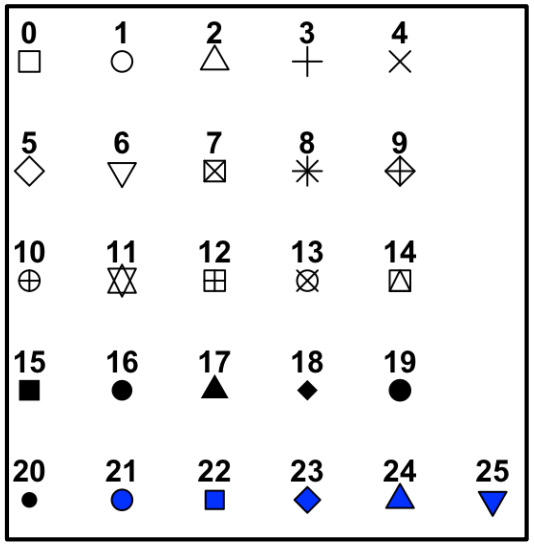
1. Change the axes labels

```
plot(x=x.pass$age, y=x.pass$read.dur,
     xlab="Age (year)", ylab="Reading duration (minutes)")
```

   - If we want to set the limits of the y-axis, we can use `ylim=c(0, 10)`

2. Change the points (type, size, colour)
   - There are 25 different symbols (see link below)
   - This is where it gets fun and you can just play around with the code till you find something that you like ☺
   - This is what I chose (for fun):

```
plot(x=x.pass$age, y=x.pass$read.dur, #ylim=c(0, 10),
     xlab="Age (year)", ylab="Reading duration (minutes)",
     pch=8, cex=1.3, lwd=2, col="blue")
```



`ylim` is commented out, so it won't be executed

`pch` means point character

`cex` means character expansion: The size of the symbol/text

`lwd` means line width: How thick do you want the line to be

http://www.sthda.com/english/wiki/r-plot-pch-symbols-the-different-point-shapes-available-in-r

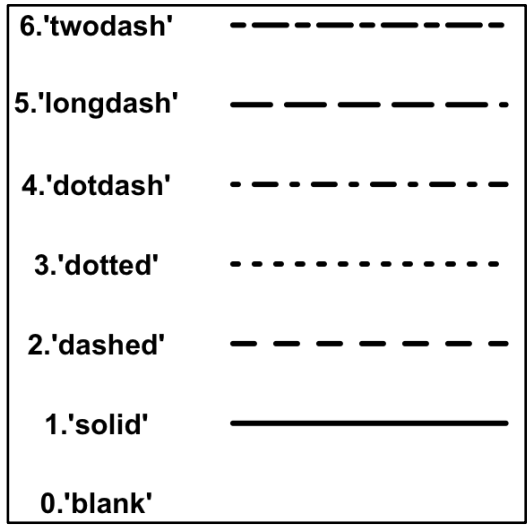
Plot 1: Scatterplot

3. Add a line of best fit

```
age.read=lm(x.pass$read.dur~x.pass$age)
abline(age.read, lty=2, lwd=4)
```

Fit a linear model

`lty` means line type: see link below for more



4. Change font size

```
plot(x=x.pass$age, y=x.pass$read.dur, #ylim=c(0, 10),
     xlab="Age (year)", ylab="Reading duration (minutes)",
     pch=8, cex=1.3, lwd=2, col="blue",
     cex.axis=1.3, cex.lab=1.8)
```

`cex.axis` means character expansion of axis ticks: How big do you want the axis ticks to be

`cex.lab` means character expansion of axis labels: How big do you want the axis labels to be

http://www.sthda.com/english/wiki/line-types-in-r-lty#google_vignette

Plot 1: Scatterplot

5. Change the space around the plot
   - To achieve this, we use plotting parameters, which should be called before plotting

```
par(mar=c(3.4, 3.4, 0.5, 0.5), mgp=c(1.8, 0.5, 0), tcl=-0.3, las=1)
```

`mar` means margin
- How much space to give outside of the plotting region
- Start from the bottom, move in a clockwise manner, i.e., B, L, T, R

Distance between plotting region &
- Axis label
- Axis tick labels
- Axis line

Length of tick marks
- Negative sign places them outside the plot

Axis label style
- 1: always horizontal
- See link below for more

https://rdocumentation.org/packages/graphics/versions/3.6.2/topics/par

Plot 1: Scatterplot

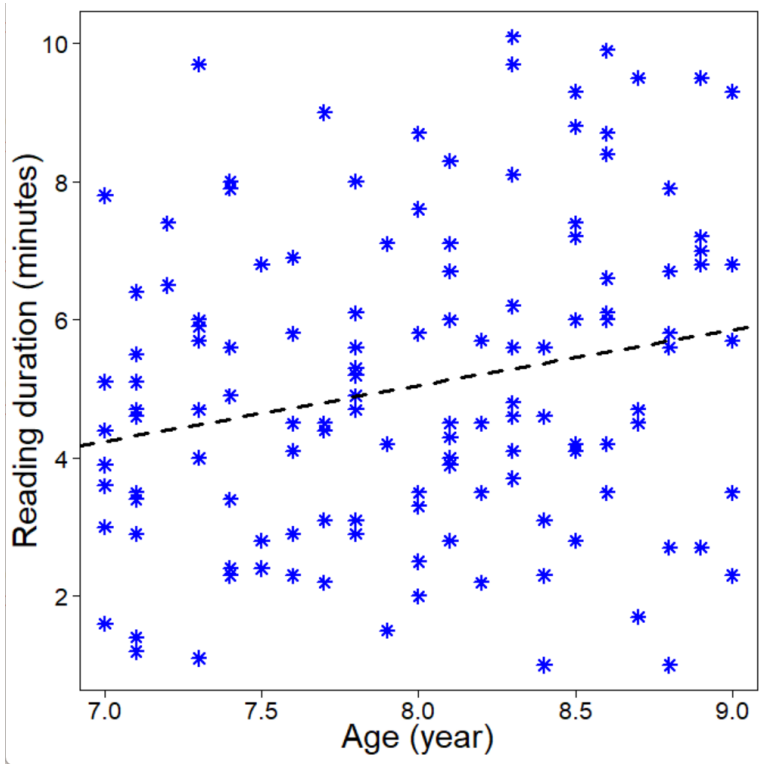
- The script will look like this:

```
##plot 1: scatterplot
par(mar=c(3.4, 3.4, 0.5, 0.5), mgp=c(1.8, 0.5, 0), tcl=-0.3, las=1)
plot(x=x.pass$age, y=x.pass$read.dur, #ylim=c(0, 10),
     xlab="Age (year)", ylab="Reading duration (minutes)",
     pch=8, cex=1.3, lwd=2, col="blue",
     cex.axis=1.3, cex.lab=1.8)
##add line
age.read=lm(x.pass$read.dur~x.pass$age)
abline(age.read, lty=2, lwd=3)
```

- To save/export the plot, you can either
  - Click plots -> export
  - Or use functions like `jpeg()` or `pdf()`

```
jpeg("plot1.jpg", width=500)
... (the plotting script)
dev.off()
```

And the plot:



Extra

- We can do a lot more, but I'll stop here and move on to other plots

- If you are interested in the correlation between the two variables, you can try this:

```
cor.test(x=x.pass$age, y=x.pass$read.dur)
##alternative="greater" ##default is two-sided
##method="spearman" ##default is pearson
```

```
> cor.test(x=x.pass$age, y=x.pass$read.dur)

        Pearson's product-moment correlation

data:  x.pass$age and x.pass$read.dur
t = 2.5251, df = 133, p-value = 0.01274
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.04661091 0.36948854
sample estimates:
      cor
0.2138837
```

A cool thing about R is that you can store and call the output:

```
c=cor.test(x=x.pass$age,
    y=x.pass$read.dur)
names(c)
p.val=round(c$p.value, 3)
r.val=round(c$estimate, 3)
```

```
> c$p.value
[1] 0.01274237
> r.val=round(c$estimate, 3)
> r.val
  cor
0.214
```

Plot 2: Boxplot

- Now, let's visualise whether children read books that they are interested in longer
- Again, we start with the basic plot:

```
plot(x=x.pass$grp, y=x.pass$read.dur)
```

- We can borrow many things from the previous script. What will you reuse?

```
##plot2: boxplot
par(mar=c(3.4, 3.4, 0.5, 0.5), mgp=c(1.8, 0.5, 0), tcl=-0.3, las=1)
plot(x.pass$grp, x.pass$read.dur,
     xlab="Age (year)", ylab="Reading duration (minutes)",
     cex.axis=1.3, cex.lab=1.8, lwd=2, col="deepskyblue")
```

  - I would go for the parameters, axis labels and cex
  - But remember to change the x-axis label

- The plot looks good, but I want to change the x-axis tick label.

Plot 2: Boxplot

1. First, we need to suppress the default tick labels:

```
xaxt="n"
```

2. Then, we insert our own labels. I want to use “interested” and “not interested”.

```
mtext(text=c("interested", "not interested"), at=c(1, 2), side=1,
line=0.4, cex=1.3)
```

What text do we want to add to the plot

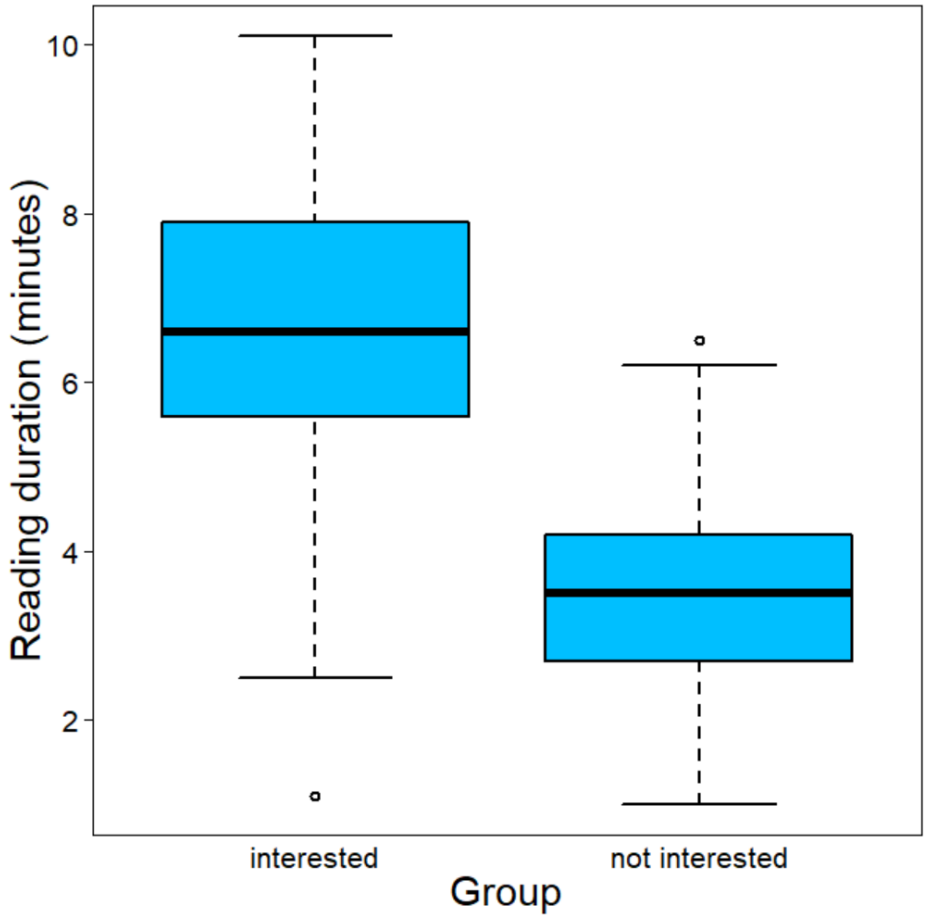
At which line do we want to insert the text?
- Positive means outside the plot
- A larger number means further from the plotting box

Where on the axis do we want to insert the text?

Which side of the plot?
- 1 means bottom
- Move in a clockwise manner, e.g., 3=top

Plot 2: Boxplot

- And this is what we get



- Let‘s compare children‘s reading duration to their brain activation during reading
- We can reuse the script and change reading duration to the average of ch1 to ch5
- How will you calculate an average score for every child?

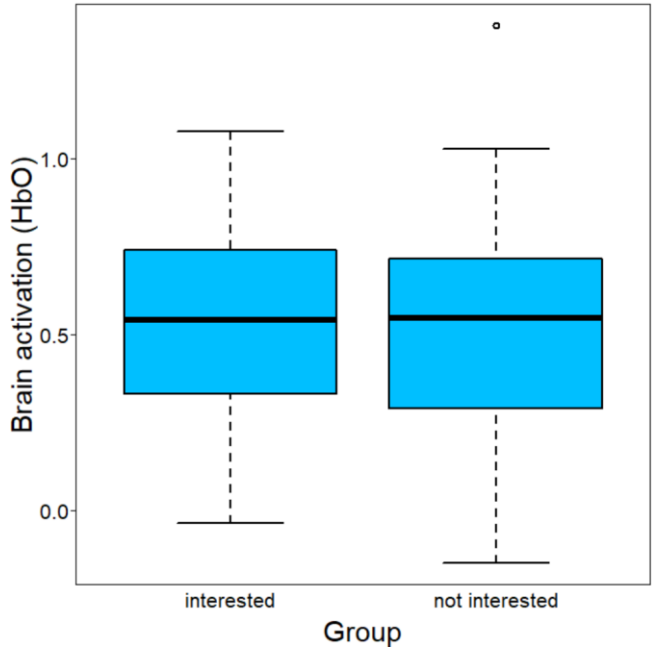
Plot 2b: Boxplot

- We can use the `apply` function and calculate the average of channels 1 to 5 for each participant (i.e., each row)

```
which(colnames(x.pass)=="ch1.hbo") ##get column number
x.pass$avg.hbo=apply(x.pass[, c(10:14)], MARGIN=1, FUN=mean, na.rm=T)
```

- Now we are ready to modify the script:

```
par(mar=c(3.4, 3.8, 0.5, 0.5), mgp=c(2.2, 0.4, 0), tcl=-0.3, las=1)
plot(x.pass$grp, x.pass$avg.hbo,
     xlab="Group", ylab="Brain activation (HbO)",
     xaxt="n", ##suppress x-axis labels
     cex.axis=1.3, cex.lab=1.8, lwd=2, col="deepskyblue")
##add x-axis labels
mtext(text=c("interested", "not interested"), at=c(1, 2), side=1,
      line=0.4, cex=1.3)
```



Plot 2: Boxplot

- Finally, let's place both plots side-by-side. To do so, we add this to the parameters:

```
mfrow=c(1, 2)
```

- And run the plotting script:

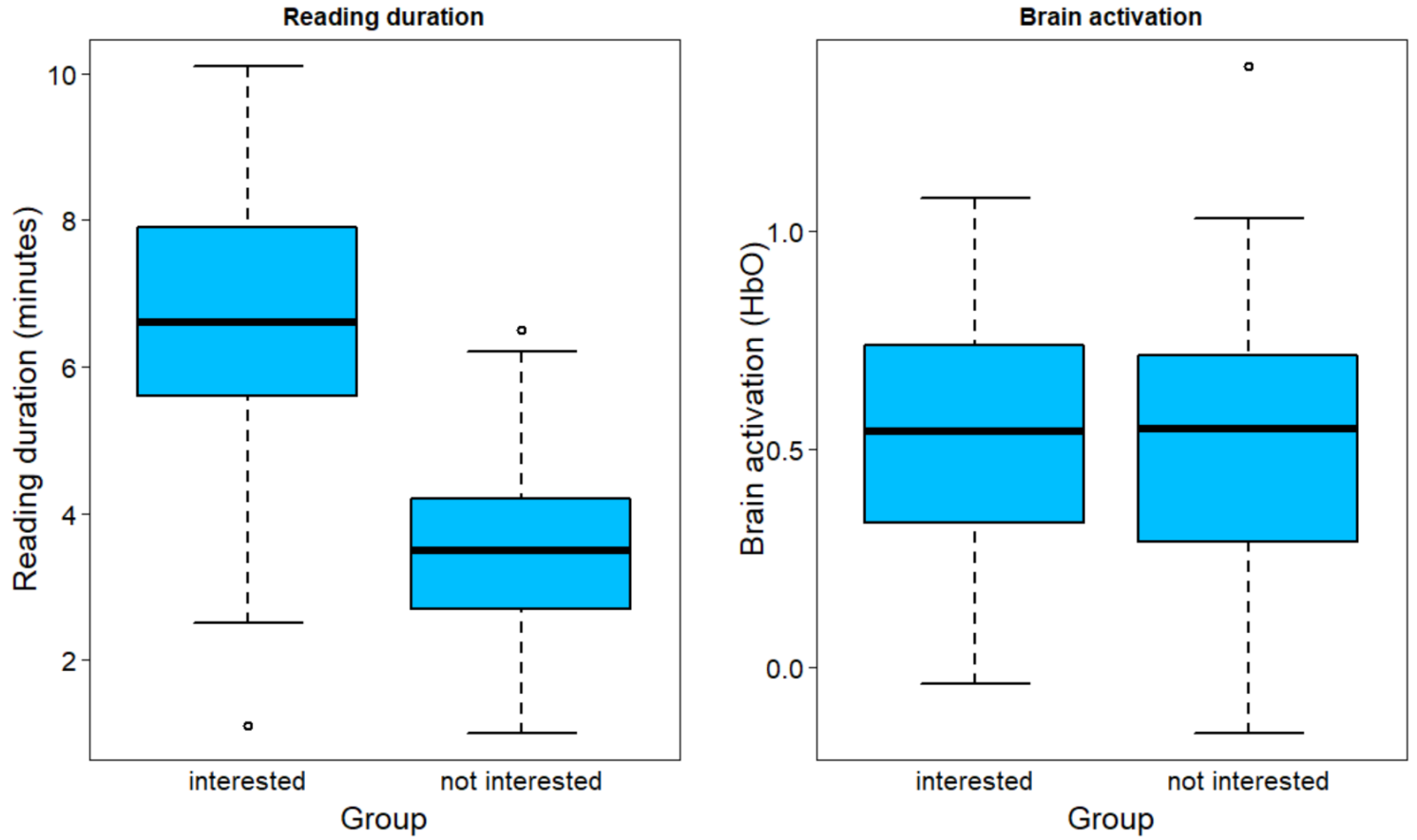
*Added more space

*Added title

```
##both plots together
par(mfrow=c(1, 2), mar=c(3.4, 3.6, 2, 2), mgp=c(2, 0.5, 0), tcl=-0.3, las=1)
##plot reading duration
plot(x.pass$grp, x.pass$read.dur, main="Reading duration",
     xlab="Group", ylab="Reading duration (minutes)", xaxt="n",
     cex.axis=1.3, cex.lab=1.6, lwd=2, col="deepskyblue")
mtext(text=c("interested", "not interested"), at=c(1, 2), side=1,
      line=0.4, cex=1.3)
##plot brain activation
plot(x.pass$grp, x.pass$avg.hbo, main="Brain activation",
     xlab="Group", ylab="Brain activation (HbO)", xaxt="n",
     cex.axis=1.3, cex.lab=1.6, lwd=2, col="deepskyblue")
mtext(text=c("interested", "not interested"), at=c(1, 2), side=1,
      line=0.4, cex=1.3)
```

Plot 2: Boxplot

- And this is what we get



Extra: t-test

```
t.test(read.dur~grp, data=x.pass)
```

```
> t.test(read.dur~grp, data=x.pass)

        Welch Two Sample t-test

data:  read.dur by grp
t = 11.427, df = 117.19, p-value < 2.2e-16
alternative hypothesis: true difference in means bet
95 percent confidence interval:
 2.583958 3.667425
sample estimates:
mean in group interest      mean in group not
              6.568116               3.442424
```

```
t.test(avg.hbo~grp, data=x.pass)
```

```
> t.test(avg.hbo~grp, data=x.pass)

        Welch Two Sample t-test

data:  avg.hbo by grp
t = 0.48976, df = 125.79, p-value = 0.6252
alternative hypothesis: true difference in means bet
95 percent confidence interval:
 -0.07173666  0.11891946
sample estimates:
mean in group interest      mean in group not
            0.5462398              0.5226484
```