Introduction to Data Analysis in R and RStudio: WORKSHOP

Harriet Cant and Dr Victoria Palin

Whilst we're waiting for everyone to join…

- Open a new script and save into the same folder as your data (file>save as…)

- At the start of your script:
  - Set your working directory to the folder you have your data saved to using `setwd(…)`
  - Load in the data you previously downloaded using `read.csv(…)`

Agenda/housekeeping

START 12:30
Intro (15 mins)
Loading and cleaning data (35 mins)
Summarising data (25 mins)
BREAK 13:45-14:00
Visualising data (25 mins)
Hypothesis testing (15 mins)
Modelling (35 mins)
Closing remarks (15 mins)
END 15:30

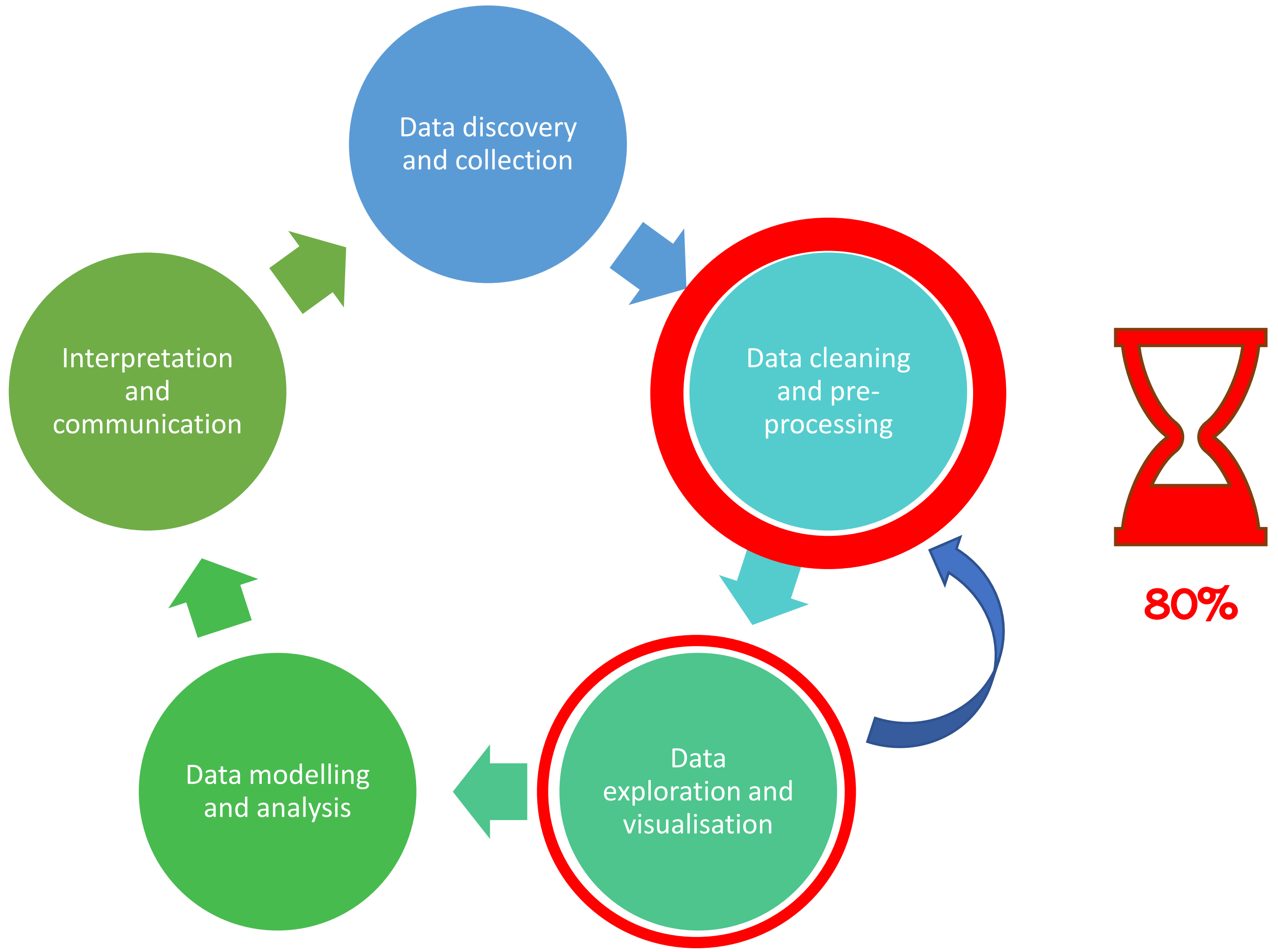
Interactive workshop: "live coding" format

Vicki

Chantelle

Hattie

Intro – the data analysis life cycle


Data discovery and collection
Data cleaning and pre-processing
Data exploration and visualisation
Data modelling and analysis
Interpretation and communication
80%

Adapted from https://www.theknowledgeacademy.com/blog/blog-data-analytics-lifecycle/

Intro – the data

- 2022 **survey** study of **410 mother-infant pairs** (Sandoz et al.)
- Aim: to better understand the link between maternal mental health and infant sleep

Maternal postpartum **depression** symptoms (Edinburgh Postnatal Depression Scale, EPDS)	EPDS_X
Maternal **anxiety** symptoms (Anxiety Subscale of the Hospital Anxiety and Depression Scale, HADS-A)	HADS_X
Maternal report of **child-birth related PTSD** (CB-PTSD) symptoms (City Birth Trauma Scale, City BiTS)	CBTS_M_X CBTS_X

Nocturnal **sleep duration** (between 7pm and 7 am)	Sleep_night_duration_bb1
How **many times the infant usually wakes** during the night	night_awakening_number_bb1
How the infant normally **falls asleep**	how_falling_asleep_bb1

Participant age	Marital status	Education	Gestational age	Pregnancy type	Infant sex	Infant age

Intro – today’s aims

Using the data collected by Sandoz et al:

- To understand how the R programming language works (understanding the ‘syntax’)
- To be able to confidently work through a data analysis lifecycle using R
    - Cleaning
    - Exploring: summarising and visualising
    - Modelling and interpreting

The session is intended to be **top-level.** We aim to equip you with a solid foundational understanding which you can then build on outside of this workshop.

Lesson 1: Loading in and cleaning data

Getting started

- Open a new script and save into the same folder as your data (file>save as…)

- At the start of your script:
  - Set your working directory to the folder you have your data saved to using `setwd(…)`
  - Load in the data you previously downloaded using `read.csv(…)`

Initial inspection of the data

```
# Inspect the 'head' (first 6 observations)
of the dataset
head(data)

# View entire dataset
View(data)

# Produce a list of column names
colnames(data)

# Investigate the 'structure' of the data
str(data)
```

ACTIVITY 1

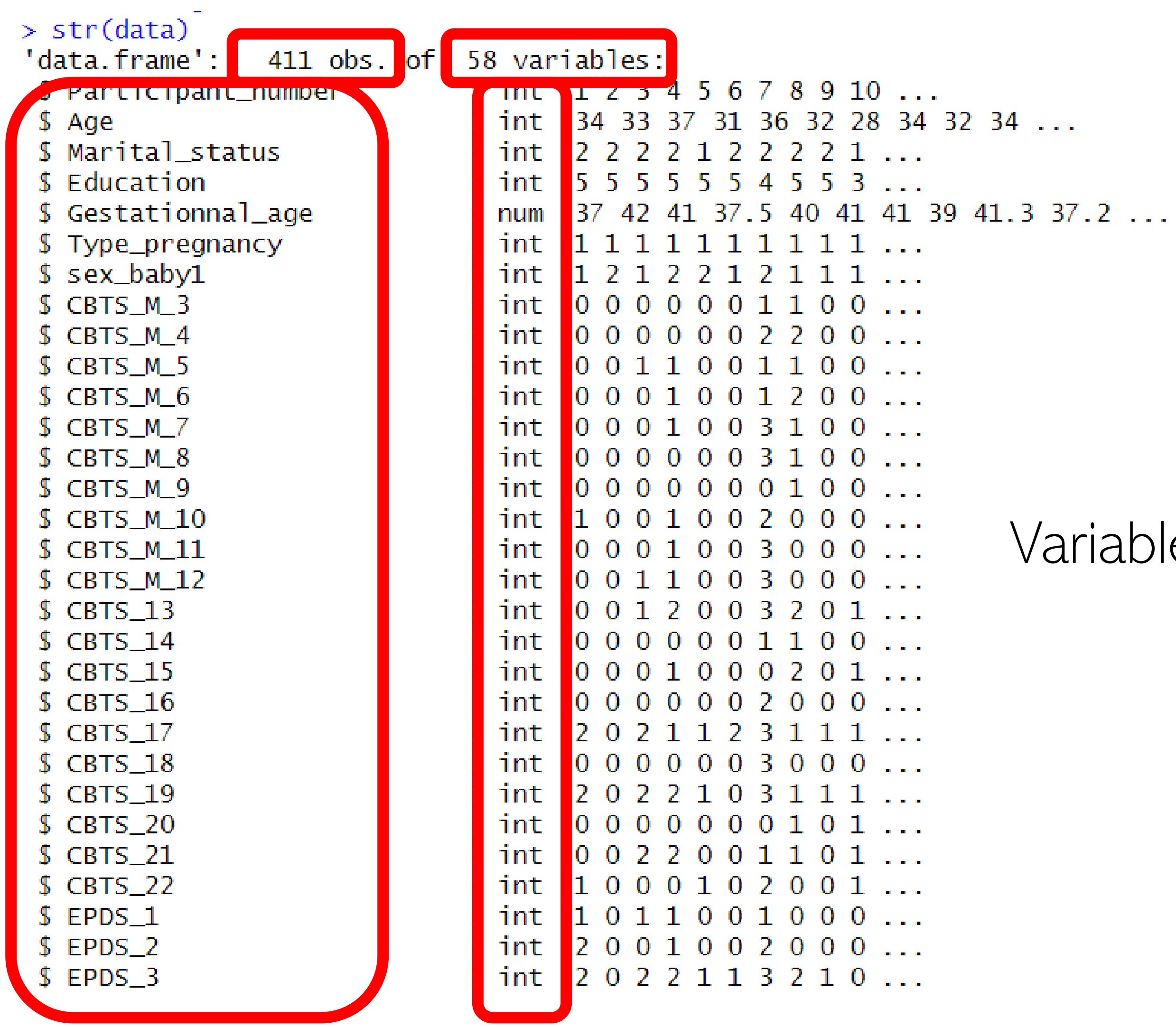
Inspect the data using the functions provided.

Is everything as you'd expect given what we know about the data?

Understanding str…

Number of rows (observations) and columns (variables) → 411 observations for 410 mother-infant pairs?

Same as colnames



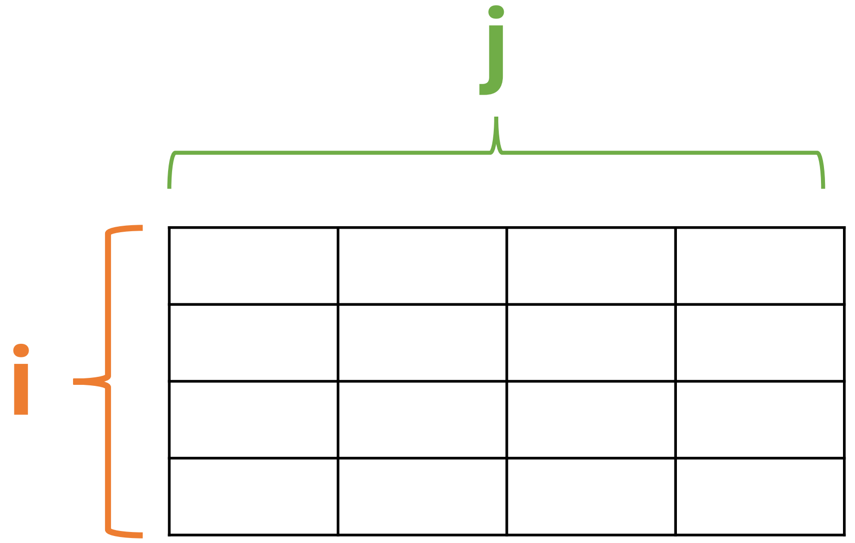
```
> str(data)
'data.frame':   411 obs. of  58 variables:
 $ Participant_number  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Age                 : int  34 33 37 31 36 32 28 34 32 34 ...
 $ Marital_status      : int  2 2 2 2 1 2 2 2 2 1 ...
 $ Education           : int  5 5 5 5 5 5 4 5 5 3 ...
 $ Gestationnal_age    : num  37 42 41 37.5 40 41 41 39 41.3 37.2 ...
 $ Type_pregnancy      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ sex_baby1           : int  1 2 1 2 2 1 2 1 1 1 ...
 $ CBTS_M_3            : int  0 0 0 0 0 0 1 1 0 0 ...
 $ CBTS_M_4            : int  0 0 0 0 0 0 2 2 0 0 ...
 $ CBTS_M_5            : int  0 0 1 1 0 0 1 1 0 0 ...
 $ CBTS_M_6            : int  0 0 0 1 0 0 1 2 0 0 ...
 $ CBTS_M_7            : int  0 0 0 1 0 0 3 1 0 0 ...
 $ CBTS_M_8            : int  0 0 0 0 0 0 3 1 0 0 ...
 $ CBTS_M_9            : int  0 0 0 0 0 0 0 1 0 0 ...
 $ CBTS_M_10           : int  1 0 0 1 0 0 2 0 0 0 ...
 $ CBTS_M_11           : int  0 0 0 1 0 0 3 0 0 0 ...
 $ CBTS_M_12           : int  0 0 1 1 0 0 3 0 0 0 ...
 $ CBTS_13             : int  0 0 1 2 0 0 3 2 0 1 ...
 $ CBTS_14             : int  0 0 0 0 0 0 1 1 0 0 ...
 $ CBTS_15             : int  0 0 0 1 0 0 0 2 0 1 ...
 $ CBTS_16             : int  0 0 0 0 0 0 2 0 0 0 ...
 $ CBTS_17             : int  2 0 2 1 1 2 3 1 1 1 ...
 $ CBTS_18             : int  0 0 0 0 0 0 3 0 0 0 ...
 $ CBTS_19             : int  2 0 2 2 1 0 3 1 1 1 ...
 $ CBTS_20             : int  0 0 0 0 0 0 0 1 0 1 ...
 $ CBTS_21             : int  0 0 2 2 0 0 1 1 0 1 ...
 $ CBTS_22             : int  1 0 0 0 1 0 2 0 0 1 ...
 $ EPDS_1              : int  1 0 1 1 0 0 1 0 0 0 ...
 $ EPDS_2              : int  2 0 0 1 0 0 2 0 0 0 ...
 $ EPDS_3              : int  2 0 2 2 1 1 3 2 1 0 ...
```

Variable types

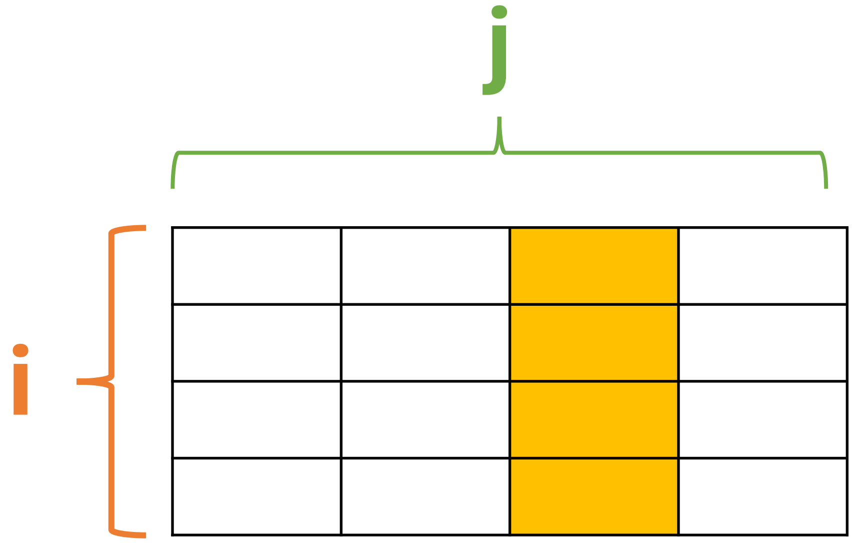
Indexing

- All data from the 5th observation
- All collected 'Age' data
- 'Age' data from the 5th observation

Indexing = the process of locating specific information within a data structure



data[i, j]

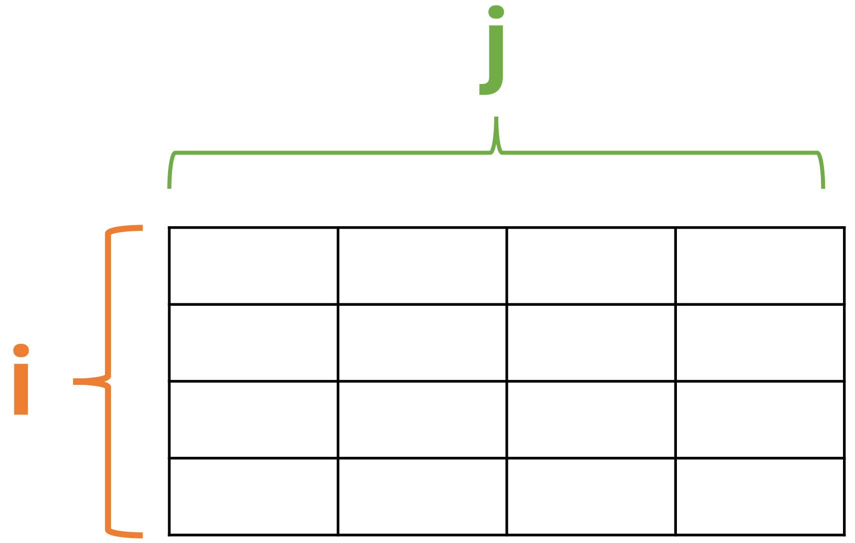


data[,3]

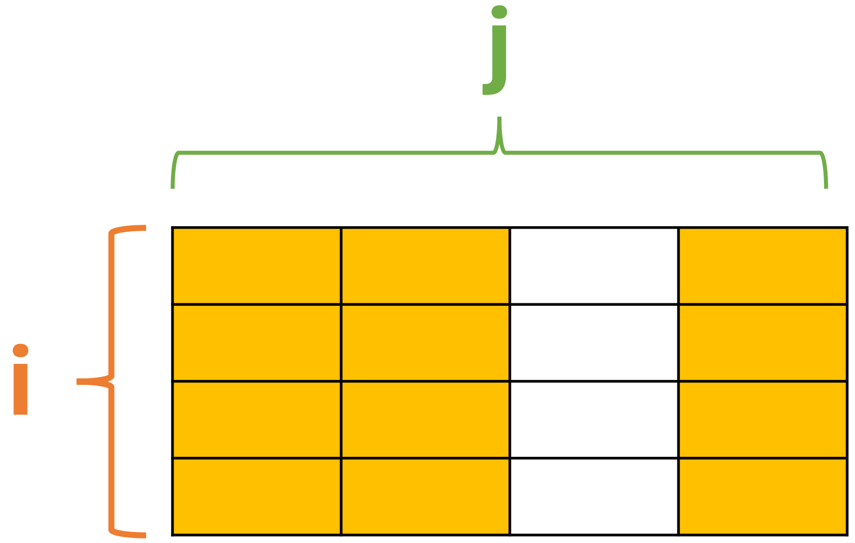
Indexing

- All data from the 5th observation
- All collected 'Age' data
- 'Age' data from the 5th observation

Indexing = the process of locating specific information within a data structure



data[i, j]

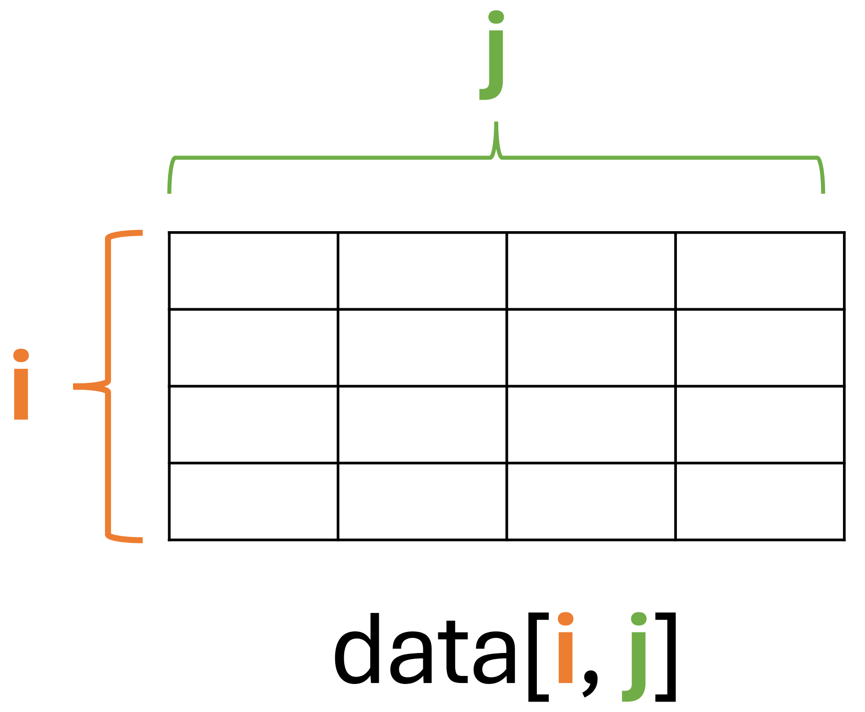


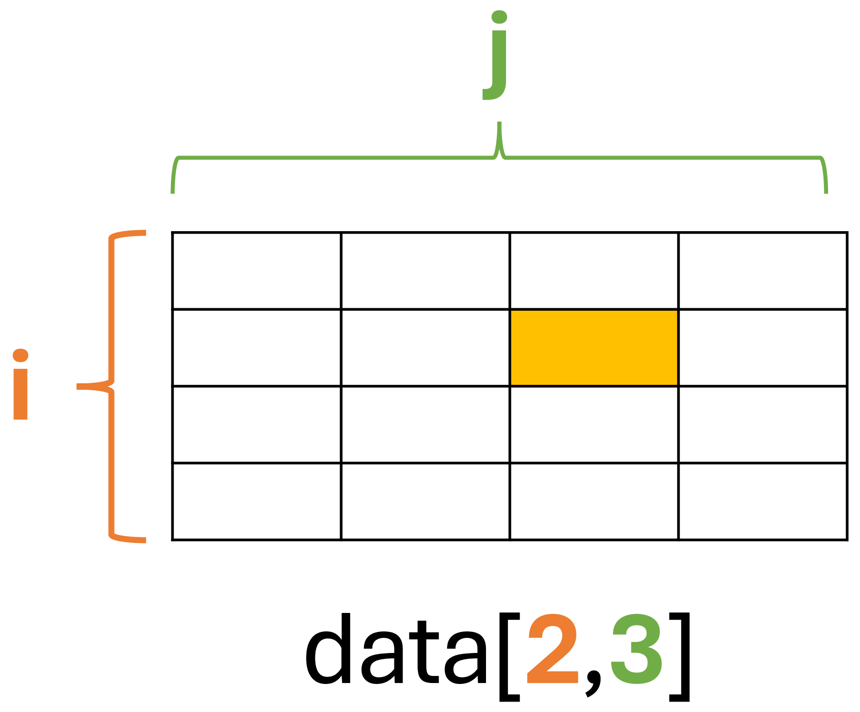
data[,-3]

Indexing

- All data from the 5th observation
- All collected 'Age' data
- 'Age' data from the 5th observation

Indexing = the process of locating specific information within a data structure

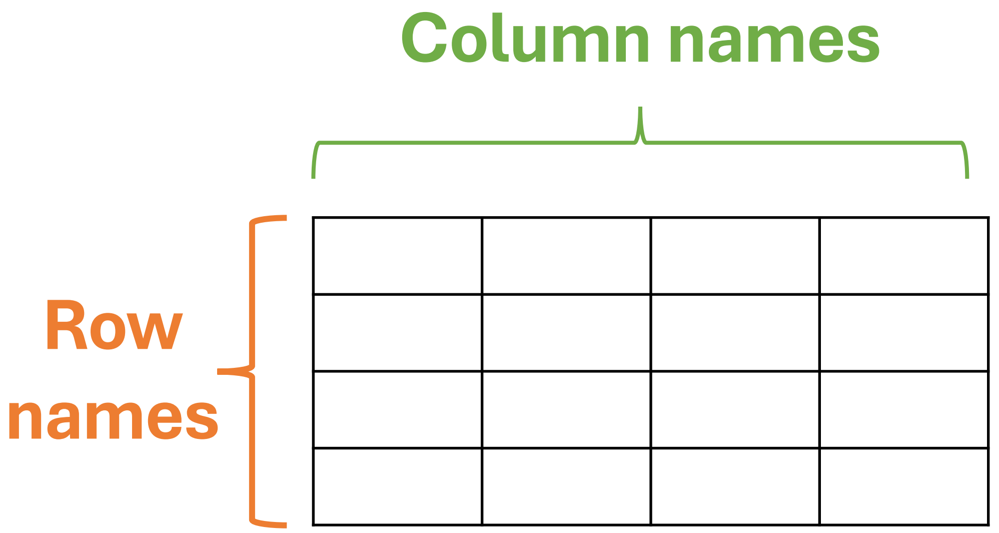




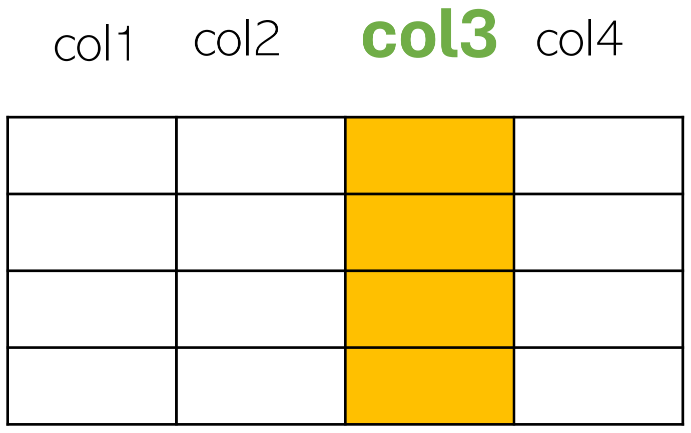
Indexing

- All data from the 5th observation
- All collected 'Age' data
- 'Age' data from the 5th observation

Indexing = the process of locating specific information within a data structure



data["row", "column"]

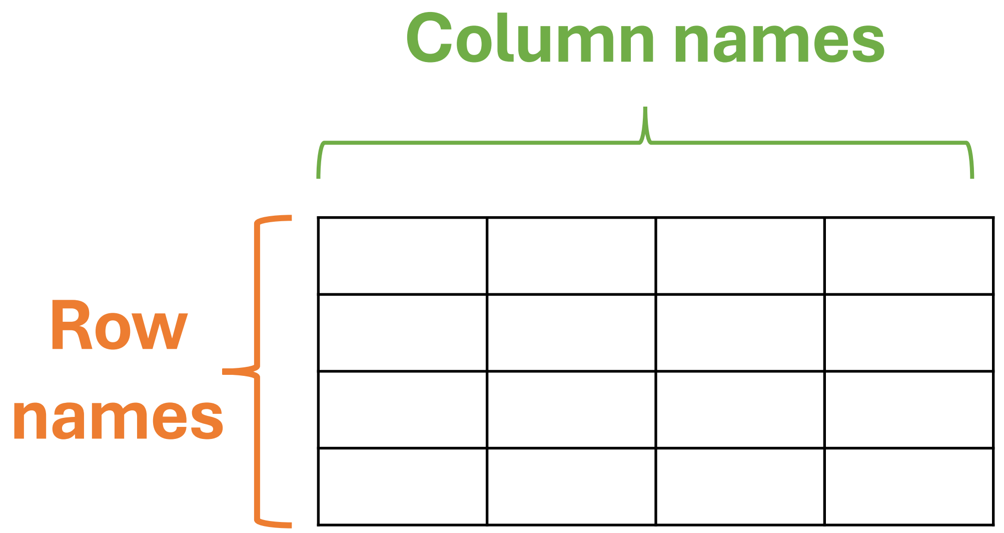


data[,"col3"]

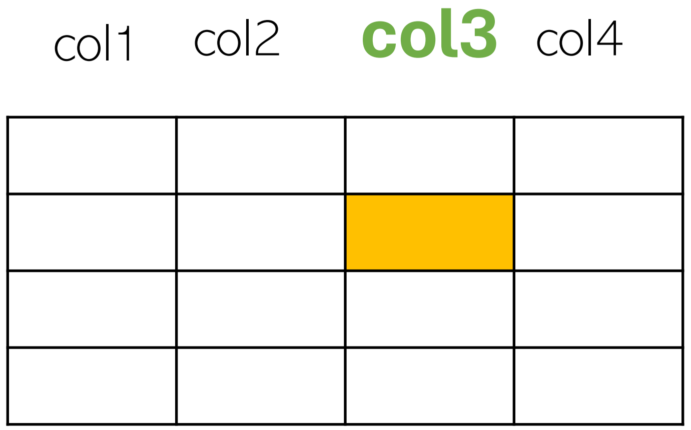
Indexing

- All data from the 5th observation
- All collected 'Age' data
- 'Age' data from the 5th observation

Indexing = the process of locating specific information within a data structure



data["row", "column"]

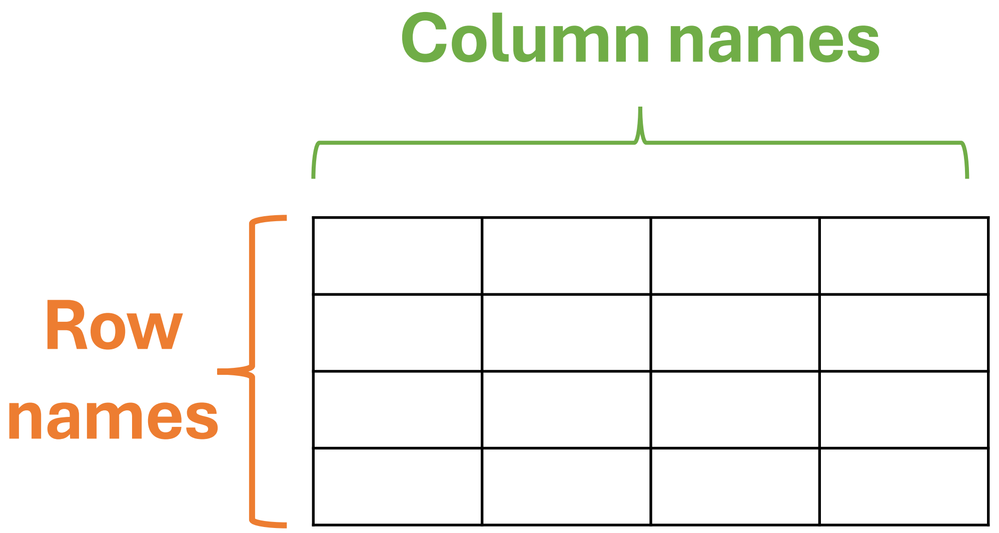


data[2,"col3"]

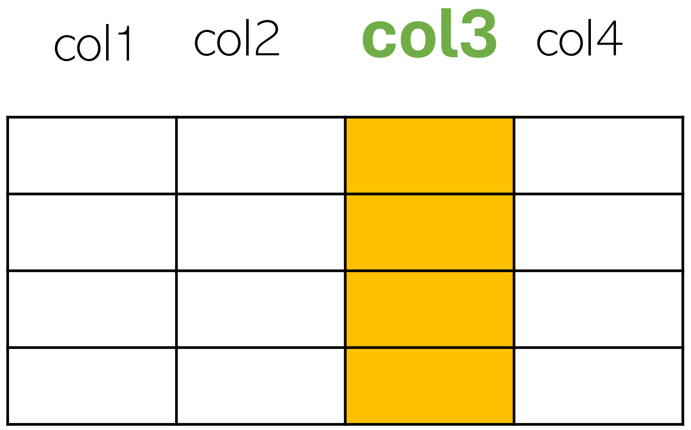
Indexing

- All data from the 5th observation
- All collected 'Age' data
- 'Age' data from the 5th observation

Indexing = the process of locating specific information within a data structure



data$**column**

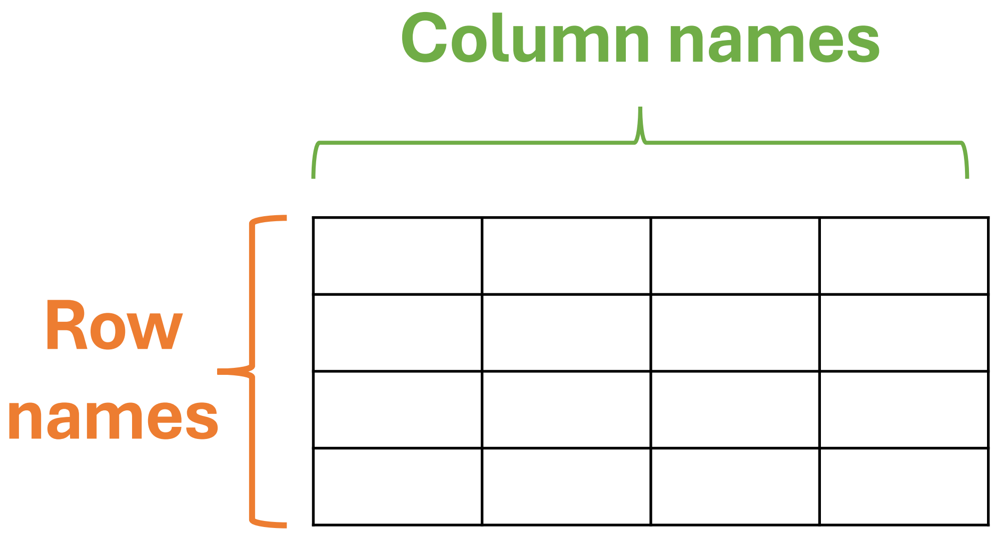


data$**col3**

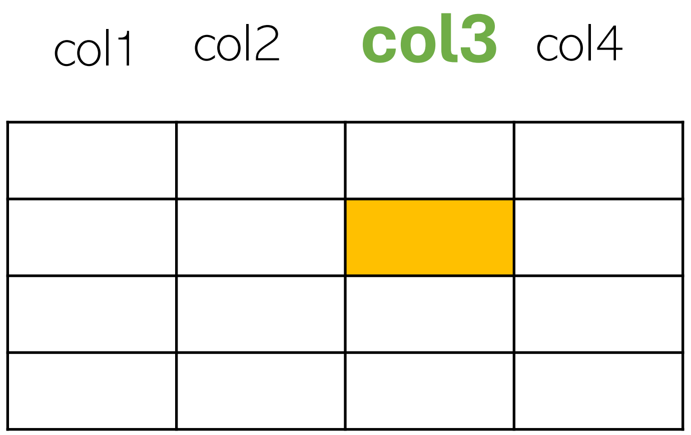
Indexing

- All data from the 5th observation
- All collected 'Age' data
- 'Age' data from the 5th observation

Indexing = the process of locating specific information within a data structure



data$column



data[2,]$col3

Indexing

<u>ACTIVITY 2</u>

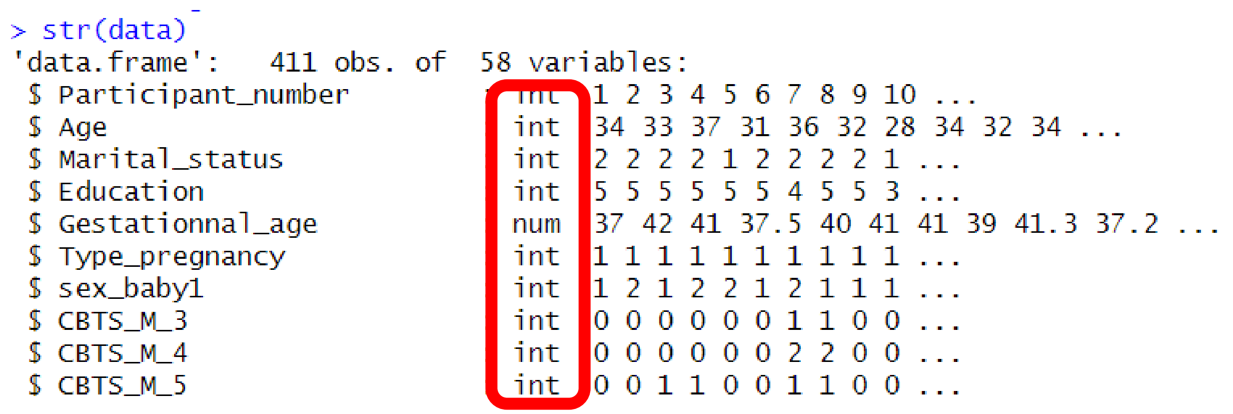
1. Can you think of why you might prefer one method of column extraction to another (e.g. by name vs by column number)?

2. Extract **age** data from the 5th **observation**

3. In the pre-requisite activity, we learned that vectors are represented by c(…) or : (e.g. c(1,2,3), c("hello", "hi"), or 1:5). Using vectors, can you try to **extract all data relating to observations 10-40**?

4. Try using the function **`which(data$sex_baby1 == 1)`**. What is the output? Can you use this to create a subset of participants with male or female infants?

Variable types



```
> str(data)
'data.frame':   411 obs. of  58 variables:
 $ Participant_number       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Age                      : int  34 33 37 31 36 32 28 34 32 34 ...
 $ Marital_status           : int  2 2 2 2 1 2 2 2 2 1 ...
 $ Education                : int  5 5 5 5 5 5 4 5 5 3 ...
 $ Gestationnal_age         : num  37 42 41 37.5 40 41 41 39 41.3 37.2 ...
 $ Type_pregnancy           : int  1 1 1 1 1 1 1 1 1 1 ...
 $ sex_baby1                : int  1 2 1 2 2 1 2 1 1 1 ...
 $ CBTS_M_3                 : int  0 0 0 0 0 0 1 1 0 0 ...
 $ CBTS_M_4                 : int  0 0 0 0 0 0 2 2 0 0 ...
 $ CBTS_M_5                 : int  0 0 1 1 0 0 1 1 0 0 ...
```

- Numeric ('num')
  - 0, 1.3, 43.231231, 1000000, Inf
- Integer ('int')
  - 0L, 1L, 2L, 3L
- Character ('chr')
  - "hello world", "…."
- Factor ('Factor')
  - "<18", "18-64", "65+" → fixed set of acceptable values
- Logical ('logi')
  - TRUE, FALSE

Variable types – useful functions

```
# Use the 'class' command to view classes for individual columns (or values)
> class(3)
[1] "numeric"

> class(3L)
[1] "integer"

> class("3")
[1] "character"


# Other functions exist which tell you whether a variable belongs to a certain
class
is.character(…)
is.numeric(…)
is.integer(…)
```

Variable types

It’s important to make sure your variable formats reflect the true nature of your data, as it can significantly alter how R interprets them.

is.character(…) → as.character(…)

```
> 1:5
[1] 1 2 3 4 5

> is.character(1:5)
[1] FALSE

> as.character(1:5)
[1] "1" "2" "3" "4" "5"
```

Change column types using <-

```
data$Participant_number <-
        as.character(data$Participant_number)

data$Type_pregnancy <-
        as.factor(data$Type_pregnancy)
```

Variable types

Change column types using the assignment command, `<-`

```
data$Participant_number <- as.character(data$Participant_number)

data$Type_pregnancy <- as.factor(data$Type_pregnancy)
```

<u>ACTIVITY 3</u>

Inspect the data using ‘str’ and identify whether there are any variables whose type may need to be changed, and then change these!

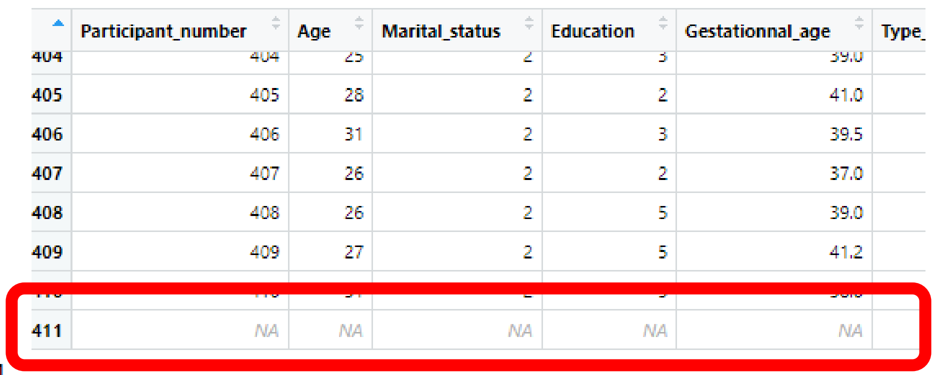
After you have changed the variables, use the ‘str’ command again to check this worked.

What do we mean by data cleaning?

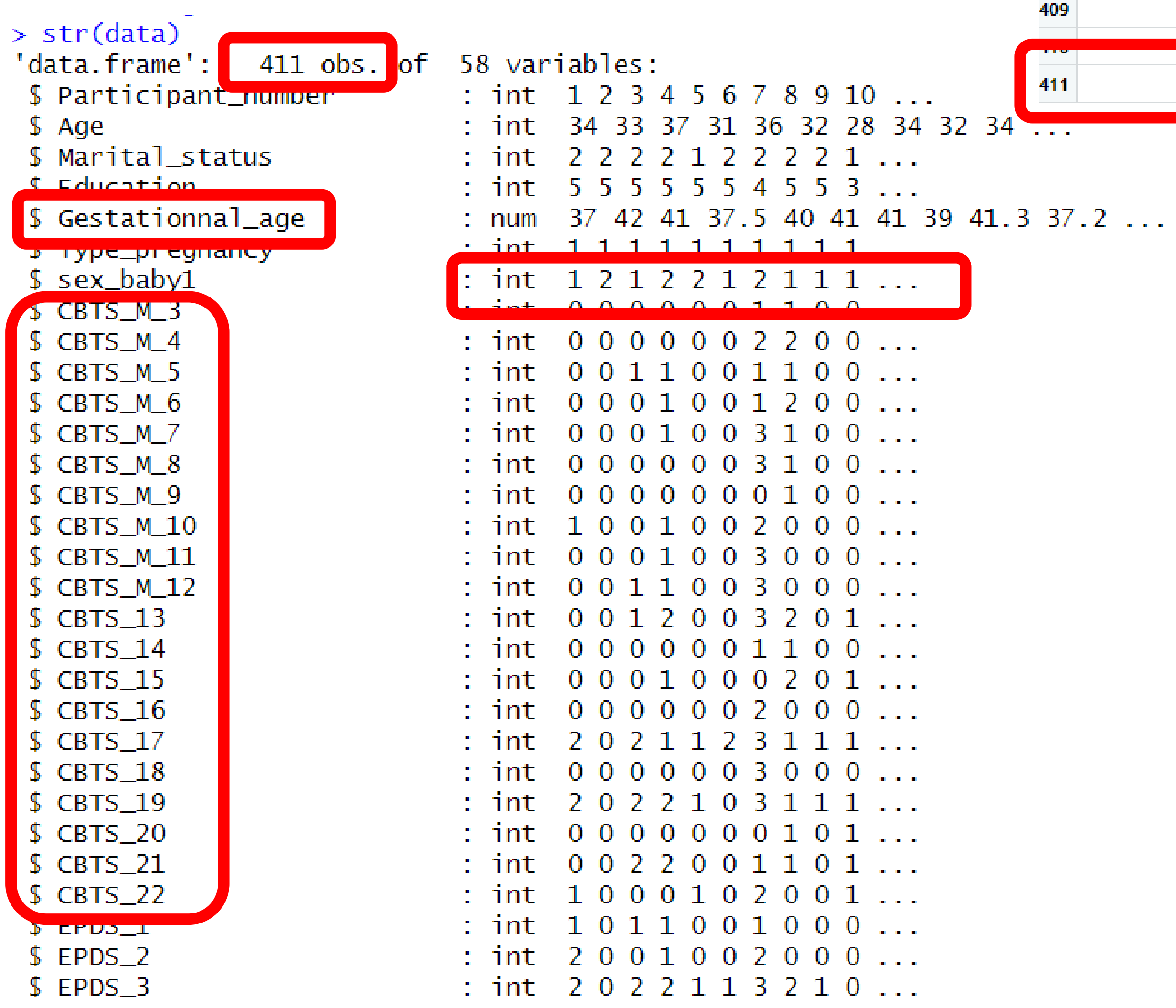
- Changing variable formats
- Renaming variables
- Removing variables or observations (e.g. removal of consent)
- Adding new variables or observations (e.g. composite variables)
- Fixing incorrect entries (e.g. typos)
- Removing outliers

GARBAGE IN = GARBAGE OUT

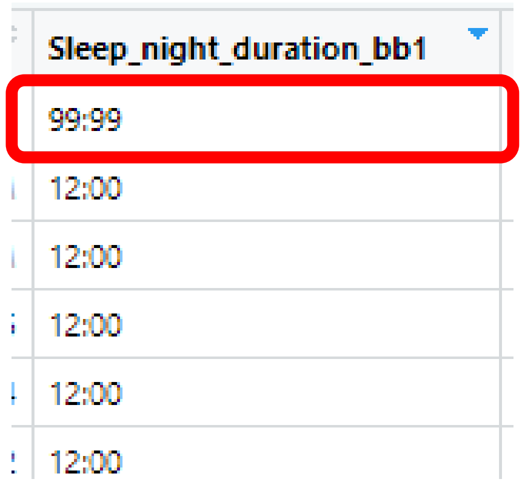
To clean or not to clean?



	Participant_number	Age	Marital_status	Education	Gestationnal_age	Type_
404	404	25	2	3	39.0	
405	405	28	2	2	41.0	
406	406	31	2	3	39.5	
407	407	26	2	2	37.0	
408	408	26	2	5	39.0	
409	409	27	2	5	41.2	
[illegible]	[illegible]	[illegible]	2	3	[illegible]	
411	NA	NA	NA	NA	NA	

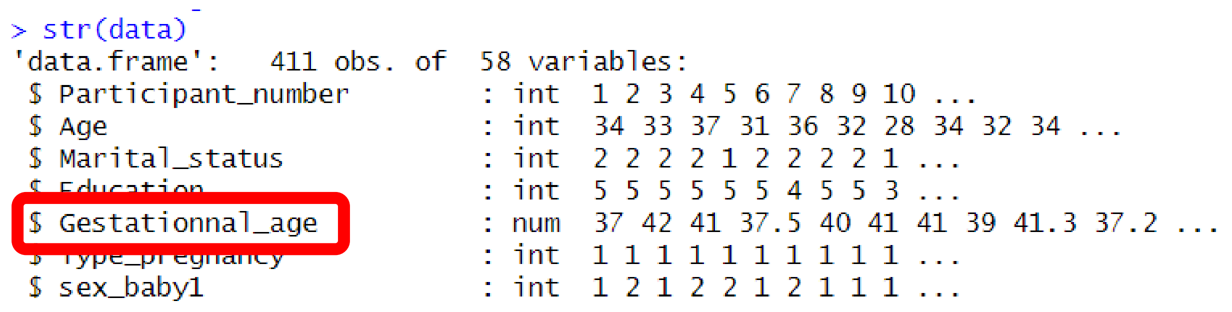


```
> str(data)
'data.frame':   411 obs. of  58 variables:
 $ Participant_number   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Age                  : int  34 33 37 31 36 32 28 34 32 34 ...
 $ Marital_status       : int  2 2 2 2 1 2 2 2 2 1 ...
 $ Education            : int  5 5 5 5 5 5 4 5 5 3 ...
 $ Gestationnal_age     : num  37 42 41 37.5 40 41 41 39 41.3 37.2 ...
 $ Type_pregnancy       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ sex_baby1            : int  1 2 1 2 2 1 2 1 1 1 ...
 $ CBTS_M_3             : int  0 0 0 0 0 0 1 1 0 0 ...
 $ CBTS_M_4             : int  0 0 0 0 0 0 2 2 0 0 ...
 $ CBTS_M_5             : int  0 0 1 1 0 0 1 1 0 0 ...
 $ CBTS_M_6             : int  0 0 0 1 0 0 1 2 0 0 ...
 $ CBTS_M_7             : int  0 0 0 1 0 0 3 1 0 0 ...
 $ CBTS_M_8             : int  0 0 0 0 0 0 3 1 0 0 ...
 $ CBTS_M_9             : int  0 0 0 0 0 0 0 1 0 0 ...
 $ CBTS_M_10            : int  1 0 0 1 0 0 2 0 0 0 ...
 $ CBTS_M_11            : int  0 0 0 1 0 0 3 0 0 0 ...
 $ CBTS_M_12            : int  0 0 1 1 0 0 3 0 0 0 ...
 $ CBTS_13              : int  0 0 1 2 0 0 3 2 0 1 ...
 $ CBTS_14              : int  0 0 0 0 0 0 1 1 0 0 ...
 $ CBTS_15              : int  0 0 0 1 0 0 0 2 0 1 ...
 $ CBTS_16              : int  0 0 0 0 0 0 2 0 0 0 ...
 $ CBTS_17              : int  2 0 2 1 1 2 3 1 1 1 ...
 $ CBTS_18              : int  0 0 0 0 0 0 3 0 0 0 ...
 $ CBTS_19              : int  2 0 2 2 1 0 3 1 1 1 ...
 $ CBTS_20              : int  0 0 0 0 0 0 0 1 0 1 ...
 $ CBTS_21              : int  0 0 2 2 0 0 1 1 0 1 ...
 $ CBTS_22              : int  1 0 0 0 1 0 2 0 0 1 ...
 $ EPDS_1               : int  1 0 1 1 0 0 1 0 0 0 ...
 $ EPDS_2               : int  2 0 0 1 0 0 2 0 0 0 ...
 $ EPDS_3               : int  2 0 2 2 1 1 3 2 1 0 ...
```



Sleep_night_duration_bb1
99:99
12:00
12:00
12:00
12:00
12:00

Renaming variables



```
> str(data)
'data.frame':   411 obs. of  58 variables:
 $ Participant_number       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Age                      : int  34 33 37 31 36 32 28 34 32 34 ...
 $ Marital_status           : int  2 2 2 2 1 2 2 2 2 1 ...
 $ Education                : int  5 5 5 5 5 5 4 5 5 3 ...
 $ Gestationnal_age         : num  37 42 41 37.5 40 41 41 39 41.3 37.2 ...
 $ Type_pregnancy           : int  1 1 1 1 1 1 1 1 1 1 ...
 $ sex_baby1                : int  1 2 1 2 2 1 2 1 1 1 ...
```

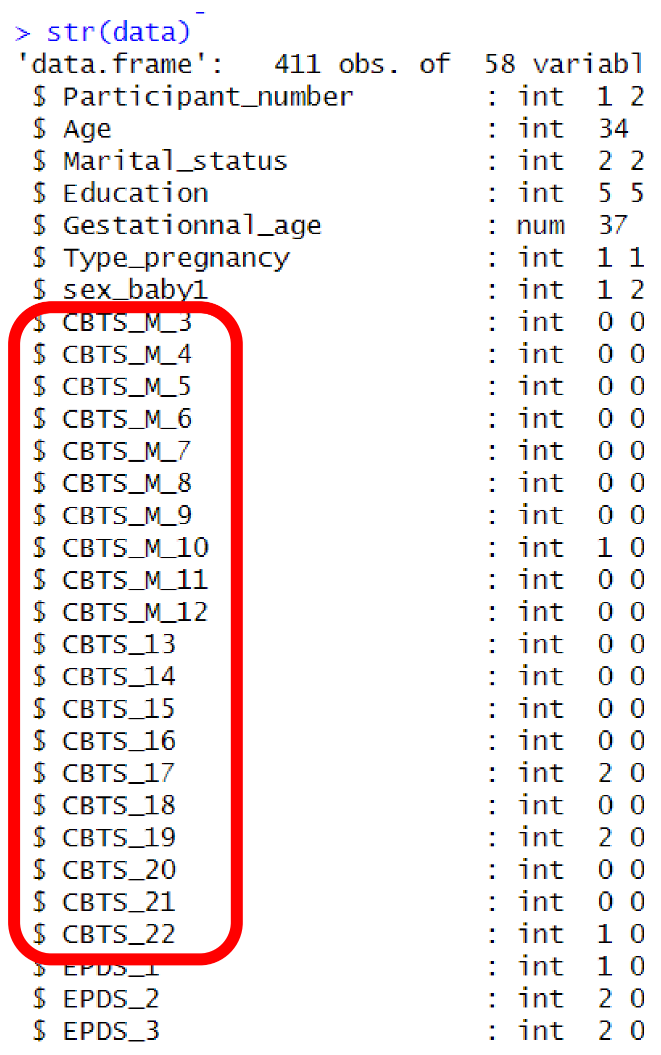
```
> colnames(data)
[1] "Participant_number" "Age" "Marital_status" "Education" "Gestationnal_age" [6]
"Type_pregnancy" "sex_baby1" "CBTS_M_3" "CBTS_M_4" "CBTS_M_5"
[11] "CBTS_M_6" "CBTS_M_7" "CBTS_M_8" "CBTS_M_9" "CBTS_M_10"
...

> which(colnames(data) == 'Gestationnal_age')
[1] 5

> colnames(data)[which(colnames(data) == 'Gestationnal_age')] <- 'Gestational_age'

> colnames(data)
[1] "Participant_number" "Age" "Marital_status" "Education" "Gestational_age"
[6] "Type_pregnancy" "sex_baby1" "CBTS_M_3" "CBTS_M_4" "CBTS_M_5"
[11] "CBTS_M_6" "CBTS_M_7" "CBTS_M_8" "CBTS_M_9" "CBTS_M_10"
...
```

Creating new variables

As well as being used to select an existing column, **$** can also be used to make a new one!

To do this we use both $ and the assignment command <-

```
> data$Age_bb_months
NULL

> data$Age_bb_months <- data$Age_bb*12

> data$Age_bb_months
[1] 12 36 12 36 36 12 36 36 12 36 36 24 24 …
```



```
> str(data)
'data.frame':	411 obs. of  58 variabl
 $ Participant_number      : int  1 2
 $ Age                     : int  34
 $ Marital_status          : int  2 2
 $ Education               : int  5 5
 $ Gestationnal_age        : num  37
 $ Type_pregnancy          : int  1 1
 $ sex_baby1               : int  1 2
 $ CBTS_M_3                : int  0 0
 $ CBTS_M_4                : int  0 0
 $ CBTS_M_5                : int  0 0
 $ CBTS_M_6                : int  0 0
 $ CBTS_M_7                : int  0 0
 $ CBTS_M_8                : int  0 0
 $ CBTS_M_9                : int  0 0
 $ CBTS_M_10               : int  1 0
 $ CBTS_M_11               : int  0 0
 $ CBTS_M_12               : int  0 0
 $ CBTS_13                 : int  0 0
 $ CBTS_14                 : int  0 0
 $ CBTS_15                 : int  0 0
 $ CBTS_16                 : int  0 0
 $ CBTS_17                 : int  2 0
 $ CBTS_18                 : int  0 0
 $ CBTS_19                 : int  2 0
 $ CBTS_20                 : int  0 0
 $ CBTS_21                 : int  0 0
 $ CBTS_22                 : int  1 0
 $ EPDS_1                  : int  1 0
 $ EPDS_2                  : int  2 0
 $ EPDS_3                  : int  2 0
```

Creating new variables

ACTIVITY 4

Try creating a new variable called EPDSsum, which is the sum of all the EPDS scores

Creating new variables – Excel analogy

	1	2	…	10	11
A			…		SUM(A1:A10)
B			…		SUM(B1:B10)
C			…		SUM(C1:C10)
D			…		SUM(D1:D10)

```
EPDS_data <- data[,28:37] # Create a subset containing the EPDS items

rowSums(EPDS_data) # Sum across the EPDS items for each row

rowSums(data[,28:37]) # Short cut!

data$EPDSsum # Compare with the manual way to reassure yourselves!
```

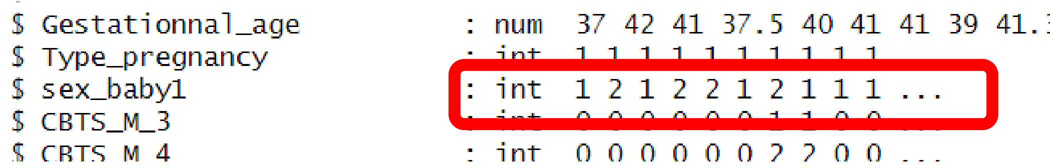
Creating new variables

ACTIVITY 5

Use rowSums to create HADSsum and CBTSsum, which sum up the scores for the HADS and CB-PTSD components (EPDS example is below).

```
EPDS_data <- data[,28:37] # Create a subset containing the EPDS items

rowSums(EPDS_data) # Sum across the EPDS items for each row

rowSums(data[,28:37]) # Short cut!

data$EPDSsum # Compare with the manual way to reassure yourselves!
```

Modifying existing values



```
$ Gestationnal_age        : num  37 42 41 37.5 40 41 41 39 41.
$ Type_pregnancy          : int  1 1 1 1 1 1 1 1 1 1
$ sex_baby1               : int  1 2 1 2 2 1 2 1 1 1 ...
$ CBTS_M_3                : int  0 0 0 0 0 0 1 1 0 0 ...
$ CBTS_M_4                : int  0 0 0 0 0 0 2 2 0 0 ...
```

If… then…

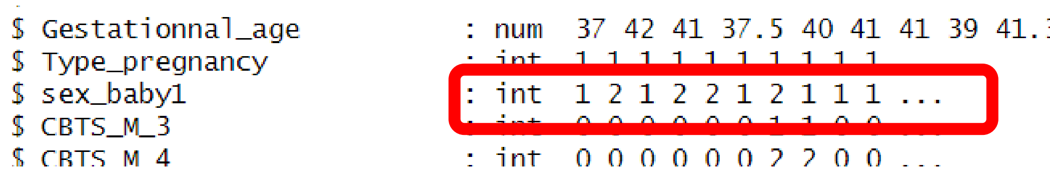
If… else…

```
ifelse(condition, action if true, action if not true)
```

If the value of sex_baby1 is 1, then change the value to "female".

ELSE (if the value is NOT 1), change the value to "male".

Modifying existing values



```
$ Gestationnal_age       : num  37 42 41 37.5 40 41 41 39 41.
$ Type_pregnancy         : int  1 1 1 1 1 1 1 1 1 1
$ sex_baby1              : int  1 2 1 2 2 1 2 1 1 1 ...
$ CBTS_M_3               : int  0 0 0 0 0 0 1 1 0 0 ...
$ CBTS_M_4               : int  0 0 0 0 0 0 2 2 0 0 ...
```

```
> data$sex_baby1
[1] 1 2 1 2 2 1 2 1 1 1 2 2 2 2 1 1 1 2 1 1 1 1 1 2 1 2 1 1 2 2 1...
Levels: 1 2

> ifelse(data$sex_baby1=="1", "female", "male")
[1] "female" "male" "female" "male" "male" "female" "male" "female" "female"
"female" "male" "male" "male" "male" "female" "female" "female"...

> data$sex_baby1 <- ifelse(data$sex_baby1=="1", "female", "male") # Reassign

> data$sex_baby1 # Check it's worked
[1] "female" "male" "female" "male" "male" "female" "male" "female" "female"
"female" "male" "male" "male" "male" "female" "female" "female"...

> data$sex_baby1 <- as.factor(data$sex_baby1)
```

Modifying existing values

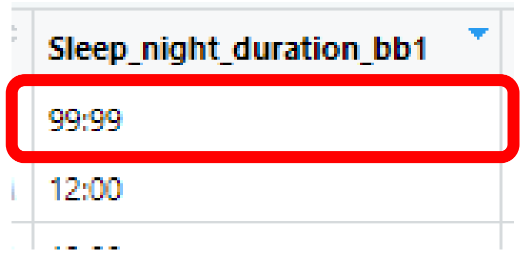
ACTIVITY 6

The HADS score can be categorised as following: 0–7 (Normal), 8–10 (Mild), 11–15 (Moderate), 16–21 (Severe).

Create a new variable which assigns whether participants have anxiety or not based on their score

TIP: your conditional statement will include the < or > operator

Coding data as missing



Sleep_night_duration_bb1
99:99
12:00

R presents missing values as ‘NA’. This is a logical value, NOT character.

```
> NA
[1] NA
```

```
> NA + 3
[1] NA
```

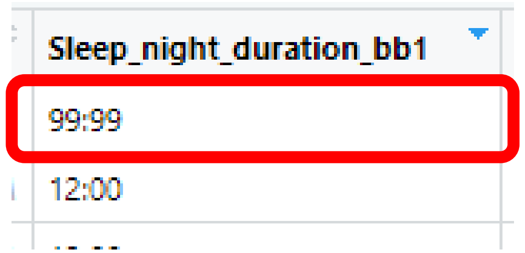
```
> is.na(NA)
[1] TRUE
```

You may have missing data which is in character or numerical format…

- Purposeful representation of missing (e.g. age = 999 years)
- Data you wish to make missing to remove it from analysis (e.g. outliers)

It’s important your missing data is in a format R recognises, otherwise it will consider it a genuine observation and include it in all analyses

Coding data as missing



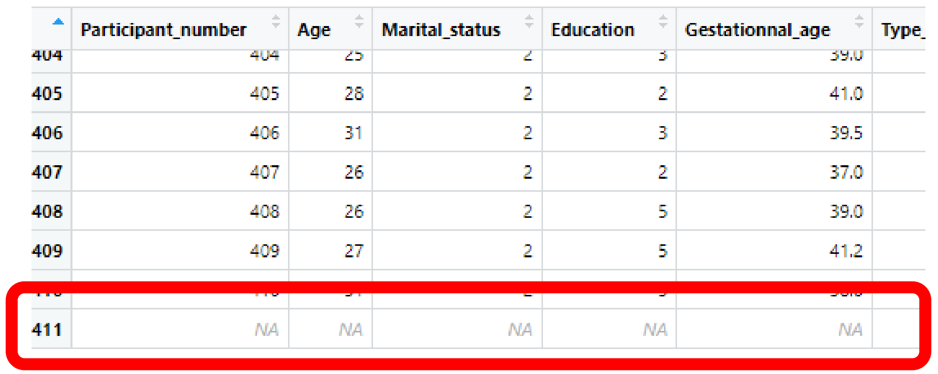
Sleep_night_duration_bb1

99:99

12:00

```
> which(data$Sleep_night_duration_bb1=="99:99")
[1] 180

> data[which(data$Sleep_night_duration_bb1=="99:99"),]$Sleep_night_duration_bb1 <- NA

> is.na(data$Sleep_night_duration_bb1)
```

Removing missing data



	Participant_number	Age	Marital_status	Education	Gestationnal_age	Type_
404	404	25	2	3	39.0	
405	405	28	2	2	41.0	
406	406	31	2	3	39.5	
407	407	26	2	2	37.0	
408	408	26	2	5	39.0	
409	409	27	2	5	41.2	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
411	NA	NA	NA	NA	NA	

If data are recorded as missing in the format that R recognises (as they are here), we can make use of R's built-in functions to remove these observations easily!

```
> data <- data[-which(is.na(data$Participant_number)),]

> View(data)
```

Other operations exist to remove rows with just some missing data.

Beware of informative missingness!

Post-cleaning processes

1. CHECK!!!!! → str, View (if small amounts of data), sense check commands (e.g. if you are not expecting any people under 18)

2. Save → `write.csv(data, file = "data_cleaned.csv", row.names=FALSE)`

↑

Only need to put the name of the csv file, as R is already going from the working directory

Tidyverse

- Collection of R packages designed for data science tasks
- Very popular!!!
- dplyr and tidyr in particular are brilliant for data manipulation and cleaning
- To appreciate these and understand how they work, you need to begin with base R

Lesson 2: Summarising data

Summarising data

```r
summary(data)
```

- Character variables: length, class
- Factor variables: counts
- Numeric variables: mean, median, quartiles, range

NAs represented separately → again, one of the benefits of coding missing data in a format R recognises

Categorical variables: tables

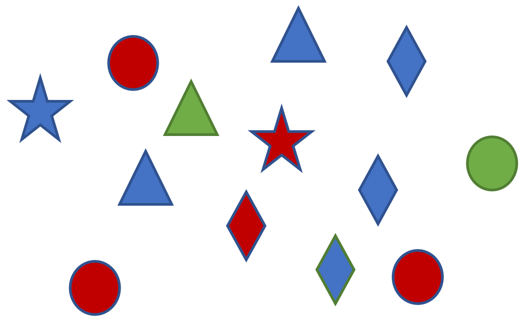
The `table()` function is used to provide a frequency table of one or two variables.

```
table(data$colour)
```

Red	Blue	Green
5	6	2

```
table(data$shape, data$colour)
```

	Red	Blue	Green
Star	1	1	0
Circle	3	0	1
Triangle	0	2	1
Diamond	0	3	0



This is the start of an **associative analysis.**

Categorical variables: tables

```
table(data$sex_baby1)

table(data$Education)

table(data$Education, data$Marital_status)

table(data$Sleep_night_duration_bb1)
```

ACTIVITY 7

Try these commands. How interpretable is your output? How are missing data summarised?

Tables of categorical variables

Often we're interested in relative numbers, not absolute...

`prop.table()` allows us to do this:

```
twowaytable <- table(data$Education, data$Marital_status)

prop.table(twowaytable)

prop.table(twowaytable, margin=1) # Proportions row-wise (margin 1)

prop.table(twowaytable, margin=2) # Proportions column-wise (margin 2)
```

Continuous (numeric) variables: distribution

The following functions give us summary statistics for continuous variables:

- `mean(data$var)`
- `sd(data$var)` for the standard deviation
- `min(data$var)` and `max(data$var)`
- `median(data$var)`
- `quantile(data$var, p=0.25)` for Q1
- `quantile(data$var, p=0.75)` for Q3
- `IQR()` for the interquartile range

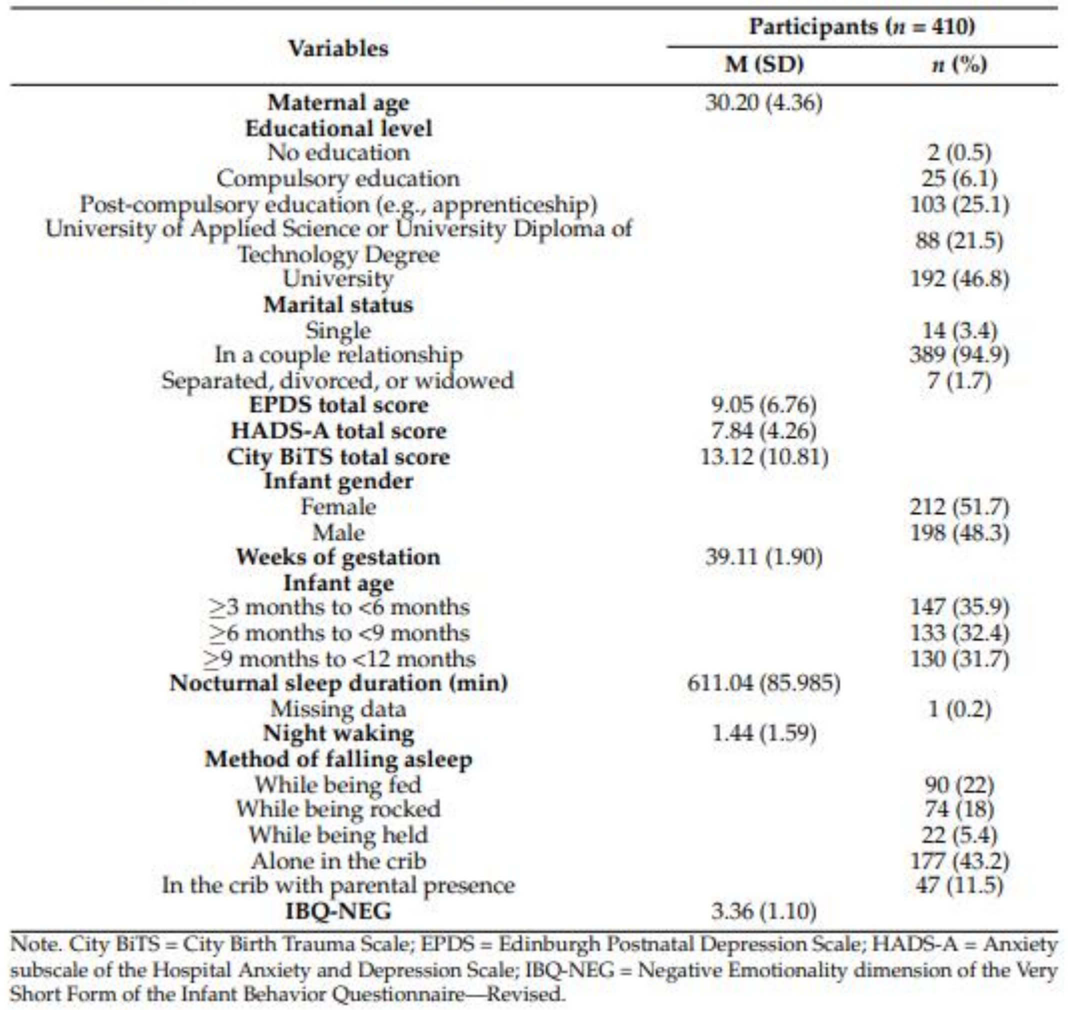
Continuous (numeric) variables: distribution

ACTIVITY 8

1. What happens if you try to take the average of a categorical variable?
2. What is the mean age and standard error of the mean? (HINT: the sqrt() function returns the square root of a number, e.g. sqrt(9)=3)
3. How many participants are less than 30 years old?

Table1 package

Table 1. Descriptive Characteristics of the Sample.



Variables	Participants (*n* = 410)	
	M (SD)	*n* (%)
Maternal age	30.20 (4.36)	
Educational level		
No education		2 (0.5)
Compulsory education		25 (6.1)
Post-compulsory education (e.g., apprenticeship)		103 (25.1)
University of Applied Science or University Diploma of Technology Degree		88 (21.5)
University		192 (46.8)
Marital status		
Single		14 (3.4)
In a couple relationship		389 (94.9)
Separated, divorced, or widowed		7 (1.7)
EPDS total score	9.05 (6.76)	
HADS-A total score	7.84 (4.26)	
City BiTS total score	13.12 (10.81)	
Infant gender		
Female		212 (51.7)
Male		198 (48.3)
Weeks of gestation	39.11 (1.90)	
Infant age		
≥3 months to <6 months		147 (35.9)
≥6 months to <9 months		133 (32.4)
≥9 months to <12 months		130 (31.7)
Nocturnal sleep duration (min)	611.04 (85.985)	
Missing data		1 (0.2)
Night waking	1.44 (1.59)	
Method of falling asleep		
While being fed		90 (22)
While being rocked		74 (18)
While being held		22 (5.4)
Alone in the crib		177 (43.2)
In the crib with parental presence		47 (11.5)
IBQ-NEG	3.36 (1.10)	

Note. City BiTS = City Birth Trauma Scale; EPDS = Edinburgh Postnatal Depression Scale; HADS-A = Anxiety subscale of the Hospital Anxiety and Depression Scale; IBQ-NEG = Negative Emotionality dimension of the Very Short Form of the Infant Behavior Questionnaire—Revised.

Table1 package

```
install.packages("table1")
library(table1)


# Create a table
table1(~ var1 + var2 + … + varX, data=data)

# E.g.
table1(~ Age + Marital_status + Education +
 night_awakening_number_bb1 +
 EPDSsum + HADSsum + CBTSsum, data=data)
```

ACTIVITY 9

Pick some variables you would like to summarise within your dataset, and try inputting these into the table1 function. Check out the output!

If you're not happy with the way something looks, feel free to clean your data further – e.g. re-naming variables, changing the format, grouping continuous variables, etc. (NB save your changes!)

Check your output against the publication. Do you get a match?

Break – return in 15 minutes

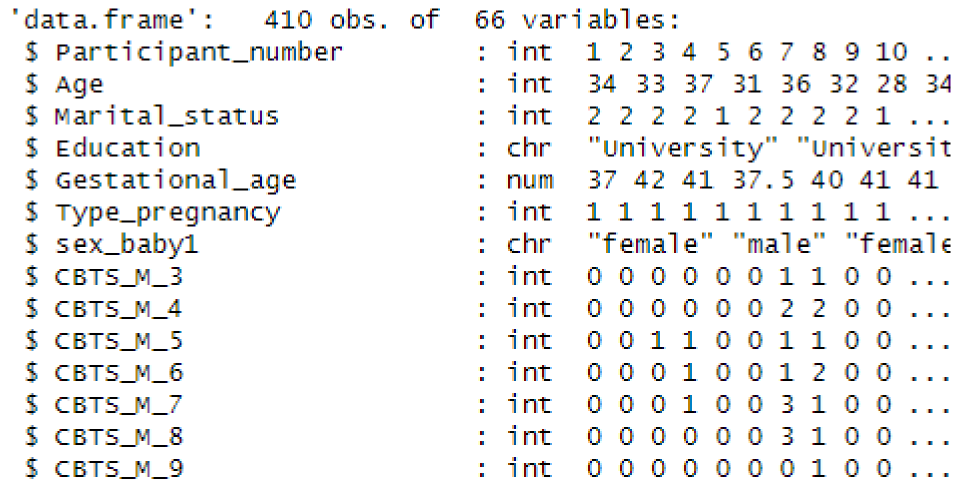
SAVE YOUR DATA!

Lesson 3: Visualising data

Load in your data…

read.csv()

Whenever loading data in. always check the format using str(). Is everything as you would expect?

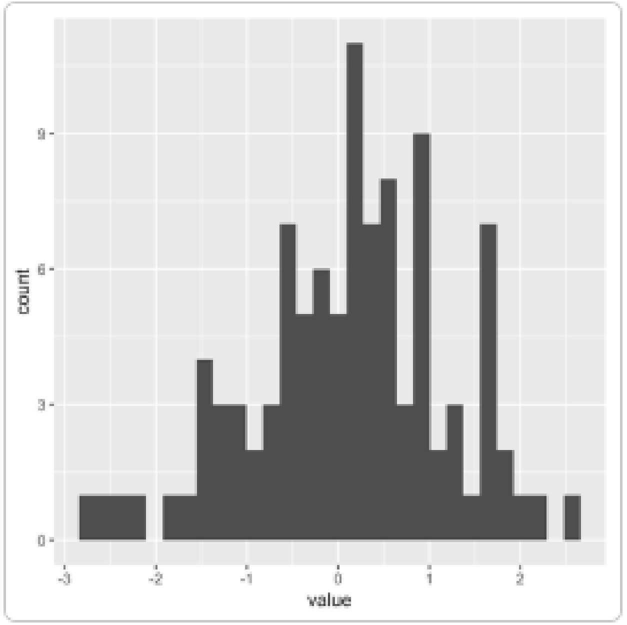


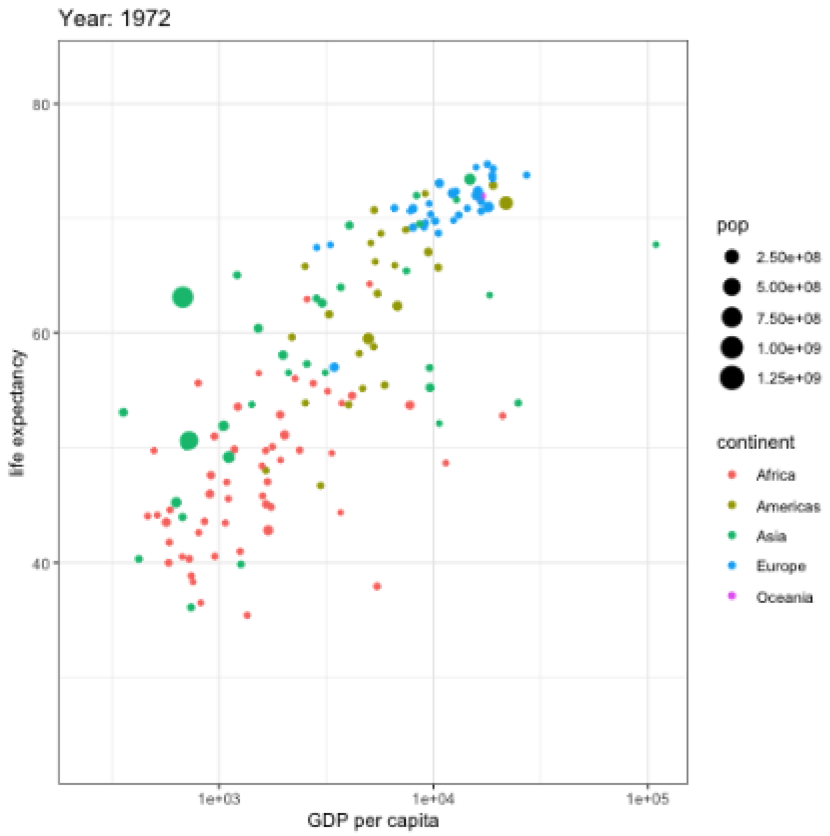
```
'data.frame':   410 obs. of  66 variables:
 $ Participant_number       : int  1 2 3 4 5 6 7 8 9 10 ..
 $ Age                      : int  34 33 37 31 36 32 28 34
 $ Marital_status           : int  2 2 2 2 1 2 2 2 2 1 ...
 $ Education                : chr  "University" "Universit
 $ Gestational_age          : num  37 42 41 37.5 40 41 41
 $ Type_pregnancy           : int  1 1 1 1 1 1 1 1 1 1 ...
 $ sex_baby1                : chr  "female" "male" "female
 $ CBTS_M_3                 : int  0 0 0 0 0 0 1 1 0 0 ...
 $ CBTS_M_4                 : int  0 0 0 0 0 0 2 2 0 0 ...
 $ CBTS_M_5                 : int  0 0 1 1 0 0 1 1 0 0 ...
 $ CBTS_M_6                 : int  0 0 0 1 0 0 1 2 0 0 ...
 $ CBTS_M_7                 : int  0 0 0 1 0 0 3 1 0 0 ...
 $ CBTS_M_8                 : int  0 0 0 0 0 0 3 1 0 0 ...
 $ CBTS_M_9                 : int  0 0 0 0 0 0 0 1 0 0 ...
```

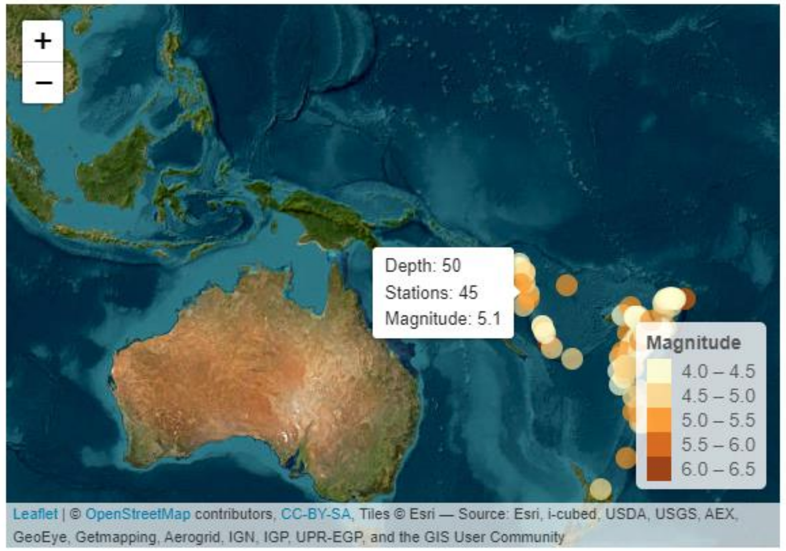
Graphs in R

1. Base R
2. Packages (e.g. ggplot2)

https://r-graph-gallery.com/





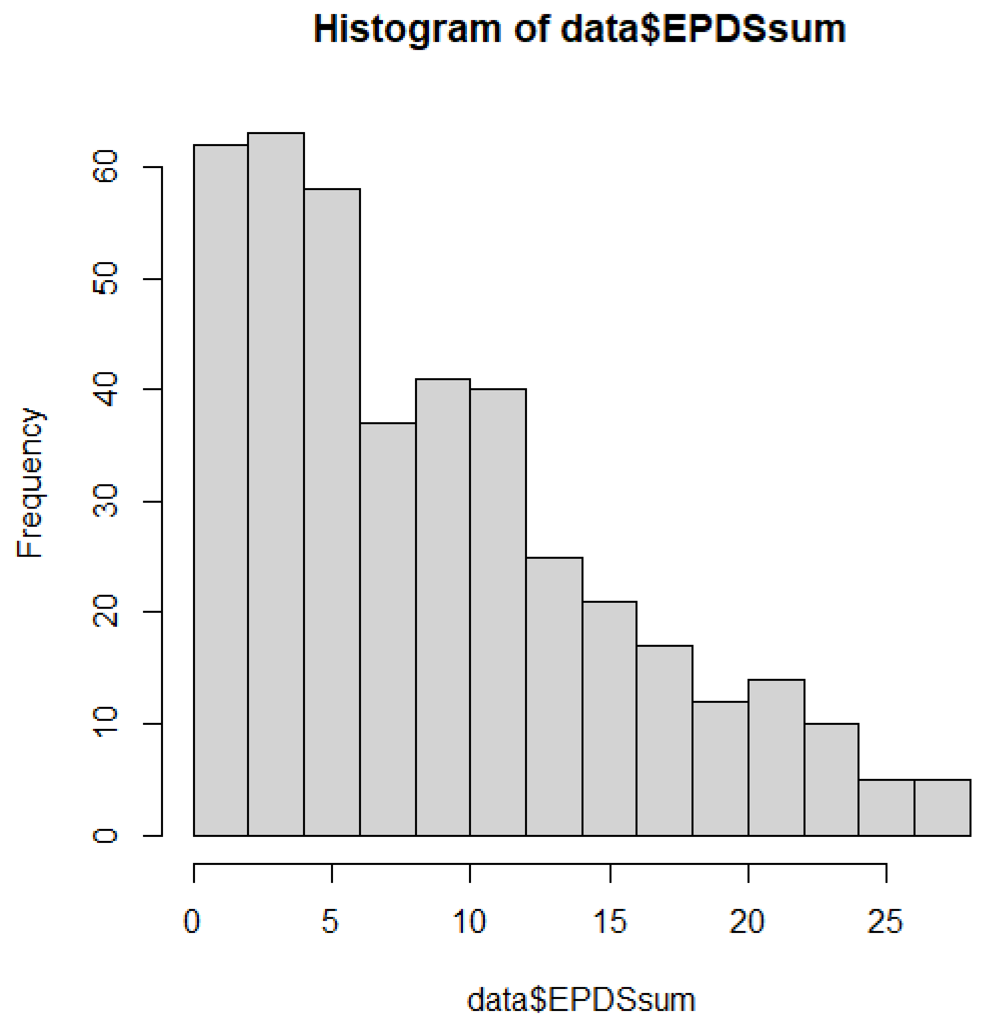


+
−
Depth: 50
Stations: 45
Magnitude: 5.1
Magnitude
4.0 – 4.5
4.5 – 5.0
5.0 – 5.5
5.5 – 6.0
6.0 – 6.5
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

function(data, optional visual parameters)

Histogram

```
hist(data$EPDSsum)
```

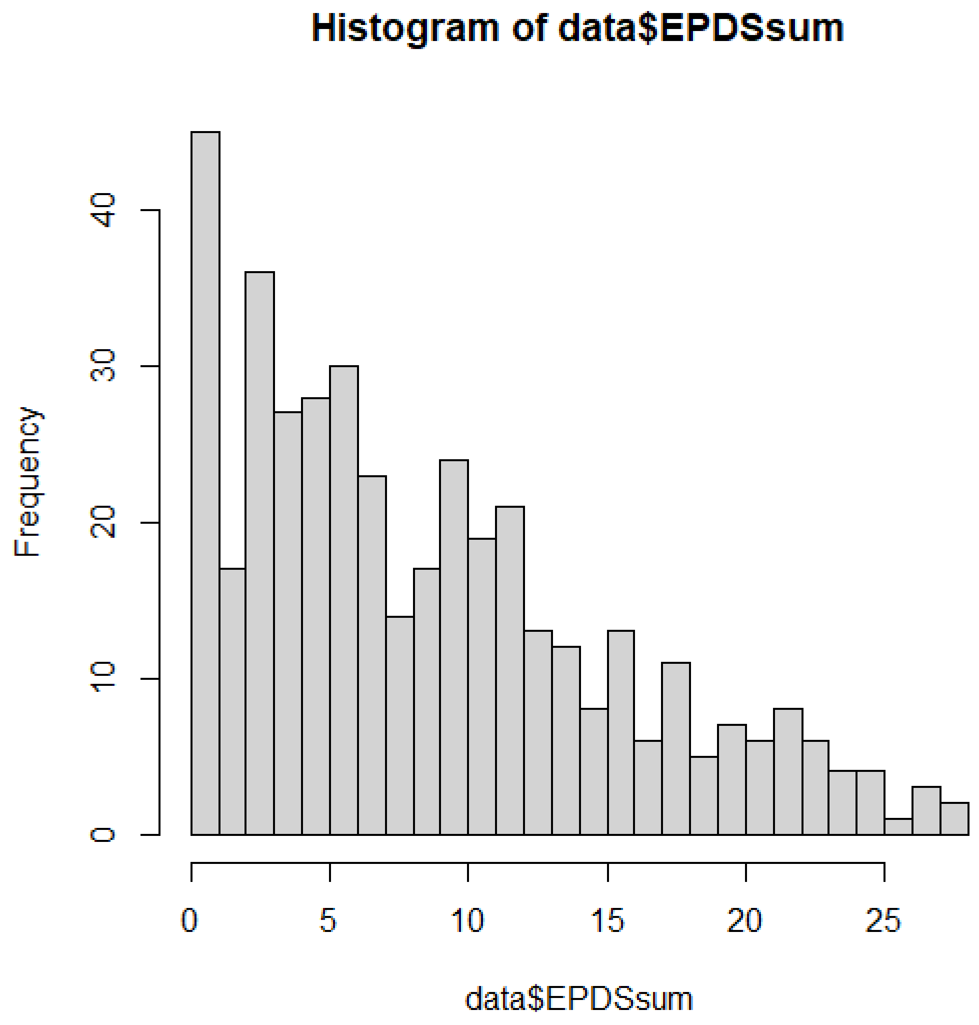


function(data, optional visual parameters)

Histogram

```
hist(data$EPDSsum)
```

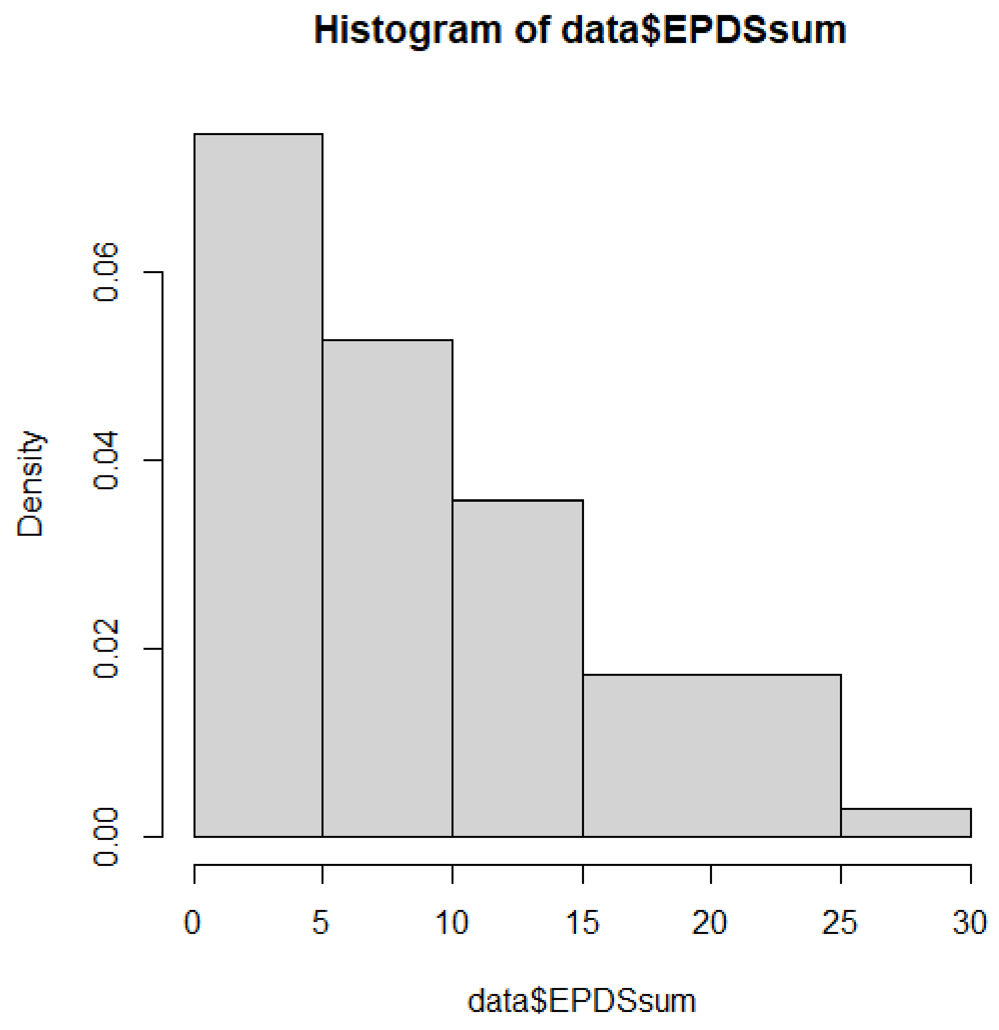
```
hist(data$EPDSsum, breaks=20)
```



function(data, optional visual parameters)

Histogram

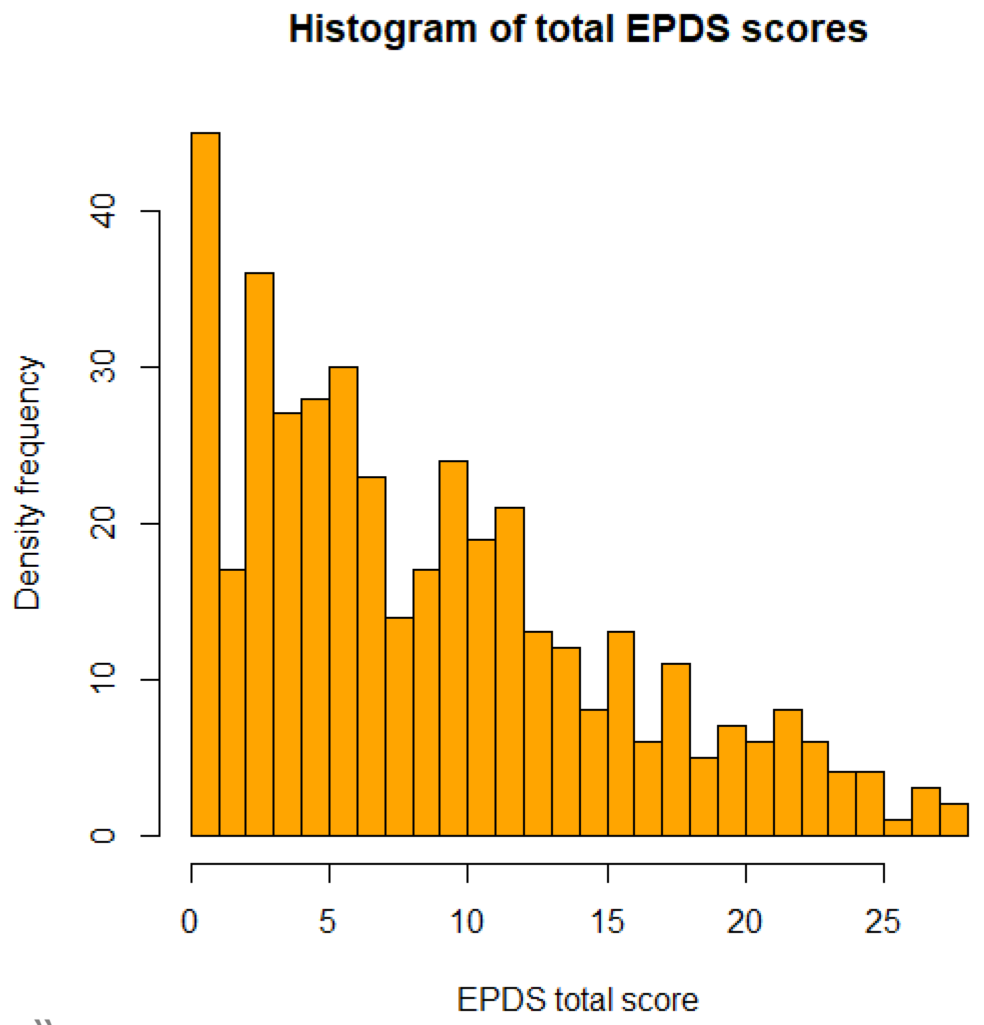
```
hist(data$EPDSsum)


hist(data$EPDSsum, breaks=20)


hist(data$EPDSsum,
        breaks=c(0, 5, 10, 15, 25, 30))
```



function(data, optional visual parameters)

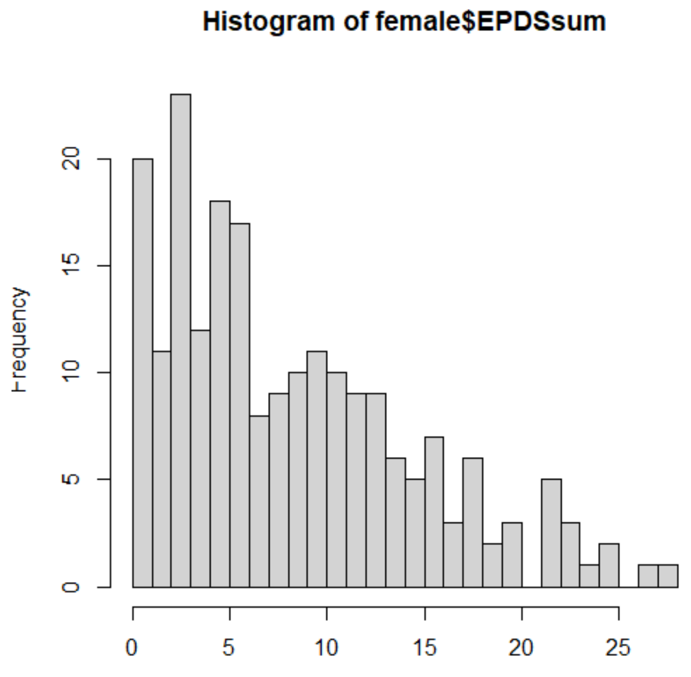
Histogram

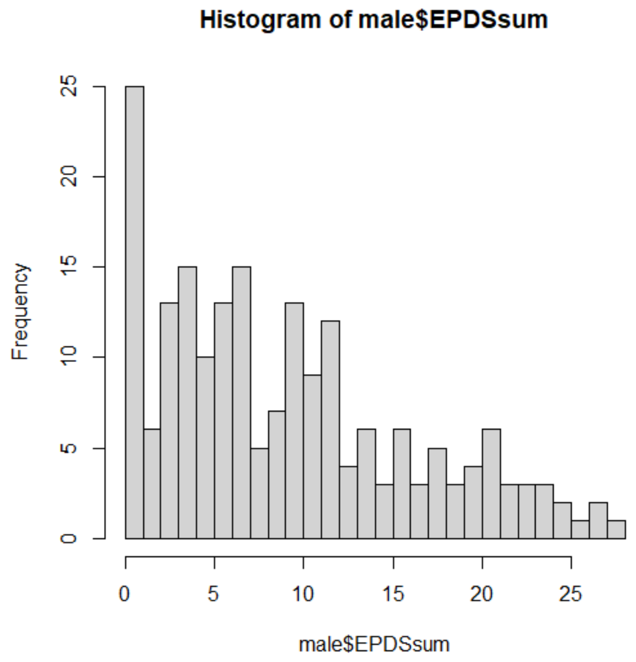
```
hist(data$EPDSsum)


hist(data$EPDSsum, breaks=20)


hist(data$EPDSsum,
         breaks=c(0, 5, 10, 15, 25, 30))


hist(data$EPDSsum, breaks=20,
         xlab = "EPDS total score",
         ylab = "Density frequency",
         main = "Histogram of total EPDS scores“,
         col="orange")
```



Histogram: two distributions

```
female <- data[which(data$sex_baby1=="female"),]
male <- data[which(data$sex_baby1=="male"),]

hist(female$EPDSsum, breaks=20)
hist(male$EPDSsum, breaks=20)
```



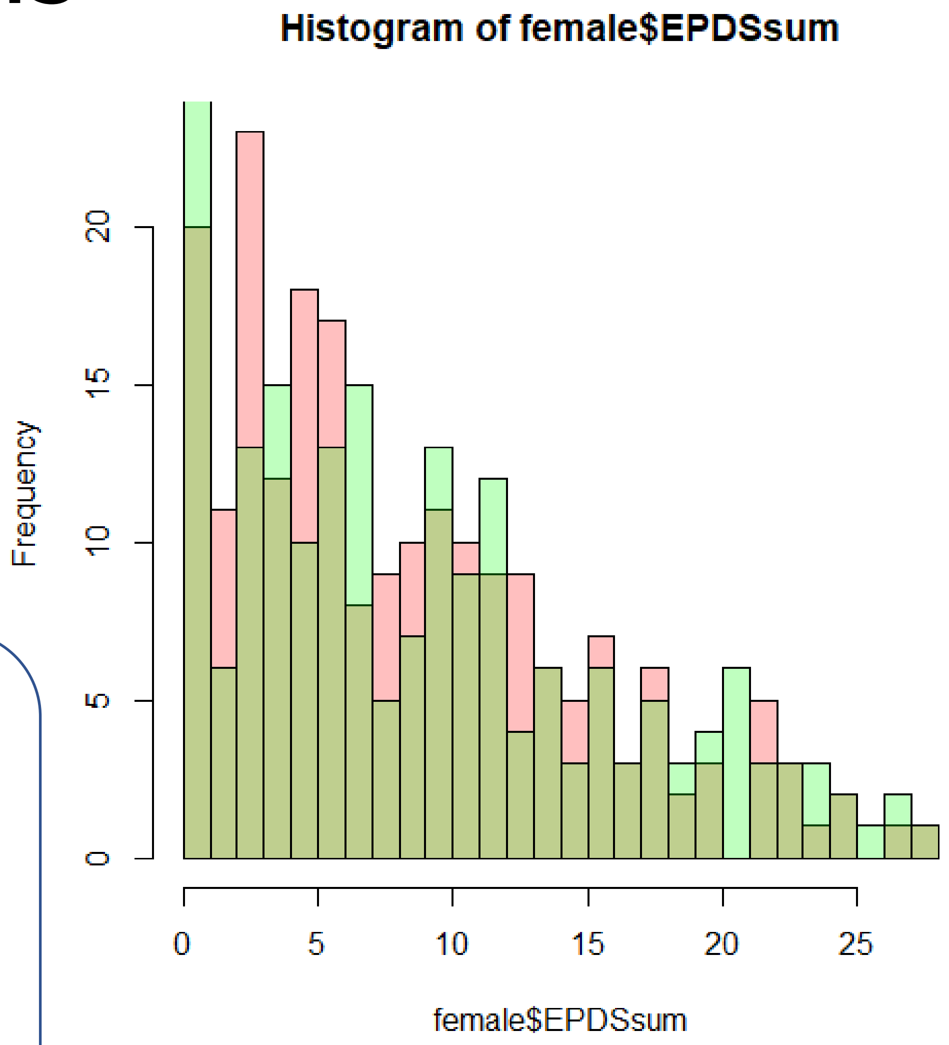


Histogram: two distributions

```
hist(female$EPDSsum, breaks=20,
      col=rgb(1,0,0,0.25))

 hist(male$EPDSsum, breaks=20,
      col=rgb(0,1,0,0.25), add=TRUE)
```

`rgb(X,Y,Z,T)` → the first three codes are RGB colour coordinates, and the last argument is a measure of transparency (needed when two graphs are being overlapped)

`add=TRUE` in the second command tells R to just add it to the pre-existing histogram



Histogram: two distributions

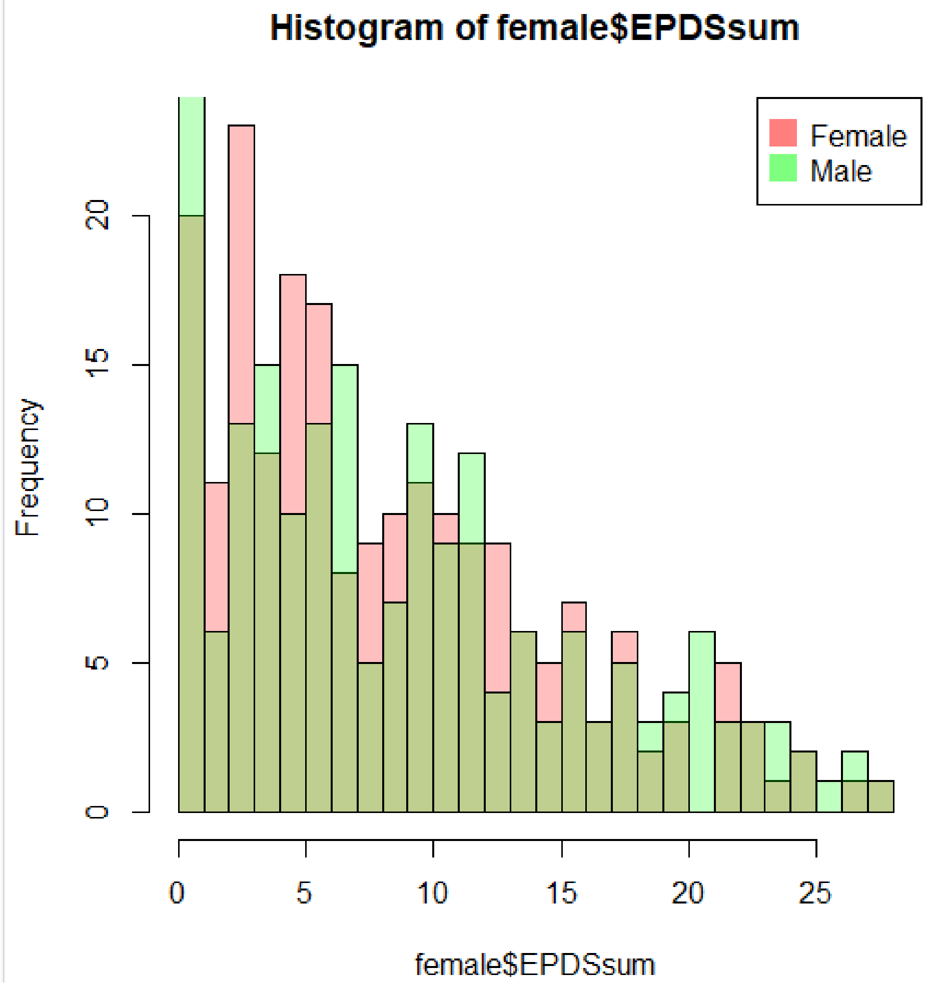
Can also add a legend:

```
legend("topright",
        legend=c("Female","Male"),
        col=c(rgb(1,0,0,0.5), rgb(0,1,0,0.5)),
        pt.cex=2, pch=15)
```

`legend` → labels

`col` → colours corresponding to labels (RGB & transparency)

`pt.cex` and **`pch`** → size and shape of the symbols (e.g. squares)



Histogram

ACTIVITY 10

1. Plot a histogram to inspect the HADS scores (sums) with 5, 20 and 50 breaks. Which do you think is the better choice and why?

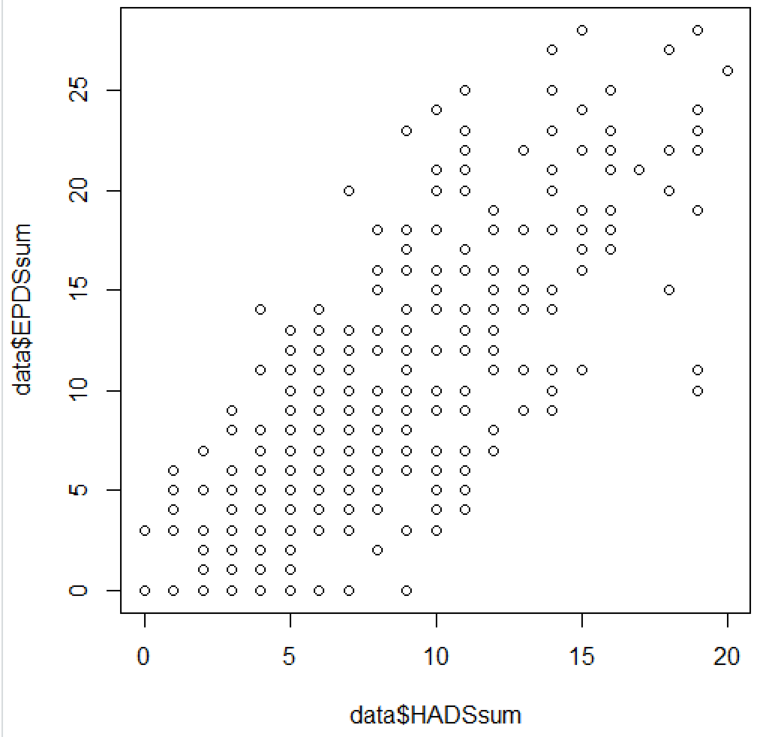
2. What is the median HADS score?

3. Using histograms, inspect the number of times an infant wakes up at night for mothers with a HADS score below vs above the median value

Scatter plots

- Histogram = distribution of one variable
  - `hist()`
- Scatter plot = distribution of up to two variables and their **association**
  - `plot()`

```
plot(x=data$HADSsum, y=data$EPDSsum)
```



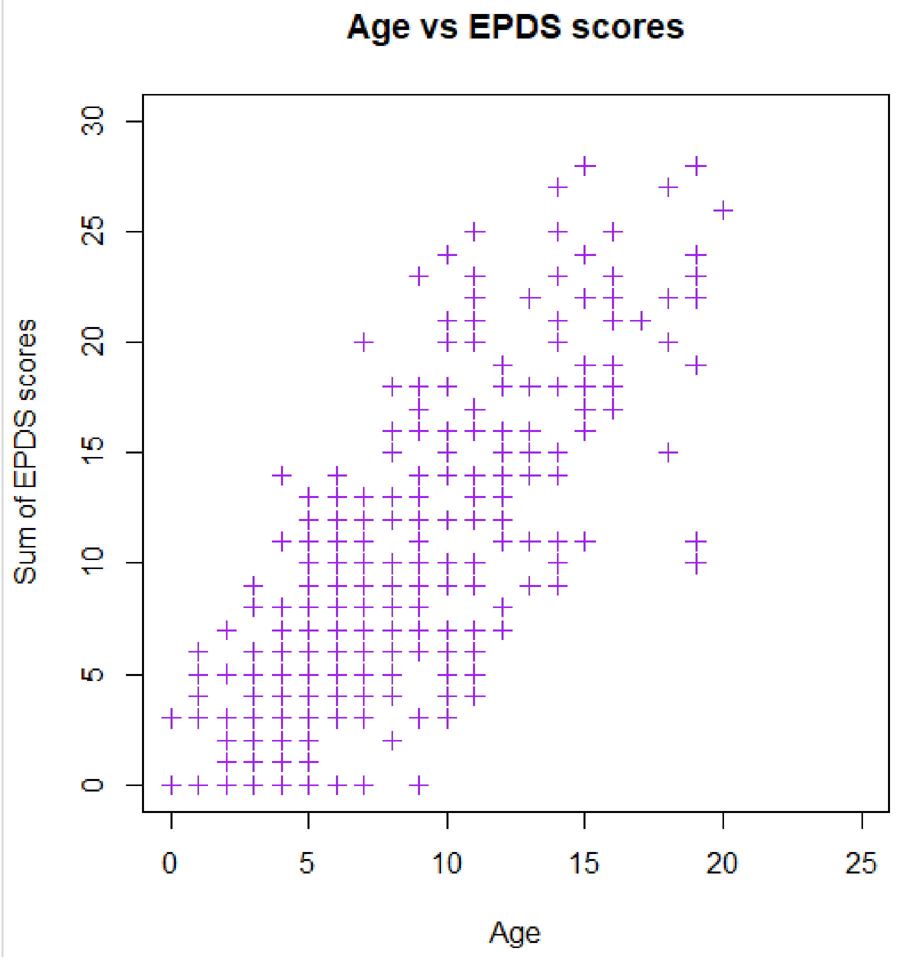
Scatter plots – changing display

```
plot(x=data$HADSsum, y=data$EPDSsum,
        xlim=c(0,25) , ylim=c(0,30),
        pch=3,
        cex=1, # larger number = larger shape
        col="purple",
        xlab="Age", ylab="Sum of EPDS scores",
        main="Age vs EPDS scores")
```

`xlim, ylim` → min and max values on axes

`pch` and `cex` → shape and size of the scatter points

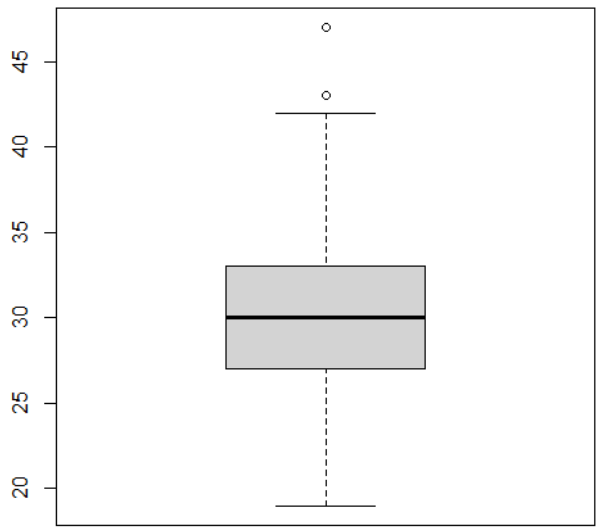
`col` → colour of shapes (R built-in colour, or RGB/hex codes)

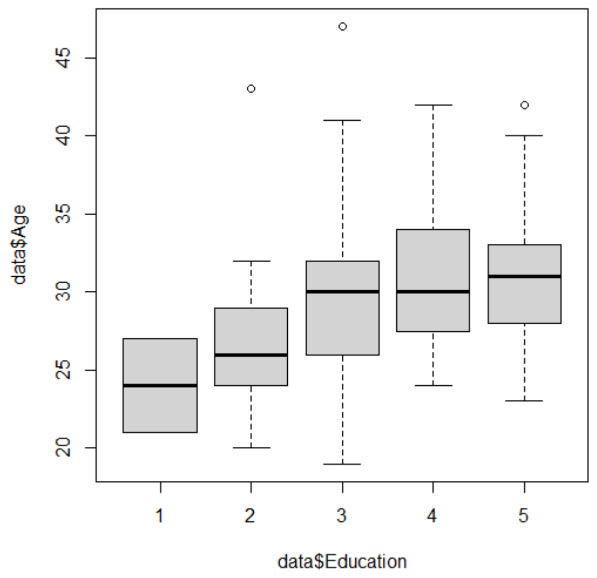


Other common plots

BOXPLOT

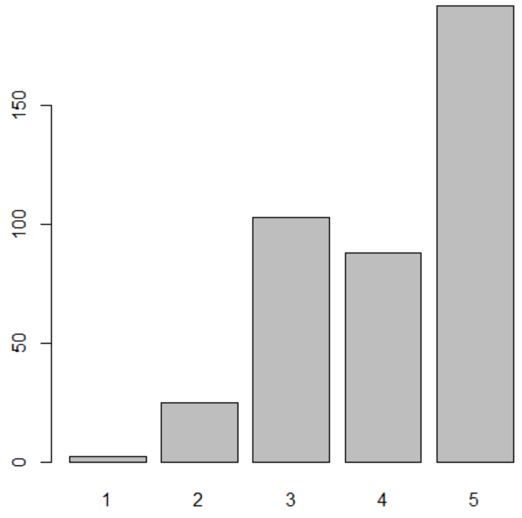
```
# Boxplot of participant age
boxplot(data$Age)

# Distribution by groups
boxplot(data$Age ~ data$Education)
```





BAR CHART

```
# Step 1: Use the table function to
obtain frequency values
table_plot <- table(data$Education)

# Step 2: feed this into the plotting
function barplot(table_plot)
```

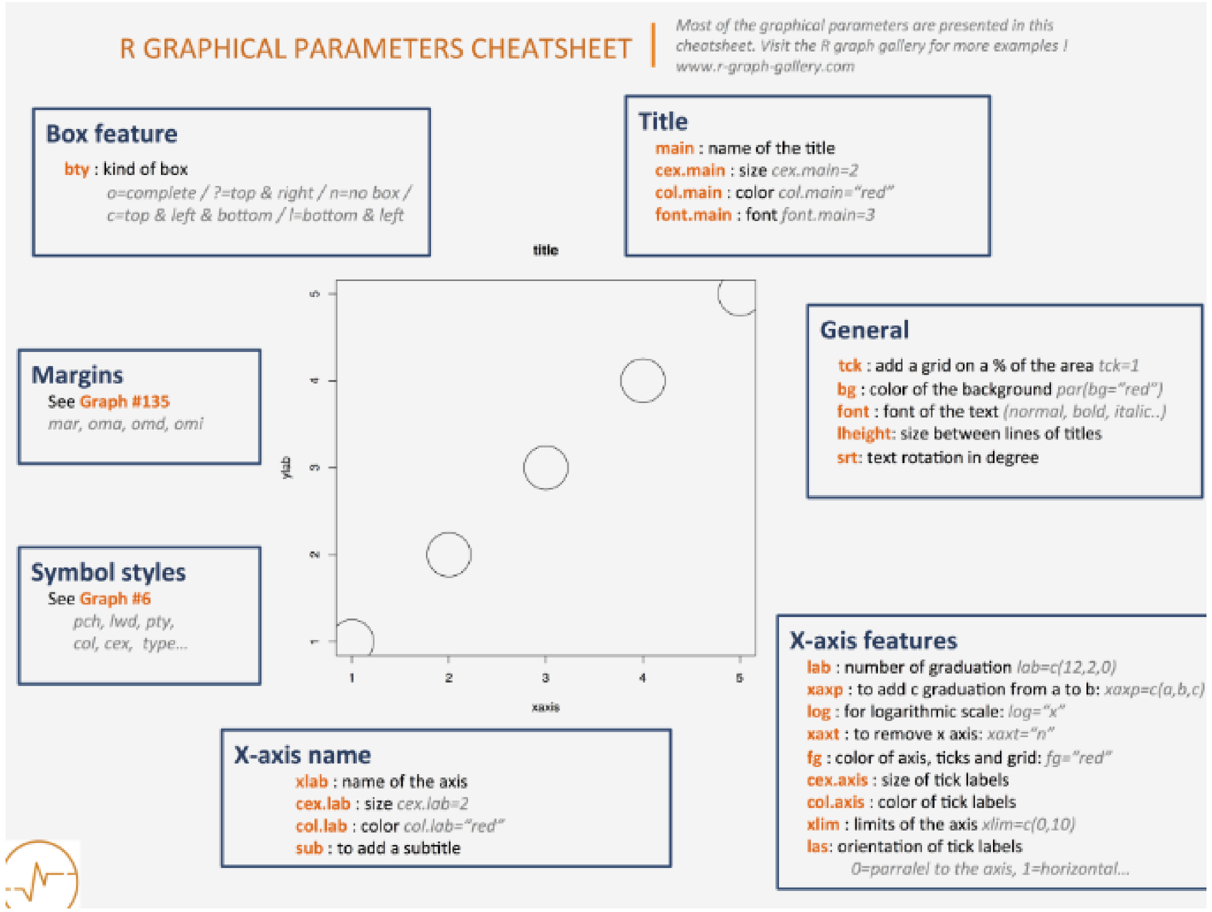


Same principles follow regarding customising colours, labels, titles, etc.

See RGraph Gallery!

Cheat sheets for Base R

https://r-graph-gallery.com/base-R.html



R GRAPHICAL PARAMETERS CHEATSHEET

Most of the graphical parameters are presented in this cheatsheet. Visit the R graph gallery for more examples ! www.r-graph-gallery.com

Box feature

bty : kind of box
o=complete / ?=top & right / n=no box / c=top & left & bottom / l=bottom & left

Title

main : name of the title
cex.main : size *cex.main=2*
col.main : color *col.main="red"*
font.main : font *font.main=3*

Margins

See Graph #135
mar, oma, omd, omi

General

tck : add a grid on a % of the area *tck=1*
bg : color of the background *par(bg="red")*
font : font of the text *(normal, bold, italic..)*
lheight: size between lines of titles
srt: text rotation in degree

Symbol styles

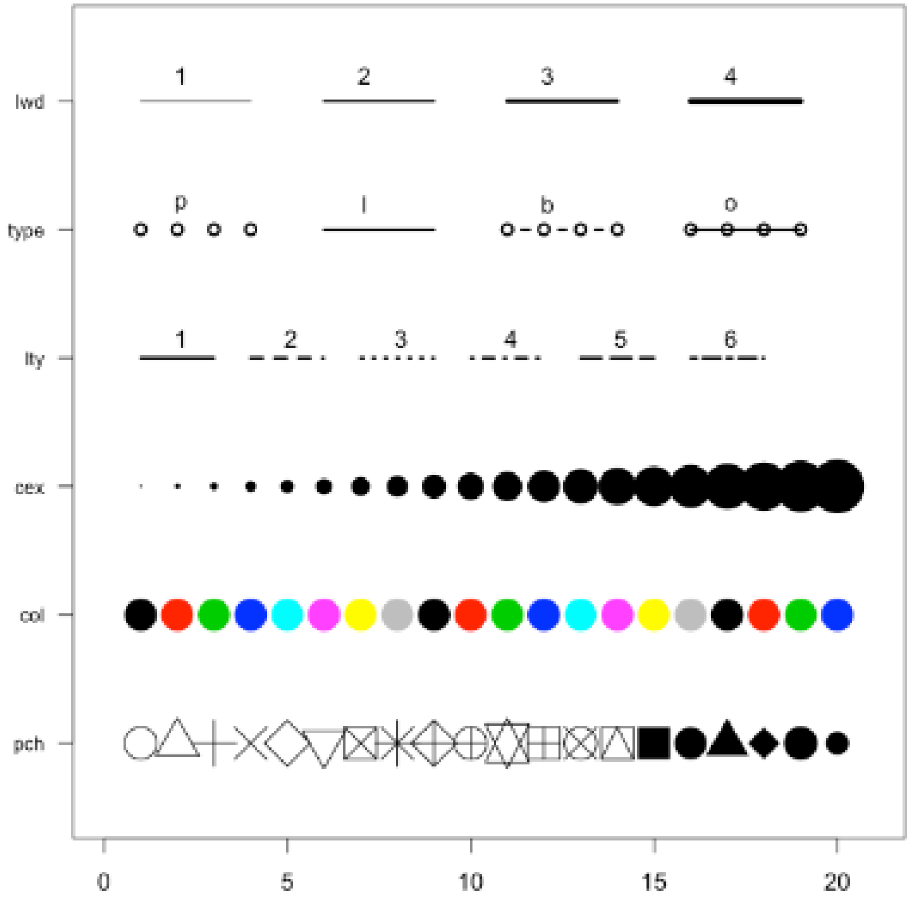
See Graph #6
pch, lwd, pty, col, cex, type...

X-axis name

xlab : name of the axis
cex.lab : size *cex.lab=2*
col.lab : color *col.lab="red"*
sub : to add a subtitle

X-axis features

lab : number of graduation *lab=c(12,2,0)*
xaxp : to add c graduation from a to b: *xaxp=c(a,b,c)*
log : for logarithmic scale: *log="x"*
xaxt : to remove x axis: *xaxt="n"*
fg : color of axis, ticks and grid: *fg="red"*
cex.axis : size of tick labels
col.axis : color of tick labels
xlim : limits of the axis *xlim=c(0,10)*
las: orientation of tick labels
0=parralel to the axis, 1=horizontal...

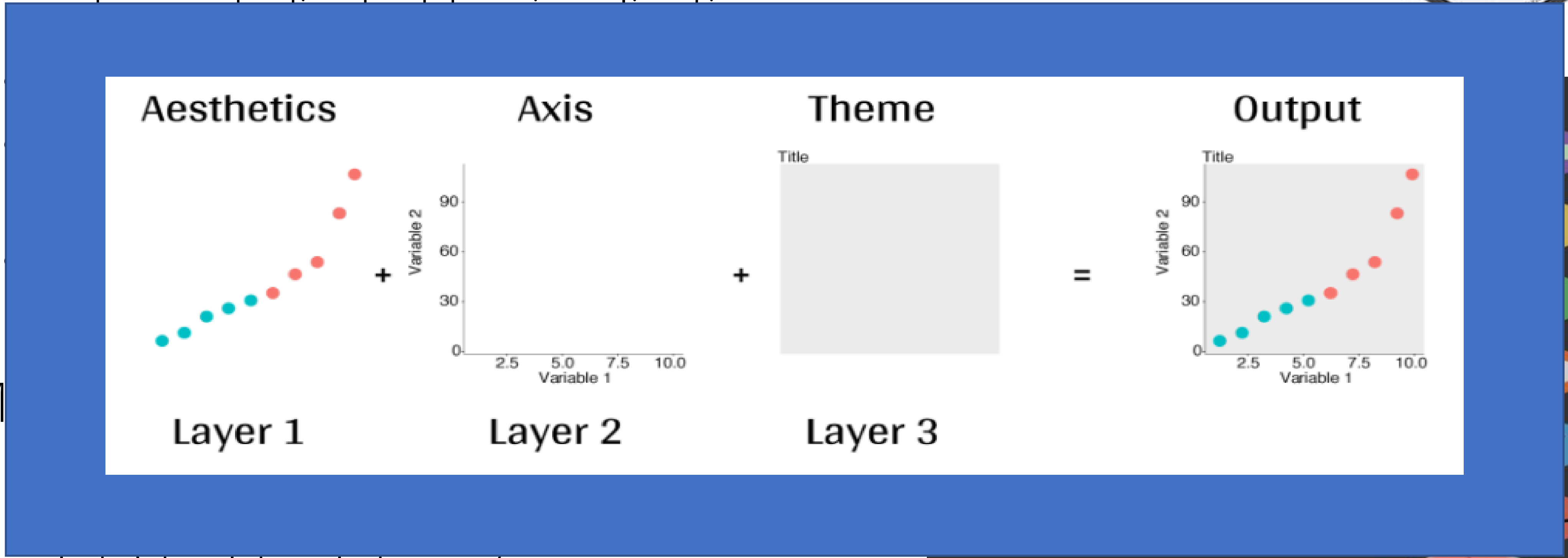


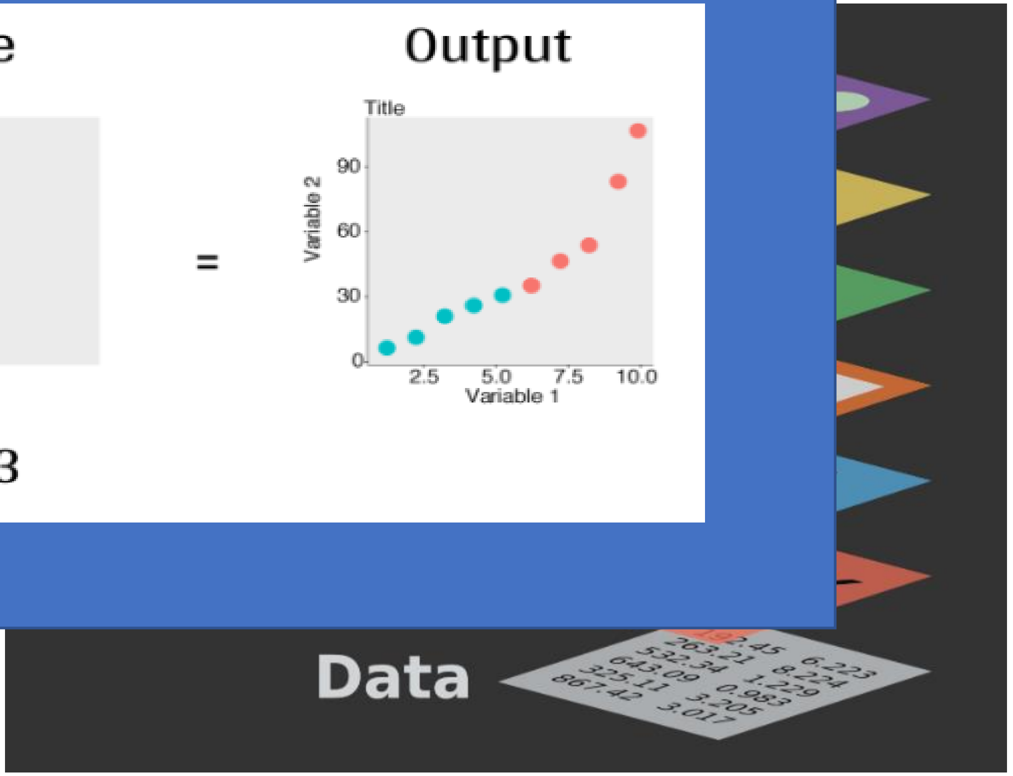
Ggplot2 (or just 'ggplot')

• R

• M

• Ggplot2 builds plots in *layers*





https://r.qcbs.ca/workshop03/

Ggplot example: histogram

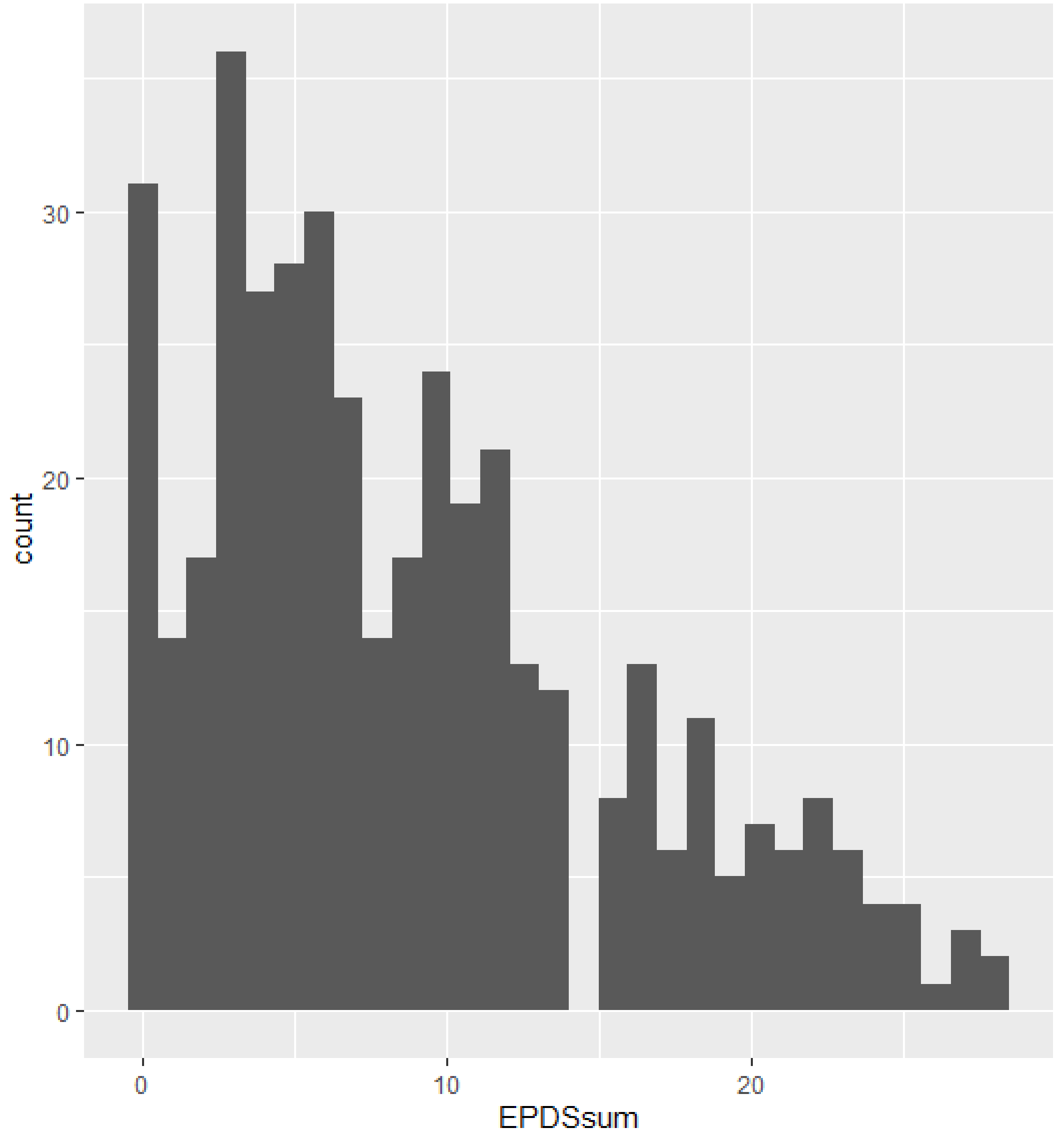
```
install.packages("ggplot2")
library(ggplot2)

# Basic plot
plot <- ggplot(data, aes(x=EPDSsum)) +
        geom_histogram()
plot
```

LAYER 1: Data and which values within this you wish to plot

LAYER 2: Which plot you want

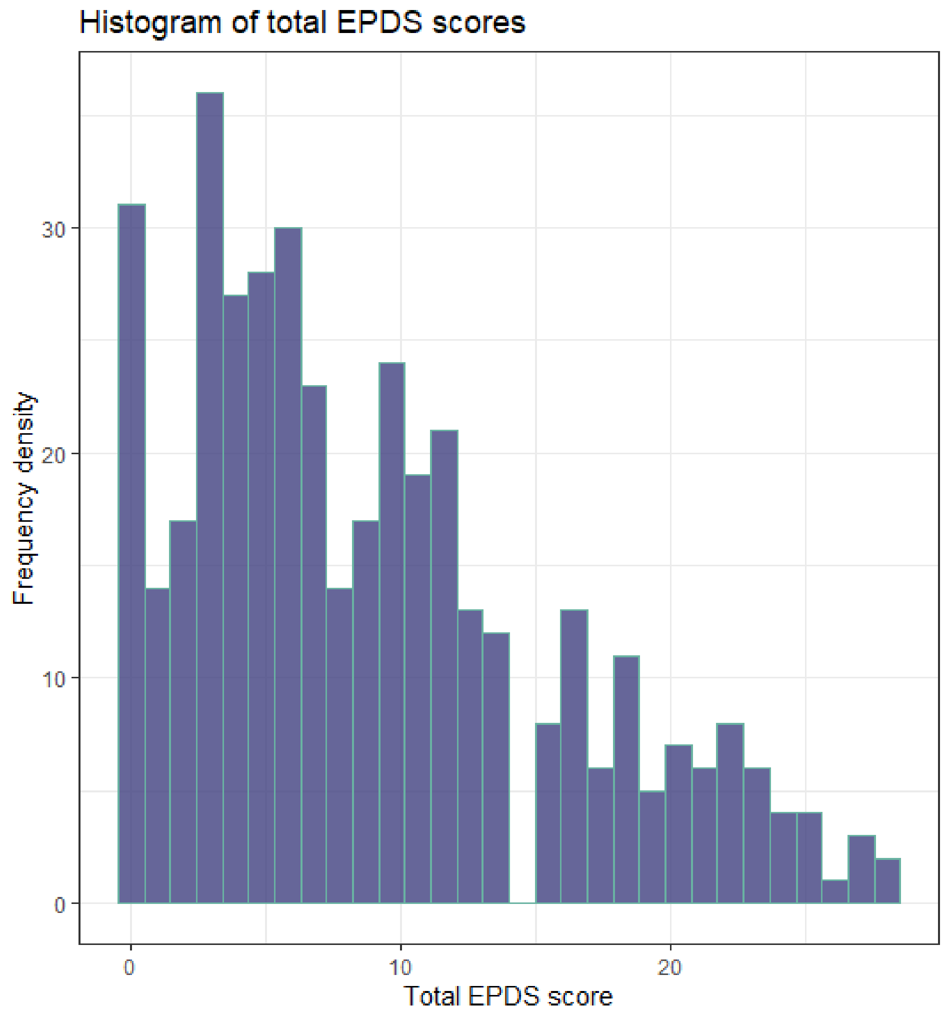
Ggplot example: histogram

```
install.packages("ggplot2")
library(ggplot2)

# Basic plot
plot <- ggplot(data, aes(x=EPDSsum)) +
      geom_histogram()
plot
```



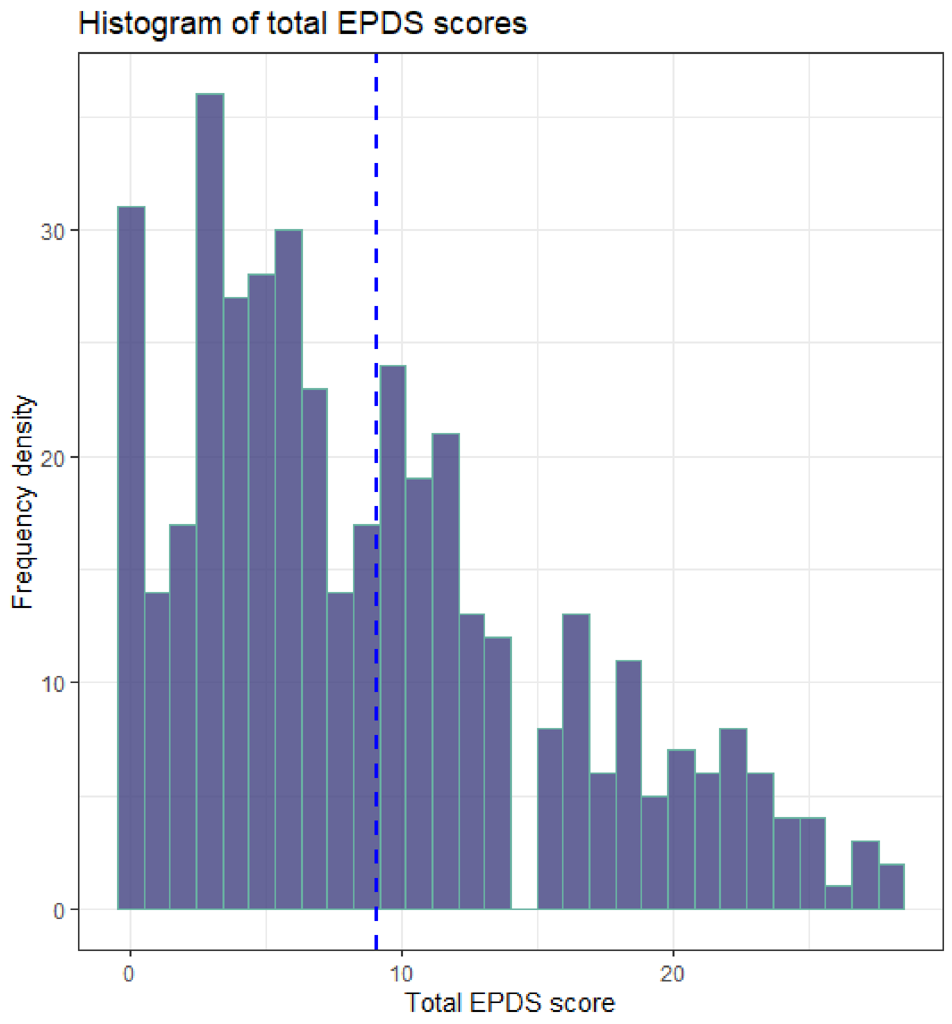
Ggplot example: histogram

```r
# Changing aesthetic values
plot <- ggplot(data, aes(x=EPDSsum)) +
      geom_histogram(color="#69b3a2",
            fill="#404080",
            alpha=0.8) +
      theme_bw() +
      xlab("Total EPDS score") +
      ylab("Frequency density") +
      ggtitle("Histogram of total EPDS scores")

plot
```



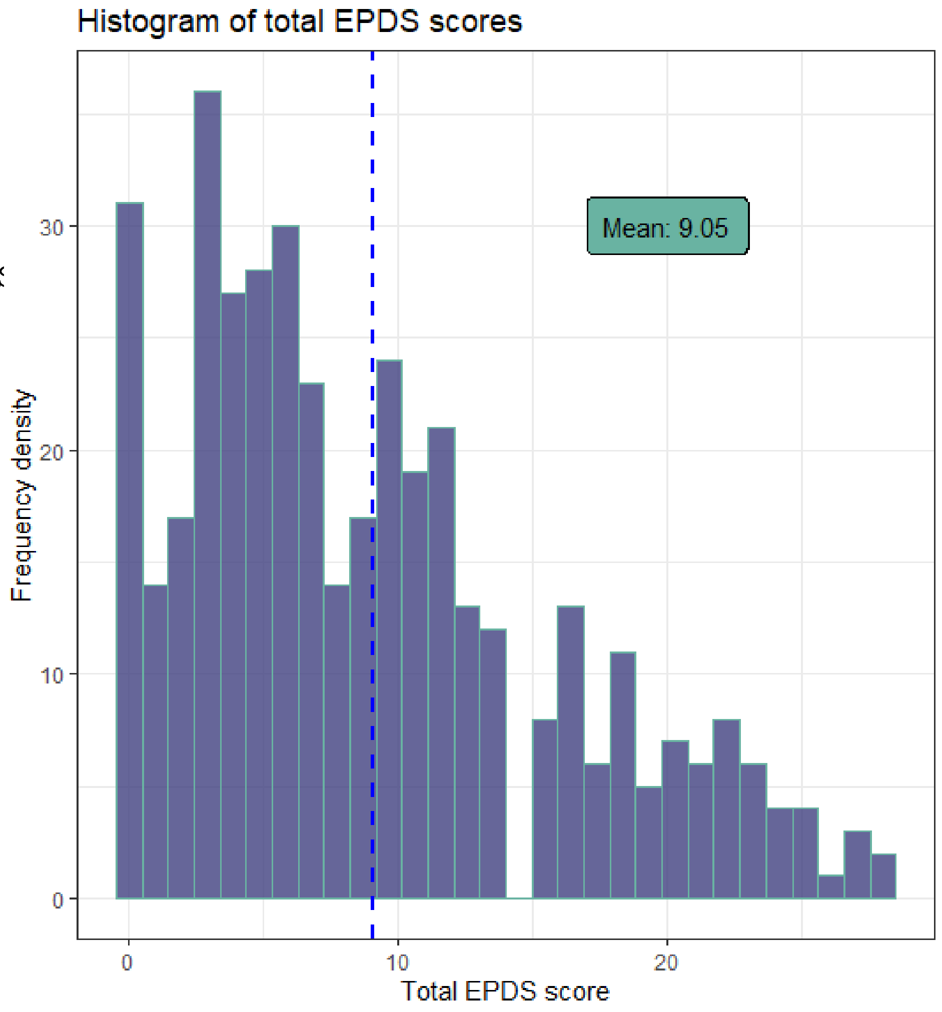
Ggplot example: histogram

```
# Changing aesthetic values
plot <- plot +
      geom_vline(aes(xintercept=mean(EPDSsum)),
            color="blue",
            linetype="dashed",
            size=1)


plot
```



Ggplot example: histogram

```
# Changing aesthetic values
plot <- plot +
        geom_label(
        label=paste0("Mean: ", round(mean(data$EPDS
        x=20, y=30,
        label.padding = unit(0.55, "lines"),
        label.size = 0.35,
        color = "black", fill="#69b3a2")

plot
```



Lesson 4: Hypothesis testing

Recap: what is a hypothesis test?

Are our observations by chance?

- Rolling 3 6s in a row… is it a weighted dice?
- Do male infants wake a greater number of times per night compared to female infants?

- Null hypothesis (H0): what is assumed to be true unless there's strong evidence against it
- Alternative hypothesis (H1): the statement alternative to H0 which is 'accepted' if the evidence is strong enough to reject the null

p-values are typically used to reject the null if <0.05

~100 tests… we'll cover a few!

Hypothesis tests: overview of commands

Comparing two means

T test

```
t.test(var ~ group, data)
```

- Parametric: var normally distributed
- H0: the difference in the mean of var across groups is zero

Mann-Whitney

```
wilcox.test(var ~ group, data)
```

- Non-parametric: no distribution assumed
- H0: the difference in the mean of var across groups is zero

Comparing two categorical variables

Chi-squared

```
table_test <- table(data$var1, data$var2)
chisq.test(table_test)
```

- H0: there is no association between var1 and var2

Fisher's exact

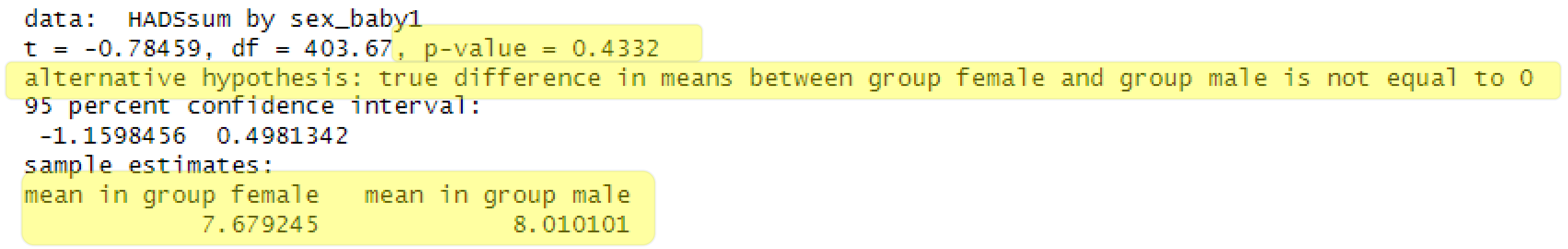
```
table_test <- table(data$var1, data$var2)
fisher.test(table_test)
```

- H0: there is no association between var1 and var2
- Better for smaller samples (cells <5)

T-test

- Parametric → normally distributed → check histograms!
- E.g. Is the total HADS score significantly different across mothers with male and female infants?
  - H0: the difference between means is zero (means are equal)
  - H1: the difference between means is not zero (two-sided)

```
t.test(HADSsum ~ sex_baby1, data = data)
```

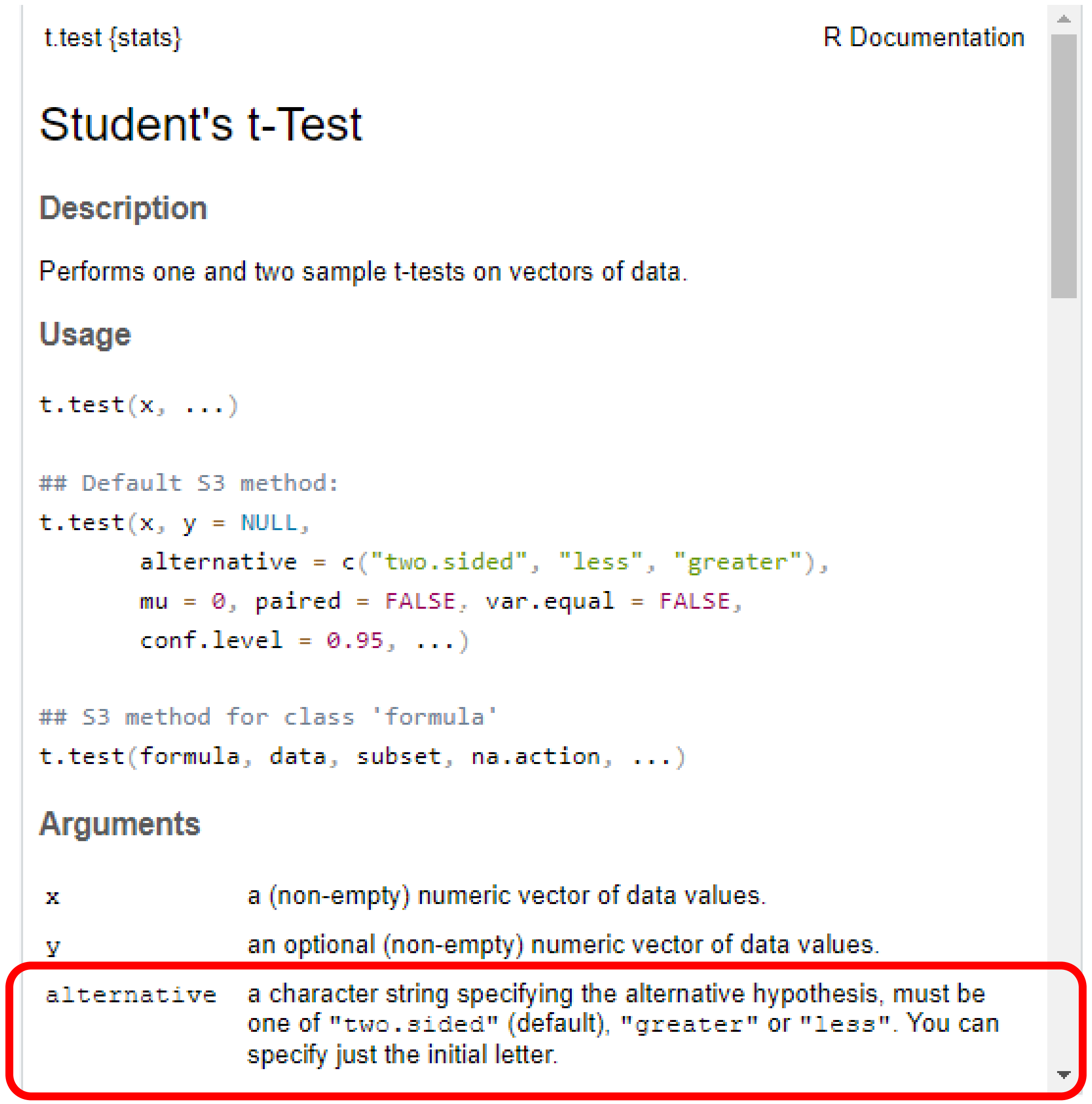


```
data:  HADSsum by sex_baby1
t = -0.78459, df = 403.67, p-value = 0.4332
alternative hypothesis: true difference in means between group female and group male is not equal to 0
95 percent confidence interval:
 -1.1598456  0.4981342
sample estimates:
mean in group female   mean in group male 
            7.679245             8.010101 
```

T-test – one-sided

```
?t.test
```

- ?function to see the documentation for your function
- Describes the function arguments and output
- Very helpful!!



t.test {stats} R Documentation

Student's t-Test

Description

Performs one and two sample t-tests on vectors of data.

Usage

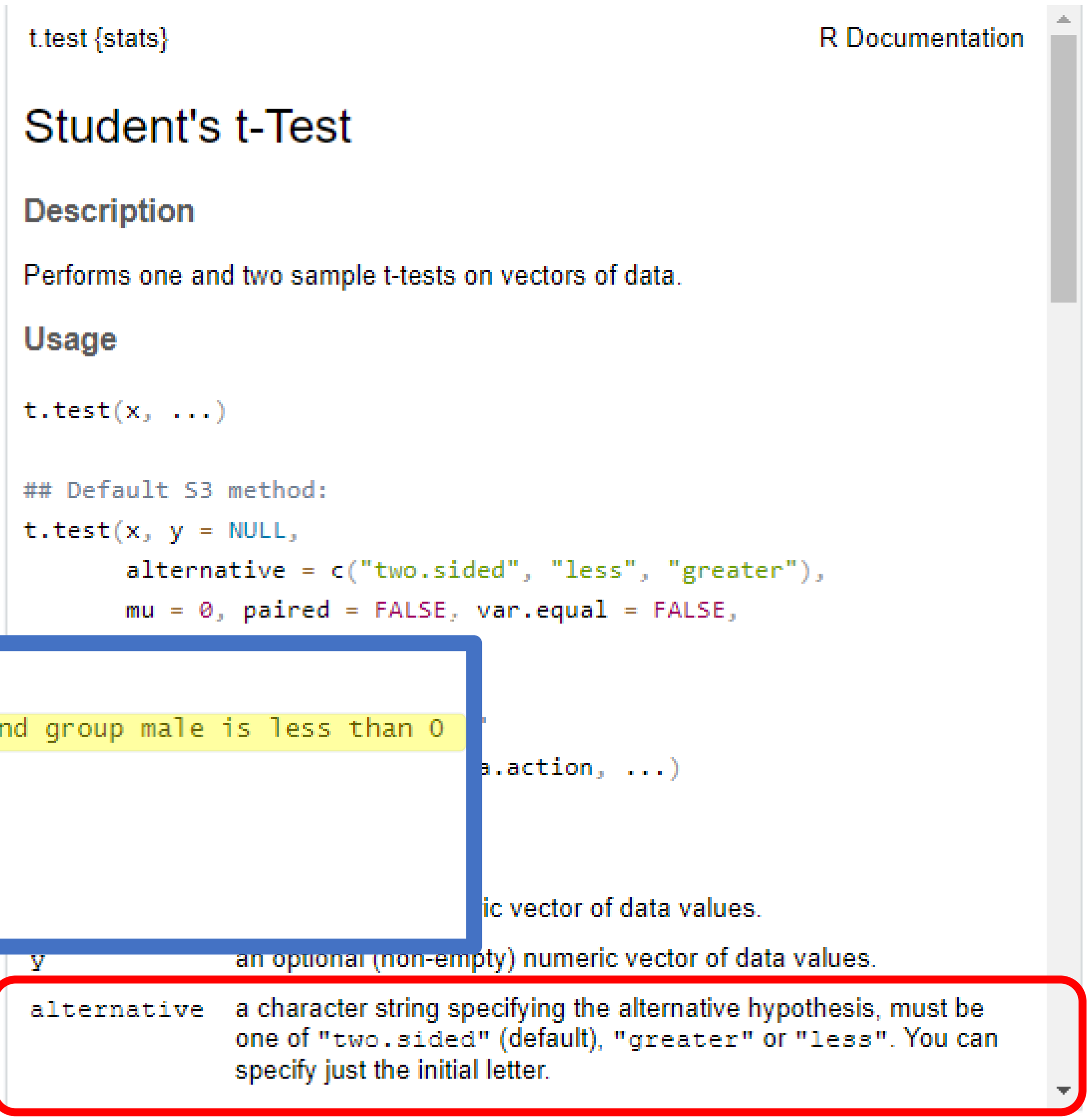
```
t.test(x, ...)

## Default S3 method:
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)

## S3 method for class 'formula'
t.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	a (non-empty) numeric vector of data values.
`y`	an optional (non-empty) numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.

T-test – one-sided

```r
t.test(HADSsum ~ sex_baby1,
       data = data,
       alternative = "greater")
```



t.test {stats} R Documentation

Student's t-Test

Description

Performs one and two sample t-tests on vectors of data.

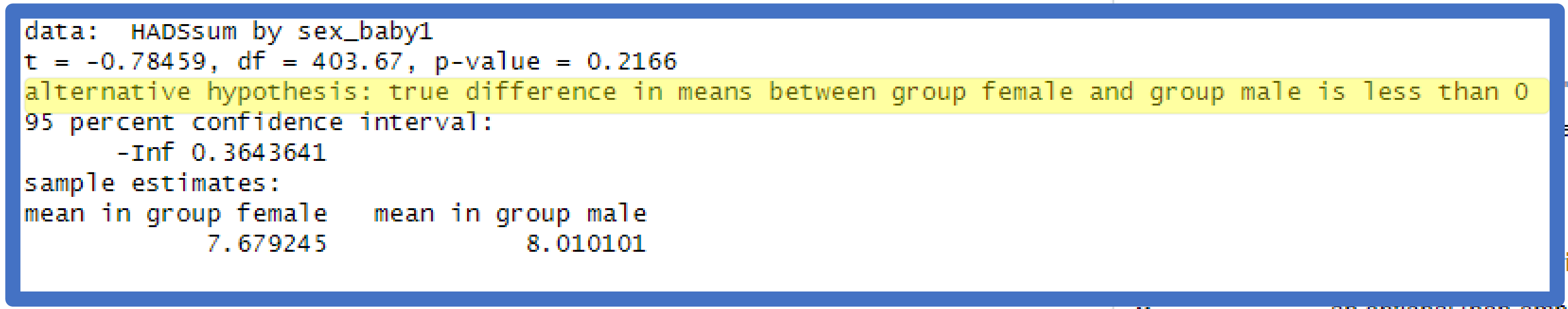
Usage

```r
t.test(x, ...)

## Default S3 method:
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
```

```
a.action, ...)
```

ic vector of data values.

y	an optional (non-empty) numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.



```
data:  HADSsum by sex_baby1
t = -0.78459, df = 403.67, p-value = 0.2166
alternative hypothesis: true difference in means between group female and group male is less than 0
95 percent confidence interval:
      -Inf 0.3643641
sample estimates:
mean in group female   mean in group male
            7.679245             8.010101
```

Hypothesis testing

ACTIVITY 11

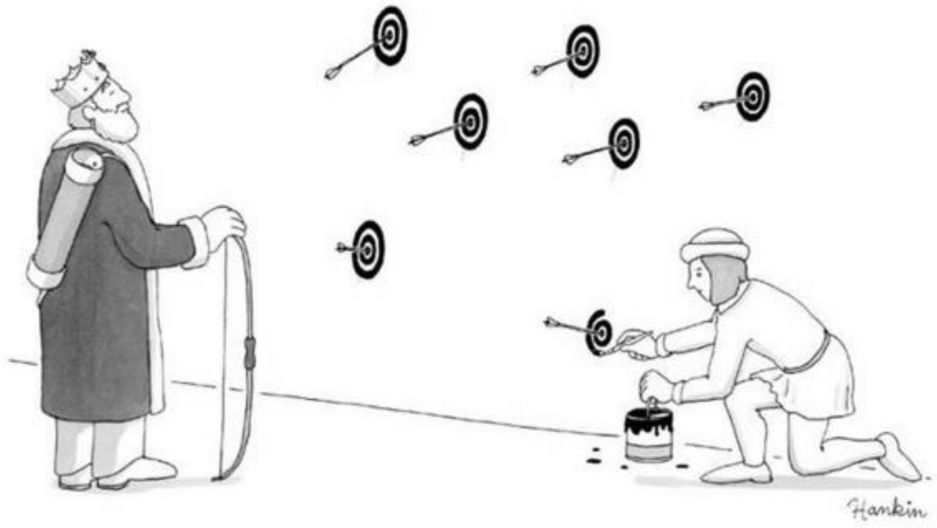
1. Run a t-test to see whether the number of times an infant wakes during the night is significantly different across male and female infants

2. Now save the t-test result as an object called ttest (ttest <- t.test(…))

3. Inspect the structure of ttest. From this, can you extract the p value?

Hypothesis testing… a warning

- Multiple hypothesis tests (p-hacking)
- Hypothesising after inspecting the data
- Statistical significance vs clinical significance…



"Bullseyes" by Charlie Hankin

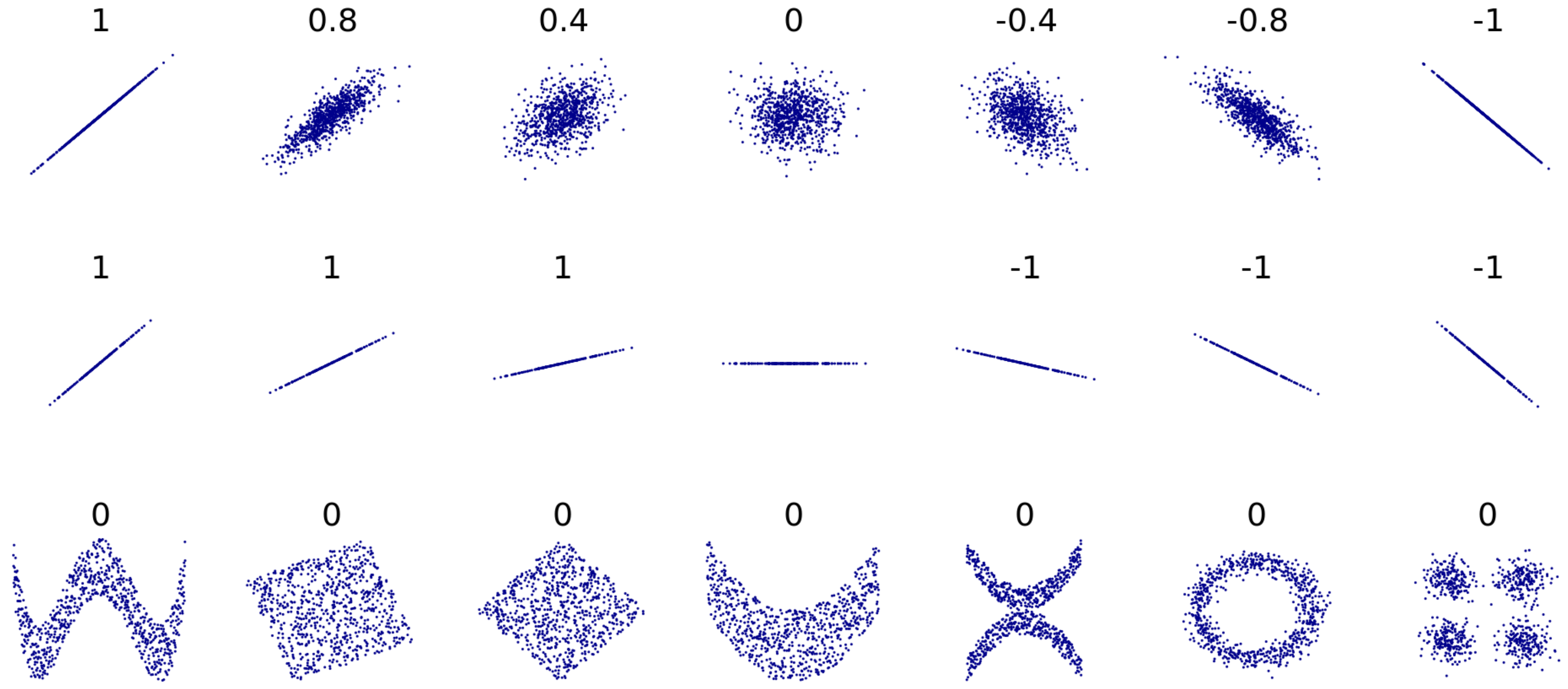
I CAN'T BELIEVE SCHOOLS ARE STILL TEACHING KIDS ABOUT THE NULL HYPOTHESIS.
I REMEMBER READING A BIG STUDY THAT CONCLUSIVELY DISPROVED IT *YEARS* AGO.

xkcd Comic number 892

Lesson 5: Analysis

Correlation

Statistical measure of the association between two *numeric* variables

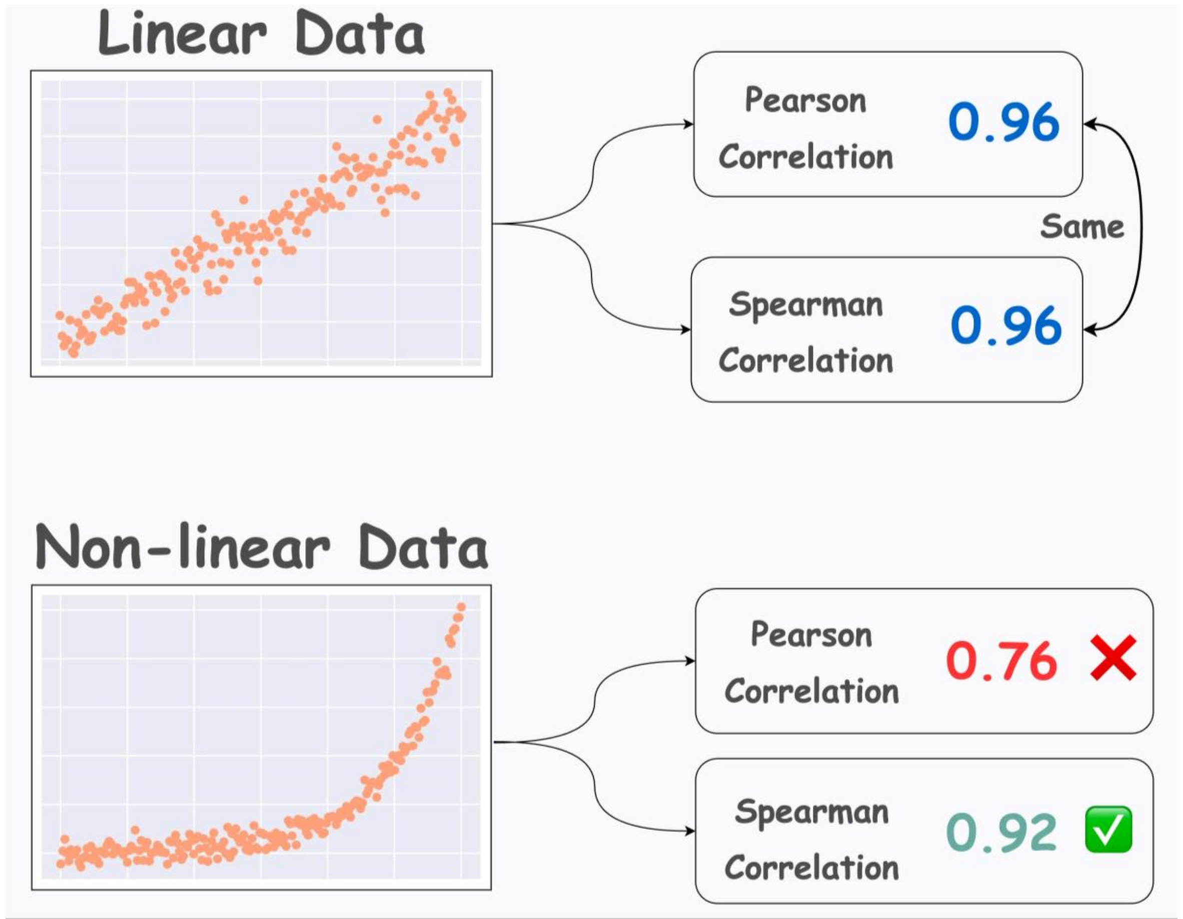


Wikipedia

Correlation

Statistical measure of the association between two *numeric* variables

- Pearson - *linear*
- Spearman - *rank*



DailyDoseofDS.com

Correlation

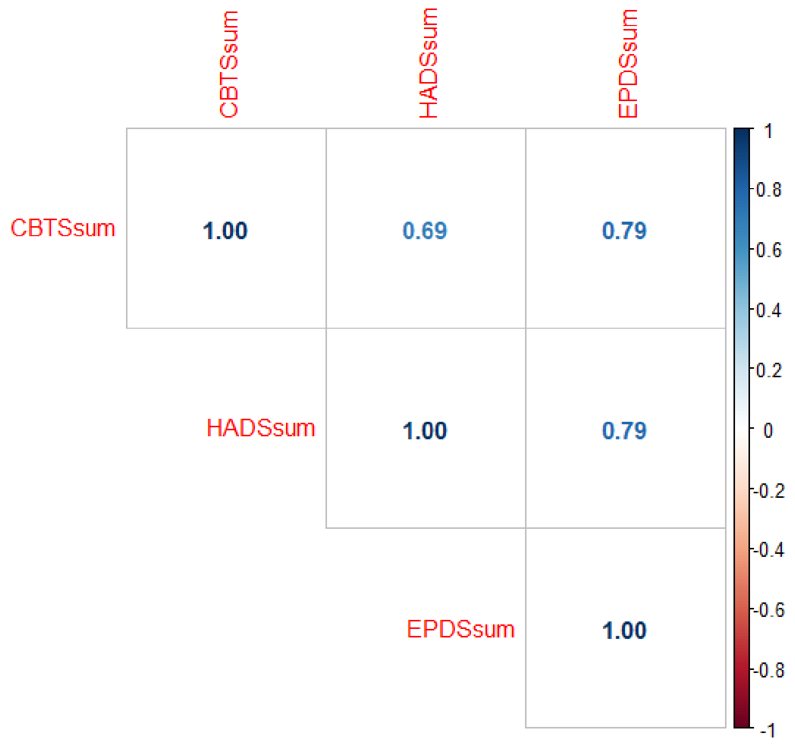
```
cor(data$var1, data$var2, method)

cor(data$CBTSsum, data$night_awakening_number_bb1, method="pearson")
cor(data$CBTSsum, data$night_awakening_number_bb1, method="spearman")
```

ACTIVITY 12

1. You can also put an entire data frame into the cor function to produce a correlation matrix – this is a 2-dimensional table showing the correlation between all columns. Try running **`cor(data)`**. What happens and why?
2. Try to create a correlation matrix for all the mental health sums – HADSsum, EPDSsum and CBTSsum (HINT: subset your data by choosing the relevant columns and then use number 1)
3. Do your results make sense? How would you visualise them?

Correlation plot

```
install.packages("corrplot")
library(corrplot)

dataplot <- data[, c("CBTSsum", "HADSsum", "EPDSsum")]

corrplot(cor(dataplot))

corrplot(cor(dataplot),
        method="number")

corrplot(cor(dataplot),
        method="number", type="upper")
```



Regression analysis

Regression analysis is a process for estimating and quantifying the **relationship** between a **dependent** variable (outcome, response) and one or more **independent** variables (predictors, explanatory variables).

We will go through how to run and interpret a regression model in R.

We will not cover the assumptions of a regression model, how to identify confounders, how to compare models etc.

Linear regression

plainenglish.io

$$Y = b*X + c$$

Dependent variable

Independent variable

Intercept

Linear regression

plainenglish.io

Regression coefficient (AKA beta) → essentially a measure of correlation!

$$Y = b * X + c$$

Dependent variable

Independent variable

Intercept

Linear regression

"Confounder adjustment"

Regression coefficient (AKA beta) → essentially a measure of correlation!

- Beta=0 → no relationship
- Beta>0 → positive relationship
- Beta<0 → negative relationship

(sounds like a hypothesis test, smells like a hypothesis test…)

$$Y = b1*X1 + b2*X2 + \ldots + c$$

Dependent variable

Independent variables

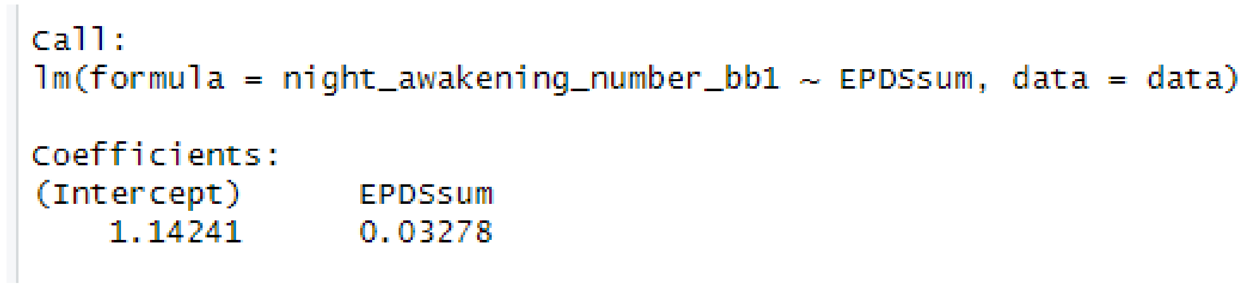
Intercept

Linear regression in R

```r
lm(Y ~ X1 + X2 + X3 + ... , data = data)
```

With EPDS scores and number of night wakes…

```r
lm(night_awakening_number_bb1 ~ EPDSsum, data = data)
```

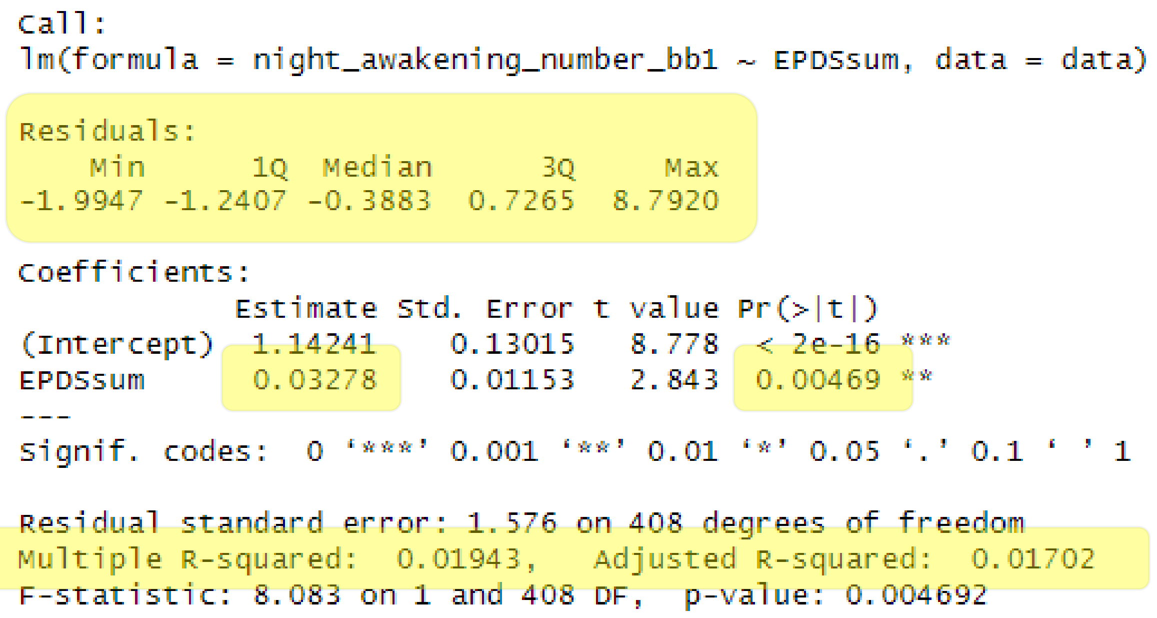


```
Call:
lm(formula = night_awakening_number_bb1 ~ EPDSsum, data = data)

Coefficients:
(Intercept)      EPDSsum
    1.14241      0.03278
```

Linear regression in R

```r
model <- lm(Y ~ X1 + X2 + X3 + ... , data = data)
summary(model)
```

```r
model <- lm(night_awakening_number_bb1 ~ EPDSsum, data = data)
summary(model)
```



```
Call:
lm(formula = night_awakening_number_bb1 ~ EPDSsum, data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.9947 -1.2407 -0.3883  0.7265  8.7920 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.14241    0.13015   8.778  < 2e-16 ***
EPDSsum      0.03278    0.01153   2.843  0.00469 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.576 on 408 degrees of freedom
Multiple R-squared:  0.01943,	Adjusted R-squared:  0.01702 
F-statistic: 8.083 on 1 and 408 DF,  p-value: 0.004692
```

Linear regression in R

ACTIVITY 13

Fit 4 regression models for the dependent variable of number of night wakes:

1. Independent variable: HADS score
2. Independent variable: EPDS score
3. Independent variable: CBTS score
4. Independent variables: HADS, EPDS and CBTS scores

Compare to Table 2 in the publication – do you get the same results? When making conclusions about the mental health scores with the number of times an infant wakes, which model(s) would you choose?

Linear regression in R

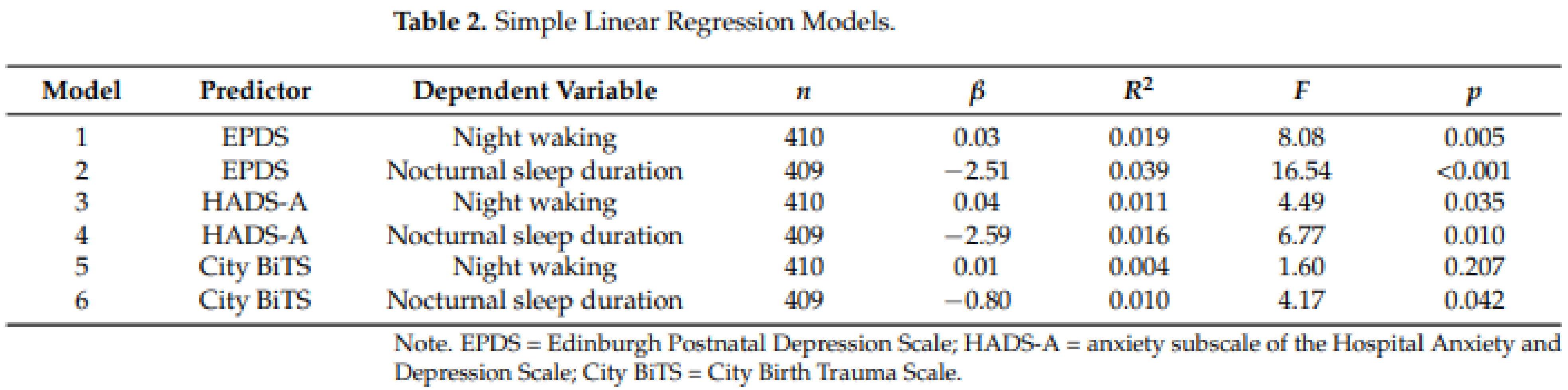


Table 2. Simple Linear Regression Models.

Model	Predictor	Dependent Variable	*n*	*β*	R^2	*F*	*p*
1	EPDS	Night waking	410	0.03	0.019	8.08	0.005
2	EPDS	Nocturnal sleep duration	409	−2.51	0.039	16.54	<0.001
3	HADS-A	Night waking	410	0.04	0.011	4.49	0.035
4	HADS-A	Nocturnal sleep duration	409	−2.59	0.016	6.77	0.010
5	City BiTS	Night waking	410	0.01	0.004	1.60	0.207
6	City BiTS	Nocturnal sleep duration	409	−0.80	0.010	4.17	0.042

Note. EPDS = Edinburgh Postnatal Depression Scale; HADS-A = anxiety subscale of the Hospital Anxiety and Depression Scale; City BiTS = City Birth Trauma Scale.

lm vs glm

- lm = linear models only
- glm = generalised linear models
  - Linear regression
  - Logistic regression
  - Poisson regression

- "gaussian" = linear regression
  - Default link = "identity"
- "binomial" = logistic regression
  - Default link = "logit"
- "poisson" = Poisson regression
  - Default link = "log"

```
lm(Y ~ X1 + X2 + ... , data = data)
```

```
glm(Y ~ X1 + X2 + ... , data = data,
        family = ”gaussian”)
```

Linear regression in R

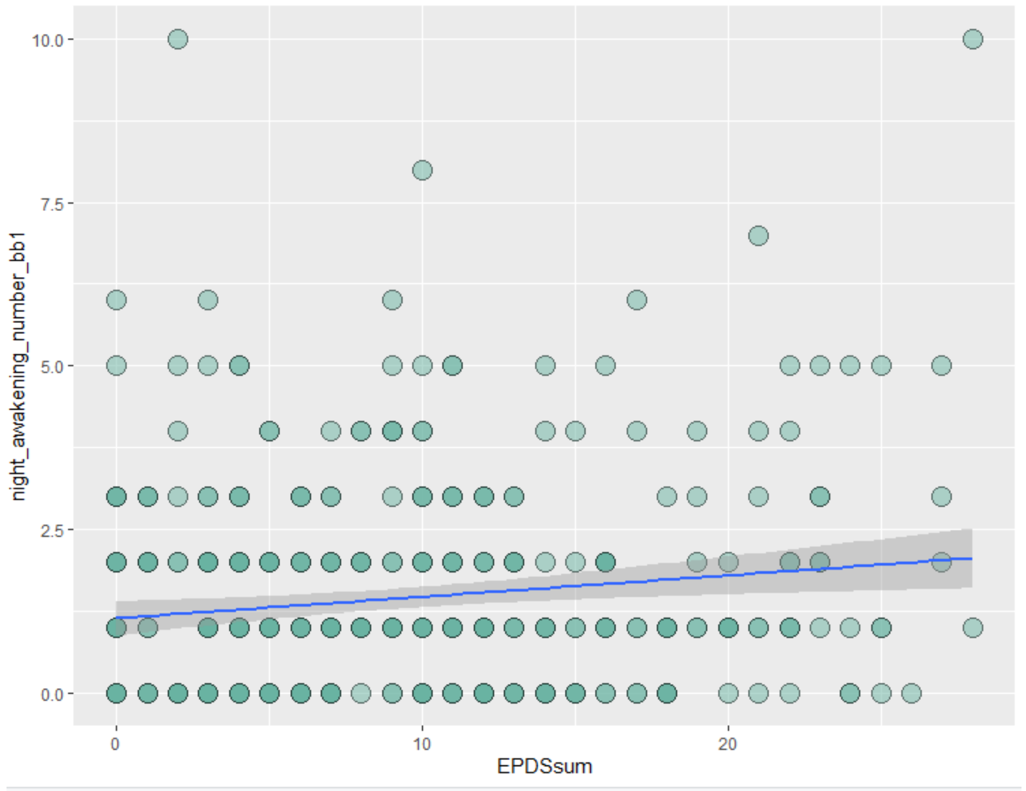
ACTIVITY 14

Re-fit your linear models using the glm function. Verify that you get the same results. Has changed about your output?

```
model <- glm(Y ~ X1 + ..., data = data, family="gaussian")
summary(model)
```

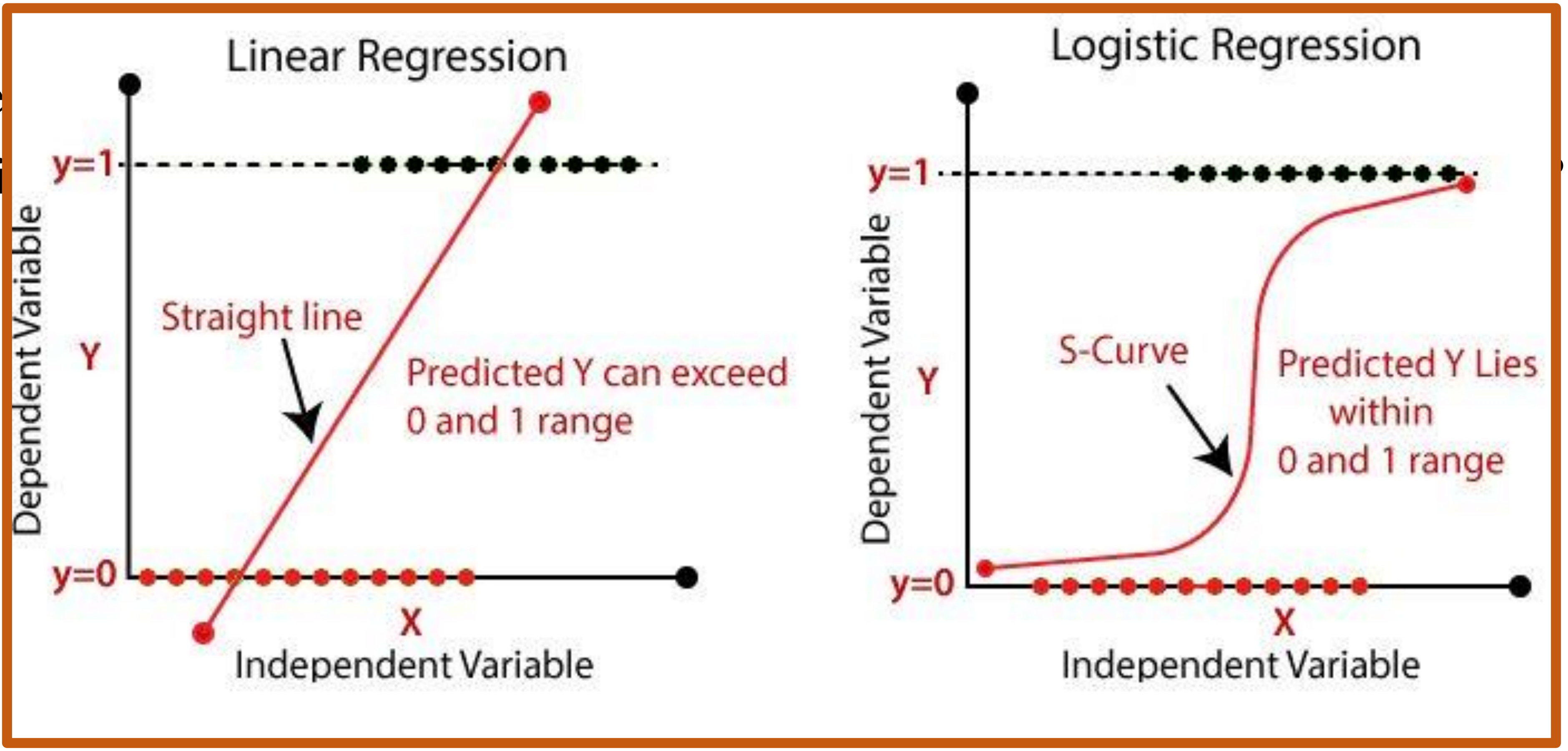
Visualising regression

```
plot <- ggplot(data, aes(x=EPDSsum, y=night_awakening_number_bb1)) +
             geom_point(fill="#69b3a2", shape=21, alpha=0.5, size=5)
             geom_smooth(method='lm', formula= y~x)
plot
```



Logistic regression

- Line
- Logi



(don't worry – you don't have to understand the maths)

Anit Choudhury on LinkedIn

Logistic regression

Coefficients in logistic regression…

- Represent the linear relationship between your independent variable and the logit probability of the dependent variable
- If you take the exponential of these, they represent odds ratios

- OR=0 → no relationship
- OR>0 → increased occurrence
- OR<0 → decreased occurrence (i.e. a protective effect)

(sounds like a hypothesis test, smells like a hypothesis test…)

Logistic regression

```
model <- glm(Y ~ X1 + ..., data = data, family="binomial")
summary(model)
```

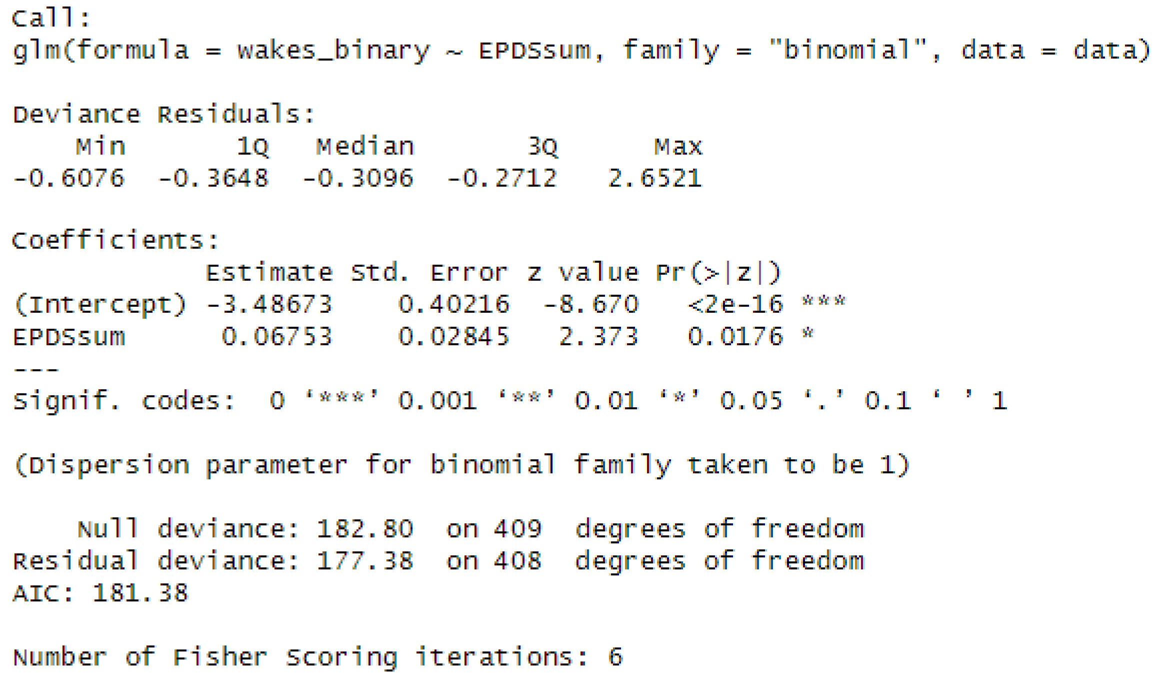
ACTIVITY 15

1. Create a new variable, wakes_binary, which categories the number of wakes during the night into 'less than 5' and '5 or more'

2. Using this new variable as your dependent variable, fit a logistic regression model with EPDS score as your independent variable. Take the exponential of the coefficient to get the odds ratio using **`exp(…)`**. What has changed compared to the linear regression? Can you think of what might have caused this?

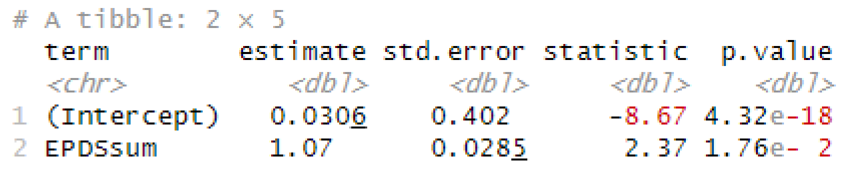
Regression – tidy outputs

```r
summary(model)
```



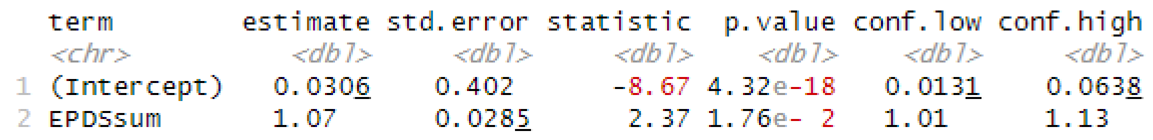
```
Call:
glm(formula = wakes_binary ~ EPDSsum, family = "binomial", data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6076  -0.3648  -0.3096  -0.2712   2.6521

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.48673    0.40216  -8.670   <2e-16 ***
EPDSsum      0.06753    0.02845   2.373   0.0176 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 182.80  on 409  degrees of freedom
Residual deviance: 177.38  on 408  degrees of freedom
AIC: 181.38

Number of Fisher Scoring iterations: 6
```

```r
install.packages("broom")
library(broom)
tidy(model, exponentiate = TRUE)
```



```
# A tibble: 2 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   0.0306    0.402      -8.67 4.32e-18
2 EPDSsum       1.07      0.0285      2.37 1.76e- 2
```

```r
tidy(model, exponentiate = TRUE,
       conf.int = TRUE)
```

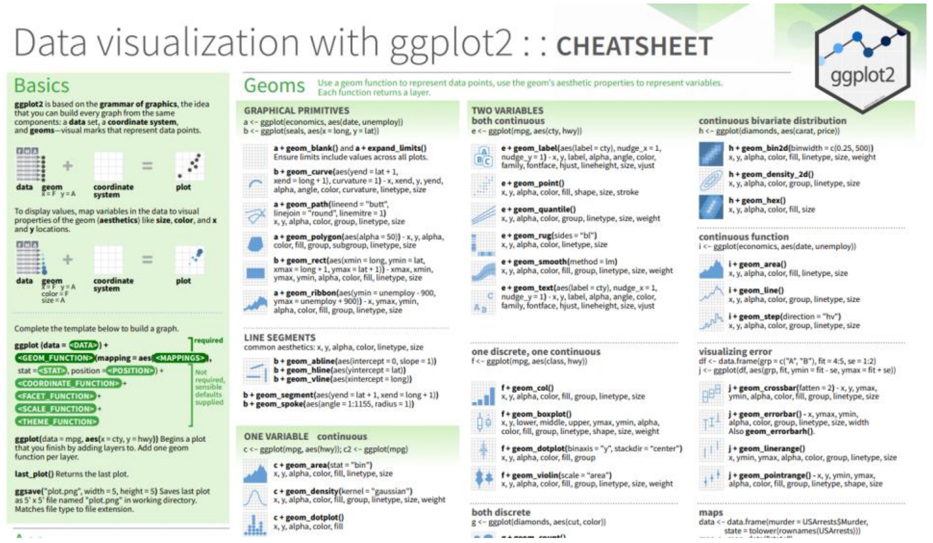


```
  term        estimate std.error statistic  p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)   0.0306    0.402      -8.67 4.32e-18   0.0131    0.0638
2 EPDSsum       1.07      0.0285      2.37 1.76e- 2   1.01      1.13
```

Closing remarks

Where can I get help?

- Google your error messages
- Stack exchange
- R Documentation → ?function
- Go down to basics
- Cheat sheets



Data visualization with ggplot2 : : CHEATSHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT>, position = <POSITION>) +                    Not
<COORDINATE_FUNCTION> +                                    required,
<FACET_FUNCTION> +                                         sensible
<SCALE_FUNCTION> +                                         defaults
<THEME_FUNCTION>                                           supplied
```

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and **a + expand_limits()**
Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS
common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

TWO VARIABLES
both continuous
e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl")
x, y, alpha, color, linetype, size

e + geom_smooth(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous
f <- ggplot(mpg, aes(class, hwy))

f + geom_col()
x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot()
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill, group

f + geom_violin(scale = "area")
x, y, alpha, color, fill, group, linetype, size, weight

both discrete
g <- ggplot(diamonds, aes(cut, color))

continuous bivariate distribution
h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, fill, size

continuous function
i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

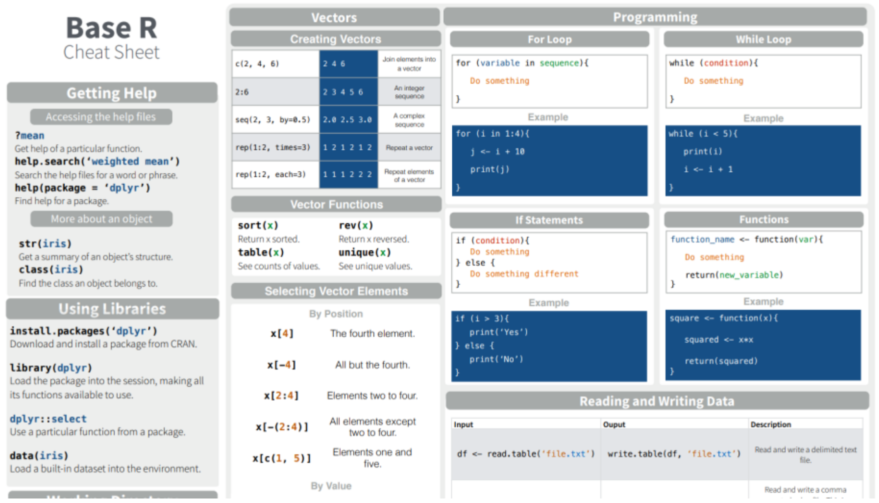
j + geom_crossbar(fatten = 2) - x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width
Also **geom_errorbarh()**.

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))



Base R
Cheat Sheet

Getting Help

Accessing the help files

`?mean`
Get help of a particular function.
`help.search('weighted mean')`
Search the help files for a word or phrase.
`help(package = 'dplyr')`
Find help for a package.

More about an object

`str(iris)`
Get a summary of an object's structure.
`class(iris)`
Find the class an object belongs to.

Using Libraries

`install.packages('dplyr')`
Download and install a package from CRAN.

`library(dplyr)`
Load the package into the session, making all its functions available to use.

`dplyr::select`
Use a particular function from a package.

`data(iris)`
Load a built-in dataset into the environment.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

`sort(x)` Return x sorted.	`rev(x)` Return x reversed.
`table(x)` See counts of values.	`unique(x)` See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

Programming

For Loop

```
for (variable in sequence){

   Do something

}
```

Example

```
for (i in 1:4){

   j <- i + 10

   print(j)

}
```

While Loop

```
while (condition){

   Do something

}
```

Example

```
while (i < 5){

   print(i)

   i <- i + 1

}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

Functions

```
function_name <- function(var){

   Do something

   return(new_variable)
}
```

Example

```
square <- function(x){

   squared <- x*x

   return(squared)
}
```

Reading and Writing Data

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
		Read and write a comma

Where do I go next?

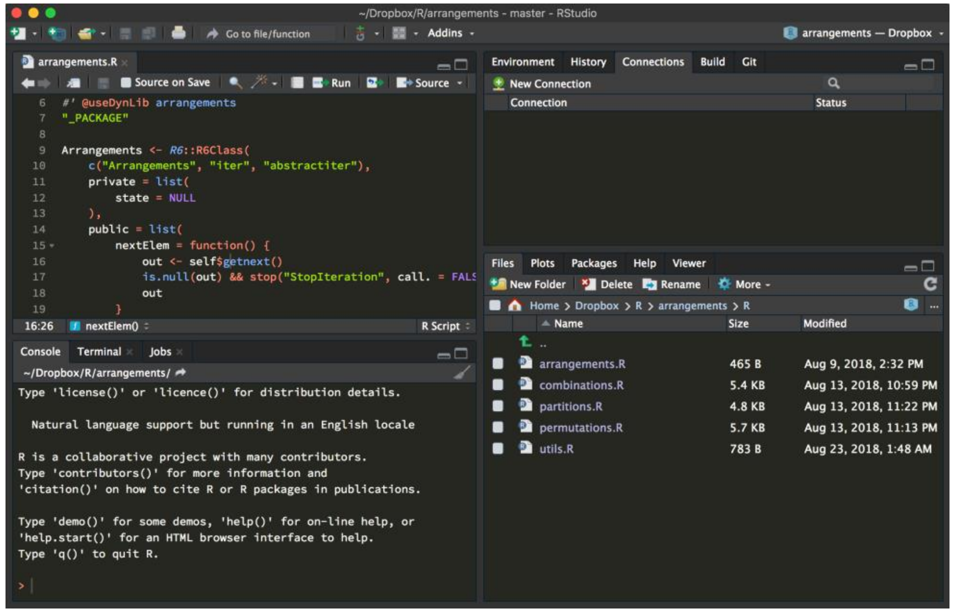
Post-workshop activity to build on today's knowledge and receive some individualised constructive feedback

- To consolidate your knowledge: base R!
- To make life easier: tidyverse (tidyr, dplyr)
- To wow your colleagues: ggplot2
- To analyse survival data: survival
- To learn more programming basics: for loops, logic operators

- Books
  - R for Data Science, Garrett Grolemund and Hadley Wickham
    - https://r4ds.had.co.nz/index.html
  - Advanced R, Hadley Wickham
  - The R Book, Michael Crawley

Tips for working with R

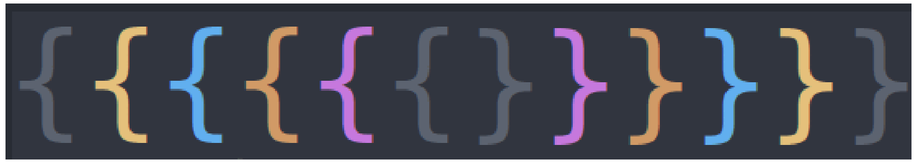
- Changing the visual display
  - Tools > global options > appearance
- Don’t let it get really out of date…
- Comment
- SENSE CHECK ALWAYS
  - Test cases
  - Esp important if using code you didn’t write
- ChatGPT
  - “The first rule of using ChatGPT to code is that you should only use it if you don’t actually need it”



```r
#' @useDynLib arrangements
"_PACKAGE"

Arrangements <- R6::R6Class(
    c("Arrangements", "iter", "abstractiter"),
    private = list(
        state = NULL
    ),
    public = list(
        nextElem = function() {
            out <- self$getnext()
            is.null(out) && stop("StopIteration", call. = FALS
            out
        }
```

```
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

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.

> 
```

Name	Size	Modified
..		
arrangements.R	465 B	Aug 9, 2018, 2:32 PM
combinations.R	5.4 KB	Aug 13, 2018, 10:59 PM
partitions.R	4.8 KB	Aug 13, 2018, 11:22 PM
permutations.R	5.7 KB	Aug 13, 2018, 11:13 PM
utils.R	783 B	Aug 23, 2018, 1:48 AM



Why R and not Excel?

- Ability to retrace steps / reproducibility
- Efficiency
  - More advanced tasks
  - Larger datasets
- Visualisations
- R has a nice balance of being programming savvy and user friendly

Excel is still very useful though ☺

Cheat sheets

A library: https://rforpoliticalscience.com/cheat-sheets-in-r/

Some favourites:

- R basics:
  - https://iqss.github.io/dss-workshops/R/Rintro/base-r-cheat-sheet.pdf
- Tidyr and dplyr
  - https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf
- Ggplot2
  - https://rstudio.github.io/cheatsheets/data-visualization.pdf

Acknowledgements

- The study which collected the data:

  Sandoz V, Lacroix A, Stuijfzand S, Bickle Graz M, Horsch A. Maternal Mental Health Symptom Profiles and Infant Sleep: A Cross-Sectional Survey. Diagnostics. 2022; 12(7):1625.

- Hannah Lennon

- Vicki and Chantelle ☺

We want to improve! Please provide (anonymous) feedback:

https://forms.office.com/e/wPCt4Zk4zA