Before we start...
following the introduction lesson
(This slide will be an exercise for the intermediate group)

- In the last exercise you opened up a new R project
  - R projects are a good practice, but why is that? answer the following questions
- Explain why analysis should save scripts rather than environments.
- Explain what a *working directory* is.
- Why is *setwd()* a bad idea to use in your scripts?
- Explain the difference between an absolute path and a relative path

3 minutes

Syntax, functions, loops, and data types

March 2019

2

Sarid Research Services

Materials by Adi Sarid https://adisarid.github.io and http://www.sarid-ins.com

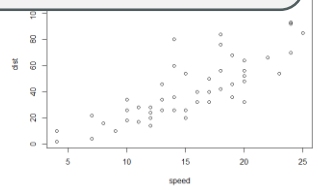
Syntax types

What we're going to focus on today

Base-

Loops (for, while)

Base plots

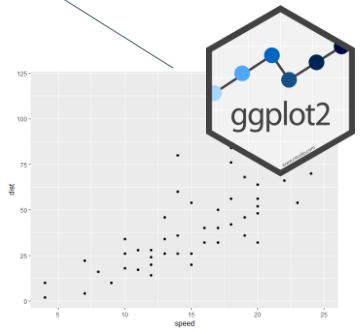


Syntax types

Functions, conditionals (if)

Data types

Functional programming



A short quiz, in pairs

- Name the **4** main base-r data types
  - Bonus if you can get that up to **6**
  - Novice: you can guess by previous experience (e.g., with R or other programming languages or just general knowledge)
  - Intermediate: you should probably know this already…

Answers will appear here after click

How data types work?

- The basic data types can join together to form more complex structures such as: vectors, data.frames, matrix, or even more complex structures called lists
  - You will cover most of these in the next exercise

Vector:

Data frame:

num	date	fact	num2	logi

Matrix:

Lists: an arbitrary combination

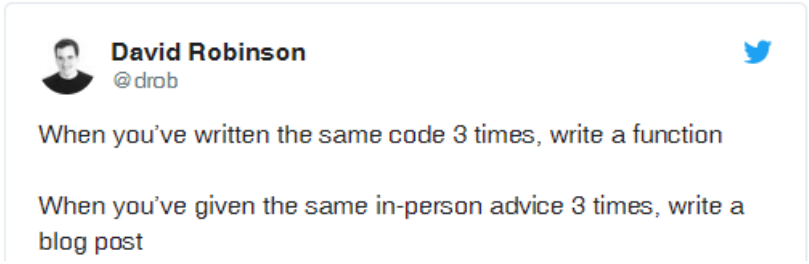
The missing value ("*NA*")

- *NA* is short for Not Available (or not applicable).
- Objects in R can have *NA* as a value, e.g., a vector like *c(1, 2, NA, 10, 3)*.
- This happens a lot with data.
- Can you guess what the following will yield?
  - *NA + 1*
  - *NA * 1*
  - *NA * 0*
  - *NA ^ 0*
  - *is.na(NA)*
- Using R's console check if you were right. Can you explain this behavior?

2 minutes

Inf, -Inf, NaN, NULL,

- R can work with infinite values *Inf*, *-Inf*
  - What are they useful for?
- *NaN* is Not a Number (i.e., *log*(*-Inf*), *Inf*0*)
- *NULL* is a null value, it can be used to initialize variables, or used as a return value for functions when they fail or yield no other result
- A set of base-r functions can help us handle these special values:
  - *is.na*
  - *is.finite*
  - *is.infinite*
  - *is.null*
  - *is.nan*



How functions work?

- A function is code you write **once** when you want to use it **many times** perhaps with some variations. For example:

```
oneplus <- function(number){
    new_number <- number + 1
    return(new_number)
}
```

function definition

```
oneplus(1)
oneplus(oneplus(1))
```

function usage

Iterations (loops)
Which of the following are iterations?

- Try to run them in an R Script and see what their output is

```
a <- 0
while (a < 100){
   a <- a+1
}
```

```
library(purrr)
map_dbl(1:100, function(a){a+1})
```

```
a <- 1:100
a <- a + 1
```

```
a <- 0
for (i in 1:10){
   a <- a+1
}
```

Iterations (loops)
Which of the following are iterations?

- Try to run them in an R Script and see what their output is

Answers will appear here after a click

Conditionals (if…else if…else)

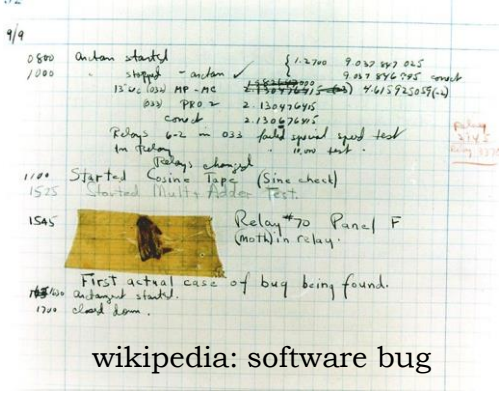
- Check a condition and decide what code should run
- Very useful from within functions

```
a <- 1:10000
if (length(a)<100){
   cat("a is a short vector")
} else if (length(a) < 1000) {
   cat("a is a medium sized vector")
} else {
   cat("a is a long vector")
}
```

Time for some exercise

- Open “01-Syntax, functions, loops, data types.Rmd” and start the exercise.

Debugging



wikipedia: software bug

- A bug is an error which makes your program/function crash or to not work properly
- Sometimes, bugs are hard to identify and to locate

Realize you have a bug → Make it repeatable (reprex) → Figure out where it is → Fix it and test it

- RStudio has some built-in tools to help us debug our code
  - Editor break points, next, step-in functions and continue

```
18  best <- 0
19  for (x in 100:999) {
20    for (y in x:999) {
21      candidate <- x * y
22      if (candidate > best && palindrome(candidate)) {
23        best <- candidate
24      }
25    }
26  }
```

Breakpoints will be activated when this file is sourced.

```
1  # Indicate whether a number is a palindrome
2  palindrome <- function(num) {
3    digits <- floor(log(num, 10))
4    for (x in 0:((digits %/% 2) + 1)) {
5      place1 <- digits - x
```