MAX PLANCK INSTITUTE FOR **BIOLOGY OF AGEING**

Introduction to R

bioinformatics@age.mpg.de

Outline

1) Introduction to R Studio
2) Overview over basic data types
3) Overview over basic functions
4) How to get help
5) Read in data
6) Basic table functions
7) Basic table manipulation
8) Loops and if queries

END OF DAY 1

R Studio

RStudio

https://rstudio.age.mpg.de

File Edit Code View Plots Session Build Debug Profile Tools Help

first_steps.R | M

Source on Save | Run | Source

```
1  # this is a script
```

1:19 (Top Level) R Script

Console ~/

```
R version 3.2.4 (2016-03-10) -- "Very Secure Dishes"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (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.

  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.

> plot(seq(1,10), seq(10,1))
> M = matrix(1,5,6)
> View(M)
> View(M)
> View(M)
>
```

Environment | History

Import Dataset | List

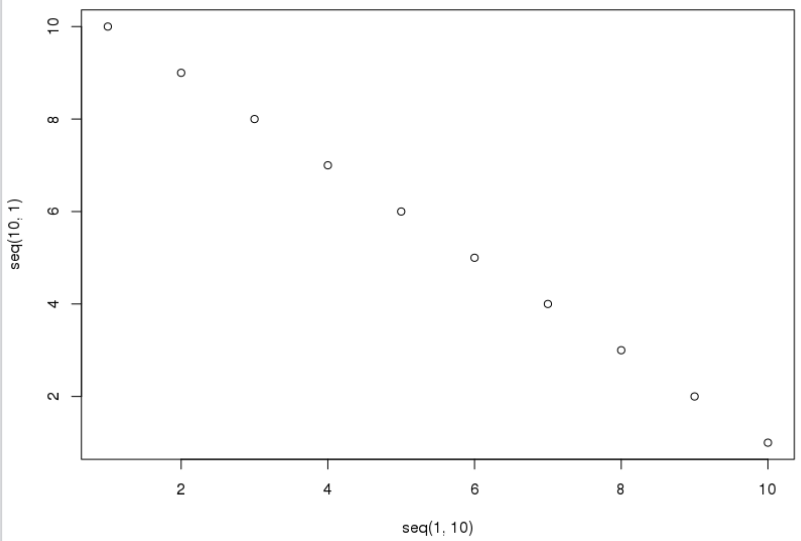
Global Environment

Data

M	num [1:5, 1:6] 1 1 1 1 1 1 1 1 1 1 ...

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish



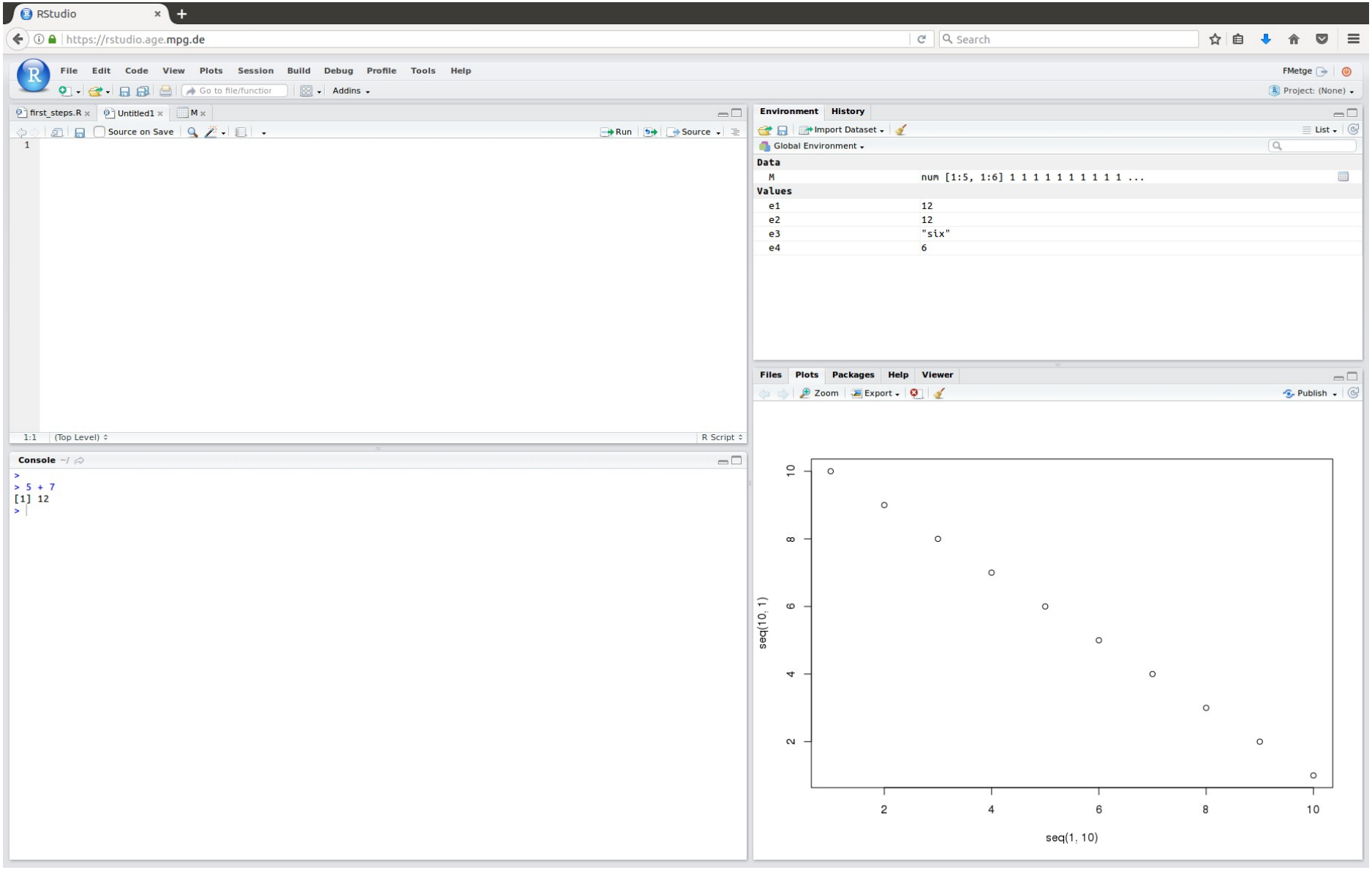
Reinforce what you have learned in the online tutorial

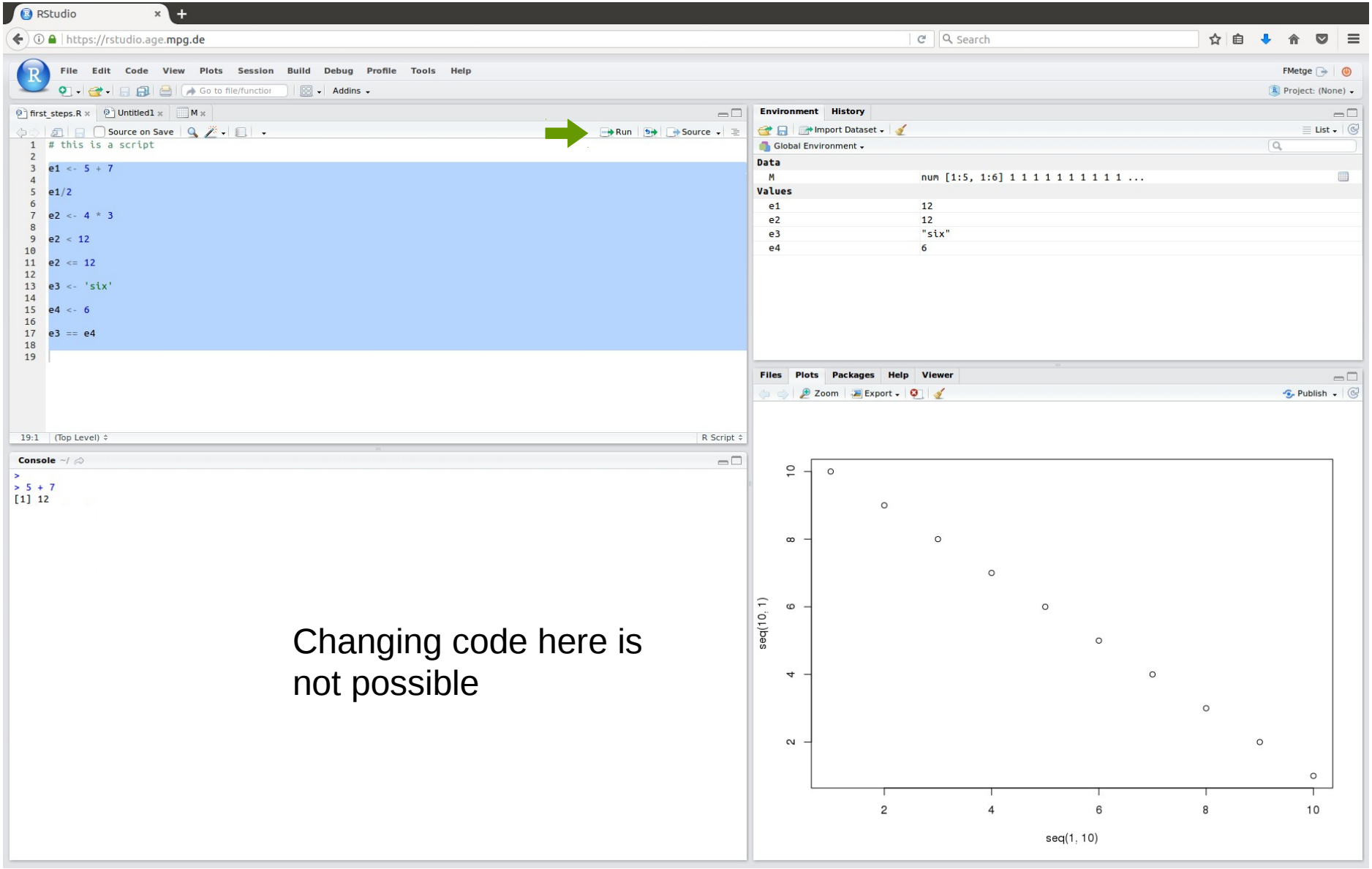
Basic calculations:

- Add 5 and 7
- Divide the results by 2
- Multiply 4 and 3
- Test if the result is smaller than 12

- Create one variable with six as character and one with 6 as number
- Compare if both variables are the same

GitHub


RStudio
https://rstudio.age.mpg.de
Search
File Edit Code View Plots Session Build Debug Profile Tools Help
FMetge
Go to file/functior
Addins
Project: (None)
first_steps.R
Untitled1
M
Source on Save
Run
Source
1
1:1 (Top Level)
R Script
Console ~/
>
> 5 + 7
[1] 12
>
Environment History
Import Dataset
List
Global Environment
Data
M num [1:5, 1:6] 1 1 1 1 1 1 1 1 1 1 ...
Values
e1 12
e2 12
e3 "six"
e4 6
Files Plots Packages Help Viewer
Zoom
Export
Publish
seq(10, 1)
seq(1, 10)
2
4
6
8
10


RStudio
https://rstudio.age.mpg.de
Search
File Edit Code View Plots Session Build Debug Profile Tools Help
FMetge
Project: (None)
Addins
first_steps.R
Untitled1
M
Source on Save
Run
Source
1 # this is a script
2
3 e1 <- 5 + 7
4
5 e1/2
6
7 e2 <- 4 * 3
8
9 e2 < 12
10
11 e2 <= 12
12
13 e3 <- 'six'
14
15 e4 <- 6
16
17 e3 == e4
18
19
19:1 (Top Level)
R Script
Environment
History
Import Dataset
List
Global Environment
Data
M num [1:5, 1:6] 1 1 1 1 1 1 1 1 1 1 ...
Values
e1 12
e2 12
e3 "six"
e4 6
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
Console ~/
>
> 5 + 7
[1] 12
Changing code here is not possible
seq(10, 1)
seq(1, 10)
2
4
6
8
10

RStudio

https://rstudio.age.mpg.de

File Edit Code View Plots Session Build Debug Profile Tools Help

FMetge

Addins

Project: (None)

first_steps.R | Untitled1 | M

Source on Save

Run Source

```
1  # this is a script
2
3  e1 <- 5 + 7
4
5  e1/2
6
7  e2 <- 4 * 3
8
9  # e2 < 12
10
11 e2 <= 14
12
13 e3 <- 'six'
14
15 e4 <- 6
16
17 e3 == e4
18
19
```

18:1 (Top Level) R Script

Environment History

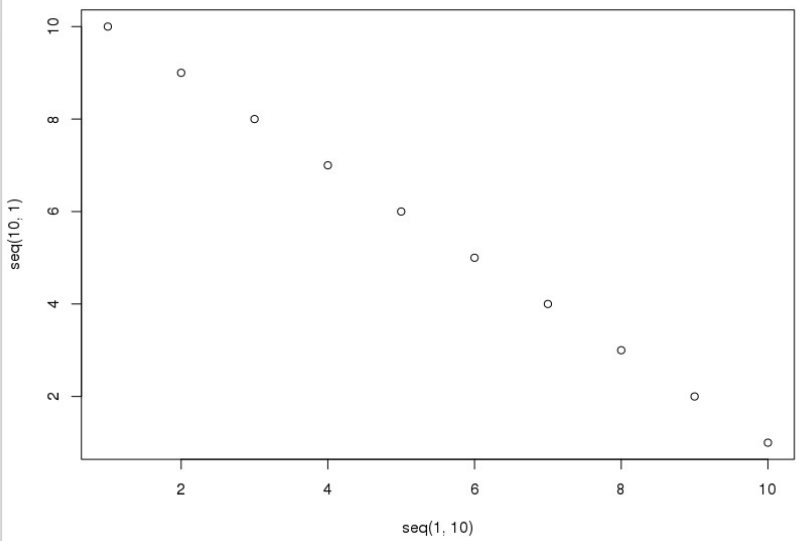
Import Dataset List

Global Environment

Data	
M	num [1:5, 1:6] 1 1 1 1 1 1 1 1 1 1 ...
Values	
e1	12
e2	12
e3	"six"
e4	6

Files Plots Packages Help Viewer

Zoom Export Publish



Console ~/

```
>
> 5 + 7
[1] 12
> e1 <- 5 + 7
>
> e1/2
[1] 6
>
> e2 <- 4 * 3
>
> e2 < 12
[1] FALSE
>
> e2 <= 12
[1] TRUE
>
> e3 <- 'six'
>
> e4 <- 6
>
> e3 == e4
[1] FALSE
>
```

RStudio

https://rstudio.age.mpg.de

File Edit Code View Plots Session Build Debug Profile Tools Help

FMetge

Addins

Project: (None)

first_steps.R | Untitled1 | M

Source on Save

Run Source

```
1  # this is a script
2
3  e1 <- 5 + 7
4
5  e1/2
6
7  e2 <- 4 * 3
8
9  # e2 < 12
10
11 e2 <= 14
12
13 e3 <- 'six'
14
15 e4 <- 6
16
17 e3 == e4
18
19
```

19:1 (Top Level) R Script

Environment History

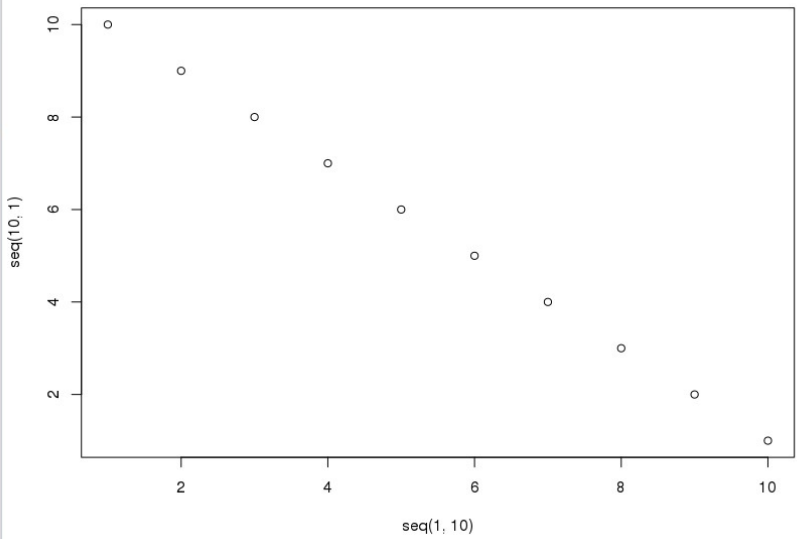
Import Dataset List

Global Environment

Data	
M	num [1:5, 1:6] 1 1 1 1 1 1 1 1 1 1 ...
Values	
e1	12
e2	12
e3	"six"
e4	6

Files Plots Packages Help Viewer

Zoom Export Publish



Console ~/

```
> e2 <- 4 * 3
>
> e2 < 12
[1] FALSE
>
> e2 <= 12
[1] TRUE
>
> e3 <- 'six'
>
> e4 <- 6
>
> e3 == e4
[1] FALSE
```

In the future, code will displayed this way

```
# this is code
> 3+7
[10]
```

Basic data types

- Numeric

```
> x <- 10.5 # or x <- 10
```

- Integer

```
> x <- as.integer(2)
```

- Complex

```
> x <- 1 + 2i
> sqrt(-1) # NaN
> sqrt(as.complex(-1)) # 0+1i
```

- Logical

```
> x <- TRUE; FALSE; T; F
```

- Character

```
> apple <- “Apple”
> x <- “6”
```

Useful functions

- Find out type of a variable

```
> class(x)
> typeof(x)
```

- Test if variable is of a certain type

```
> is.numeric(x)
> is.logical(x)
```

- Force a variable to be in a certain type

```
> as.numeric(x)
> as.character(x)
> as.integer(5.7)       # 5
> as.integer(“apple”)   # NA
> as.integer(TRUE)      # 1
```

More complex data structures (Vector)

A vector is a sequence of elements of the same basic type

```
> # numerical vector
> a <- c(2, 3, 4)

> # logical vector
> b <- c(TRUE, FALSE, TRUE)

> # character vector
> c <- c(“apple”, “six”)

> # mixed vector?
> d <- c(“six”, a, 5)

> length(d)
```

More complex data structures (Matrix)

A matrix is a collection of elements of the same data type arranged in a two-dimensional rectangular layout.

```
> # 3 columns, fill matrix column wise (default)
> A <- matrix(c(2,3,4,5,6,7), ncol = 3)

> # 2 rows, fill matrix row wise
> A <- matrix(c(2,3,4,5,6,7), nrow = 2, byrow = T)

> # alternative matrix generation
> B <- matrix(1, 2, 3)
> C <- matrix(NA, 3, 2)

> # transpose a matrix
> t(A)

> # concatenate two matrices row or column wise
> D <- cbind(A, B) # rbind(A, B)
```

More complex data structures (List)

A list is a generic vector containing objects of the same or different data types

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc", "dd")
> b = c(TRUE, FALSE, TRUE)

> # list without names
> x = list(n, s, b)

> # same list with names
> xn = list(Numbers = n, Strings = s, Boolean = b)

> # access list elements
> x[2] # vs. x[[2]]
> xn$Strings # vs. xn["Strings"] or xn[["Strings"]]
```

More complex data structures (Data Frame)

A data frame is list of vectors of equal length organized in rows and columns

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> b = c(TRUE, FALSE, TRUE)

> # data frame witout names
> df = data.frame(n, s, b)

> # data frame with column names
> names(df) <- c("Numbers", "Strings", "Boolean")
> dfn = data.frame(Numbers = n, Strings = s, Boolean = b)

> # access elements in a data frame
> dfn$Numbers # whole first column
> dfn[,1]     # whole first column
> dfn[2,3]    # second element of the third column
```

5min Break

Basic functions

- Sum

```
> M = matrix(c(5,6,7,8), 2)
> sum(M)       # 26
> rowSums(M)   # 12 14
> colSums(M)   # 11 15
```

- Mean

```
> mean(M)      # 6.5
> rowMeans(M) # 6 7
> colMeans(M) # 5.5 7.5
```

- Variation and standard deviation

```
> var(as.numeric(M))    # 1.67
> sd(M)                 # 1.29
```

Basic functions

- Median

```
> mean(c(5,6,7,89))     # 26.75
> median(c(5,6,7,89))   # 6.5
```

- Quantiles

```
> quantile(c(5,6,7,89), probs
   = c(.25, .5, .75))
#  25%   50%   75%
# 5.75  6.50 27.50
```

Basic functions

- Sequence

```
> seq(from = 1, to = 10, by = 1)
[1] 1 2 3 4 5 6 7 8 9 10
```

- Repeat

```
> rep(x = NA, times = 3)
[1] NA NA NA
> rep(c(0,1), 5)
[1] 0 1 0 1 0 1 0 1 0 1
```

- Random Numbers

```
> rnorm(n = 5, mean = 5, sd = 1)
[1] 4.62 5.46 6.58 4.60 5.42
> runif(n = 5, min = 1, max = 5)
[1] 1.46 1.07 3.55 4.96 3.28
```

Distribution	R function
Normal	rnorm(n, mean, sd)
Uniform	runif(n, min, max)
Binomial	rbinom(n, size, prob)
Beta	rbeta(n, shape1, shape2, ncp)
Exponential	rexp(n, rate)
Poisson	rpois(n, lambda)
Chi^2	rchisq(n, df, ncp)
Student's t	rt(n, df, ncp)
...	

How to get help (inside R)

- If you have a rough idea what the function might be called

```
> ??rowmean
```

- If you know what the function is called

```
> ?seq
```

All R help functions are structured the same way:

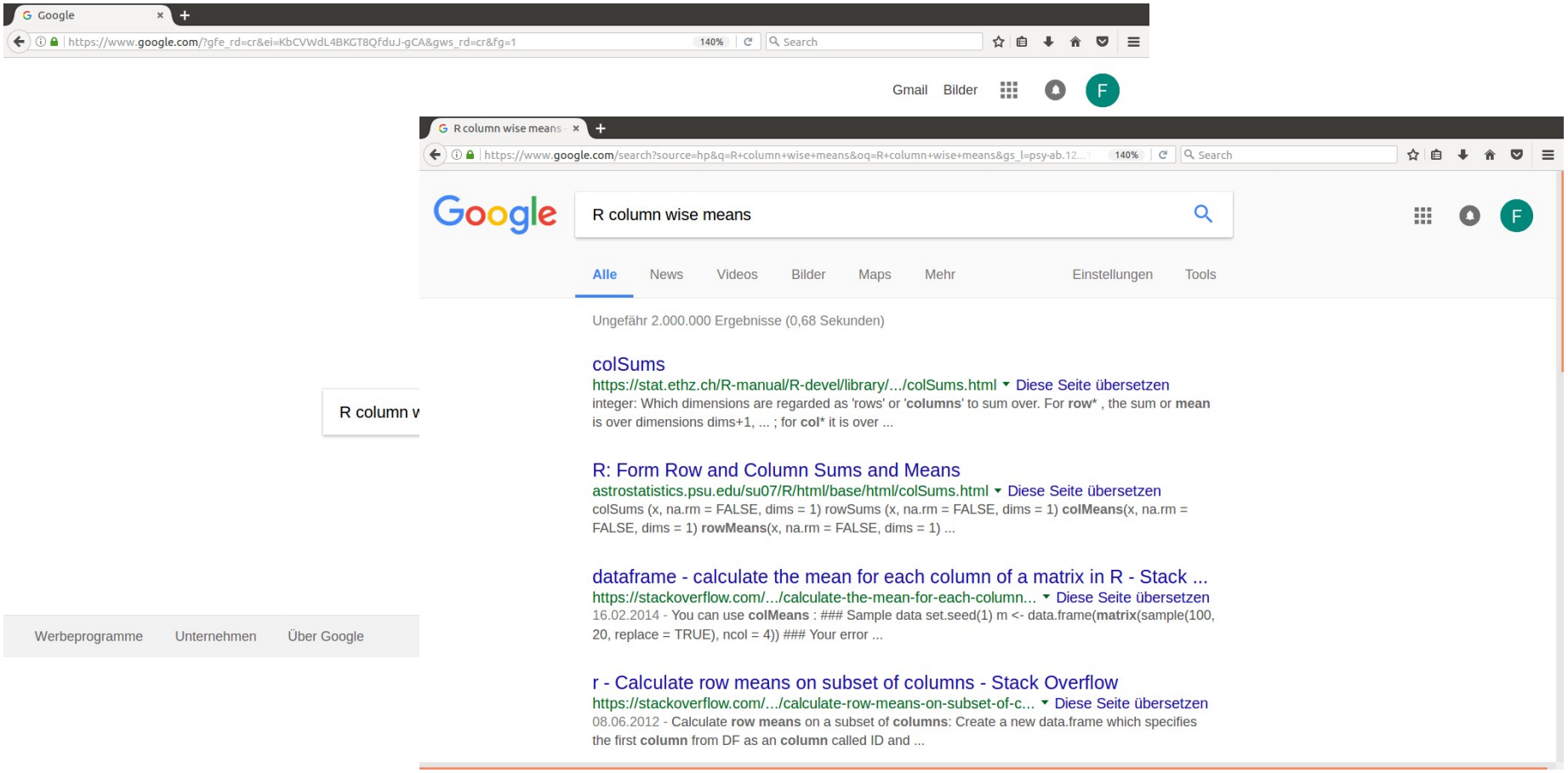
Description	A short overview of what the function intends to do
Usage	How to call this function and which arguments may be supplied. The order of arguments is meaningful
Arguments	A detailed description of the arguments that are passed to the function
Details	A more or less detailed description of the function
Value	The value which is returned by the function
References/See Also	Citation and related functions
Examples	Different examples on how to use the function

```
seq                     package:base                     R Documentation

Sequence Generation

Description:

     Generate regular sequences.  'seq' is a standard generic with a
     default method.  'seq.int' is a primitive which can be much faster
     but has a few restrictions.  'seq_along' and 'seq_len' are very
     fast primitives for two common cases.

Usage:

     seq(...)

     ## Default S3 method:
     seq(from = 1, to = 1, by = ((to - from)/(l
         length.out = NULL, along.with = NULL,

     seq.int(from, to, by, length.out, along.wi

     seq_along(along.with)
     seq_len(length.out)

Arguments:

     ...: arguments passed to or from methods.

from, to: the starting and (maximal) end values
          length '1' unless just 'from' is supp
          argument.

      by: number: increment of the sequence.

length.out: desired length of the sequence.  A
          which for 'seq' and 'seq.int' will be
          fractional.

along.with: take the length from the length of

Details:

     Numerical inputs should all be finite (tha
     'NaN' or 'NA').

     The interpretation of the unnamed argument
     is _not_ standard, and it is recommended always to name the
     arguments when programming.

     'seq' is generic, and only the default method is described here.
:
```

```
Help files with alias or concept or title matching 'rowmean' using
fuzzy matching:

base::colSums           Form Row and Column Sums and Means
  Aliases: rowMeans, .rowMeans
Matrix::colSums         Form Row and Column Sums and Means
  Aliases: rowMeans, rowMeans,diagonalMatrix-method,
    rowMeans,CsparseMatrix-method, rowMeans,TsparseMatrix-method,
    rowMeans,RsparseMatrix-method, rowMeans,dgCMatrix-method,
    rowMeans,igCMatrix-method, rowMeans,lgCMatrix-method,
    rowMeans,ngCMatrix-method, rowMeans,denseMatrix-method,
    rowMeans,ddenseMatrix-method
Matrix::dgeMatrix-class
                        Class "dgeMatrix" of Dense Numeric (S4 Class)
                        Matrices
  Aliases: rowMeans,dgeMatrix-method
Matrix::indMatrix-class
                        Index Matrices
  Aliases: rowMeans,indMatrix-method

Type '?PKG::FOO' to inspect entries 'PKG::FOO', or 'TYPE?PKG::FOO' for
entries like 'PKG::FOO-TYPE'.
~
~
~
~
~
(END)
```

TICS

How to get help (outside R)



stackoverflow.com

5min Break

Read in data

Table

```
> D = read.table(“mouse.tab”, as.is = T, header = F )
```

CSV

```
> D = read.csv(“mouse.csv”, as.is = T, header = T)
> # default separator “,”, decimals as “.”
```

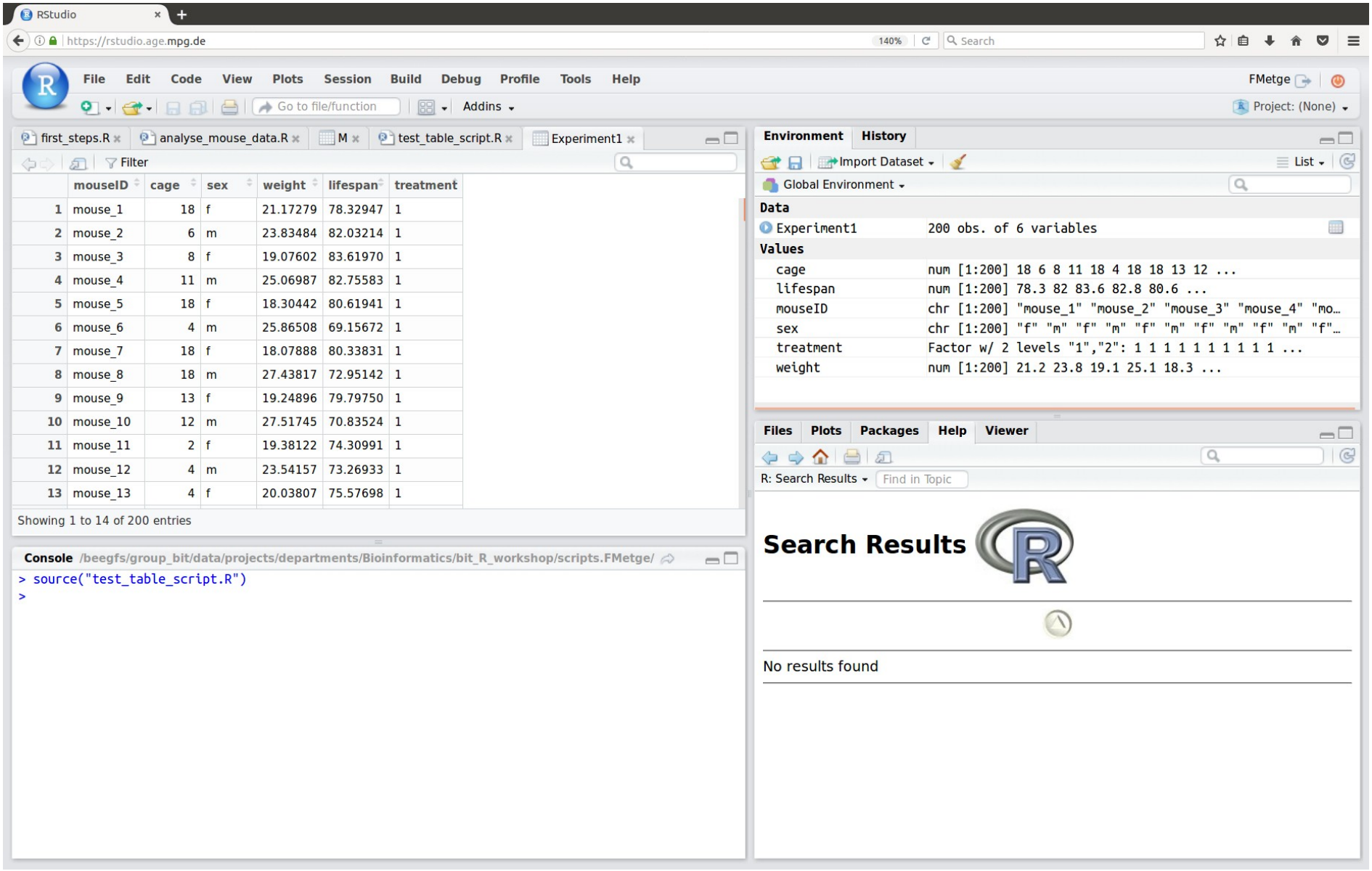
EXCEL

- Not in the standard library, but various packages are available

```
> library(xlsx)
> D = read.xlsx(“mouse.xlsx”, sheetName = “first”)
```

script

```
> source(“test_table_script.R”)
> ls()
```


RStudio
https://rstudio.age.mpg.de
140%
Search
File Edit Code View Plots Session Build Debug Profile Tools Help
FMetge
Go to file/function
Addins
Project: (None)
first_steps.R
analyse_mouse_data.R
M
test_table_script.R
Experiment1
Filter
mouseID cage sex weight lifespan treatment
1 mouse_1 18 f 21.17279 78.32947 1
2 mouse_2 6 m 23.83484 82.03214 1
3 mouse_3 8 f 19.07602 83.61970 1
4 mouse_4 11 m 25.06987 82.75583 1
5 mouse_5 18 f 18.30442 80.61941 1
6 mouse_6 4 m 25.86508 69.15672 1
7 mouse_7 18 f 18.07888 80.33831 1
8 mouse_8 18 m 27.43817 72.95142 1
9 mouse_9 13 f 19.24896 79.79750 1
10 mouse_10 12 m 27.51745 70.83524 1
11 mouse_11 2 f 19.38122 74.30991 1
12 mouse_12 4 m 23.54157 73.26933 1
13 mouse_13 4 f 20.03807 75.57698 1
Showing 1 to 14 of 200 entries
Console /beegfs/group_bit/data/projects/departments/Bioinformatics/bit_R_workshop/scripts.FMetge/
> source("test_table_script.R")
>
Environment History
Import Dataset
List
Global Environment
Data
Experiment1 200 obs. of 6 variables
Values
cage num [1:200] 18 6 8 11 18 4 18 18 13 12 ...
lifespan num [1:200] 78.3 82 83.6 82.8 80.6 ...
mouseID chr [1:200] "mouse_1" "mouse_2" "mouse_3" "mouse_4" "mo…
sex chr [1:200] "f" "m" "f" "m" "f" "m" "f" "m" "f" "m" "f"…
treatment Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
weight num [1:200] 21.2 23.8 19.1 25.1 18.3 ...
Files Plots Packages Help Viewer
R: Search Results
Find in Topic
Search Results
No results found

Basic table functions

Summary

```
> summary(Experiment1)
       mouseID         cage       sex         weight          lifespan     treatment
 mouse_1  :  1   Min.   : 1.00   f:100   Min.   :17.48   Min.   :60.46   1:100
 mouse_10 :  1   1st Qu.: 6.00   m:100   1st Qu.:21.41   1st Qu.:71.59   2:100
 mouse_100:  1   Median :10.00           Median :24.50   Median :76.17
 mouse_101:  1   Mean   :10.36           Mean   :24.53   Mean   :75.38
 mouse_102:  1   3rd Qu.:15.00           3rd Qu.:27.06   3rd Qu.:79.40
 mouse_103:  1   Max.   :19.00           Max.   :34.01   Max.   :93.09
 (Other)  :194
```

Table

```
> table(Experiment1$cage)

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
 4 11  7 11 10  8 13 14 10 16  8 11 13 13 13  8 11 13  6
```

Basic table functions

Correlation

```
> cor(Experiment1$cage, Experiment1$weight,
        method = 'spearman')
[1] 0.01028674
> cor(Experiment1$lifespan, Experiment1$weight)
[1] -0.6827497
```

Wilcoxon Test

```
> wilcox.test(Experiment1$lifespan ~ Experiment1$treatment)

    Wilcoxon rank sum test with continuity correction

data:  Experiment1$lifespan by Experiment1$treatment
W = 7296, p-value = 2.037e-08
alternative hypothesis: true location shift is not equal to 0
```

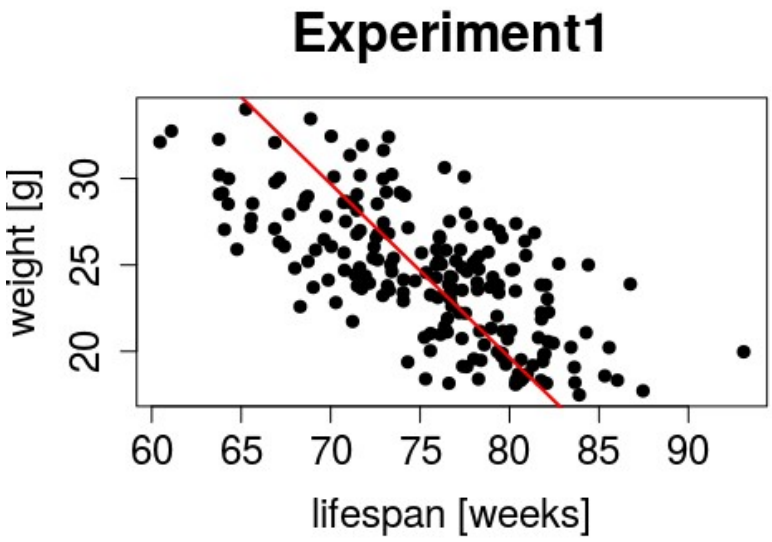
Basic table functions

Because both groups are normally distributed we can use a t-test

```
> t.test(Experiment1$lifespan ~ Experiment1$treatment)

    Welch Two Sample t-test

data:  Experiment1$lifespan by Experiment1$treatment
t = 6.0342, df = 193.78, p-value = 7.972e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.968538 5.851315
sample estimates:
mean in group 1 mean in group 2
       77.58123        73.17130
```

Basic Table functions

```
> plot(Experiment1$lifespan,
       Experiment1$weight)

> L = lm(Experiment1$lifespan
       ~ Experiment1$weight)

> L

Call:
lm(formula = Experiment1$lifespan
       ~ Experiment1$weight)

Coefficients:
       (Intercept)  Experiment1$weight
           100.028              -1.005

> abline(L)
```



Basic table manipulations

You have a second treatment you want to add to the bottom of the existing data frame (rbind)

```
> Experiment <- rbind(Experiment1, Treatment3)
```

You have the body size of each mouse and you want to add it to the left of the exisiting data frame (cbind)

```
> ExperimentS <- cbind(Experiment, size)
```

Basic table manipulations

Merge

```
> ExperimentP <- merge(ExperimentS, Exp2,
    by.x = 'mouseID', by.y = 'mouseID_pupps', all = T)
```

Subset

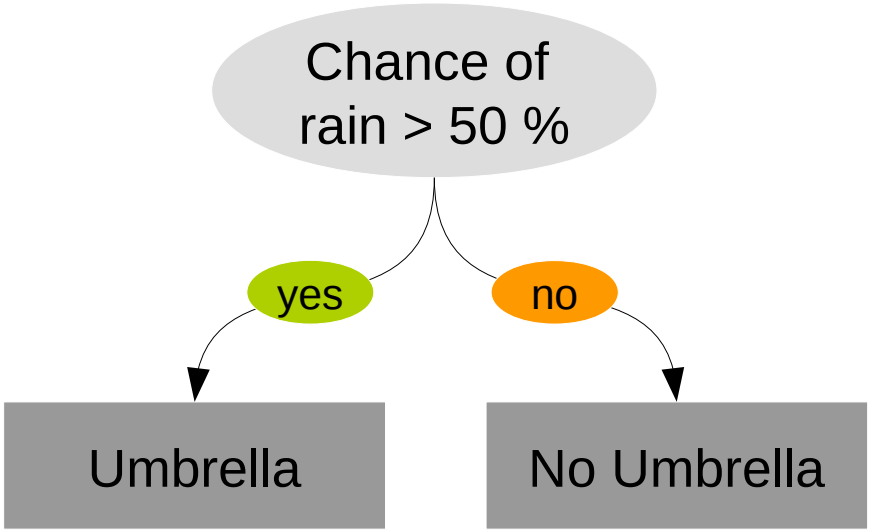
```
> Exp_cage10 <- subset(ExperimentP, cage == 10)
> Exp_cage10_11 <- subset(ExperimentP,
    cage %in% c(10, 11))[,1:8]
```

Names

```
> names(Exp_cage10_11)
[1] "mouseID" "cage" "sex.x" "weight" "lifespan" "treatment" "size" "num_pupps"

> names(Exp_cage10_11)[c(3, 9)] <- "sex"
```

5min Break

Conditional procedures




```
> chance_of_rain = 0.3

> if(chance_of_rain > 0.5){
    print(“take umbrella”)
} else {
    print(“you don’t need an
           umbrella”)
}

[1] “you don’t need an
      umbrella”
```

if(condition = True){

- do something

} else {

- do something different

}

if / else if / else

```
> x = 5
> y = NA
> z = 'six'

if(x == 5){
  print("x is equal to 5")
}

if(x < 5){
  print('x is smaller than 5')
}else{
  print('x is equal or larger than 5')
}
```

if / else if / else

```
> x = 5
> y = NA
> z = 'six'

> if(x < 5){
+  print("x is smaller than 5")
+ }else if(x == 5){
+   print("x is equal to 5")
+ }else{
+   print("x is larger than 5")
+ }
```

if / else if / else

```
> x = 5
> y = NA
> z = 'six'


> if(is.na(y)){
+  print("y is not available")
+ }

> if(!is.character(z)){
+  print("z is not a character")
+ } else {
+  print("z is a character")
+ }
```

if / else if / else

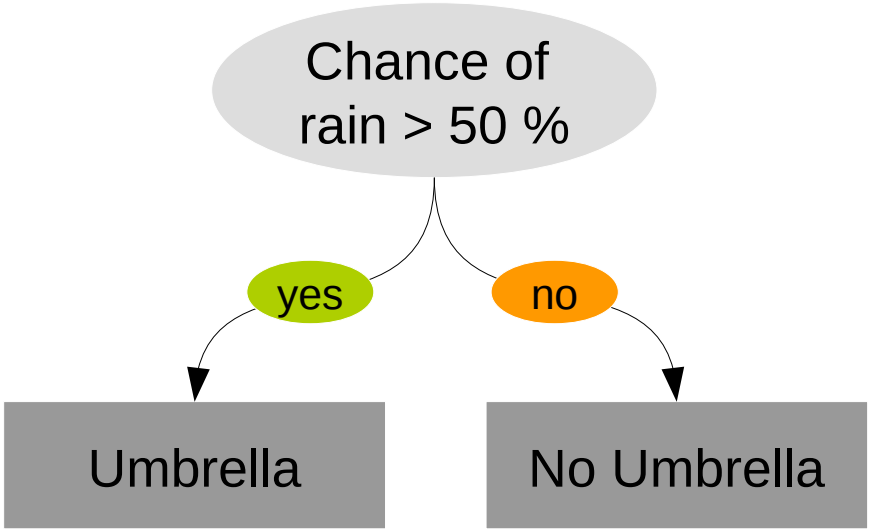
```
> a = c(1,2,3)
> b = c(1,2,3)
> c = c(1,3)


> if(a == b){
+   print(paste("a and b agree on", sum(a == b),
+   "elements", sep = ' '))
+ }
[1]    "a and b agree on 3 elements" # (warning)

> if(a == c){
+   print(paste("a and c agree on", sum(a == c),
+   "elements", sep = ' '))
+ }
[1]    ERROR
```

if / else if / else

```
> a = c(1,2,3)
> b = c(1,2,3)
> c = c(1,3)

> if(length(a) == length(c) & a == c){
+  print(paste("a and c agree on", sum(a == c),
+  "elements", sep = ' '))
+} else {
+  print("a and c cannot be compared, they differ in length")
+}

> if(length(a) == length(c) && a == c){
+  print(paste("a and c agree on", sum(a == c),
"elements", sep = ' '))
+ } else {
+  print("a and c cannot be compared, they differ in length")
+}
```

5min Break

Loops


Chance of rain > 50 %
yes
no
Umbrella
No Umbrella


Köln
Mittwoch, 15:00
Überwiegend sonnig
28 °C | °F
Niederschlag: 0%
Luftfeuchte: 39%
Wind: 16 km/h
Temperatur
Niederschlag
Wind
0% 0% 15% 23% 5% 6% 3%
16:00 19:00 22:00 01:00 04:00 07:00 10:00 13:00
Mi. 28° 15°
Do. 24° 12°
Fr. 26° 17°
Sa. 27° 16°
So. 24° 16°
Mo. 25° 16°
Di. 25° 14°
Mi. 22° 14°

Loops

Loops are useful if you want to repeat the same command over and over again

Different type of loops

While loop

- Repeat while condition is true

For loop

- Iterate over index
- No conditional statement

Most procedures can be written as for or while loops. For loops are almost always the saver choice

Day	1	2	3	4	5	6	7	8
Chance of rain [%]	0	0	15	23	5	6	3	0

For-loop

```
> for(i in something){
+ do something
+ }
```

Use if you have a fixed number of iterations

Will almost always finish

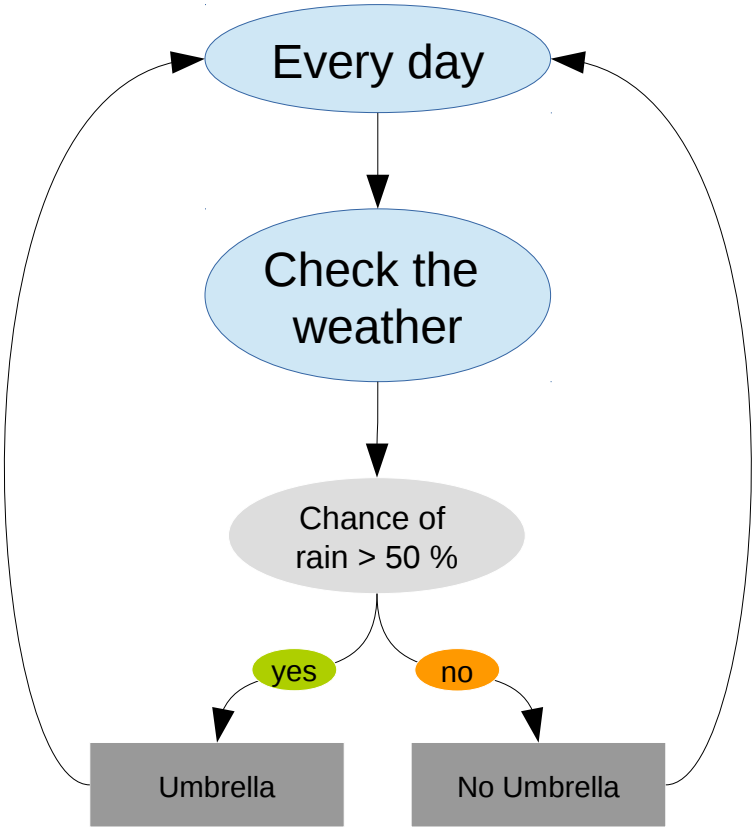
While-loop

```
> while(condition = TRUE){
+ do something
+ }
```

Use if you do not know how often you need to repeat something

Could run infinitely

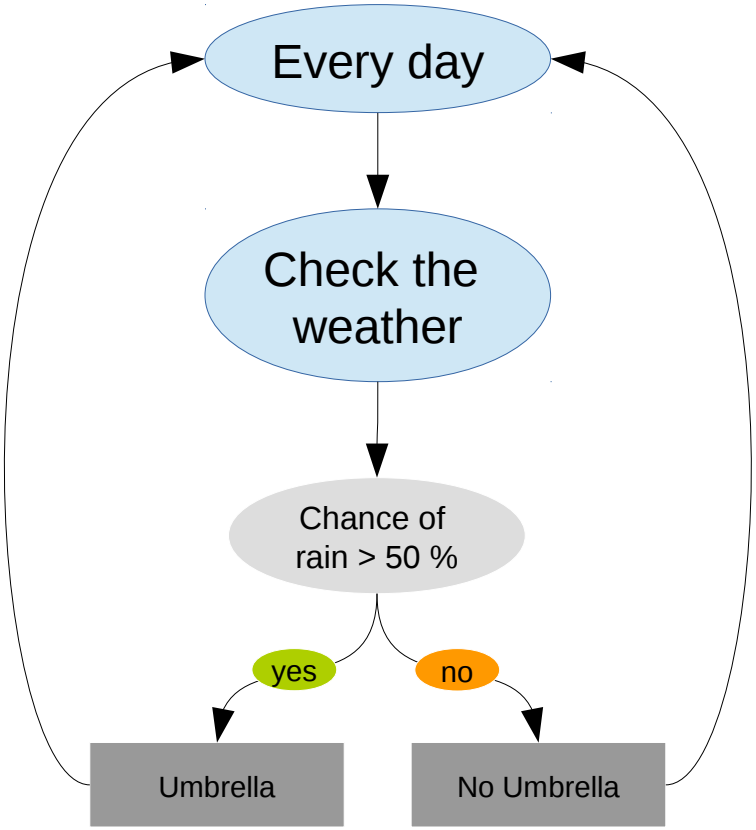
While - loops



day	1	2	3	4	5	6	7	8
check_ weather	0	0	15	23	5	6	3	0

```
> day = 1
> while(day < 9){
+ chance_of_rain =
+          check_weather(day)
+
+ if(chance_of_rain > 10){
+   print(“take umbrella”)
+ } else {
+   print(“you don’t need an
+          umbrella”)
+ }
+ day = day + 1
+ }

[1] “you don’t need an
       umbrella”

…
[8] “you don’t need an
       umbrella”
```

For - loops



day	1	2	3	4	5	6	7	8
check_ weather	0	0	15	23	5	6	3	0

```
>
> for(day in 1:8){
+ chance_of_rain =
+         check_weather(day)
+
+ if(chance_of_rain > 10){
+   print(“take umbrella”)
+ } else {
+   print(“you don’t need an
+         umbrella”)
+ }
+
+ }

[1] “you don’t need an
      umbrella”

…
[8] “you don’t need an
      umbrella”
```

More examples

for

```
> for(x in 2:4){
+ print(c(x, x * x))
+ }

[1] 2 4
[1] 3 9
[1]  4 16
```

while

```
> x = 1
> while(x < 4){
+ x <- x + 1
+ print(c(x,x * x))
+ }

[1] 2 4
[1] 3 9
[1]  4 16

> x = 1
> while(x < 4){
+ print(c(x,x * x))
+ x <- x + 1
+ }
```

Exit a loop

```
> for(x in 2:4){
+   if(x == 3){
+       break
+   } else {
+   print(x)
+   }
+ }

[1] 2
```

```
> x = 1
> while(x < 5){
+   x = x + 1
+   if(x == 3){
+       break
+   } else {
+   print(x)
+   }
+ }

[1] 2
```

Skip one iteration in a loop

```
> for(x in 2:4){
+   if(x == 3){
+       next
+   } else {
+   print(x)
+   }
+ }

[1] 2
[1] 4
[1] 5
```

```
> x = 1
> while(x < 5){
+   x = x + 1
+   if(x == 3){
+       next
+   } else {
+   print(x)
+   }
+ }

[1] 2
[1] 4
[1] 5
```

Example

You have several items you want to pack

Each Item has a weight

You are only allowed to pack 10kg
because you are flying Ryanair

Problem:

Add Items to your bag until you reached
the maximum weight

```
> items = c('shoes', 'shirt',
+   'pants', 'underwear',
+   'book', 'tooth brush',
+   'pillow', 'head phones',
+   'hair dryer')

> sizes = c(2.4, 2, 3.1, 1.5,
      1.1, 0.3, 0.8, 0.5, 1)

> bag = 0
```

Example

```
> bag = 0
> things_packed = character(0)
> for(i in 1:length(items)){
+   if(bag + sizes[i] > 10){
+     break
+   }
+   bag = bag + sizes[i]
+   things_packed =
+   c(things_packed, items[i])
+   print(c(bag, things_packed))
+ }

[1] "2.4" "shoes"
[1] "4.4" "shoes" "shirt"
[1] "7.5" "shoes" "shirt" "pants"
[1] "9" "shoes" "shirt" "pants" "underwear"
```

```
> i = 1
> bag = 0
> things_packed = character(0)
> while( bag + sizes[i] < 10){
+   bag = bag + sizes[i]
+   things_packed =
+   c(things_packed, items[i])
+   i = i + 1
+   print(c(bag, things_packed))
+ }

[1] "2.4" "shoes"
[1] "4.4" "shoes" "shirt"
[1] "7.5" "shoes" "shirt" "pants"
[1] "9" "shoes" "shirt" "pants" "underwear"
```

Example improved

```
> bag = 0
> things_packed = character(0)
> for(i in 1:length(items)){
+   if(bag + sizes[i] > 10){
+     next
+   }
+   bag = bag + sizes[i]
+   things_packed =
+   c(things_packed, items[i])
+   print(c(bag, things_packed))
+ }


[1] "2.4" "shoes"
[1] "4.4" "shoes" "shirt"
[1] "7.5" "shoes" "shirt" "pants"
[1] "9" "shoes" "shirt" "pants" "underwear"
[1] "9.3" "shoes" "shirt" "pants" "underwear" "tooth brush"
[1] "9.8" "shoes" "shirt" "pants" "underwear" "tooth brush"
       "head phones"
```

```
> i = 1
> bag = 0
> things_packed = character(0)
> while( bag + sizes[i] < 10){
+   if(bag + sizes[i] > 10){
+     next
+   }
+   bag = bag + sizes[i]
+   things_packed =
+   c(things_packed, items[i])
+   i = i + 1
+   print(c(bag, things_packed))
+ }

[1] "2.4" "shoes"
[1] "4.4" "shoes" "shirt"
[1] "7.5" "shoes" "shirt" "pants"
[1] "9" "shoes" "shirt" "pants" "underwear"

# infinite - BAD !!!!!
```

Example while loop corrected

```
> i = 1
> bag = 0
> things_packed = character(0)
> while( bag + sizes[i] < 10){
+   if(bag + sizes[i] > 10){
+     i = i + 1
+     next
+   }
+   bag = bag + sizes[i]
+   things_packed =
+   c(things_packed, items[i])
+   i = i + 1
+   print(c(bag, things_packed))
+ }


[1] "2.4" "shoes"
[1] "4.4" "shoes" "shirt"
[1] "7.5" "shoes" "shirt" "pants"
[1] "9" "shoes" "shirt" "pants" "underwear"
[1] "9.3" "shoes" "shirt" "pants" "underwear" "tooth brush"
[1] "9.8" "shoes" "shirt" "pants" "underwear" "tooth brush"
      "head phones"

# exits on an ERROR (ran out of the array)
```

```
> i = 1
> bag = 0
> things_packed = character(0)
> while( bag < 10 &
+           i <= length(sizes)){
+   if(bag + sizes[i] > 10){
+     i = i + 1
+     next
+   }
+   bag = bag + sizes[i]
+   things_packed =
+   c(things_packed, items[i])
+   i = i + 1
+   print(c(bag, things_packed))
+ }


[1] "2.4" "shoes"
[1] "4.4" "shoes" "shirt"
[1] "7.5" "shoes" "shirt" "pants"
[1] "9" "shoes" "shirt" "pants" "underwear"
[1] "9.3" "shoes" "shirt" "pants" "underwear" "tooth brush"
[1] "9.8" "shoes" "shirt" "pants" "underwear" "tooth brush"
      "head phones"
```

Loops for real application

gene	R1	R2	R3	R4	R5	R6	Diff?
AA	10	10	6	9	7	11	
BB	11	12	11	11	13	9	
CC	10	12	9	11	9	10	
DD	11	6	12	28	20	17	
EE	12	5	9	10	10	11	

You want to know which genes are differentially expressed

Perform the T-test **for** each gene

```
> gene[1,9] = t.test(gene[1,2:4], gene[1,5:7])$p.value
> gene[2,9] = t.test(gene[2,2:4], gene[2,5:7])$p.value
> gene[3,9] = t.test(gene[3,2:4], gene[3,5:7])$p.value
> gene[4,9] = t.test(gene[4,2:4], gene[4,5:7])$p.value
…
> gene[8,9] = t.test(gene[8,2:4], gene[8,5:7])$p.value
> gene[9,9] = t.test(gene[9,2:4], gene[9,5:7])$p.value
```

Loops for real application

```
> gene<- read.csv("gene_exp1.csv")
> pvals = numeric(0)
> for(i in 1:9){
+   pvals[i] <- t.test(gene[i,2:4], gene[i,5:7])$p.value
+ }
> gene$Diff <- pvals < 0.05
> gene

  gene R1 R2 R3 R4 R5 R6  Diff
1   AA 10 10  6  9  7 11 FALSE
2   BB 11 12 11 11 13  9 FALSE
3   CC 10 12  9 11  9 10 FALSE
4   DD 11  6 12 28 20 17  TRUE
5   EE 12  5  9 10 10 11 FALSE
6   FF 11  8 12  1  3  4  TRUE
7   GG 23 12 11 11  9 11 FALSE
8   HH  6  7  8  8 11  8 FALSE
9   II 11  8 13 21 18 20  TRUE
```

Exercise

Load the table generated in “create_gene_expression_table.R” using source()

Return all genes which are significant differentially expressed

Use the t.test()$p.value and a for loop

END OF DAY 1