Differential analysis of quantitative proteomics data using R

Karin Schork and Michael Turewicz

Medical Bioinformatics
Medizinisches Proteom-Center
Ruhr-University Bochum
karin.schork@rub.de michael.turewicz@rub.de

Organisation

10:00	-	10:30	Welcome and de.NBI overview
10:30	-	11:15	First steps in R
11:15	-	12:00	Hands-on (and sample solution)
12:00	-	12:45	Theory: statistical inference
12:45	-	13:30	Hands-on (and sample solution)
13:30	-	14:15	Lunch break
14:15	-	15:00	Presentation: data import, conducting tests, plots
15:00	-	17:30	Hands-on (and sample solution)
17:30	-	18:00	Discussion and questions

About **R**

- '**R** is a free software environment for statistical computing and graphics'
- '**R** is available as Free Software under the terms of the Free Software Foundations GNU General Public License'
- R is an alternative to S and S-Plus (Insightful Corporation)
- active development of **R**

Where to get **R**

All things for **R** can be found here

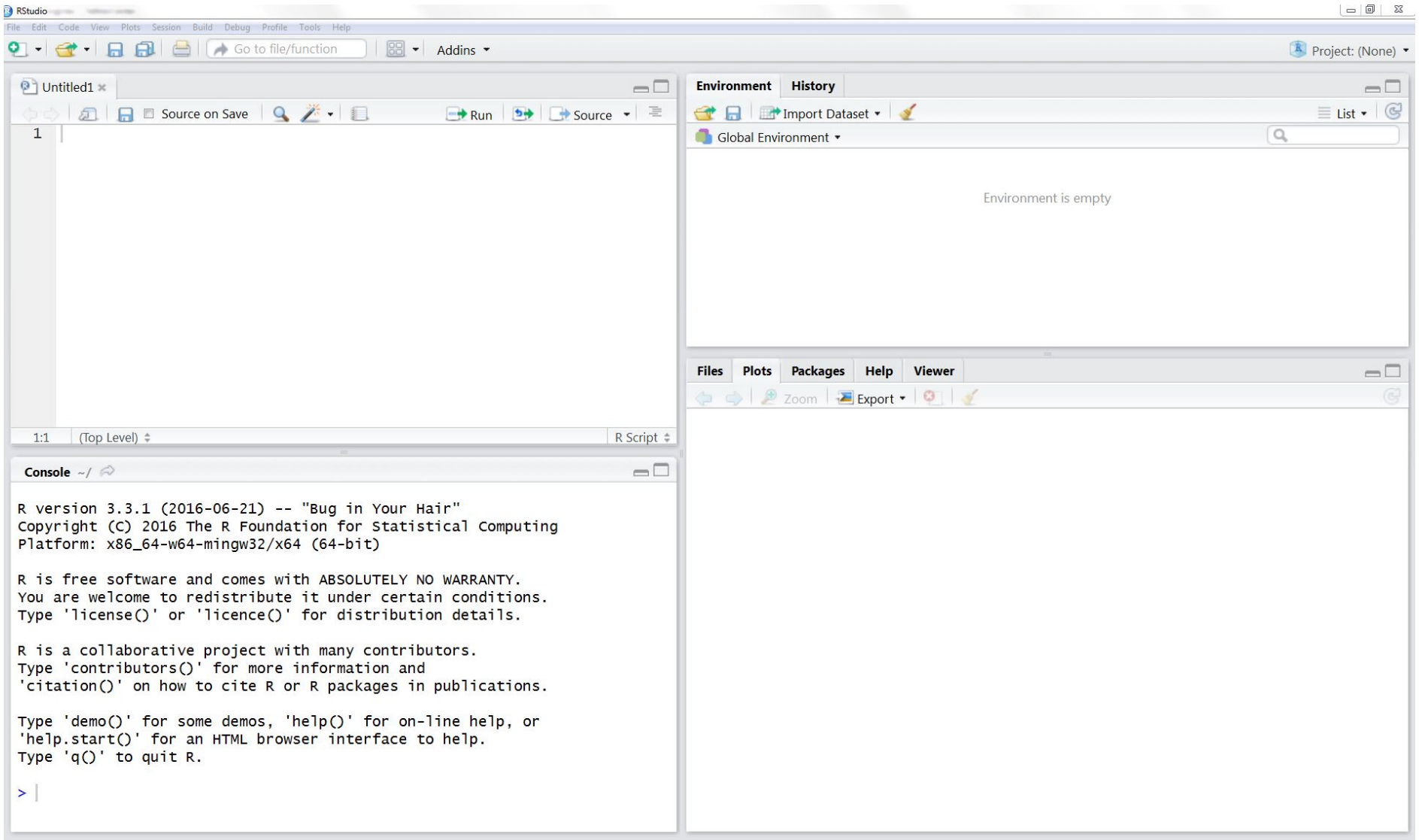
- `http://www.R-Project.org`
- `http://CRAN.R-Project.org`

Among others you find

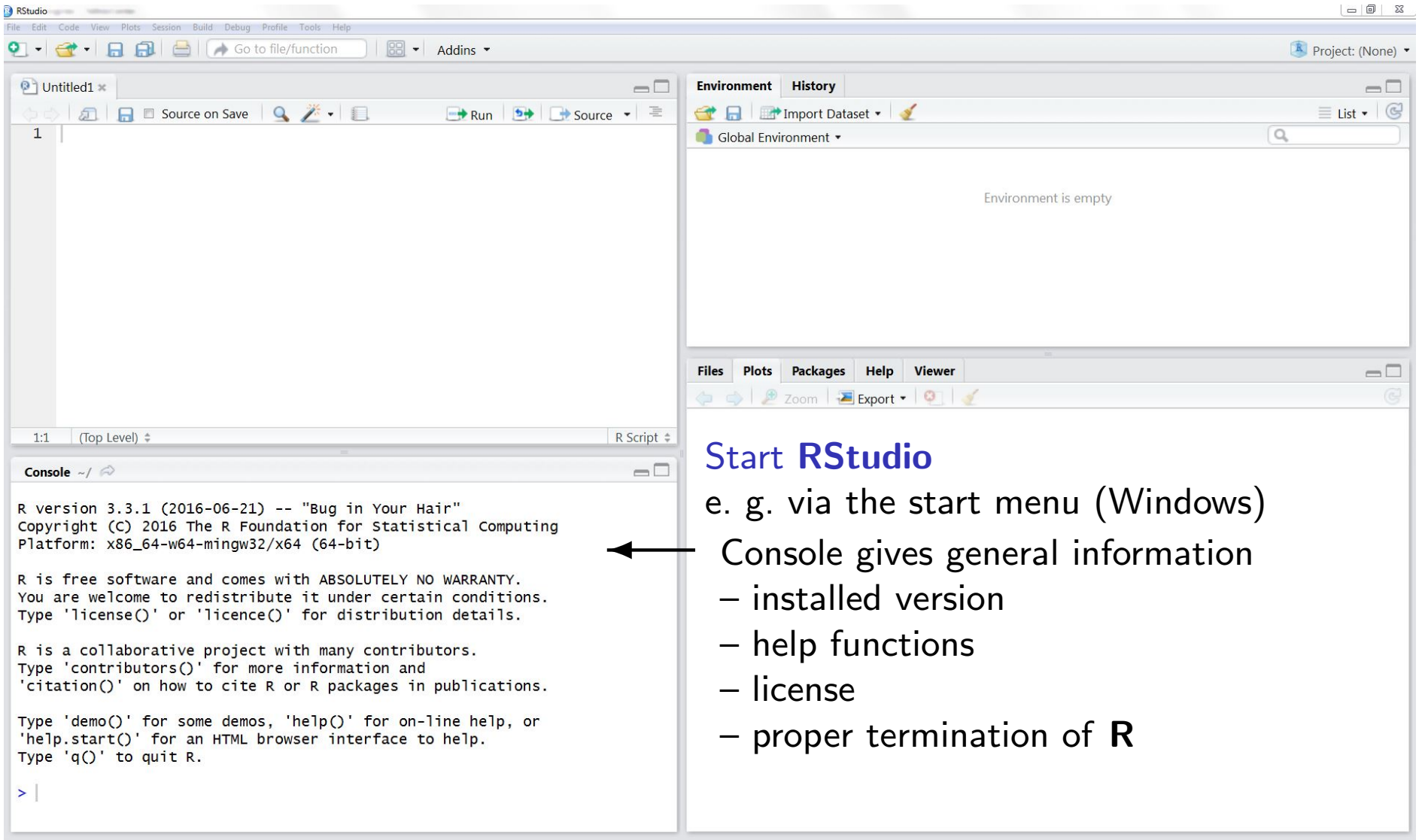
- **R** sources and binaries for different operating systems,
- **R** packages with diverse (statistical) methods,
- online manuals, FAQs, documentation and
- support (mailing list)

Live demo:
First steps in R

Getting started: RStudio



Getting started: RStudio



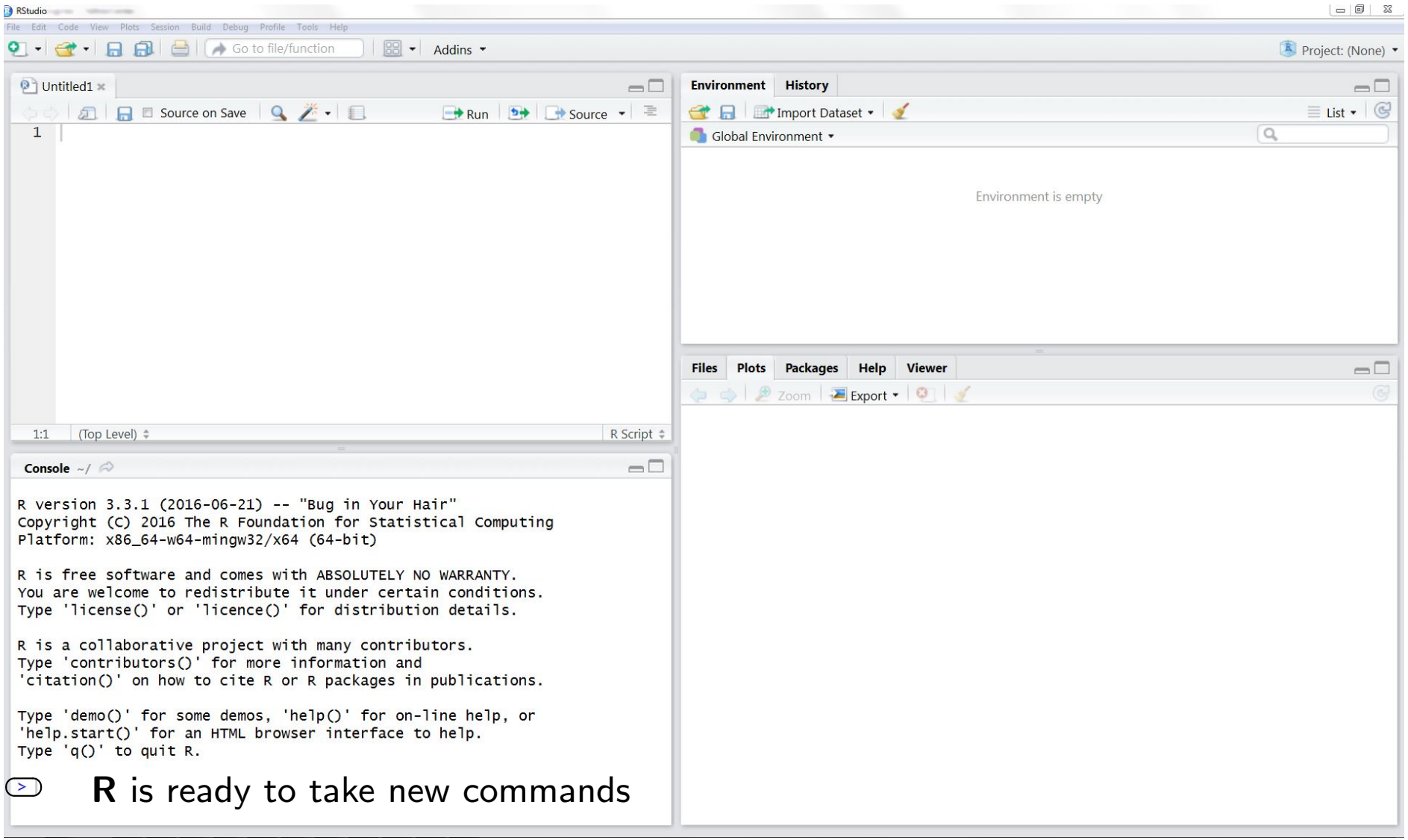
Start **RStudio**

e. g. via the start menu (Windows)

Console gives general information

- installed version
- help functions
- license
- proper termination of **R**

Getting started: RStudio



> **R** is ready to take new commands

Editors

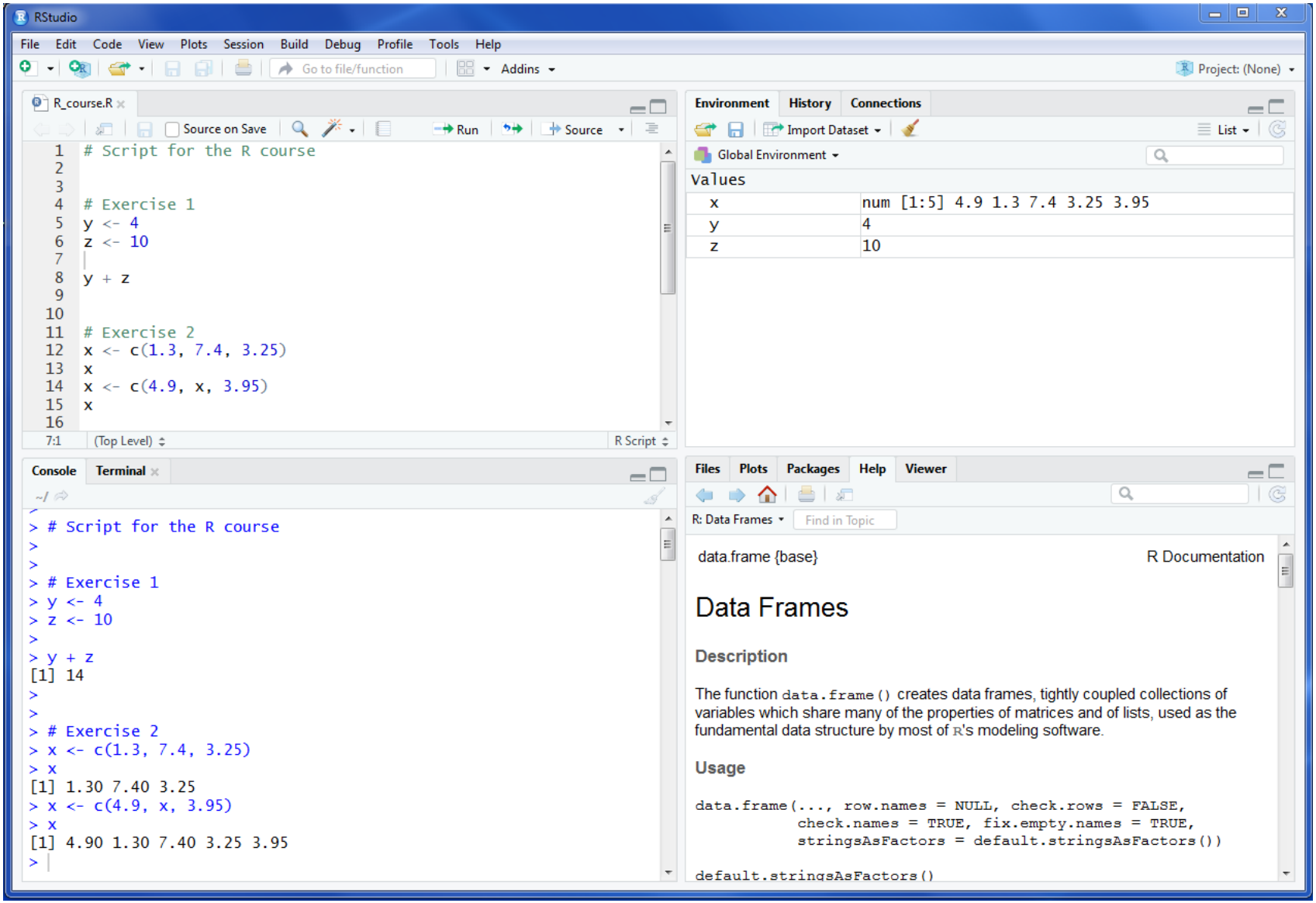
Especially for more complex analyses, it is advisable to use an editor to prepare the code before pasting it to **R**.

- transparency
- reproducibility
- transferability to other data sets

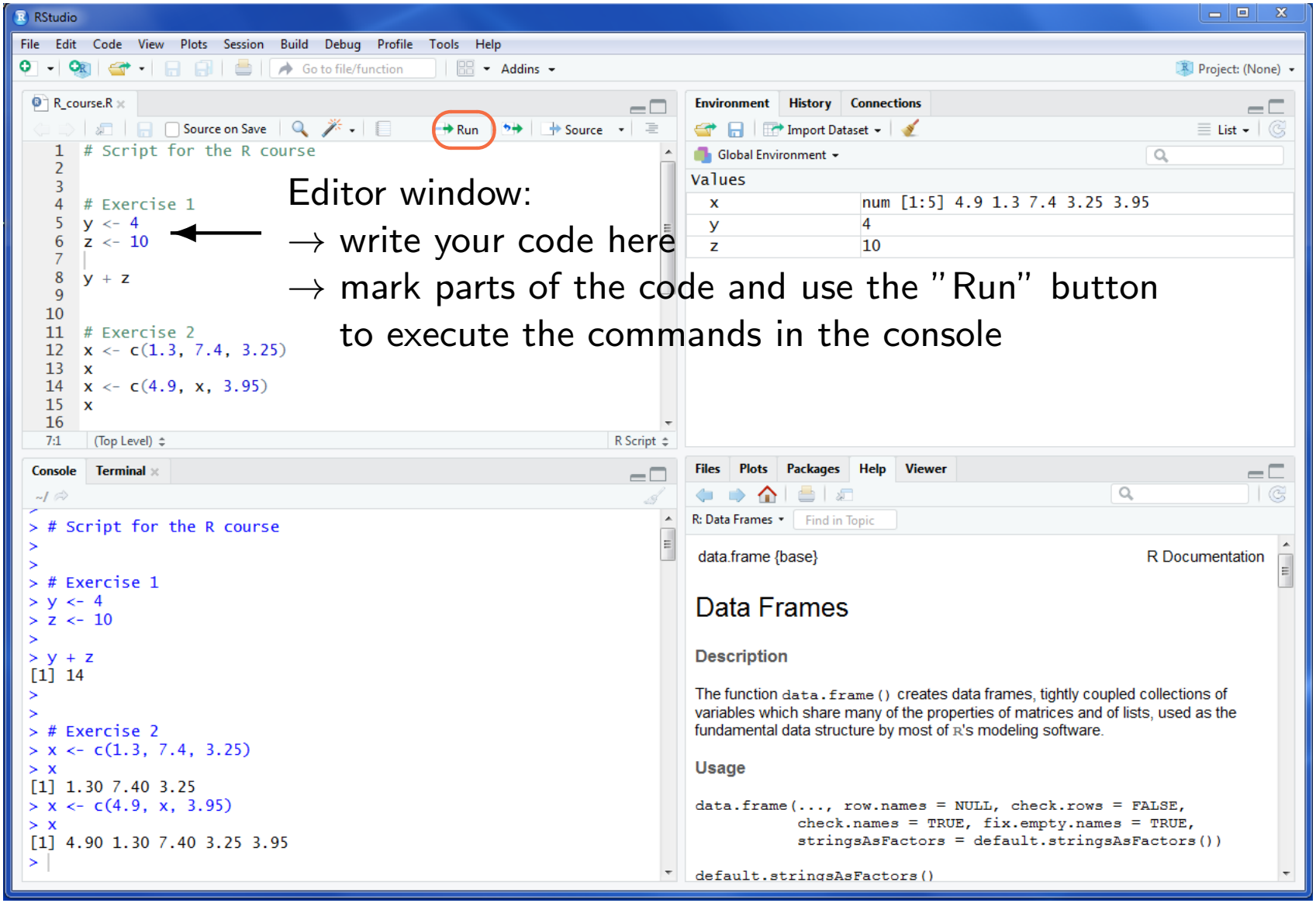
Some editors are specifically tailored to **R**, e. g.,

- Tinn-R (`http://sourceforge.net/projects/tinn-r`)
- RStudio (`https://www.rstudio.com/`)

Editors: RStudio



Editors: RStudio



R as pocket calculator

```
1 + 3
6 / 2      # This is a comment (after '#')
7 - 3 * 2
0 / Inf
0 / 0      # not defined -> NaN (not-a-number)
1.5 + 2.7
sqrt(9)
```

- multiplication and division are calculated before addition and subtraction ('Punkt-vor-Strich')
- plus (+) at beginning of line (instead of >): current command is not complete

Assignments

Save results in objects

```
a.sum <- 3 + 5      # assignment

a_number <- 4

a.sum + a_number

a.sum / 2

difference <- a.sum - a_number
```

names of objects in **R**

- may contain letters, numbers, period '.' and underscore '_'
- have to start with a letter
- are case-sensitive

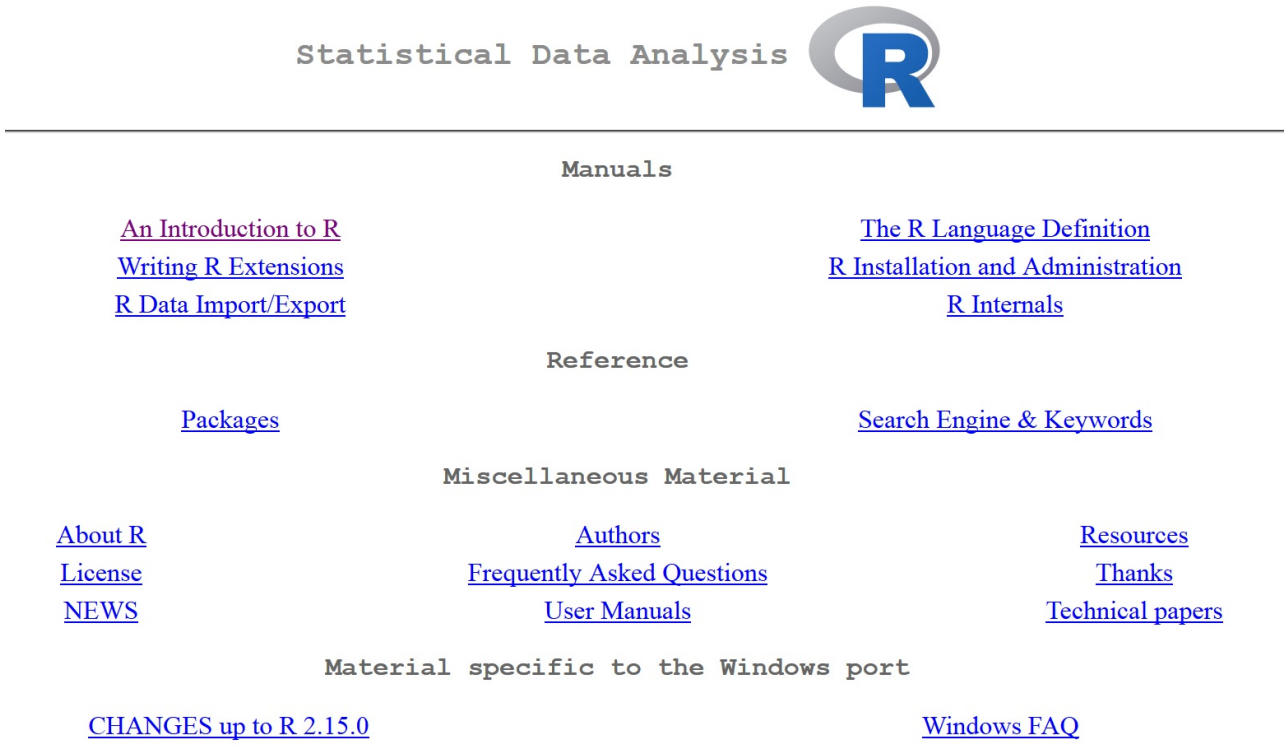
Listing and removing objects

R saves objects in the so-called workspace.

`ls()`	list all objects in the workspace
`rm(<name of object>)`	remove single object permanently
`rm(list = ls())`	remove entire workspace

End of session (close **R** window or type `q()`):
Question to save current workspace (including all objects)

Help



`help.start()` — open browser, start help system

`help.search(<keyword>)`, `??<keyword>`, e. g. `??"t-test"` — show all help pages with topic <keyword>

`help(<function>)`, `?<function>`, e. g. `?t.test` — help on a function (exact name)

Help pages

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.
...	

Help pages

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.
...	

Sections of a help page (functions)

Description	short(est) description
Usage	e. g. order of arguments
Arguments	description of parameters
Details	find most important information here
Value	output
See Also	similar functions
Examples	executable code

- Always have a look at help pages for functions you use
- 'Using R-help is a skill, and like any other skill it requires practice'

http://isites.harvard.edu/fs/docs/icb.topic662693.files/Using_R_and_Zelig/R-help.version1.pdf

Basic operators, functions & specific values

`+, -, *, /, ^`	basic arithmetic operators
`max(), min()`	extreme values
`abs()`	absolute value
`sqrt()`	square root
`round(), floor(), ceiling()`	round (commercially, down, up)
`sin(), cos(), tan()`	trigonometrical functions
`log(), log10(), log2(), exp()`	logarithms
`sum(), prod()`	sum, product
`length()`	length of a vector (number of elements)
`Inf, -Inf`	infinity
`NaN`	not defined (Not a Number)
`NA`	missing values (Not Available)
`NULL`	'the null object in R', empty set

R - all vectors

- **R** is a vector-based language.
- A vector gathers data of a specific type in an ordered manner.
- Data types in **R**:

`logical`	`TRUE` or `FALSE`
`numeric [integer]`	integers
`numeric [double]`	real number (double precision)
`character`	letters and character strings

Vectors

The function `c()` (combine/concatenate) creates vectors.

```
x <- c(1.3, 7.4, 3.25)
x <- c(4.9, x, 3.95)

y <- c(TRUE, FALSE)                          # logical
z <- c("Alpha", "Bravo", "Charlie")    # character
```

All arguments are coerced to a common type.

```
c(4, 3, y)                  # -> 4 3 1 0
c("Alpha", 3, TRUE)    # -> "Alpha" "3" "TRUE"
```

more detailed: highest type in following hierachy:

`NULL` < `raw` < `logical` < `integer` < `double` < `complex` < `character` < `list` < `expression`

Sequences and repetitions

```
1:10
x <- seq(1, 10, by=2)
rep(1, 10)
rep(x, 3)
rep(5:9, each=2)
```

sequences:
```
seq(start, end, by = increment)
seq(along = object)
start:end
```

repetitions:
```
rep(object, times to repeat)
```
(repeat whole vector)
```
rep(object, each = times to repeat)
```
(repeat each element of the vector)

Matrices

To collect tables of the same data type use `matrix()`.

Arguments of `matrix()`

`data`	vector of data, sorted by columns or rows
`nrow, ncol`	number of rows and columns, resp.
`byrow`	if `TRUE`, matrix is filled by rows, else by column
`dimnames`	`list(rows, columns)`

Usage

```
matrix(data = NA, nrow = 1, ncol = 1,
       byrow = FALSE, dimnames = NULL)
```

```
X <- matrix(c(4, 2, 5, 7, 1, 6), nrow = 3)
```

```
Y <- matrix(c(3, 5, 6, 2, 4, 1), ncol = 3, byrow = TRUE)
```

Matrices - Argument byrow example

```
matrix(1:9, ncol = 3, byrow = TRUE)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

```
matrix(1:9, ncol = 3, byrow = FALSE)
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Matrices - data.frame

For data sets consisting of different types of variables use `data.frame()`:

```
Z <- data.frame("Gender" = c("f","f","m","f","m"),
                "Age"    = c(10,9,10,8,9),
                "Height" = c(120,118,104,109,111))
```

Gender	Age	Height	
f	10	120	← row
f	9	118	
m	10	104	
f	8	109	
m	9	111	
	↑ column		

Logic and related functions

Data type logical allows assessment of logical statements.

comparisons

```
==, !=, >, <, >=, <=
```

constants

```
TRUE, T, FALSE, F
```

Don't use `T` or `F` to name objects!

operators

```
!                                   logical negation, NOT
&&, ||                              AND, OR
&, |                                AND, OR (for vectors)
(4 <= 1) && (6 == (3 * 2))
c(TRUE, TRUE) & c(FALSE, TRUE)
FALSE || TRUE
FALSE + TRUE + TRUE                 # TRUE = 1, FALSE = 0
```

functions

`any()`, `all()`, `which()`, `sum()`

Indexing vectors

In order to navigate through vectors, matrices or data frames (indexing) use `[]`:

```
x[3]                # single entry
x[1:3]              # set of entries
x[c(1,3,12)]

x[-1]               # exclude single entry via '-'
x[-(1:3)]           # exclude set of entries
x[-c(1,3,12)]

x[x<15]             # entries meeting a logical condition
index <- x>=16
x[index]
```

Indexing and handling of data frames and matrices

Y	1	2	...	C
1	1,1	1,2	...	1,C
2	2,1	2,2		
⋮	⋮			
R	R,1	R,2	...	R,C

Indexing

```
Y[2,1]                  # 2nd row, 1st column
Y[1,]                   # complete 1st row
Y[1:2,c(1,3)]
Z[,"Gender"]            # selection by column name
Z$Gender
```

General handling

`dimnames()`	get/set list of row and column names
`rownames(), colnames()`	get/set row/column names
`ncol(), nrow()`	number of columns/rows
`t()`	transpose (switch rows with columns)
`head(), tail()`	show first/last lines of a matrix

Indexing and handling of data frames and matrices

Y	1	2	...	C
1	1,1	1,2	...	1,C
2	2,1	2,2		
⋮	⋮			
R	R,1	R,2	...	R,C

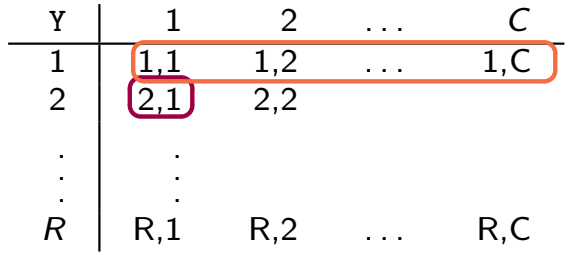
Indexing

```
Y[2,1]                  # 2nd row, 1st column
Y[1,]                   # complete 1st row
Y[1:2,c(1,3)]
Z[,"Gender"]            # selection by column name
Z$Gender
```

General handling

`dimnames()`	get/set list of row and column names
`rownames(), colnames()`	get/set row/column names
`ncol(), nrow()`	number of columns/rows
`t()`	transpose (switch rows with columns)
`head(), tail()`	show first/last lines of a matrix

Indexing and handling of data frames and matrices



Y	1	2	...	C
1	1,1	1,2	...	1,C
2	2,1	2,2		
⋮	⋮			
R	R,1	R,2	...	R,C

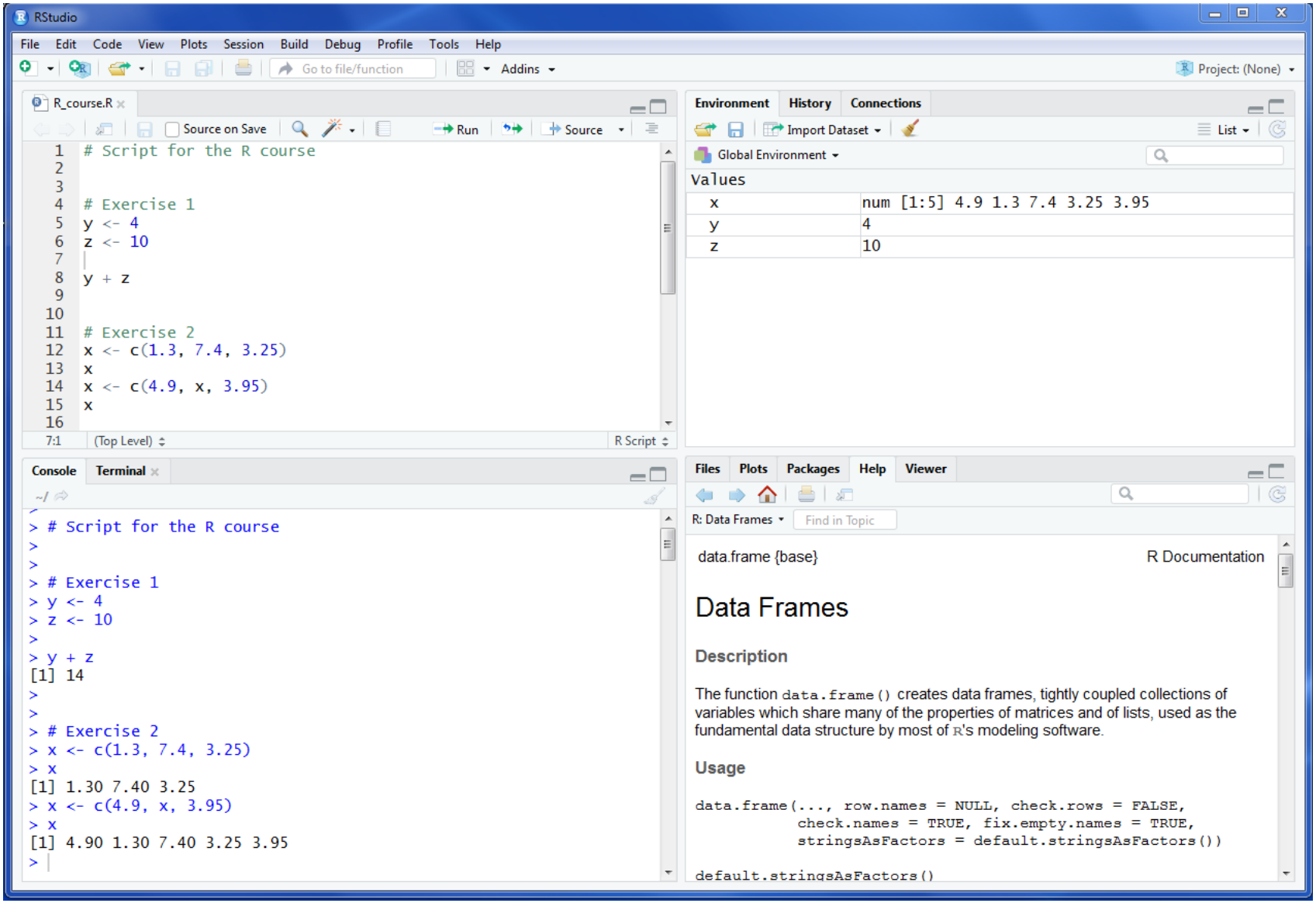
Indexing

```
Y[2,1]                  # 2nd row, 1st column
Y[1,]                   # complete 1st row
Y[1:2,c(1,3)]
Z[,"Gender"]            # selection by column name
Z$Gender
```

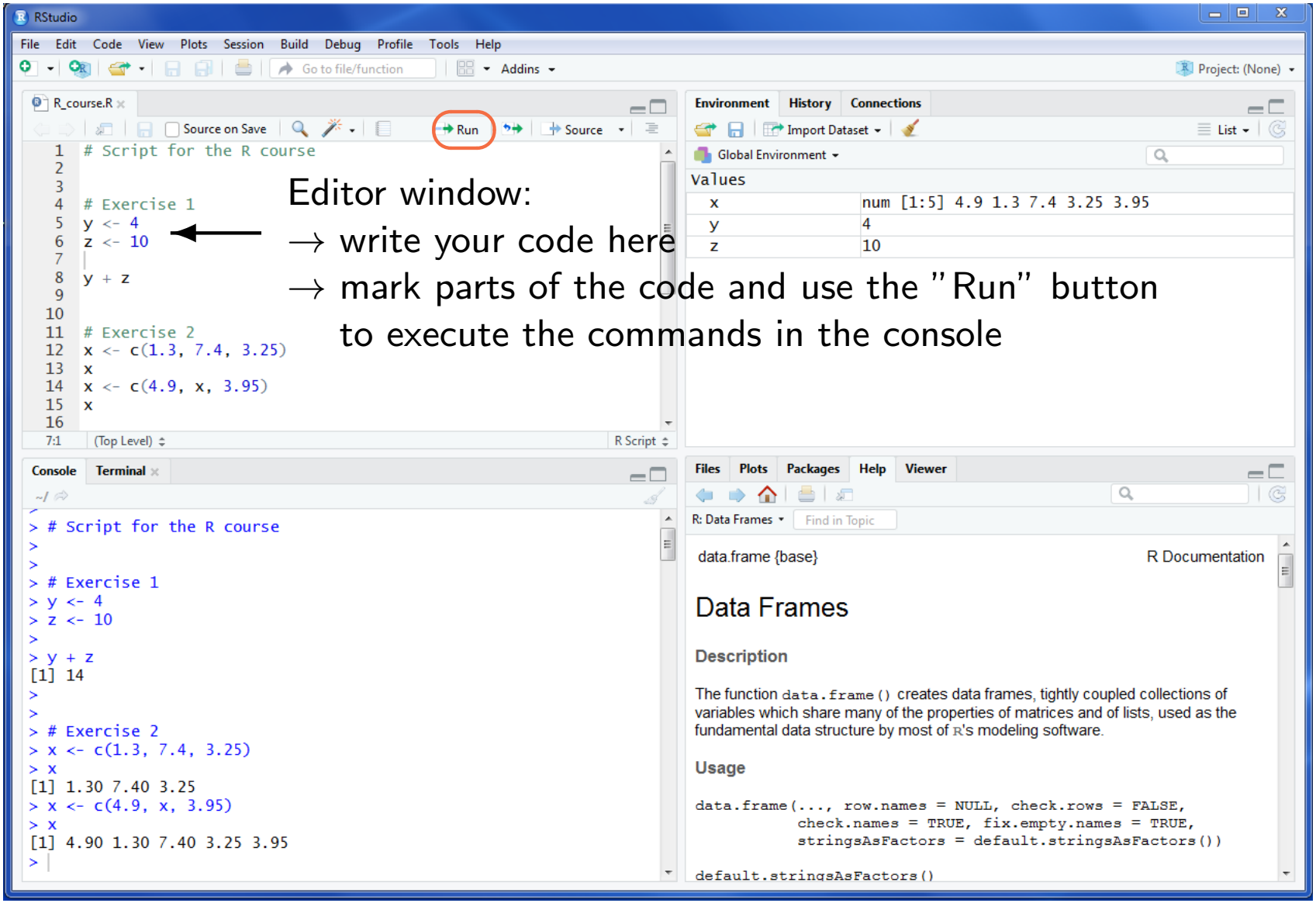
General handling

`dimnames()`	get/set list of row and column names
`rownames(), colnames()`	get/set row/column names
`ncol(), nrow()`	number of columns/rows
`t()`	transpose (switch rows with columns)
`head(), tail()`	show first/last lines of a matrix

Editors: RStudio



Editors: RStudio



EXERCISE

Theory:
Statistical inference

Construction of test problem: Example t-test

- Test problem: H_0 vs. H_1
  H_0, null hypothesis
  H_1, alternative hypothesis
- t-test is commonly used to assess differences in location
- Assumption of underlying normality (amongst others)
- Variations of the t-test:
  - **two samples** / one sample
  - **independent** / dependent groups (**unpaired**/paired design)
  - **unequal** / equal variances
  - **two-sided** / one-sided
- H_0: No difference in true means of the compared samples X and Y.
  $\Leftrightarrow H_0 : \mu_X = \mu_Y \quad \Leftrightarrow \quad H_0 : \delta = \mu_X - \mu_Y = 0$
- $H_1 : \mu_X \neq \mu_Y$

Error types

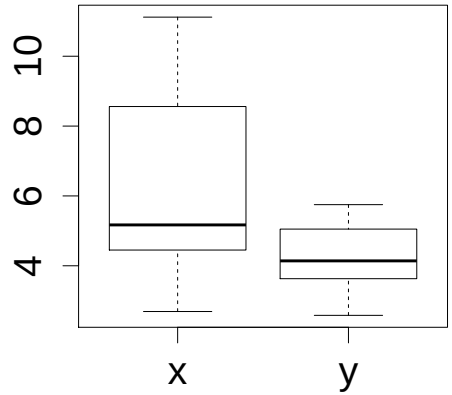
Types of errors when trying to infer from a (limited) sample to the whole population:

	Test decision	
Truth	do not reject H_0	reject H_0
H_0 true	✓	type I error (α)
H_0 false $= H_1$ true	type II error (β)	✓ ($1-\beta$)

- One cannot control both error types simultaneously.
- Each statistical test controls the type I error.
  Type I error is made with probability α.
- Type II error is affected by sample size.
  Type II error is made with probability β.
  Probability $1-\beta$ is called power.

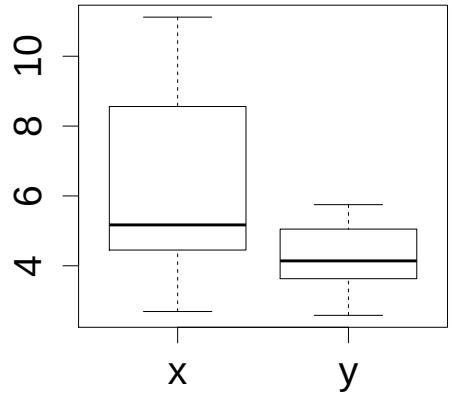
t-test in **R**

```
x <- c(  4.88,  8.56, 2.69, 4.89, 5.45,
        11.12, 10.43, 5.85, 4.31, 4.45)
y <- c(  3.99,  5.75, 5.05, 2.58, 3.63,
         4.20,  5.03, 5.23, 4.08, 2.77)
```



t-test in **R**

```
x <- c(  4.88,  8.56, 2.69, 4.89, 5.45,
        11.12, 10.43, 5.85, 4.31, 4.45)
y <- c(  3.99,  5.75, 5.05, 2.58, 3.63,
         4.20,  5.03, 5.23, 4.08, 2.77)
```

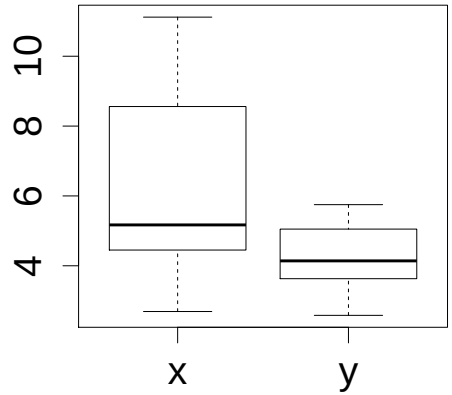


Assess $H_0 : \mu_X = \mu_Y$ based on observed difference of means and variance:

$$t = \frac{mean(X) - mean(Y)}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}, \quad \text{for equal sample sizes } n.$$

t-test in **R**

```
x <- c(  4.88,  8.56, 2.69, 4.89, 5.45,
        11.12, 10.43, 5.85, 4.31, 4.45)
y <- c(  3.99,  5.75, 5.05, 2.58, 3.63,
         4.20,  5.03, 5.23, 4.08, 2.77)
```



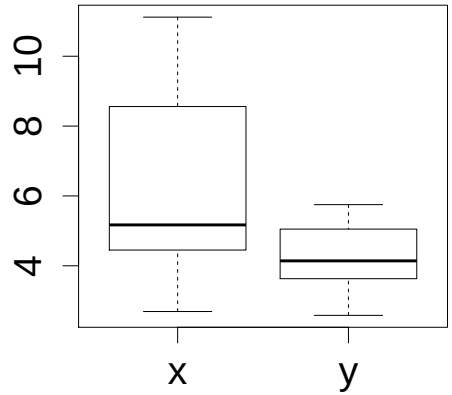
Assess $H_0 : \mu_X = \mu_Y$ based on observed difference of means and variance:

$$t = \frac{mean(X) - mean(Y)}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}, \quad \text{for equal sample sizes } n.$$

```
> t.test(x, y, var.equal = FALSE, paired = FALSE)

        Welch Two Sample t-test

data:  x and y
t = 2.1466, df = 11.475, p-value = 0.05397
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.04097837  4.10497837
sample estimates:
mean of x mean of y
    6.263     4.231
```

t-test in **R**

```
x <- c(  4.88,  8.56, 2.69, 4.89, 5.45,
        11.12, 10.43, 5.85, 4.31, 4.45)
y <- c(  3.99,  5.75, 5.05, 2.58, 3.63,
         4.20,  5.03, 5.23, 4.08, 2.77)
```



Assess $H_0 : \mu_X = \mu_Y$ based on observed difference of means and variance:

$$t = \frac{mean(X) - mean(Y)}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}, \quad \text{for equal sample sizes } n.$$

```
> t.test(x, y, var.equal = FALSE, paired = FALSE)

        Welch Two Sample t-test

data:  x and y
t = 2.1466, df = 11.475, p-value = 0.05397
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.04097837  4.10497837
sample estimates:
mean of x mean of y
    6.263     4.231
```

p-value calculation

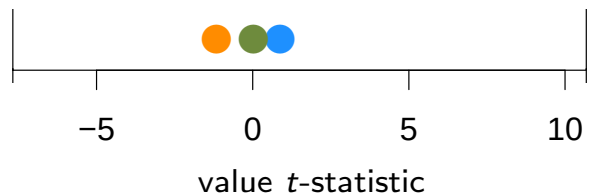
- $$t = \frac{\delta}{\sqrt{\frac{1}{n} \cdot (\sigma_X^2 + \sigma_Y^2)}}$$

  $$\begin{aligned} \delta &= mean(X) - mean(Y) \\ \sigma_X^2 &= var(X) = sd(X)^2 \end{aligned}$$

- Let all assumptions and H_0 hold.
  $\Rightarrow$ distribution of t is known
  $\Rightarrow$ is your t value 'typical'?

- draw random numbers from $N(0,1)$[a]
  that represent variables with $\mu_X = \mu_Y$,
  i. e. under H_0

	x_1	x_2	x_3	y_1	y_2	y_3	t(x,y)$t
1	0.33	-0.19	-2.23	-0.18	1.95	-0.19	-1.16
2	1.51	0.50	0.48	1.41	-0.28	-0.42	0.87
3	0.46	1.22	0.24	1.45	-0.65	1.08	0.02



[a] standard normal distribution

p-value calculation

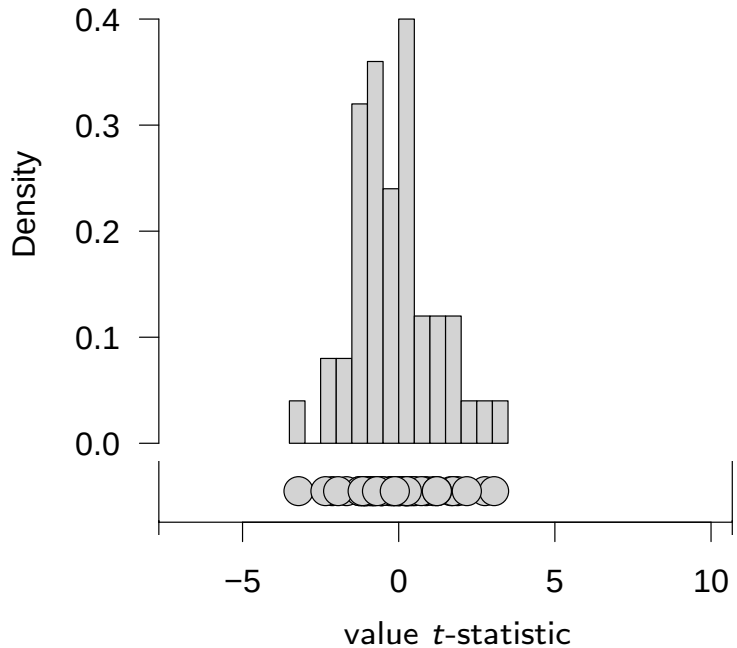
- $t = \frac{\delta}{\sqrt{\frac{1}{n} \cdot (\sigma_X^2 + \sigma_Y^2)}}$

$$\delta = \quad mean(X) - mean(Y)$$
$$\sigma_X^2 = \quad var(X) = sd(X)^2$$

- Let all assumptions and H_0 hold.
  $\Rightarrow$ distribution of t is known
  $\Rightarrow$ is your t value 'typical'?
- draw random numbers from $N(0,1)$[a] that represent variables with $\mu_X = \mu_Y$, i. e. under H_0

	x_1	x_2	x_3	y_1	y_2	y_3	t(x,y)$t
1	0.33	-0.19	-2.23	-0.18	1.95	-0.19	-1.16
2	1.51	0.50	0.48	1.41	-0.28	-0.42	0.87
3	0.46	1.22	0.24	1.45	-0.65	1.08	0.02

...



[a] standard normal distribution

p-value calculation

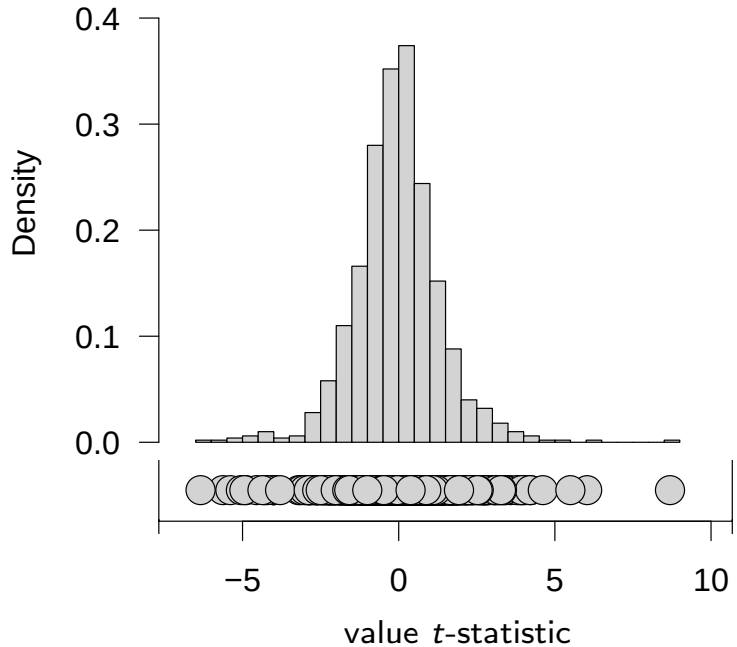
- $$t = \frac{\delta}{\sqrt{\frac{1}{n} \cdot (\sigma_X^2 + \sigma_Y^2)}}$$

$$\begin{aligned} \delta &= mean(X) - mean(Y) \\ \sigma_X^2 &= var(X) = sd(X)^2 \end{aligned}$$

- Let all assumptions and H_0 hold.
  $\Rightarrow$ distribution of t is known
  $\Rightarrow$ is your t value 'typical'?
- draw random numbers from $N(0,1)$[a] that represent variables with $\mu_X = \mu_Y$, i. e. under H_0

	x_1	x_2	x_3	y_1	y_2	y_3	t(x,y)$t
1	0.33	-0.19	-2.23	-0.18	1.95	-0.19	-1.16
2	1.51	0.50	0.48	1.41	-0.28	-0.42	0.87
3	0.46	1.22	0.24	1.45	-0.65	1.08	0.02

...



[a] standard normal distribution

p-value calculation

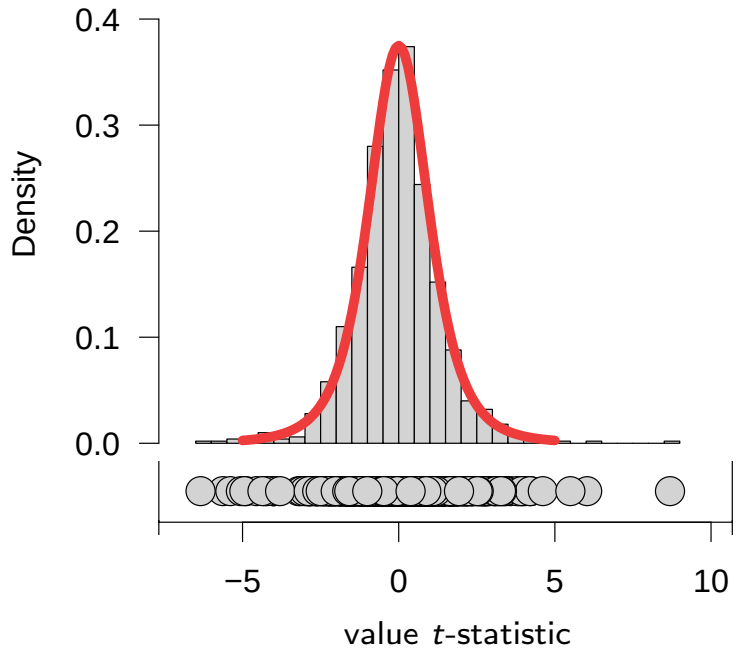
- $t = \frac{\delta}{\sqrt{\frac{1}{n} \cdot (\sigma_X^2 + \sigma_Y^2)}}$

$$\delta = mean(X) - mean(Y)$$
$$\sigma_X^2 = var(X) = sd(X)^2$$

- Let all assumptions and H_0 hold.
  $\Rightarrow$ distribution of t is known
  $\Rightarrow$ is your t value 'typical'?
- draw random numbers from $N(0,1)$[a] that represent variables with $\mu_X = \mu_Y$, i. e. under H_0

	x_1	x_2	x_3	y_1	y_2	y_3	t(x,y)$t
1	0.33	-0.19	-2.23	-0.18	1.95	-0.19	-1.16
2	1.51	0.50	0.48	1.41	-0.28	-0.42	0.87
3	0.46	1.22	0.24	1.45	-0.65	1.08	0.02

...



[a] standard normal distribution

p-value calculation

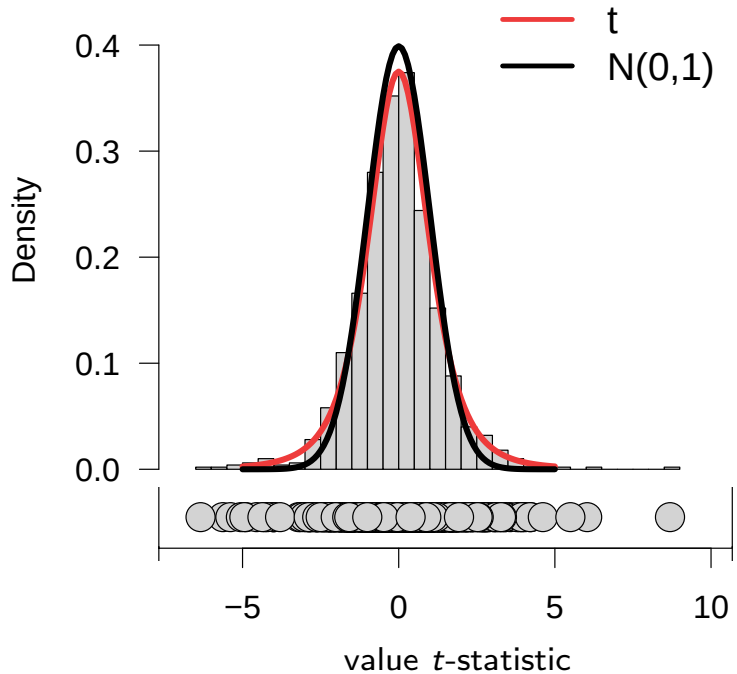
- $$t = \frac{\delta}{\sqrt{\frac{1}{n} \cdot (\sigma_X^2 + \sigma_Y^2)}}$$

$$\delta = \quad mean(X) - mean(Y)$$
$$\sigma_X^2 = \quad var(X) = sd(X)^2$$

- Let all assumptions and H_0 hold.
  $\Rightarrow$ distribution of t is known
  $\Rightarrow$ is your t value 'typical'?
- draw random numbers from $N(0,1)$[a] that represent variables with $\mu_X = \mu_Y$, i. e. under H_0

	x_1	x_2	x_3	y_1	y_2	y_3	t(x,y)$t
1	0.33	-0.19	-2.23	-0.18	1.95	-0.19	-1.16
2	1.51	0.50	0.48	1.41	-0.28	-0.42	0.87
3	0.46	1.22	0.24	1.45	-0.65	1.08	0.02

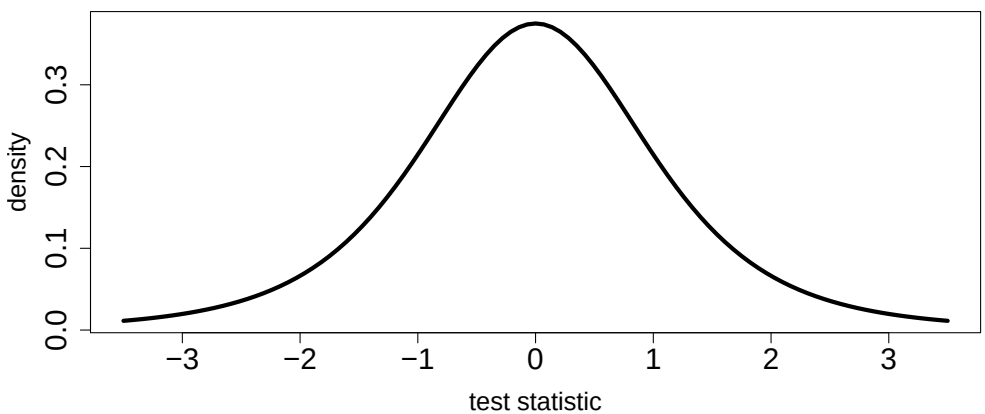
...



[a] standard normal distribution

t-test: From test statistic to p-value

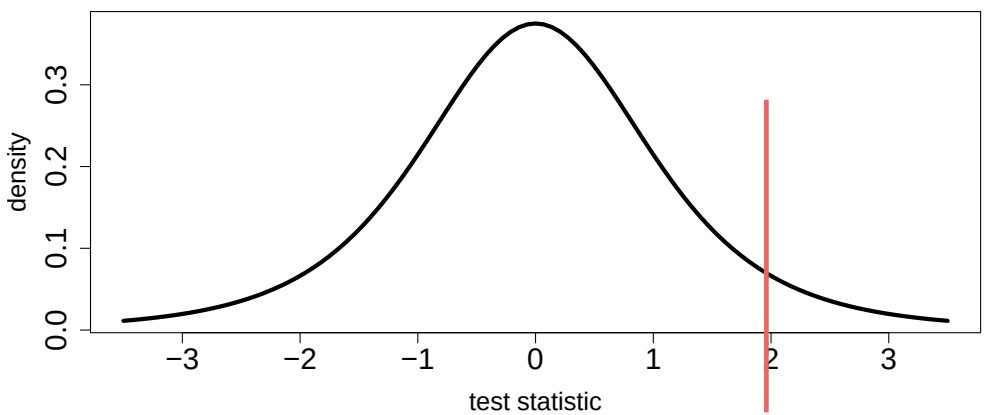
- Area under density curve equals one.
- There is no value that proves that the null is wrong.
- If the observed value of the test statistic is 'too extreme' in terms of the assumed distribution (under H_0), we do not believe that this is the truely underlying distribution (and reject H_0).
- The null hypothesis is rejected at the ($\alpha =$)5% level, if the p-value is less than the predefined (!) significance level α.



t(x, y)	p_t	x_1	x_2	x_3	y_1	y_2	y_3
1.77		4.20	1.22	3.30	2.08	0.79	0.49

t-test: From test statistic to p-value

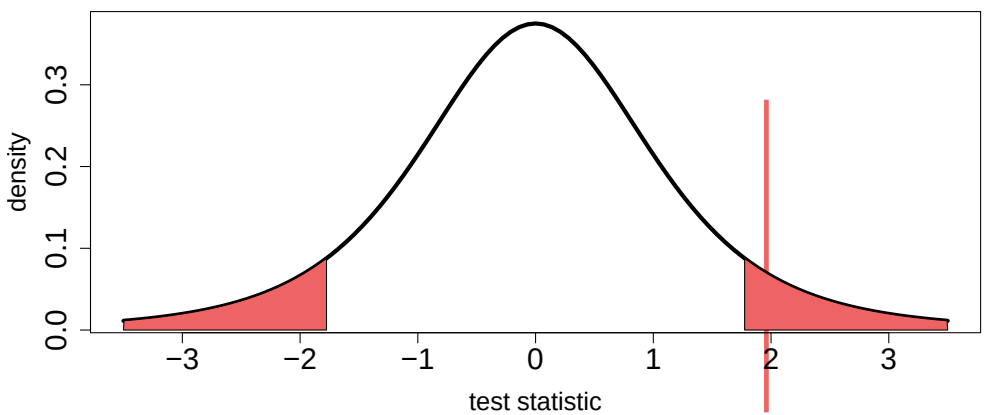
- Area under density curve equals one.
- There is no value that proves that the null is wrong.
- If the observed value of the test statistic is 'too extreme' in terms of the assumed distribution (under H_0), we do not believe that this is the truely underlying distribution (and reject H_0).
- The null hypothesis is rejected at the ($\alpha =$)5% level, if the p-value is less than the predefined (!) significance level α.



t(x, y)	p_t	x_1	x_2	x_3	y_1	y_2	y_3
1.77		4.20	1.22	3.30	2.08	0.79	0.49

t-test: From test statistic to p-value

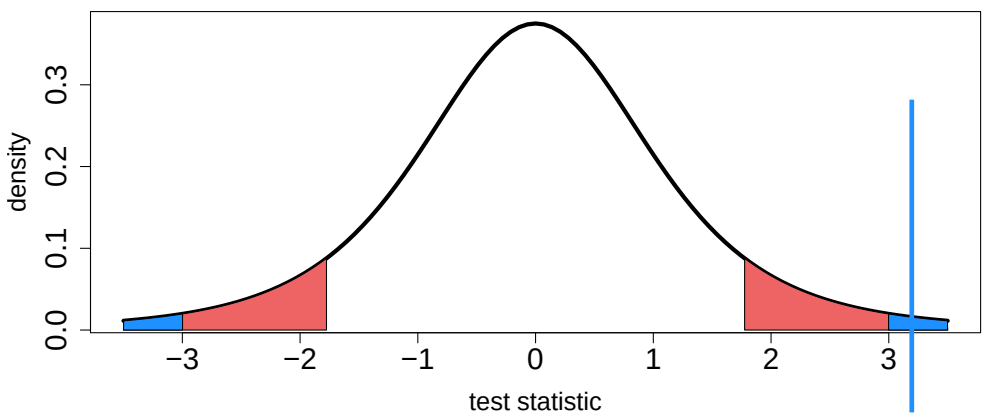
- Area under density curve equals one.
- There is no value that proves that the null is wrong.
- If the observed value of the test statistic is 'too extreme' in terms of the assumed distribution (under H_0), we do not believe that this is the truely underlying distribution (and reject H_0).
- The null hypothesis is rejected at the ($\alpha =$)5% level, if the p-value is less than the predefined (!) significance level α.



t(x, y)	p_t	x_1	x_2	x_3	y_1	y_2	y_3
1.77	0.15	4.20	1.22	3.30	2.08	0.79	0.49

t-test: From test statistic to p-value

- Area under density curve equals one.
- There is no value that proves that the null is wrong.
- If the observed value of the test statistic is 'too extreme' in terms of the assumed distribution (under H_0), we do not believe that this is the truely underlying distribution (and reject H_0).
- The null hypothesis is rejected at the ($\alpha =$)5% level, if the p-value is less than the predefined (!) significance level α.



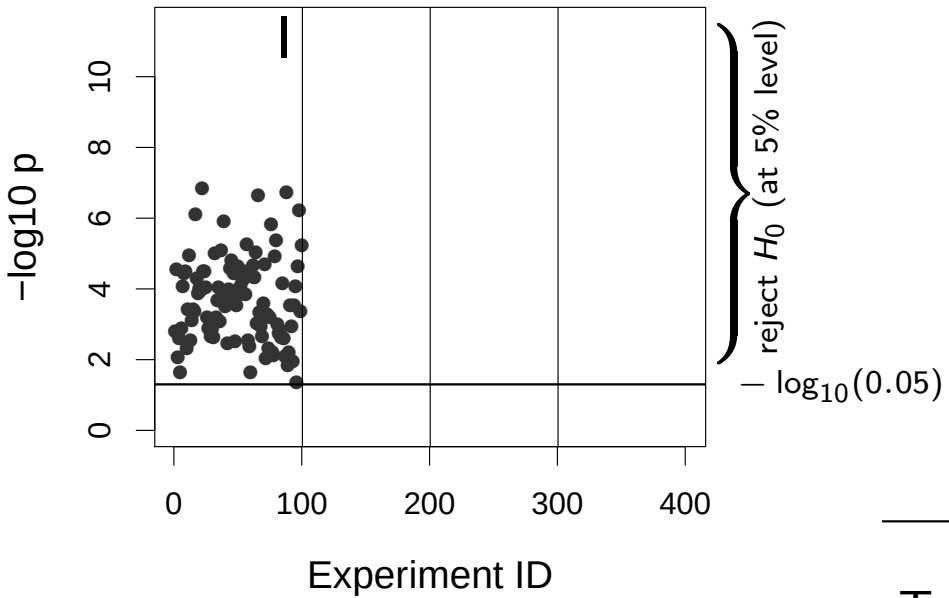
t(x, y)	p_t	x_1	x_2	x_3	y_1	y_2	y_3
1.77	0.15	4.20	1.22	3.30	2.08	0.79	0.49
3.00	0.04	2.26	6.07	4.98	1.82	-0.54	-0.02

t-test: statistic to p-value

- The p-value is the probability that under H_0 the value of the test statistic is at least as extreme as the one observed (in terms of H_1) in the particular experiment of interest.
- The p-value is not
  - the probability of the null hypothesis.
  - the probability of the observed value of the statistic.
- The decision to make is to either reject H_0 or not.
  It is not about the acception of H_0.

t-test and power

$$t = \frac{mean(X) - mean(Y)}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}$$



draw random numbers from

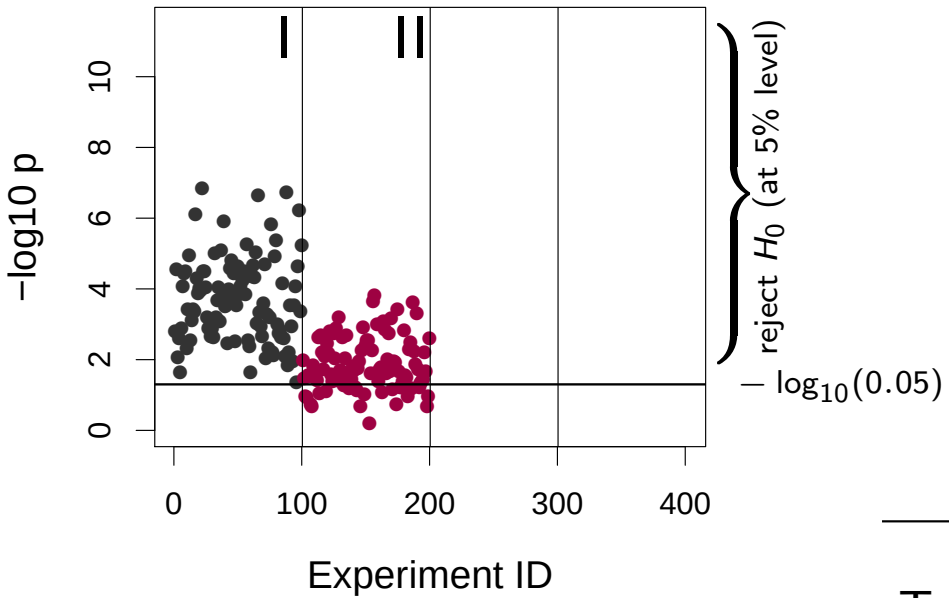
Setting	n	δ	σ
I	**10**	**2**	**1**

Each point represents:
the p-value of one experiment
with n samples per group X and Y
where $0 \neq \delta = \mu_X - \mu_Y \Rightarrow H_1$

Truth	Test decision: do not reject H_0	Test decision: reject H_0
H_0 true	✓	α
H_0 false	β	✓ $(1-\beta)$

t-test and power

$$t = \frac{mean(X) - mean(Y)}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}$$



draw random numbers from

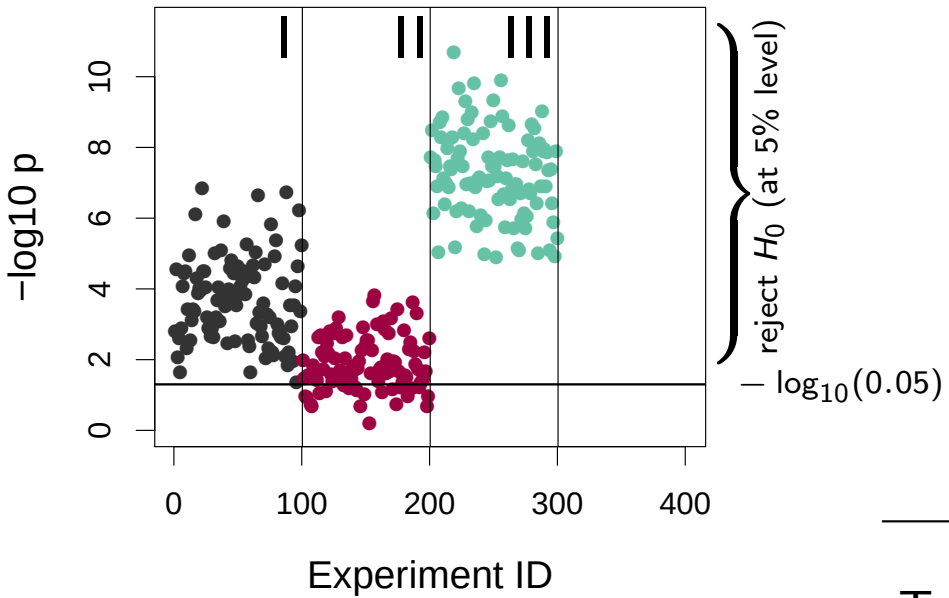
Setting		n	δ	σ
I		**10**	**2**	**1**
II	$n \downarrow$	5	2	1

Each point represents:
the p-value of one experiment
with n samples per group X and Y
where $0 \neq \delta = \mu_X - \mu_Y \Rightarrow H_1$

	Test decision	
Truth	do not reject H_0	reject H_0
H_0 true	✓	α
H_0 false	β	✓ $(1-\beta)$

t-test and power

$$t = \frac{\overbrace{mean(X) - mean(Y)}^{\delta}}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}$$



Each point represents:
the p-value of one experiment
with n samples per group X and Y
where $0 \neq \delta = \mu_X - \mu_Y \Rightarrow H_1$

draw random numbers from

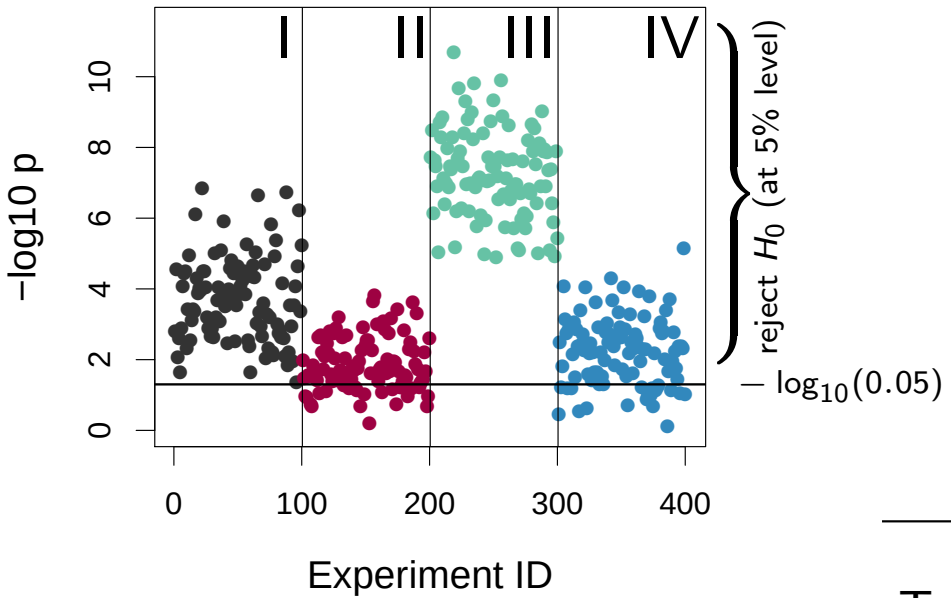
Setting		n	δ	σ
I		**10**	**2**	**1**
II	$n \downarrow$	5	2	1
III	$\delta \uparrow$	10	4	1

Truth	Test decision: do not reject H_0	Test decision: reject H_0
H_0 true	✓	α
H_0 false	β	✓ $(1-\beta)$

t-test and power

$$t = \frac{mean(X) - mean(Y)}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}$$

$$\sigma = sd = \sqrt{var}, \sigma_X = \sigma_Y$$



draw random numbers from

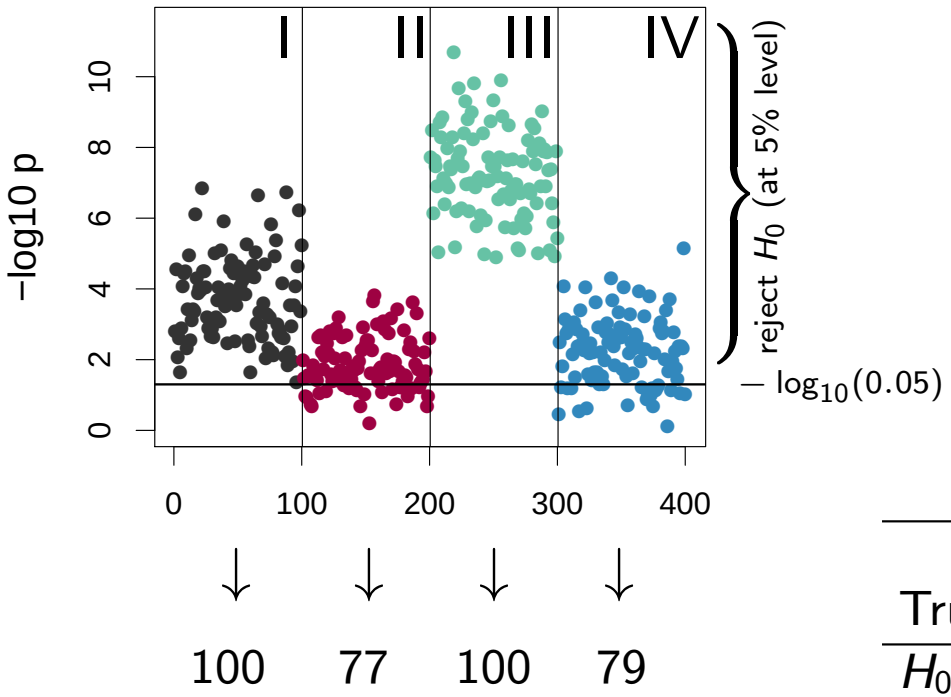
Setting		n	δ	σ
I		**10**	**2**	**1**
II	$n \downarrow$	5	2	1
III	$\delta \uparrow$	10	4	1
IV	$\sigma \uparrow$	10	2	1.5

Each point represents:
the p-value of one experiment
with n samples per group X and Y
where $0 \neq \delta = \mu_X - \mu_Y \Rightarrow H_1$

	Test decision	
Truth	do not reject H_0	reject H_0
H_0 true	✓	α
H_0 false	β	✓ $(1-\beta)$

t-test and power

$$t = \frac{mean(X) - mean(Y)}{\sqrt{\frac{1}{n} \cdot (var(X) + var(Y))}}$$



Power estimate per setting:
correct rejections %

Setting		n	δ	σ	Power *est*	Power *theo*
draw random numbers from						
I		**10**	**2**	**1**	1.00	0.99
II	$n \downarrow$	5	2	1	0.77	0.79
III	$\delta \uparrow$	10	4	1	1.00	1.00
IV	$\sigma \uparrow$	10	2	1.5	0.79	0.80

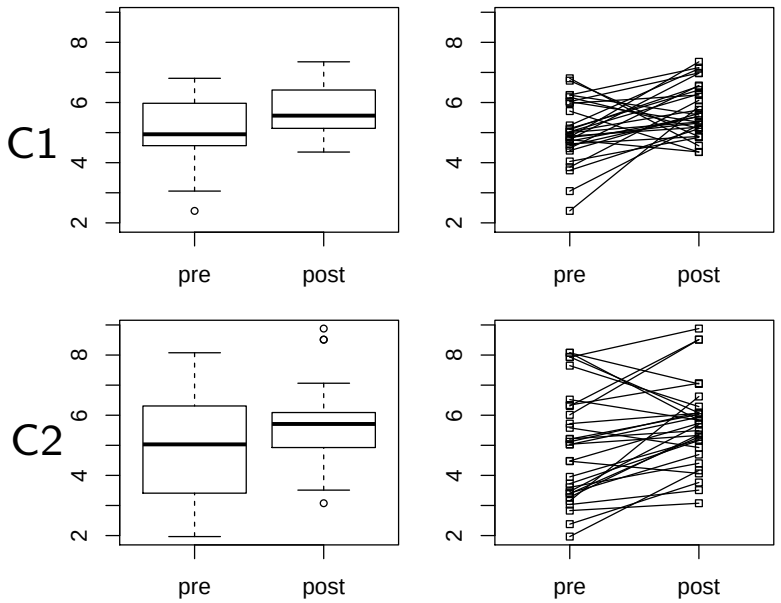
Truth	Test decision: do not reject H_0	Test decision: reject H_0
H_0 true	✓	α
H_0 false	β	✓ $(1-\beta)$

Paired *t*-test

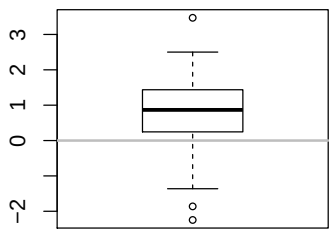
- Especially in medical applications paired designs are used
- Common examples: measurements on the same patients
  - pre/post treatment
  - tumerous/healthy tissue
- Common situation:
  - high variance between patients (inter-individual)
  - low differences of the measurements of the same patient (intra-individual)
- An unpaired t-test might not recognise existing differences between groups in this situation
- A paired t-test performs better because it only uses the intra-individual differences
- In **R**: `t.test(x, y, paired = TRUE)`

Paired *t*-test - Example

- In situations C1 and C2 there are the same intra-individual differences, only the pairs are shifted (more inter-individual variance)
- In C1 both test results are significant
- In C2 only the paired t-test recognises the differences between groups
- Paired t-test gives the same result for C1 and C2 (inter-individual variance doesn't matter)



intra-individual differences for C1 and C2



	paired	
	TRUE	FALSE
C1	0.006	0.004
C2	0.006	0.088

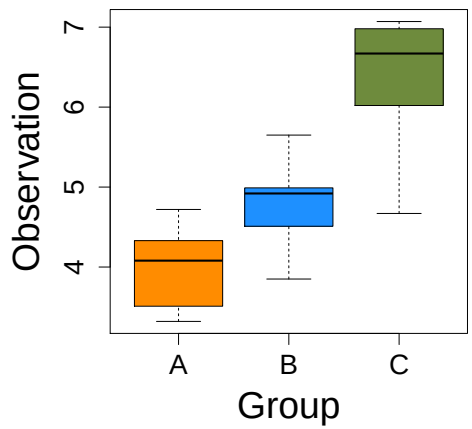
Analysis of variance: one-way ANOVA

```
X <- data.frame(
        "Group"    =  rep(c("A","B","C"), each = 5),
  "Observation"    =  c(4.33, 4.72, 3.32, 4.08, 3.51,
                        4.51, 3.85, 4.99, 4.92, 5.65,
                        6.98, 7.07, 6.67, 4.67, 6.02))
```

- Comparison of multiple groups
- $H_0 : \mu_1 = \ldots = \mu_I$
- A commonly used model

  $Y_{ik} = \mu + \alpha_i + e_{ik}$, where

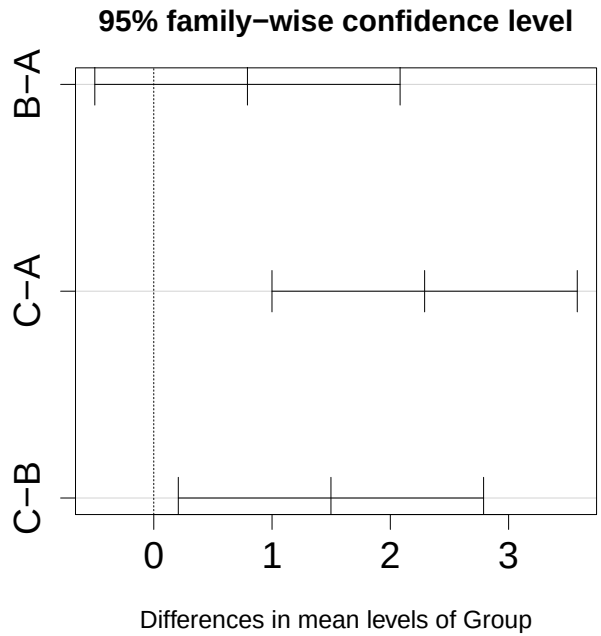
  Y_{ik} observed value for dependent variable
  μ overall mean (grand mean)
  α_i group effect, e. g. treatment effect, $i = 1, \ldots, I$
  e_{ik} random errors ($\sim N(0, \sigma^2)$ amongst others)



Analysis of variance: one-way ANOVA

```
aov.res <- aov(Observation ~ Group, data = X)
summary(aov.res)
TukeyHSD(aov.res)
```

- $H_1 : \mu_i \neq \mu_j$ for a pair (i, j)
- Rejection of H_0 gives no indication on which groups actually differ in means
- In case of rejection (only!), conduct a post-hoc test, e. g. using Tukey's honest significant difference



EXERCISE

LUNCH BREAK

High-throughput analysis

High-throughput analysis in R: Exemplary workflow

Preparations

- platform-dependent quality control, normalization from appropriate software
  $\rightarrow$ 'analysis-ready' quantifications
- get your data in R

Analysis workflow

1. Conduct t-test for each feature (protein) $\rightarrow$ p-value
2. Identify most promising candidates by means of volcano plot, i. e. considering effect size and significance.
3. Adjust p-value to control false discovery rate.
4. Inspect candidate features.

Exemplary workflow

Within the exemplary workflow you will come across

- data import
- descriptive statistics
- statistical inference
- create and export plots
- loops
- data export

File paths

- Function `getwd()` shows the current working directory
- `setwd()` changes your working directory
- **R** uses a forward slash '/' as a separator between folders (Windows normally uses backshlashes '\')
- **R** on Windows: use forward slashes or double backslashes
- Examples:
  "C:/Users/Paul/Documents/Data/data.txt"
  or "C:\\Users\\Paul\\Documents\\Data\\data.txt"

Preparations: Data import via `read.table()`

- Text files (.txt) can be imported via the read.table() command.
- Some options for the `read.table()` command (see also `?read.table`):

`file`	path and file name
`header = FALSE`	column names present?
`sep ="\t"`	separator, e. g. tabulator
`quote = ""`	disable quoting
`comment.char = ""`	turn off the interpretation of comments
`dec = "."`	decimal separator
`skip`	number of lines to skip before the actual data

- Check result using `dim()`, `head()`, `tail()` and `str()`.
- Special functions for other data formats (e.g. `read.csv()`)
- Functionality of RStudio (Import Dataset button in the upper right window)

Example `read.table`

The text file `example_import.txt` has the following structure

row at the very beginning containing additional information

	name	age	number
1	Carl	48	4,2
2	Ben	27	3,4
3	Josh	39	9,1
4	Berta	32	5,2
5	Kim	63	4,7

```
data <- read.table(file = "C:\\...\\example_import.txt",
         header = TRUE,    # header line (2nd row)
         sep = "\t",       # separator = tabulator
         dec = ",",        # decimal separator = ','
         skip = 1)         # skip first row
```

Descriptive statistics - numeric measures

Functions may return either a single statistic or a vector of measures on the data

`mean(), median()`	arithmetic mean, median
`var(), sd()`	variance, standard deviation
`min(), max()`	extrema
`quantile()`	quantiles
`cor()`	correlation
`range()`	vector of minimum und maximum
`summary()`	vector of extrema, mean, quartiles

Loops

- Loops allow the repeated application of the same operation(s), e. g., conduction of statistical test for each protein in a data set.
- Syntax of the most important type of loop:

  `for`(*var* `in` *vector*) { *statements* }

- In turn *var* is assigned the value of each element of *vector*.
- For each iteration, the complete set of *statements* is executed.

```
x <- c(4, 1, 16, 9)
for (i in x) {
  print(sqrt(i))
}
```

Loops - Example

- dat: Matrix with 100 rows
- Calculate the mean for each row and save them into a result vector

```
result <- rep(NA, 100)     # empty result vector
for (row in 1:100) {       # start loop
  m <- mean(dat[row,])     # calculate mean
  result[row] <- m         # save result
}                          # end of loop
```

Comparisons of means - two samples

Broadly speaking

Assuming that two samples `x` and `y` (series of measurements) come from a normal distribution (each), the respective means can be compared by means of the *t*-test.
In **R** this is easily done using the function

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

Important options:

`alternative`	one-sided or two-sided
`paired`	paired or independent samples
`var.equal`	equal or unequal variances in X and Y

Example: *t*-test

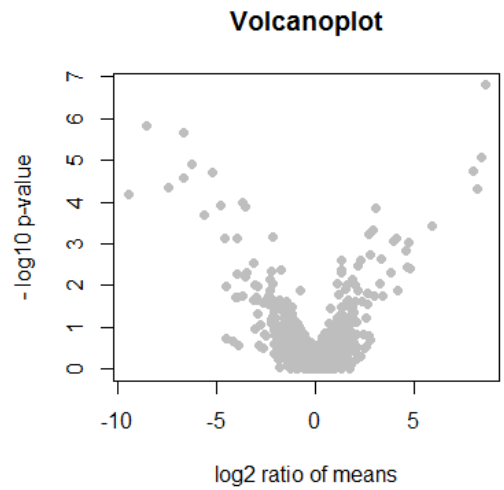
```
x <- rnorm(20)
y <- rnorm(20, mean = 2, sd = 1)
test <- t.test(x, y, alternative = "two.sided",
               paired = FALSE, var.equal = TRUE)

names(test)
test$p.value
```

Candidate selection

- For each variable the differential analysis yields a pair of effect size (fold change, FC) and significance measure (p-value).
- (There are multiple versions for the computation of the FC.)
- In most settings, a variable is considered a 'promising/good' candidate, if the (absolute) FC is large and the p-value small.

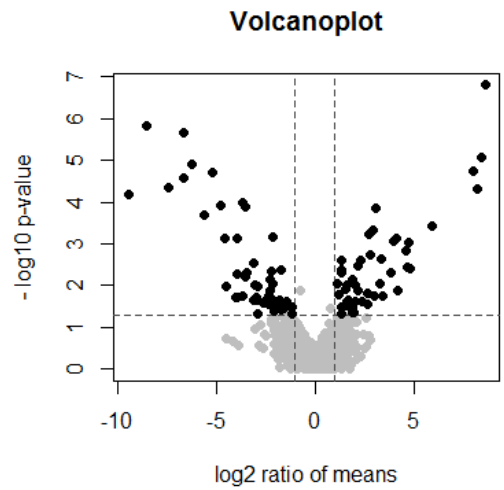
Candidate selection

- For each variable the differential analysis yields a pair of effect size (fold change, FC) and significance measure (p-value).
- (There are multiple versions for the computation of the FC.)
- In most settings, a variable is considered a 'promising/good' candidate, if the (absolute) FC is large and the p-value small.
- Assessment of volcano plot allows identification of these candidates: scatter plot of $-log_{10}p$ versus 'log_2 FC' (log_2(ratio of means))



Candidate selection

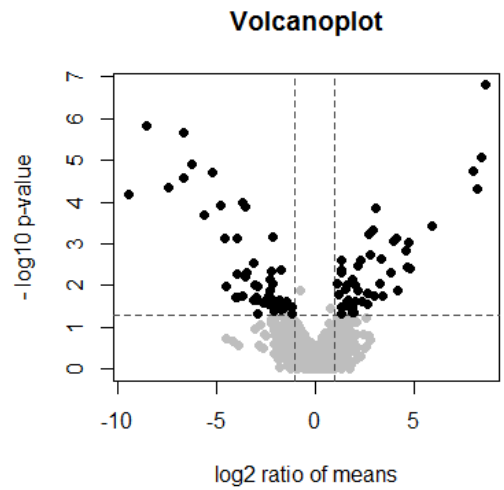
- For each variable the differential analysis yields a pair of effect size (fold change, FC) and significance measure (p-value).
- (There are multiple versions for the computation of the FC.)
- In most settings, a variable is considered a 'promising/good' candidate, if the (absolute) FC is large and the p-value small.
- Assessment of volcano plot allows identification of these candidates: scatter plot of $-log_{10}p$ versus 'log_2 FC' (log_2(ratio of means))



Candidate selection

- For each variable the differential analysis yields a pair of effect size (fold change, FC) and significance measure (p-value).
- (There are multiple versions for the computation of the FC.)
- In most settings, a variable is considered a 'promising/good' candidate, if the (absolute) FC is large and the p-value small.

- Assessment of volcano plot allows identification of these candidates: scatter plot of $-log_{10}p$ versus 'log_2 FC' (log_2(ratio of means))
- Most promising candidates are those in the upper outer corners



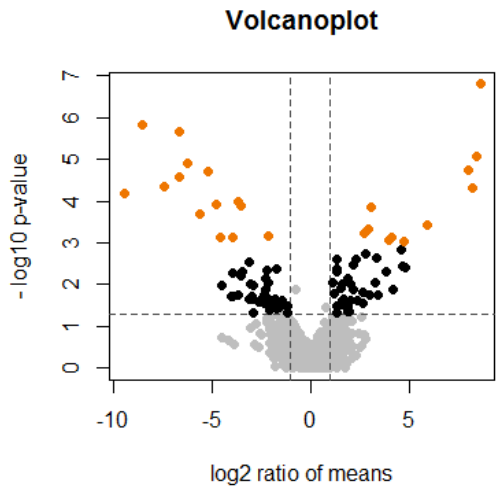
Adjustment for multiple testing

- Type I error is controlled for a single test on a sample.
- Conducting multiple tests on the same sample leads to type I error inflation.
- Example: 1010 proteins (10 differentially expressed, 1000 not)
  - Good power $\rightarrow$ 10 differentially expressed proteins are significant
  - you expect about 50 (5% of 1000) false positive findings.
  - $\rightarrow$ from $\approx$ 60 significant results only 10 are true.
  - $\rightarrow$ high false discovery rate (FDR)

Adjustment for multiple testing

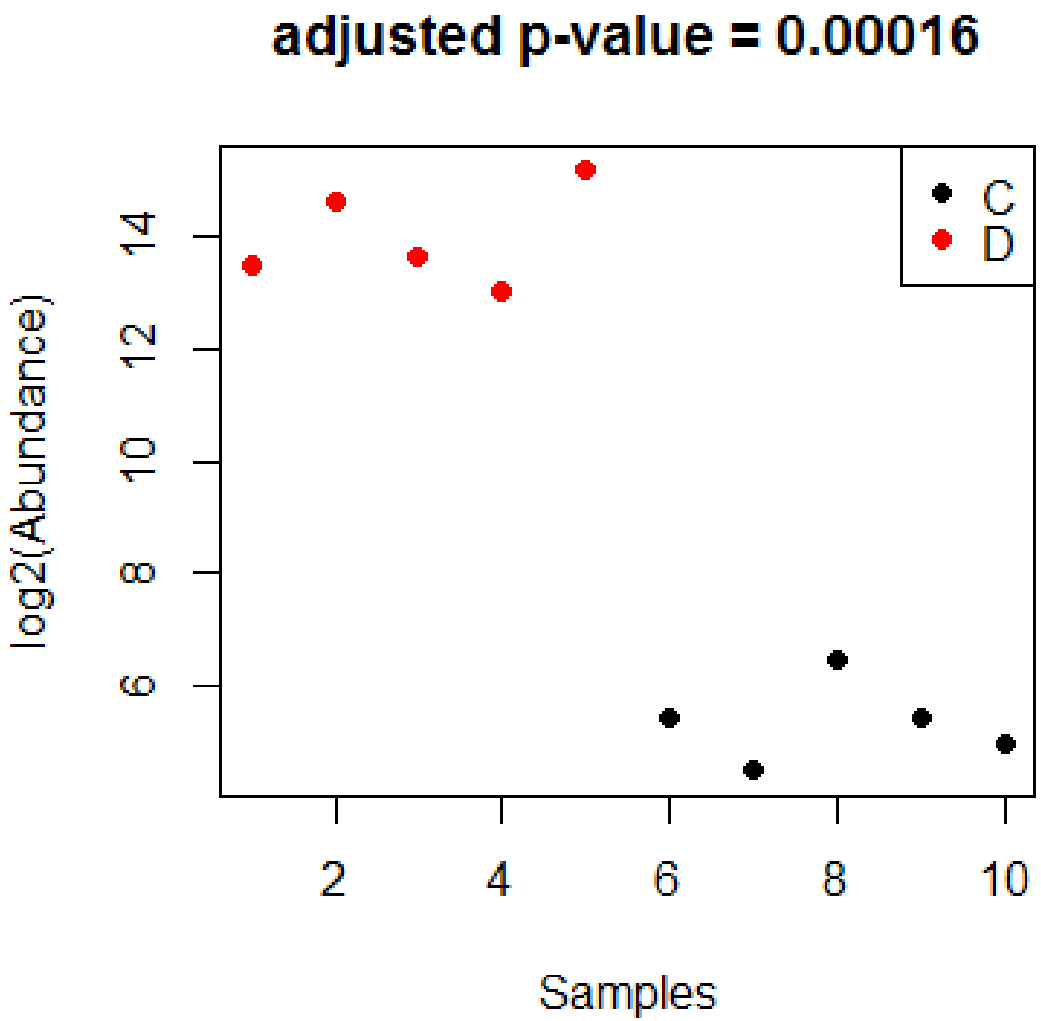
- Type I error is controlled for a single test on a sample.
- Conducting multiple tests on the same sample leads to type I error inflation.
- Example: 1010 proteins (10 differentially expressed, 1000 not)
  - Good power $\rightarrow$ 10 differentially expressed proteins are significant
  - you expect about 50 (5% of 1000) false positive findings.
  - $\rightarrow$ from $\approx$ 60 significant results only 10 are true.
  - $\rightarrow$ high false discovery rate (FDR)

- In **R** use `p.adjust(p, method='fdr')` to adjust according to the method of Benjamini and Hochberg (1995).



Inspection of chosen candidates

- choose candidate proteins from the volcano plot
- look at abundance profiles



Save, load and export

- Use commands `save()` and `load()` to save/load single objects from/to your workspace.
- Data format: `.RData`

`save()`	save objects
`load()`	load objects
`save.image()`	save complete workspace

- The `write.table()` command allows to export **R** data sets into text files.

`x`	object to write
`file`	path and file name
`col.names = TRUE`	use column names of `x` as a header
`sep ="\t"`	separator, e. g. tabulator
`quote = FALSE`	disable quoting
`dec = "."`	decimal separator

Graphics

- Even though you cannot 'click together' your graphics in **R**, only few commands are necessary for the preparation of overview plots.
- A large number of graphics parameters is available to adapt the presentation to the user's requirements.
- **R** provides comprehensive functionality for exporting your graphics.
- See `?plotmath` on usage of mathematical symbols for labeling.
- Some basic high-level plots ('stand-alone' plots):

`plot()`	scatter plot (e. g.)
`boxplot()`	box plot
`barplot()`	bar plot
`hist()`	histogram

Graphics: Parameter

- Most often, the created plots need some adaptions.
- `par()` allows to change settings in a more general manner than a function's own build-in parameters.
  (see `mfcol, mfrow` for multiple frames in one array)

Some of the most used plot() parameters are

`main, sub`	plot title
`xlab, ylab`	axis labels
`xlim, ylim`	range on axes
`type`	plot type (l=line, p=point, b=both, ...)
`cex`	magnification for text and symbols
`col`	color specification
`lty, lwd`	line type (1=solid, 2=dashed, ...) and width resp.
`pch`	point character for plotting

Low-level plots

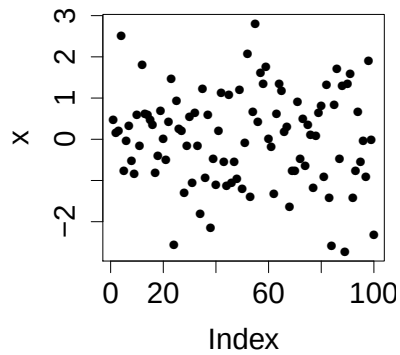
Low level functions are used to add additional elements to an already existing plot.

`points()`	points (e. g. for highlighting)
`lines()`	lines
`abline()`	(straight) lines
`legend()`	add a legend to the plot
`axis()`	control x and y-axis (e. g. ticks and labels)
`text()`	strings
`...`	

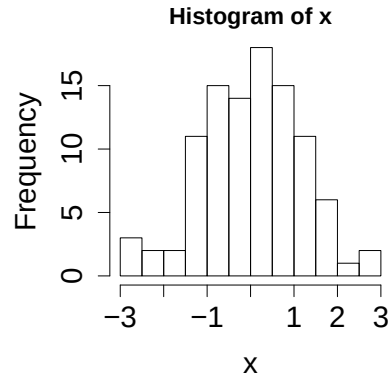
First graphics in R

```
x <- rnorm(100)       # draw 100 random numbers from N(0,1)
y <- x + rnorm(100)
```

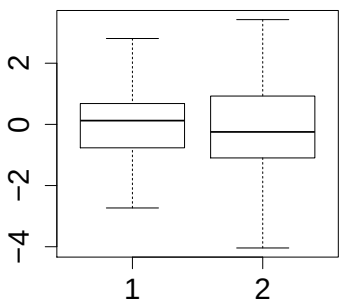
`plot(x)`



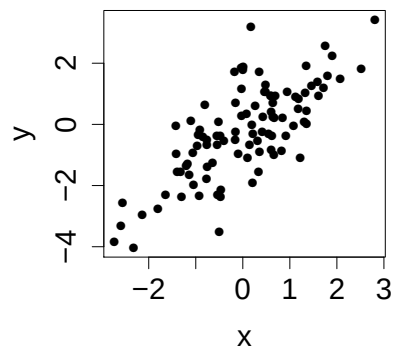
`hist(x)`



`boxplot(x,y)`

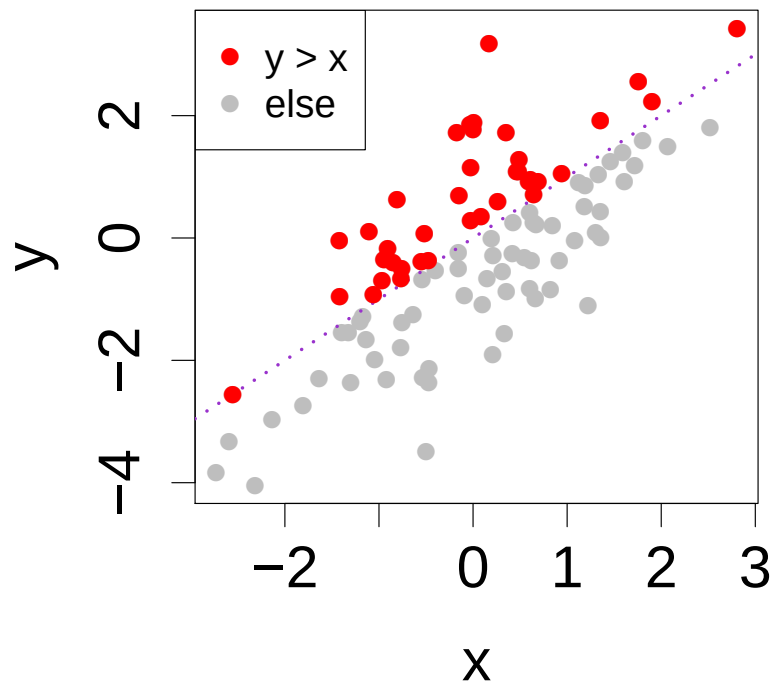


`plot(x,y)`



Graphics

```
plot(x,y, col = "grey", pch = 16)
abline(0, 1, lwd = 3, lty = 3, col = "darkorchid")
y.greater.x <- which(x < y)
points(x[y.greater.x], y[y.greater.x], pch = 16, col = "red")
legend("topleft", pch = c(16, 16), col = c("red", "grey"),
   legend = c("y > x", "else"), cex = 1)
```



Devices

- By default, graphics are only shown in a separate window without being saved.
- Graphics can be saved as `pdf()`, `jpeg()`, `bmp()`, `png()`, `tiff()` for example.
- In a device one can save either a single or multiple graphics.
- how it works:
  1. open device — `pdf("C:/test.pdf")` (or the like)
  2. plot — `plot(1:5)`
     `...`
  3. shut down device — `dev.off()`
- One can also use the functionality of RStudio (`Plots > Export`).

EXERCISE