Advanced analysis of quantitative proteomics data using R

Karin Schork and Michael Turewicz

Ruhr-University Bochum
Medizinisches Proteom-Center
Medical Bioinformatics

Organisation

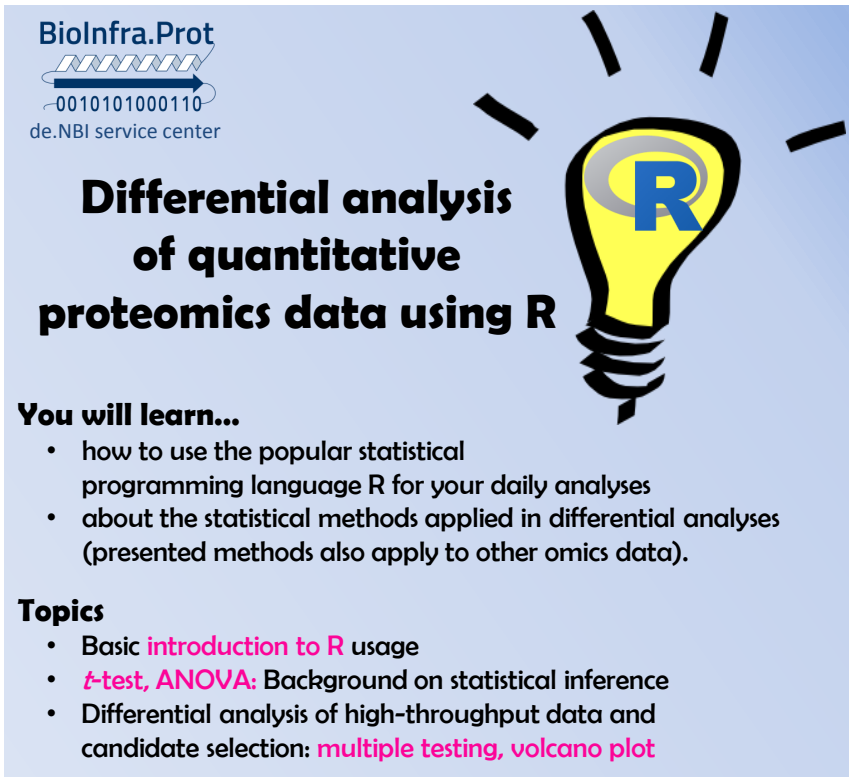
All times are given in UTC+1!

10:00	-	10:20	Welcome and de.NBI overview
10:20	-	10:50	Advanced usage of R
10:50	-	11:30	Data preprocessing
11:30	-	12:45	Hands-on (and sample solution)
12:45	-	13:30	Clustering
13:30	-	14:15	Lunch break
14:15	-	15:00	Hands-on (and sample solution)
15:00	-	15:45	Principal component analysis (PCA) and ROC analysis
15:45	-	16:30	Hands-on (and sample solution)
16:30	-	17:00	Writing own R functions
17:10	-	17:45	Hands-on (and sample solution)
17:45	-	18:00	Discussion and questions

Advanced usage of R by using additional R packages

What is "basic usage" of R?

- Basic usage: Using basic R packages and their functions.
- Basic packages: Packages coming with the standard installation of R that can be directly used after starting the R console.
- This was the scope of our basic R course.



Basic R packages

- Listing basic packages via "`sessionInfo()`"...
- ...or "`sessionInfo()$basePkgs`" in the console.

```
> sessionInfo()
R version 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 18362)

Matrix products: default

locale:
[1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252    LC_MONETARY=German_Germany.1252 LC_NUMERIC=C

[5] LC_TIME=German_Germany.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
[1] compiler_3.6.1 tools_3.6.1
> |
```

```
> sessionInfo()$basePkgs
[1] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
> |
```

Basic R packages

- **basic**: The basic functions which let R function as a language (e.g., arithmetic, input/output, basic programming support). Its contents are available through inheritance from any environment.
- **datasets**: Base R datasets (e.g., cars, iris).
- **graphics**: R functions for base graphics (e.g., plot).
- **grDevices**: Graphics devices and support for base and grid graphics (e.g., pdf, png, colors).
- **methods**: Formally defined methods and classes for R objects, plus other programming tools, as described in the references.
- **stats**: Functions for statistical calculations (e.g., statistical tests) and random number generation.
- **utils**: Collection of utility functions (e.g., for package installation and updates).

What is "advanced usage" of R?

- Installing additional (= not basic) R packages from repositories & using them for specific tasks.
- Writing one's own functions (and packages)
  $\rightarrow$ at the end of today's course.

R package repositories

- A **software repository** is a storage location from which software packages can be downloaded and installed.
- Most of them are searchable online resources containing also package metadata (e.g., download statistics).
- Quality control: Package tests and / or reviews & quality measures.
- Some important software repositories:
  - Comprehensive Perl Archive Network (Perl)
  - Comprehensive-TeX-Archive-Network (TeX)
  - Python Package Index (PyPI)
  - GitHub
- An **R package repository** is an online software repository containing R packages.
- Most important R package repositories:
  - Comprehensive R Archive Network (CRAN)
  - Bioconductor
  - GitHub

Comprehensive R Archive Network (CRAN)

- https://cran.r-project.org/

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

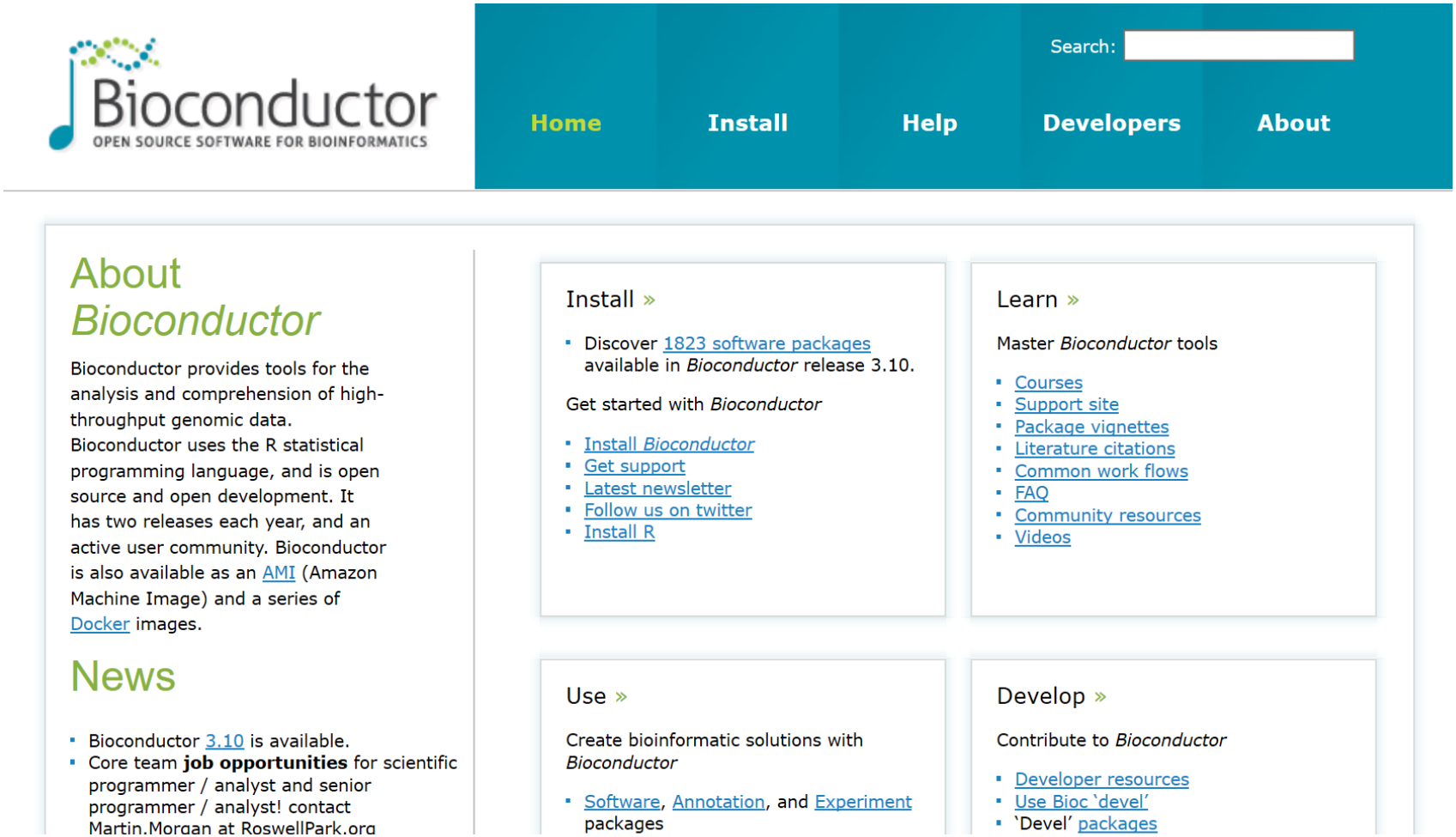
- The latest release (2019-07-05, Action of the Toes) R-3.6.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.

Comprehensive R Archive Network (CRAN)

- The repository that provides the basic R packages.
- Contains currently $15,247$ packages from various disciplines (no restriction).
- There are currently 101 mirrors (= servers containing copies of repository) in 49 countries/regions around the world (4 in Germany).
- Quality: Automatic tests (R CMD check) on different platforms (Linux, Mac & Windows), review during submission and compliance check with the "CRAN Repository Policy" guidelines.

Bioconductor

- https://www.bioconductor.org/

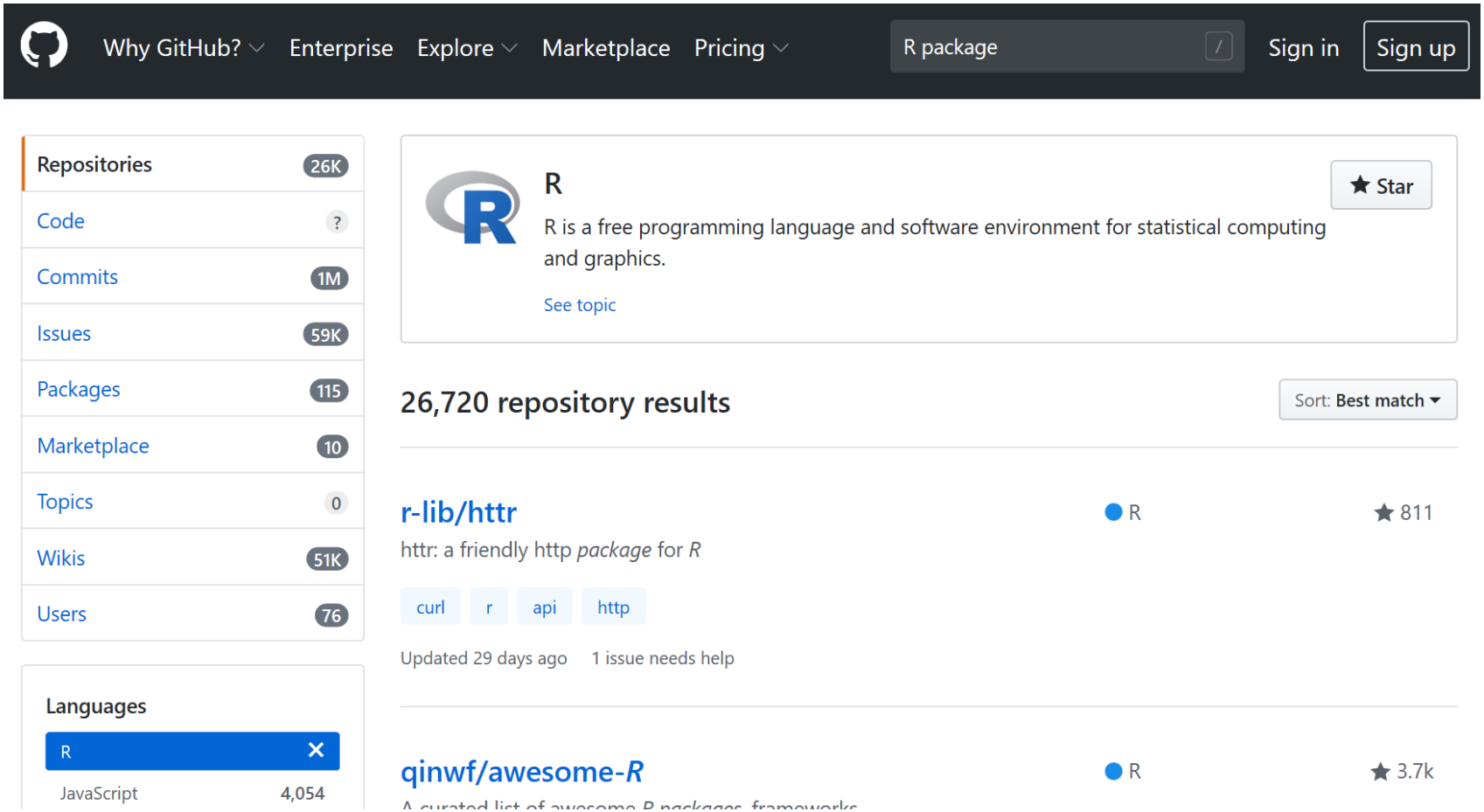


Bioconductor

- Provides R packages for bioinformatics.
- Contains currently $1,823$ software packages.
- There are currently 8 mirrors in 7 countries/regions around the world (1 at TU Dortmund in Germany).
- Some relevant packages: limma, topGO, mzR, MSnbase, xcms, MSstats, (...)
- Quality: Automatic tests (R CMD check) on different platforms (Linux, Mac & Windows), expert review during submission, compliance check with the Bioconductor package guidelines and various quality measures (e.g., last update, update frequency, number of downloads, coverage with unit tests).

GitHub

- https://github.com/



GitHub

- General software package repository.
- Contains currently $> 25,000$ R packages.
- All packages from CRAN & Bioconductor are automatically in GitHub.
- Quality: No review, no R CMD check tests, no R-specific guidelines. Only general code quality measures (e.g., unit test coverage).
- Scope: Good for package development before CRAN or Bioconductor submission.

Package installation: Set repositories

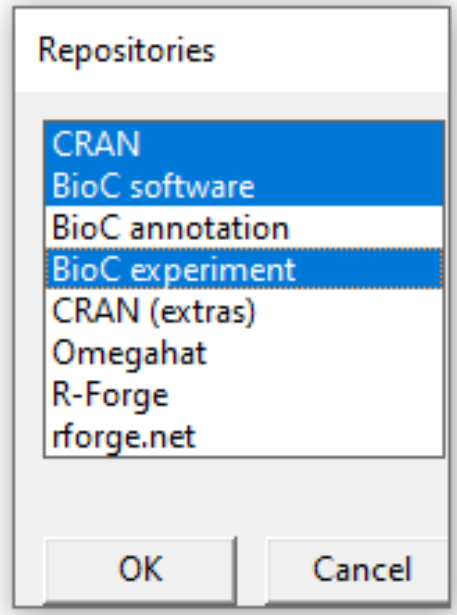
- `setRepositories()`

```
> setRepositories()
--- Please select repositories for use in this session ---


1: + CRAN
2:   BioC software
3:   BioC annotation
4:   BioC experiment
5:   CRAN (extras)
6:   Omegahat
7:   R-Forge
8:   rforge.net

Enter one or more numbers separated by spaces, or an empty line to cancel
1: 1 2 4
>
```

Package installation: Set repositories

- `setRepositories(graphics = TRUE)`

```
>
>
>
>
>
>
>
>
> setRepositories(graphics = TRUE)
```



Package installation: Details about CRAN mirrors

- `getCRANmirrors(all = TRUE)[35:50,1:3]`

```
> getCRANmirrors(all = TRUE)[35:50,1:3]
                              Name     Country          City
35                 El Salvador [https] El Salvador San Salvador
36                Estonia [https]     Estonia         Tartu
37      France (Lyon 1) [https]      France          Lyon
38      France (Lyon 2) [https]      France          Lyon
39   France (Marseille) [https]      France     Marseille
40 France (Montpellier) [https]      France   Montpellier
41               France (Paris 1)      France         Paris
42   Germany (Erlangen) [https]     Germany      Erlangen
43  Germany (Göttingen) [https]     Germany     Göttingen
44    Germany (Münster) [https]     Germany       Münster
45 Germany (Regensburg) [https]     Germany    Regensburg
46                Greece [https]      Greece         Crete
47                Hungary [https]     Hungary      Budapest
48                Iceland [https]     Iceland     Reykjavik
49  Indonesia (Jakarta) [https]   Indonesia       Jakarta
50                  Iran [https]        Iran       Mashhad
> 
```

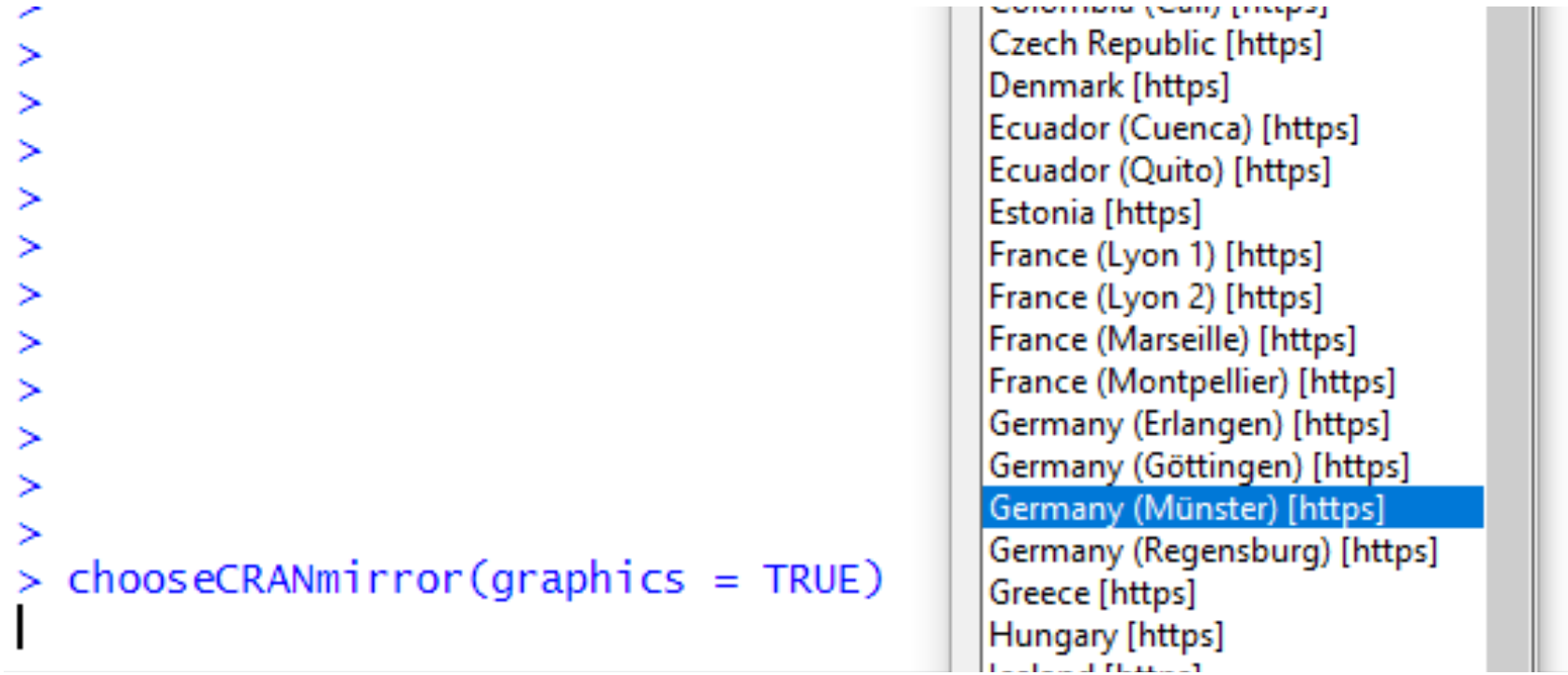
Package installation: Choose secure CRAN mirror

- `chooseCRANmirror()`

```
> chooseCRANmirror()
Secure CRAN mirrors

 1: 0-Cloud [https]                  2: Algeria [https]                  3: Australia (Canberra) [https]
 4: Australia (Melbourne 1) [https]  5: Australia (Melbourne 2) [https]  6: Australia (Perth) [https]
 7: Austria [https]                  8: Belgium (Ghent) [https]          9: Brazil (BA) [https]
10: Brazil (PR) [https]             11: Brazil (RJ) [https]             12: Brazil (SP 1) [https]
13: Brazil (SP 2) [https]           14: Bulgaria [https]                15: Chile (Santiago) [https]
16: China (Hong Kong) [https]       17: China (Lanzhou) [https]         18: China (Shanghai) [https]
19: Colombia (Cali) [https]         20: Czech Republic [https]          21: Denmark [https]
22: Ecuador (Cuenca) [https]        23: Ecuador (Quito) [https]         24: Estonia [https]
25: France (Lyon 1) [https]         26: France (Lyon 2) [https]         27: France (Marseille) [https]
28: France (Montpellier) [https]    29: Germany (Erlangen) [https]      30: Germany (Göttingen) [https]
31: Germany (Münster) [https]       32: Germany (Regensburg) [https]    33: Greece [https]
34: Hungary [https]                 35: Iceland [https]                 36: Indonesia (Jakarta) [https]
37: Ireland [https]                 38: Italy (Padua) [https]           39: Japan (Tokyo) [https]
40: Japan (Yonezawa) [https]        41: Korea (Busan) [https]           42: Korea (Gyeongsan-si) [https]
43: Korea (Seoul 1) [https]         44: Korea (Ulsan) [https]           45: Malaysia [https]
46: Mexico (Mexico City) [https]    47: Morocco [https]                 48: Norway [https]
49: Philippines [https]             50: Russia [https]                  51: Spain (Madrid) [https]
52: Sweden [https]                  53: Switzerland [https]             54: Turkey (Denizli) [https]
55: Turkey (Mersin) [https]         56: UK (Bristol) [https]            57: UK (London 1) [https]
58: USA (CA 1) [https]              59: USA (IA) [https]                60: USA (KS) [https]
61: USA (MI 1) [https]              62: USA (MI 2) [https]              63: USA (OR) [https]
64: USA (TN) [https]                65: USA (TX 1) [https]              66: Uruguay [https]
67: (other mirrors)

Selection: 31
>
```

Package installation: Choose secure CRAN mirror

- `chooseCRANmirror(graphics = TRUE)`



Package installation: Choose secure Bioconductor mirror

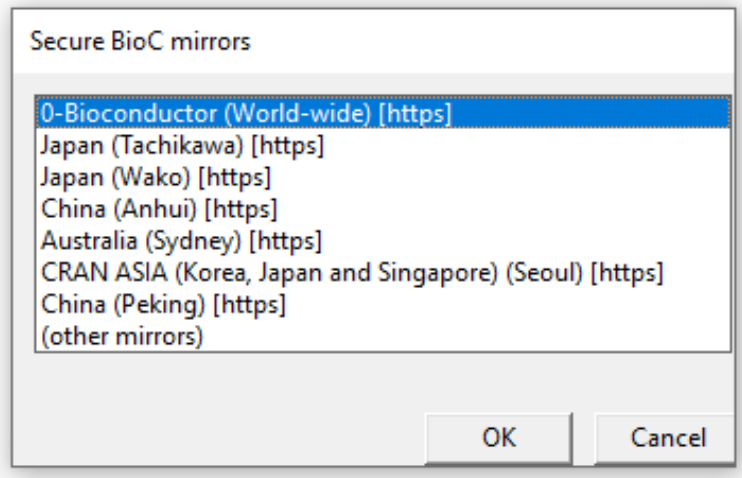
- `chooseBioCmirror()`

```
> chooseBioCmirror()
Secure BioC mirrors

1: 0-Bioconductor (World-wide) [https]
2: Japan (Tachikawa) [https]
3: Japan (Wako) [https]
4: China (Anhui) [https]
5: Australia (Sydney) [https]
6: CRAN ASIA (Korea, Japan and Singapore) (Seoul) [https]
7: China (Peking) [https]
8: (other mirrors)

Selection: 1
```

Package installation: Choose secure Bioconductor mirror

- `chooseBioCmirror(graphics = TRUE)`

```
>
>
>
>
>
>
>
>
> chooseBioCmirror(graphics = TRUE)
```



Package installation: Already installed?

- `installed.packages()` returns matrix of installed packages (each row one package).
- Provides information on package name, version, license, etc.

```
installed.packages()[1:10, c("Package","LibPath","Version","License")]
```

```
> installed.packages()[1:10, c("Package", "LibPath", "Version", "Built", "License")]
             Package        LibPath                 Version    Built   License
assertthat   "assertthat"   "C:/R/R-3.6.1/library" "0.2.1"    "3.6.1" "GPL-3"
backports    "backports"    "C:/R/R-3.6.1/library" "1.1.5"    "3.6.1" "GPL-2 | GPL-3"
base         "base"         "C:/R/R-3.6.1/library" "3.6.1"    "3.6.1" "Part of R 3.6.1"
BH           "BH"           "C:/R/R-3.6.1/library" "1.69.0-1" "3.6.0" "BSL-1.0"
BiocGenerics "BiocGenerics" "C:/R/R-3.6.1/library" "0.30.0"   "3.6.0" "Artistic-2.0"
bit          "bit"          "C:/R/R-3.6.1/library" "1.1-14"   "3.6.0" "GPL-2"
bit64        "bit64"        "C:/R/R-3.6.1/library" "0.9-7"    "3.6.0" "GPL-2"
bitops       "bitops"       "C:/R/R-3.6.1/library" "1.0-6"    "3.6.0" "GPL (>= 2)"
blob         "blob"         "C:/R/R-3.6.1/library" "1.2.0"    "3.6.1" "GPL-3"
boot         "boot"         "C:/R/R-3.6.1/library" "1.3-22"   "3.6.1" "Unlimited"
>
```

`nrow(installed.packages())` gives package number
`rownames(installed.packages())` gives package names

```
> nrow(installed.packages())
[1] 115
> rownames(installed.packages())[1:10]
 [1] "assertthat"   "backports"    "base"         "BH"           "BiocGenerics"
 [6] "bit"          "bit64"        "bitops"       "blob"         "boot"
>
```

Package installation: install.packages()

`install.packages()` downloads and installs packages from currently chosen repositories (or local files).

Most important arguments:

`pkgs`	`Character vector of names of packages.`
`repos`	`Character vector of repository URLs.`
`dependencies`	`Install also dependencies of packages? TRUE\FALSE or character vector, subset of c("Depends", "Imports", "LinkingTo","Suggests", "Enhances").`
`verbose`	`TRUE\FALSE for giving progress report.`
`quiet`	`TRUE\FALSE for reducing amount of output.`

Package installation: available.packages()

- `available.packages()` returns **matrix of all packages** available at currently selected repositories.
- Provides information on package name, version, license, etc.
- Package names can be searched using `grep()` even if exact name is unknown.
- E.g., you can't remember the name of a nice package you want to test - `jimma`, `rimma`, `yimma`, ... ?!

```
> packages <- available.packages()
> packages[grep("imma", packages[,"Package"]), 1:2]
          Package    Version
timma     "timma"    "1.2.1"
Glimma    "Glimma"   "1.12.0"
limma     "limma"    "3.40.6"
limmaGUI "limmaGUI" "1.60.0"
> |
```

Package installation: new.packages()

- `new.packages()` returns **vector of not already installed packages** available at currently selected repositories.
- Alternative to `available.packages()`.
- Advantage: Shorter list including only not installed packages.
- However, provides only package names.
- Can be also searched using `grep()`.

```
> packages <- new.packages()
> packages[grep("imma",packages)]
[1] "Glimma"   "limma"    "limmaGUI" "timma"
```

Package installation: install.packages()

Example: installing `limma`.

```
> install.packages(pkgs="limma", dependencies = TRUE)
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the ap
propriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Warning in install.packages :
  dependencies 'GO.db', 'org.Hs.eg.db' are not available
also installing the dependencies 'sys', 'askpass', 'openssl', 'affyio', 'BiocManager', 'preprocessCore', 'zlibbio
c', 'IRanges', 'S4Vectors', 'base64', 'affy', 'AnnotationDbi', 'BiasedUrn', 'Biobase', 'ellipse', 'illuminaio', 'l
ocfit', 'statmod', 'vsn'


  There is a binary version available but the source version is
  later:
            binary  source needs_compilation
BiocManager 1.30.9 1.30.10             FALSE

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/sys_3.3.zip'
Content type 'application/zip' length 59886 bytes (58 KB)
downloaded 58 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/askpass_1.1.zip'
```

(...)

Package installation: old.packages()

`old.packages()` indicates packages which have a later version at currently selected repositories.

```
> old.packages()
           Package      LibPath                   Installed  Built   ReposVer
boot       "boot"       "C:/R/R-3.6.1/library" "1.3-22"   "3.6.1" "1.3-23"
foreign    "foreign"    "C:/R/R-3.6.1/library" "0.8-71"   "3.6.1" "0.8-72"
KernSmooth "KernSmooth" "C:/R/R-3.6.1/library" "2.23-15"  "3.6.1" "2.23-16"
mgcv       "mgcv"       "C:/R/R-3.6.1/library" "1.8-28"   "3.6.1" "1.8-31"
nlme       "nlme"       "C:/R/R-3.6.1/library" "3.1-140"  "3.6.1" "3.1-142"
R6         "R6"         "C:/R/R-3.6.1/library" "2.4.0"    "3.6.1" "2.4.1"
Rcpp       "Rcpp"       "C:/R/R-3.6.1/library" "1.0.2"    "3.6.1" "1.0.3"
survival   "survival"   "C:/R/R-3.6.1/library" "2.44-1.1" "3.6.1" "3.1-7"
           Repository
boot       "https://cran.uni-muenster.de/src/contrib"
foreign    "https://cran.uni-muenster.de/src/contrib"
KernSmooth "https://cran.uni-muenster.de/src/contrib"
mgcv       "https://cran.uni-muenster.de/src/contrib"
nlme       "https://cran.uni-muenster.de/src/contrib"
R6         "https://cran.uni-muenster.de/src/contrib"
Rcpp       "https://cran.uni-muenster.de/src/contrib"
survival   "https://cran.uni-muenster.de/src/contrib"
> 
```

Package installation: update.packages()

`update.packages()` downloads & updates packages indicated by `old.packages()`. Use argument `ask=FALSE` to be not asked for every package separately whether to install it. Alternatively, `ask="graphics"` allows package selection via a pop-up menu.

```
> update.packages(ask=FALSE)

  There is a binary version available but the source version is later:
   binary source needs_compilation
R6  2.4.0  2.4.1             FALSE

trying URL 'https://cran.uni-muenster.de/bin/windows/contrib/3.6/boot_1.3-23.zip'
Content type 'application/zip' length 639824 bytes (624 KB)
downloaded 624 KB

trying URL 'https://cran.uni-muenster.de/bin/windows/contrib/3.6/foreign_0.8-72.zip'
Content type 'application/zip' length 325973 bytes (318 KB)
downloaded 318 KB

trying URL 'https://cran.uni-muenster.de/bin/windows/contrib/3.6/KernSmooth_2.23-16.zip'
Content type 'application/zip' length 115200 bytes (112 KB)
downloaded 112 KB

trying URL 'https://cran.uni-muenster.de/bin/windows/contrib/3.6/mgcv_1.8-31.zip'
Content type 'application/zip' length 3042876 bytes (2.9 MB)
downloaded 2.9 MB
```

Package installation: remove.packages()

`remove.packages()` removes packages indicated by the argument `pkgs`.

```
> remove.packages(pkgs="limma")
Removing package from 'C:/R/R-3.6.1/library'
(as 'lib' is unspecified)
> library(limma)
Error in library(limma) : there is no package called 'limma'
```

Loading packages: library()

`library()` loads and attach add-on packages.

Most important arguments:

`package`	`Name of a package.`
`logical.return`	`TRUE\FALSE for indicating success by TRUE\FALSE.`
`verbose`	`TRUE\FALSE for printing additional diagnostics.`

`search()`, `ls()`, `objects()` give lists of attached packages and/or R objects.

Loading packages: library() & search()

Examples:

```
> library(package="limma", logical.return=TRUE, verbose=TRUE)
[1] TRUE
> library(limma)
> library(package="limma", logical.return=TRUE, verbose=TRUE)
[1] TRUE
Warning message:
In library(package = "limma", logical.return = TRUE, verbose = TRUE) :
  package 'limma' already present in search()
>
```

```
> search()
 [1] ".GlobalEnv"        "package:limma"     "tools:rstudio"
 [4] "package:stats"     "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"  "package:methods"
[10] "Autoloads"         "package:base"
>
```

Using packages: help() pages of packages

```
help("limma")
```

01.Introduction {limma} R Documentation

Introduction to the LIMMA Package

Description

LIMMA is a library for the analysis of gene expression microarray data, especially the use of linear models for analysing designed experiments and the assessment of differential expression. LIMMA provides the ability to analyse comparisons between many RNA targets simultaneously in arbitrary complicated designed experiments. Empirical Bayesian methods are used to provide stable results even when the number of arrays is small. The linear model and differential expression functions apply to all gene expression technologies, including microarrays, RNA-seq and quantitative PCR.

Details

There are three types of documentation available:

1. The *LIMMA User's Guide* can be reached through the "User Guides and Package Vignettes" links at the top of the LIMMA contents page. The function `limmaUsersGuide` gives the file location of the User's Guide.

2. An overview of limma functions grouped by purpose is contained in the numbered chapters at the foot of the LIMMA package index page, of which this page is the first

Using packages: help() pages of packages

```
help("normalizeBetweenArrays")
```

normalizeBetweenArrays {limma} R Documentation

Normalize Between Arrays

Description

Normalizes expression intensities so that the intensities or log-ratios have similar distributions across a set of arrays.

Usage

```
normalizeBetweenArrays(object, method=NULL, targets=NULL, cyclic.method="fast", ...)
```

Arguments

object	a numeric matrix, EListRaw, RGList or MAList object containing un-normalized expression data. If a matrix, then it is assumed to contain log-transformed single-channel data.
method	character string specifying the normalization method to be used. Choices for single-channel data are "none", "scale", "quantile" or "cyclicloess". Choices for two-color data are those previously mentioned plus "Aquantile", "Gquantile", "Rquantile" or "Tquantile". A partial string

Using packages: help() pages of packages

```
browseVignettes(package="limma")
```

```
        Vignettes found by "browseVignettes(package = "limma")"

Vignettes in package limma
```

- Limma One Page Introduction - PDF source

Using packages: Function name ambiguities

- Some functions from various packages have the same name.
- This may be very confusing!
- If you want to use explicitly a function from a specific package use the `::` operator.
- Usage: `package::function`
- Example: `openxlsx::read.xlsx()`

Data preprocessing

Data preprocessing

- now: preprocessing for quantitative proteomics datasets
- important preprocessing steps:
  - handling of missing values
  - normalization
- we will not cover preprocessing for raw files/spectra

Missing Values

- missing values occur when no data value is available for a variable
- how missing values are handled can have a strong effect on the analysis results
- different types of missingness of a variable X:
  - missing completely at random (independent of X and other variables)
  - missing at random (probability of missingness depends on other variables)
  - missing not at random (value of X is related to the reason it is missing)

Reasons for missing values in proteomics datasets

Biological reasons:

- protein is not present in the sample
- protein is present, but below detection limit

Experimental reasons:

- peptides derives from the protein are not selected for MS/MS
- MS/MS spectra produced for the corresponding peptides are not identified with high enough confidence

Data preprocessing reasons:

- additional filter, e.g. at least 2 unique peptides
- values may be filtered out due to lack of confidence (e.g. protein level FDR)
- e.g. MaxQuant: protein quantity set to 0, if there are not enough peptide ratios for the LFQ-normalization step

Missing values - MaxQuant

	A	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT
1	Protein IDs	LFQ intensity	LFQ intensity	LFQ intensity	LFQ intensity	LFQ intensity	LFQ intensity	LFQ intensity	LFQ intensity	LFQ intensity	LFQ intensity
2	A0A024QZX5;A0A	103120000	63538000	106440000	87570000	73349000	118490000	122110000	147700000	57634000	44445000
3	A0A024R4M0;P46	5790400	11242000	0	0	5865100	0	0	8130200	7725800	64144000
4	A0A024R571;Q9H	21333000	21231000	6750900	33134000	61434000	37726000	27669000	29360000	34895000	36724000
5	A0A1B0GWI2;A0A	6483900	4971000	5407600	0	0	0	15560000	0	0	0
6	A0A024R6I7;A0A0	273110000	0	0	0	415720000	231880000	170450000	0	77606000	0
7	A0A024RA52;P25	27201000	18075000	20193000	34648000	25855000	57069000	17675000	6078100	0	14778000
8	A0A0G2JL47;A0A0	0	0	0	0	4455500	0	0	0	0	0
9	A0A0C4DH90;A0A	0	0	1034100000	0	0	0	0	0	0	0
10	A0A075B6I0	93741000	0	248030000	0	0	0	273300000	0	0	0
11	A0A075B6I9;P042	0	0	103170000	0	339850000	79526000	92197000	79838000	0	0
12	A0A075B6J9	0	5006300	0	0	24838000	0	120130000	0	4247200	0
13	A0A075B6K2	0	0	34257000	0	0	0	0	0	0	0
14	A0A075B6K4;P017	183290000	142910000	339380000	27835000	426830000	41551000	125020000	53773000	17543000	76506000
15	A0A075B6K5;P807	154030000	161710000	223690000	0	236050000	270420000	0	257760000	260630000	0
16	A0A075B6Q5	8969300	0	0	0	0	0	0	0	0	0
17	A0A075B6R2	20478000	61140000	107980000	13775000	11235000	53081000	41233000	0	0	7177100
18	A0A0C4DH68;A0A	0	0	0	0	0	0	0	590390000	0	0
19	A0A0C4DH67;A0A	40041000	60561000	108000000	0	115260000	52592000	17819000	0	61582000	36194000
20	A0A075B730;P581	0	0	0	0	0	0	0	0	0	2661600
21	A0A075B785;A0A1	0	0	0	0	0	0	0	0	0	265000
22	A0A075B7B1;O15(	0	0	0	0	0	0	0	0	1955000000	0
23	A0A075B7B8	146510000	94722000	156260000	35300000	85495000	51199000	175450000	85534000	0	0
24	A0A075B7D8	29282000	55143000	75176000	12130000	67264000	28192000	109770000	19428000	35409000	4770600
25	A0A075B7D9;H3B	5767000	0	0	0	4577100	6169700	6582700	4150200	0	5979700
26	A0A075B7E8	0	0	0	0	0	0	0	0	2689800	0
27	A0A087WSV8;P80	1034200000	1366900000	1836300000	800310000	776780000	527980000	2053500000	1147500000	1285400000	980880000
28	E9PIR7;F8W809;A	2272200	0	0	7492200	7253800	4113300	0	3018100	0	0
29	A0A087WT27;J3KI	0	0	0	13819000	0	0	0	0	0	0
30	E9PQS1;A0A087W	0	0	0	0	0	0	0	10814000	0	0

Missing values - Spectronaut Pulsar

	A	G	H	I	J	K	L	M	N	O	P
1	PG.ProteinAc	[1] QExHF06	[2] QExHF06	[3] QExHF06	[4] QExHF06	[5] QExHF06	[6] QExHF06	[7] QExHF06	[8] QExHF06	[9] QExHF06	[10] QExHF0
2	A0A075B5N9	618140.63	470208.91	Filtered	Filtered	781414.56	Filtered	Filtered	Filtered	Filtered	1058049.1
3	A0A075B5P2	45745.563	41125.871	Filtered	23042.35	44224.266	10818.64	13955.237	Filtered	Filtered	74740.023
4	A0A075B5P3	Filtered	19667.855	Filtered	490833.19	Filtered	Filtered	Filtered	Filtered	Filtered	9440.125
5	A0A075B5P4	88045.828	47362.102	Filtered	Filtered	89857.43	Filtered	Filtered	Filtered	Filtered	97300.547
6	A0A075B5P6	Filtered	6831.8857	10210.884	Filtered	Filtered	236460.86	225424.78	61737.527	9862.832	Filtered
7	A0A075B5T4	258741.03	318471.03	Filtered	168419.63	410943.59	95607.055	110008.93	Filtered	Filtered	762567.06
8	A0A087WNZ	NaN	Filtered	Filtered	14055.865	NaN	Filtered	Filtered	Filtered	Filtered	Filtered
9	A0A087WNZ	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered
10	A0A087WPL	Filtered	NaN	NaN	NaN	Filtered	NaN	Filtered	NaN	NaN	NaN
11	A0A087WQ9	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	NaN	Filtered	29038.57
12	A0A087WR4	4986.2119	3685.49	Filtered	5831.5195	5395.8906	Filtered	Filtered	Filtered	Filtered	6739.0107
13	A0A087WR5	Filtered	NaN	Filtered	NaN	NaN	Filtered	Filtered	Filtered	Filtered	NaN
14	A0A087WR9	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered
15	A0A087WRI	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered
16	A0A087WS1	24580.932	37076.348	7916.3228	13286.828	10575.333	5486.104	Filtered	14163.547	23873.461	81431.703
17	A0A087WS4	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	NaN
18	A0A0A0MQ7	Filtered	Filtered	5811.3013	3885.6641	Filtered	3827.0281	11074.24	Filtered	4450.0674	Filtered
19	A0A0A0MQA	65496.52	93936.945	111184.16	140451.75	122204.32	124442.08	142380.7	191849.08	122788.52	147775.2
20	A0A0A0MQC	9144.5059	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	11517.721
21	A0A0A0MQN	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	NaN	Filtered	Filtered
22	A0A0A6YW6	219568.98	101649.59	135950.09	112404.94	181963.38	241055.64	249178.69	438168.69	161391.84	313495.5
23	A0A0A6YWE	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered
24	A0A0A6YX26	14257.832	10519.646	12048.069	5314.4165	9227.3584	18353.398	24446.9	33099.422	15562.273	10709.649
25	A0A0A6YX73	NaN	NaN	NaN	NaN	NaN	Filtered	NaN	NaN	NaN	NaN
26	A0A0A6YXF6	NaN	NaN	1623.8696	NaN	NaN	Filtered	Filtered	2155.5095	1027.8027	NaN
27	A0A0A6YXH3	95685.984	74517.148	148655.63	185537.63	80724.891	104292.41	376379.25	149180.09	81319.219	129515.31
28	A0A0A6YXV1	15454.54	9918.833	18011.652	10534.896	13475.811	29441.141	28358.637	23006.416	18081.477	5689.3311
29	A0A0B4J1H7	Filtered	Filtered	Filtered	229111.88	Filtered	Filtered	Filtered	Filtered	Filtered	Filtered
30	A0A0G2JDI9	4655.3018	2489.9185	Filtered	2871.1023	6735.7471	Filtered	Filtered	Filtered	Filtered	5270.5479

Missing values

- Different codings for missing values depending on software and output settings (NA, NaN, 0, Filtered, ?, empty cell)
- Number of missing values often very high
- Proteins with missing values might be interesting (on/off-proteins)

Handling of missing values

- remove proteins with missing values
- perform analysis only on valid values
- impute missing values

How does R represent missing values?

- Missing values are represented by `NA` (Not Available)
- `NA`s are present in data sets or returned by functions
- different versions of `NA` depending on vector type (`NA_integer_`, `NA_real_`, `NA_complex_`, `NA_character_`)
- in character vectors, missing `NA` is shown without quotation marks (to distinguish it from the character "NA"):

```
c("x", "NA", NA)  [1] "x" "NA" NA
```

- results of impossible calculations (e.g. `0/0`, `log(-1)`) are represented by `NaN` (Not a Number)
- `NULL`: is empty an object and is returned when an expression or function results in an undefined value

How to read in datasets with missing values?

Most reading functions have an argument to control which entries are recognized as missing values

```
read.table(file, na.strings = "NA")
```

Values that match to one of the values given in `na.strings` show up as a missing value (`NA`) in R.

Warning! If `na.strings` is not set properly, numeric columns containing missing values may be read in as characters or factors! (see live presentation)

How do R functions handle missing values?

In general, NAs are treated as a unknown value, that could range from $-\infty$ to ∞ (in case of numeric values):

```
NA == 3                          [1] NA
x <- c(1,2,3,NA); x + 1          [1] 2 3 4 NA
mean(x)                          [1] NA
mean(x, na.rm = TRUE)            [1] 2
```

The argument `na.rm` is available in many R functions (e.g. mean, median, min, max, sum). If set to `TRUE`, missing values in the vector are deleted before calculation.

Missing values - correlation

Different options to handle missing values for calculation of correlation matrices

`cor(x, use = ...)`

- `"everything"`: returns NAs for all comparisons with a variable containing NAs
- `"complete.obs"`: deletes all rows with missing values before calculation
- `pairwise.complete.obs`: use complete pairs of observations for each combination of variables

"everything"

X_1	X_2	X_3
9	8	7
3	9	1
8	3	NA
8	4	6

"complete.obs"

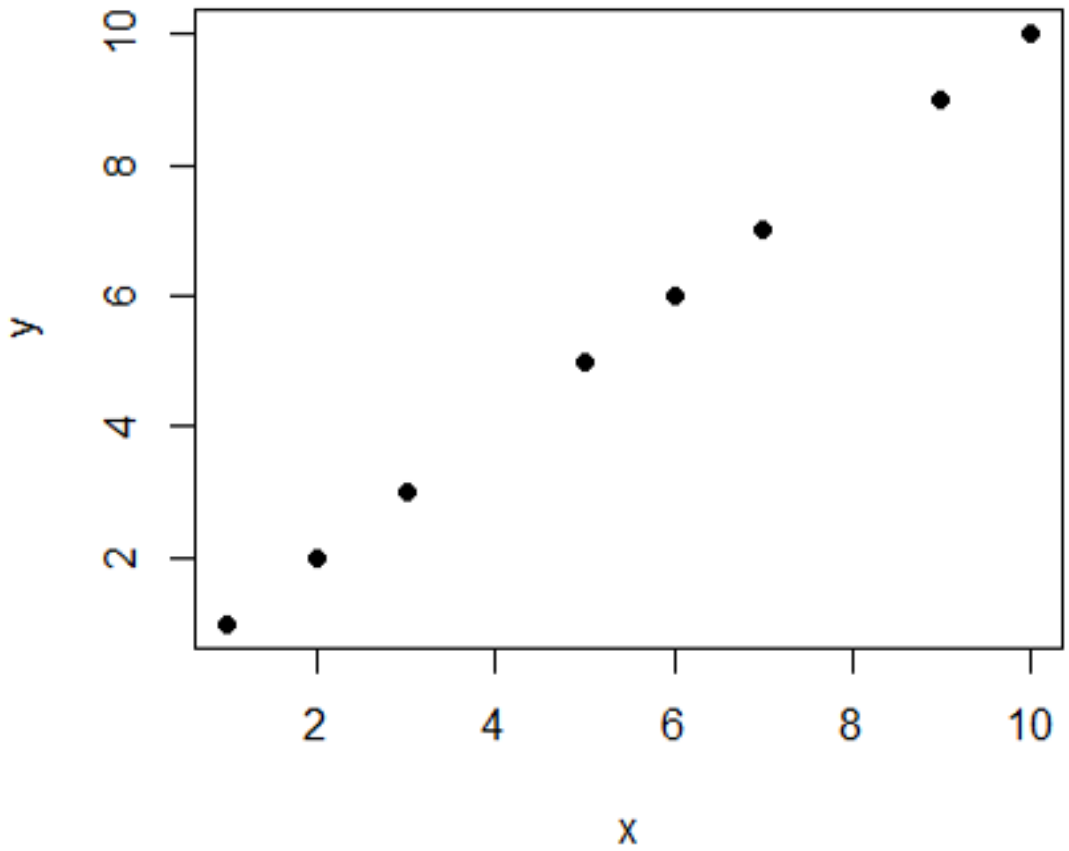
X_1	X_2	X_3
9	8	7
3	9	1
8	3	NA
8	4	6

"pairwise.complete.obs"

X_1	X_2	X_3
9	8	7
3	9	1
8	3	NA
8	4	6

Missing values for plotting

```
x <- 1:10
y <- c(1:3, NA, 5:7, NA, 9:10)
plot(x, y, pch = 16)
```



Missing values are often ignored without a warning!

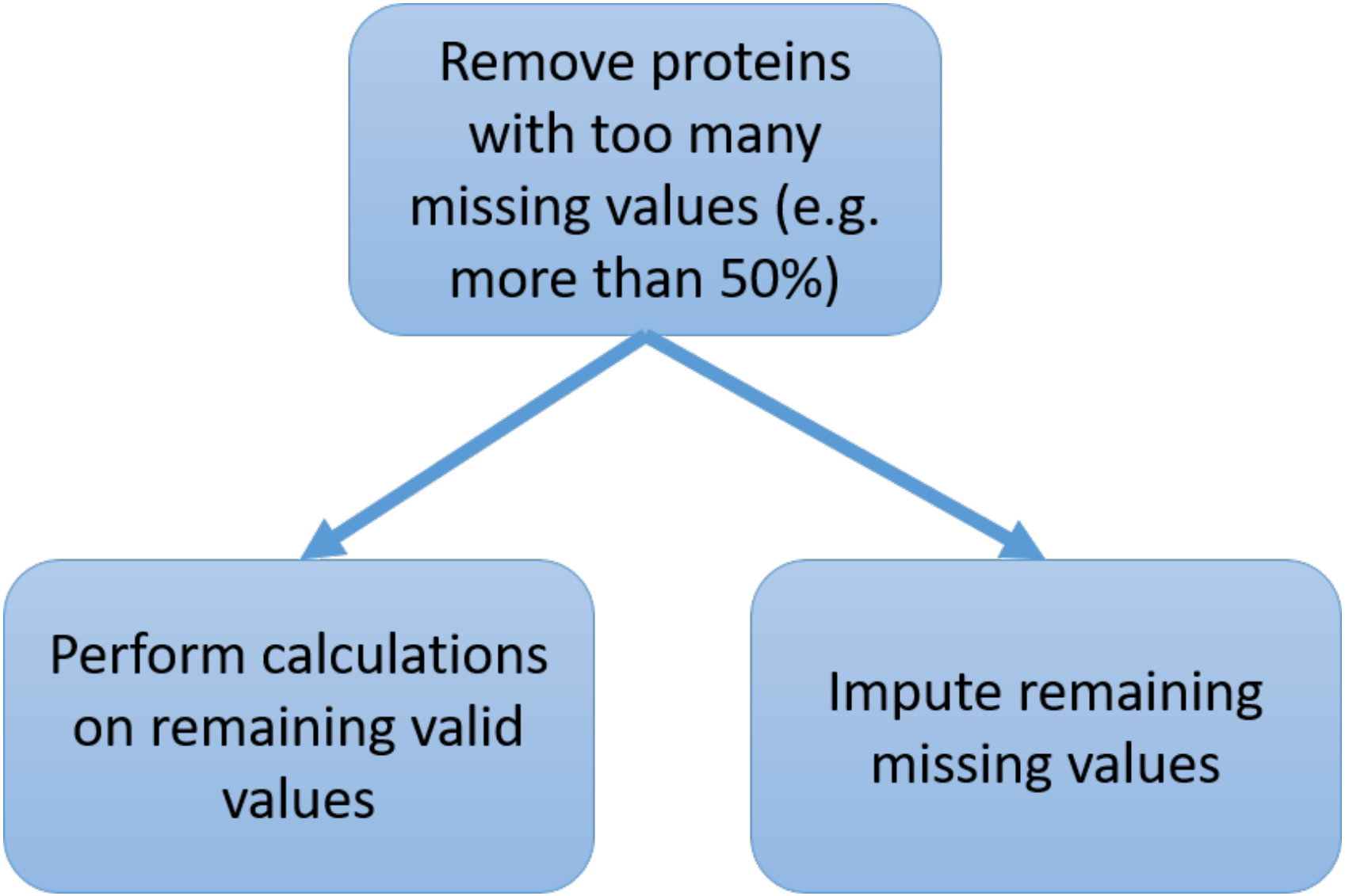
Useful R functions to handle missing values

```
x <- c(1,2,3,NA)
na.omit(x)
[1] 1 2 3
attr(,"na.action")
[1] 4
attr(,"class")
[1] "omit"
```

Warning! For matrices and data.frames, `na.omit` will delete all rows that contain at least one missing value.

`is.na(x)` `[1] FALSE FALSE FALSE TRUE` (why not `x == NA`?)
`anyNA(x)` `[1] TRUE`
`complete.cases()`

Handling of missing values



Data Imputation

Data Imputation = replace missing values with valid values

Imputation methods

- mean or median of the protein
- random value based on distribution of non-missing values
- small values (e.g. 0 or LOD/2, LOD = limit of detection)
- machine learning based

Disadvantages

- imputation can have a huge impact on result
- imputation of constant value can lead to underestimated variance $\rightarrow$ risk of false positives
- biomarker candidates with too many imputed values may be worthless

on/off proteins

- proteins that are present in one group and absent in the other group
- $\approx$ proteins that have valid values in one group and missing values in the other group
- higher confidence for found on/off proteins with high sample size
- on/off proteins are often forgotten or filtered out by the software
- t-test not possible $\rightarrow$ not p-value
- fold change $= \infty$?
- cannot be displayed in volcano plot $\rightarrow$ separate list

Normalization - Motivation

Data contains technical and biological variation (but we are only interested in biological differences)

Reasons for technical bias:

- small variations in experimental conditions and sample handling (temperature, age of column, pipetting)
- often exact reasons for bias are unknown

Aims of normalization

- reduce/remove technical bias while keeping biological differences
- make samples more comparable
- make following statistical analysis more reliable

Normalization

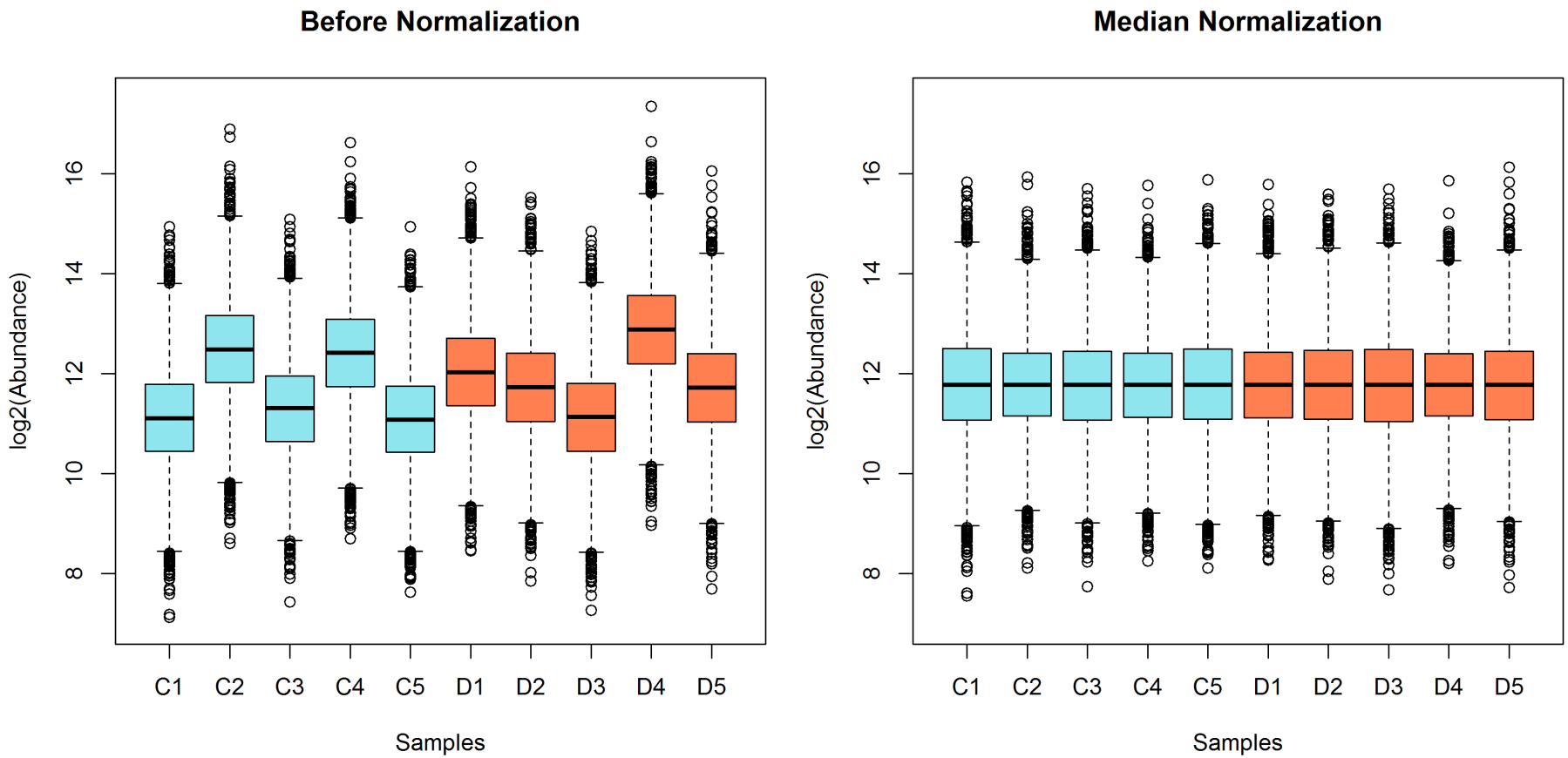
Assumptions:

- high-troughput data
- "true" intensity distribution is similar over all samples
- most proteins are not differentially expressed between groups

- most normalization methods were developed for genomics and later adapted to proteomics data
- often, data are log-transformed before normalization

Median normalization

shift or scale samples to have the same median



Quantile Normalization

Original dataset

	S1	S2	S3
Prot1	100	50	115
Prot2	85	140	45
Prot3	150	70	80
Prot4	95	65	160

1) Sort Values in each column

S1	S2	S3
85	50	45
95	65	80
100	70	115
150	140	160

2) Replace values with row mean

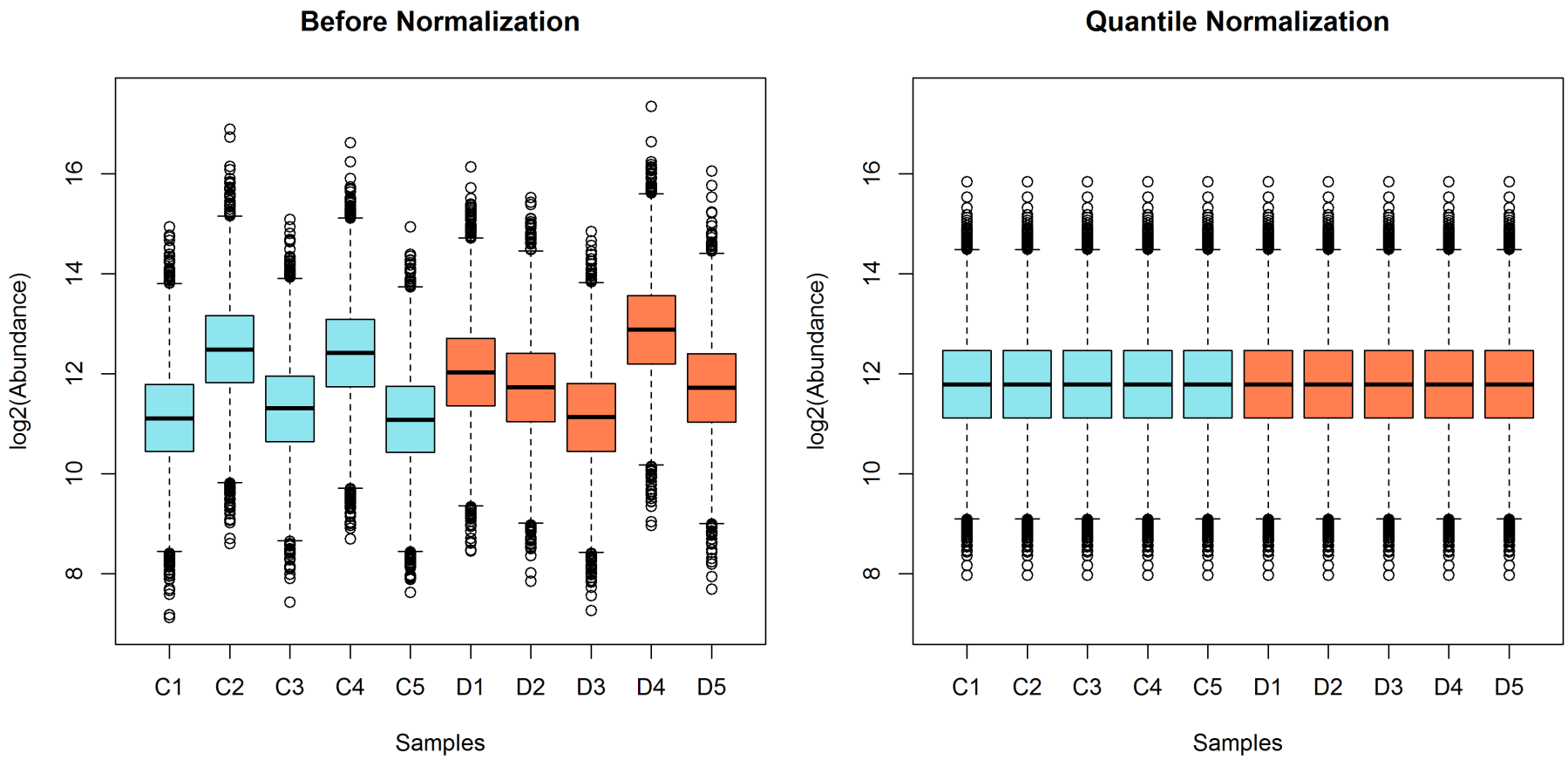
S1	S2	S3
60	60	60
80	80	80
95	95	95
150	150	150

3) Reconstruct original order

	S1	S2	S3
Prot1	95	60	95
Prot2	60	150	60
Prot3	150	95	80
Prot4	80	80	150

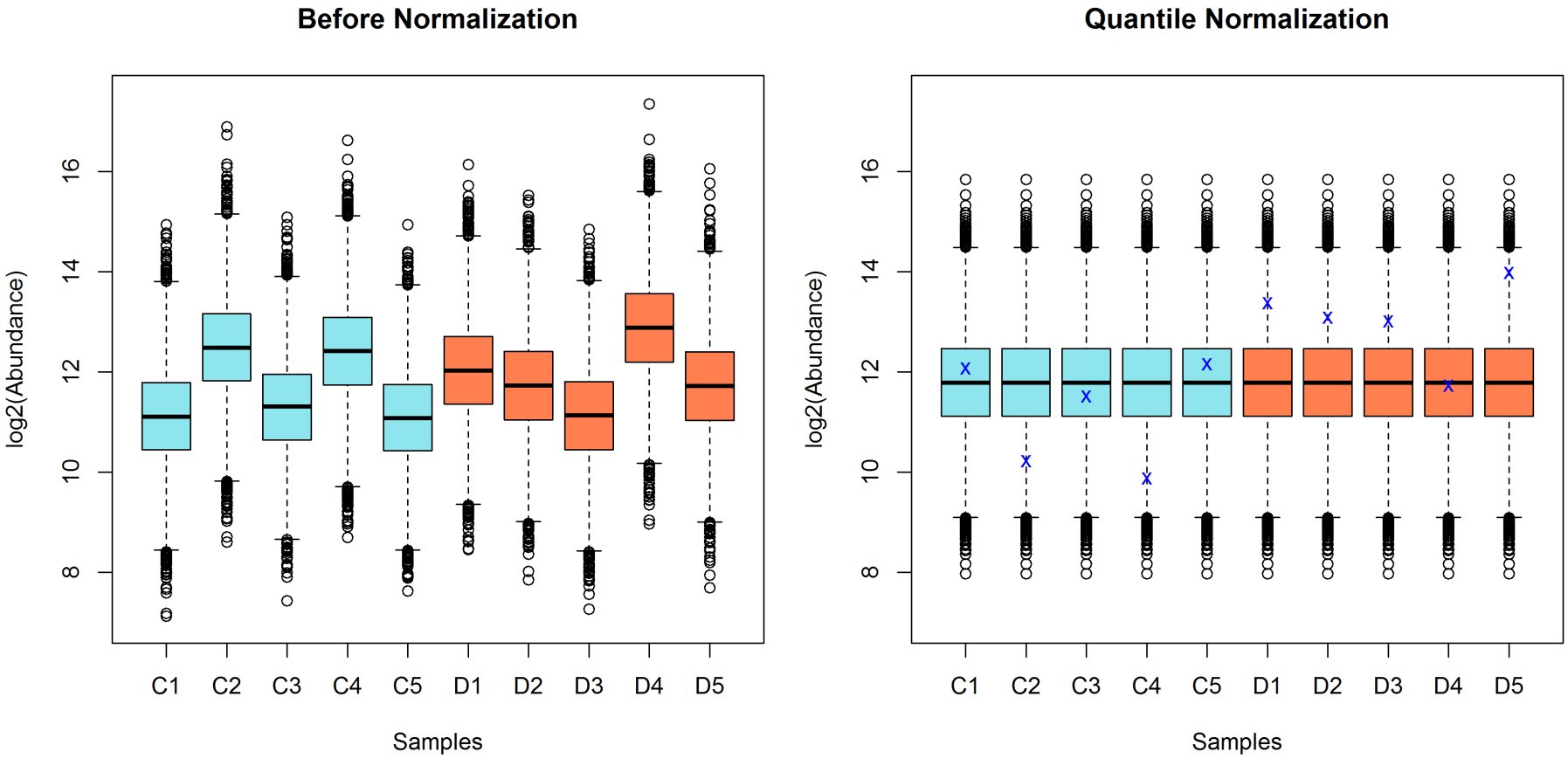
Quantile normalization

normalize all samples to the same distribution



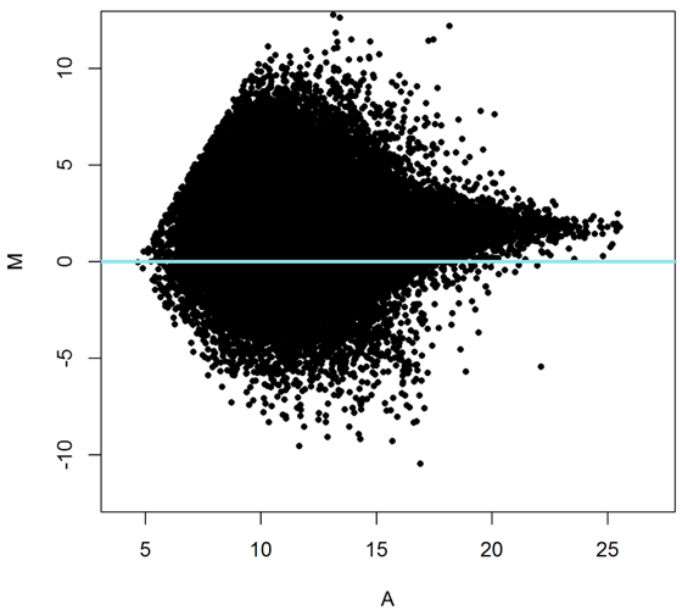
Quantile normalization

normalize all samples to the same distribution



LOESS Normalization

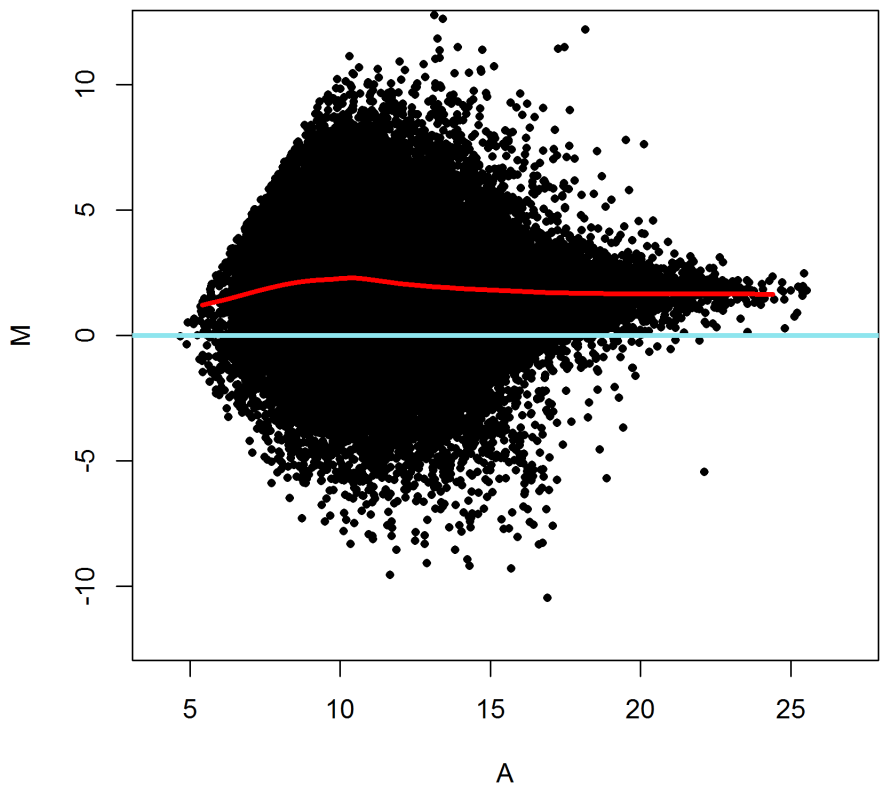
- LOESS = LOcal regrESSion
- alternative name: LOWESS = LOcally WEighted Scatterplot Smoothing
- based on MA-Plot (MA = Minus vs. Average)
- X-axis: average of log2-abundances (A)
- Y-axis: difference of log2-abundances (M)

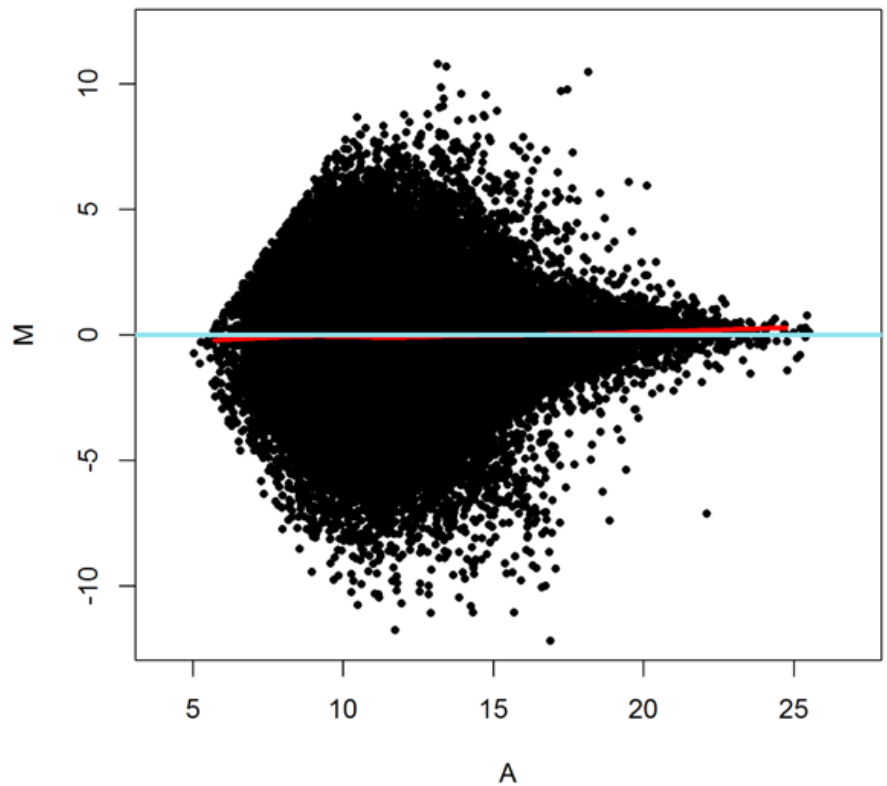


$$A = \frac{1}{2}(log2(X_1) + log2(X_2))$$
$$= \frac{1}{2} log2(X_1 \cdot X_2)$$
$$M = log2(X_1) - log2(X_2) = log2(X_1/X_2)$$

LOESS Normalization

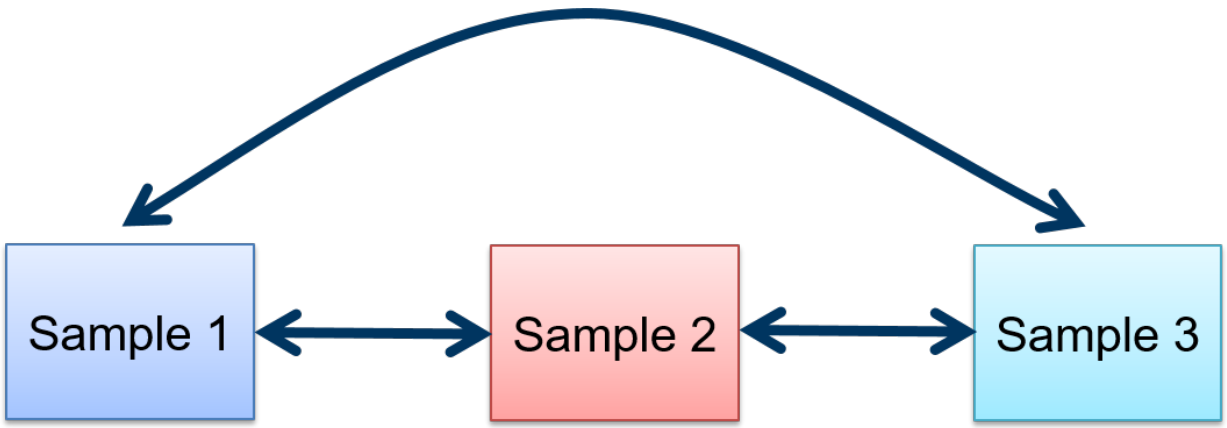
- Aim: symmetry of the scatterplot around M = 0
- Estimation of local linear regression curve to remove bias
- Back-transformation of M and A values to intensities
- left: before normalization, right: after normalization





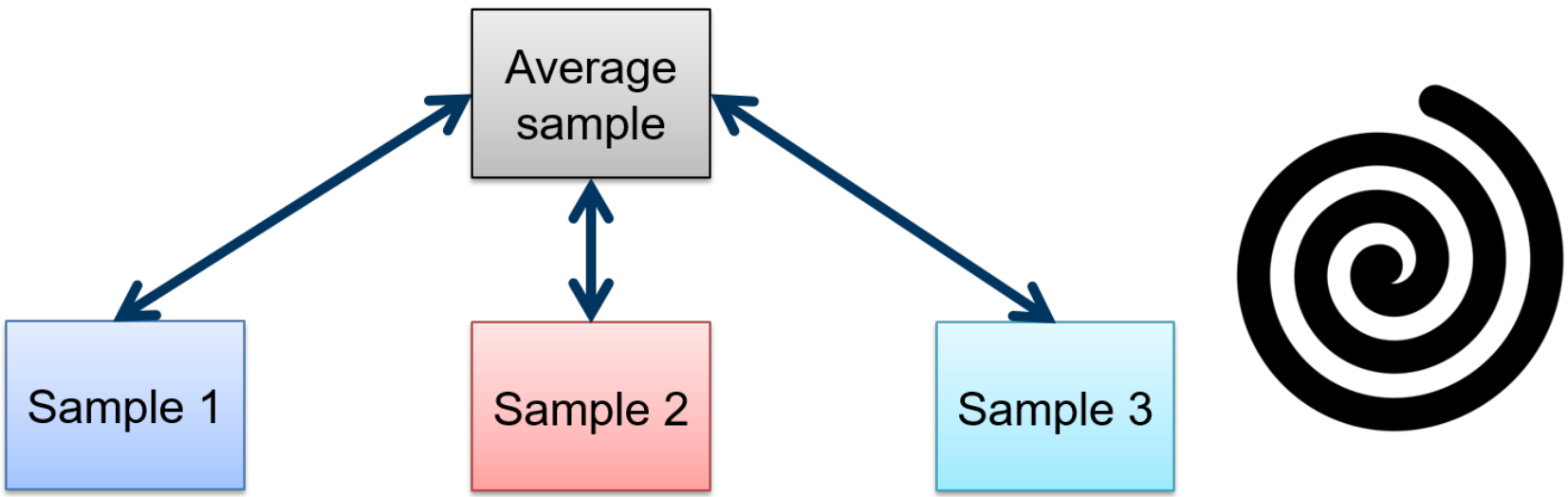
cyclic LOESS

- with LOESS, only two samples can be normalized at once
- *cyclic LOESS*: normalization is cycled through all possible pairs of samples and repeated for several iterations
- number of pairs: $\binom{n}{2}$



fast cyclic LOESS

- compute average of all samples and normalize all samples against this reference sample
- usually faster than cyclic loess, because less comparisons are necessary



Normalization in R

R package `limma` (Linear Models for Microarray Data)
`normalizeBetweenArrays(object, method, cyclic.method)`

`object`: data matrix
`method`: `"scale"` (median), `"quantile"`, `"cyclicloess"`
`cyclic.method`: `"fast"` or `"pairs"` (for LOESS)

MA Plots in R

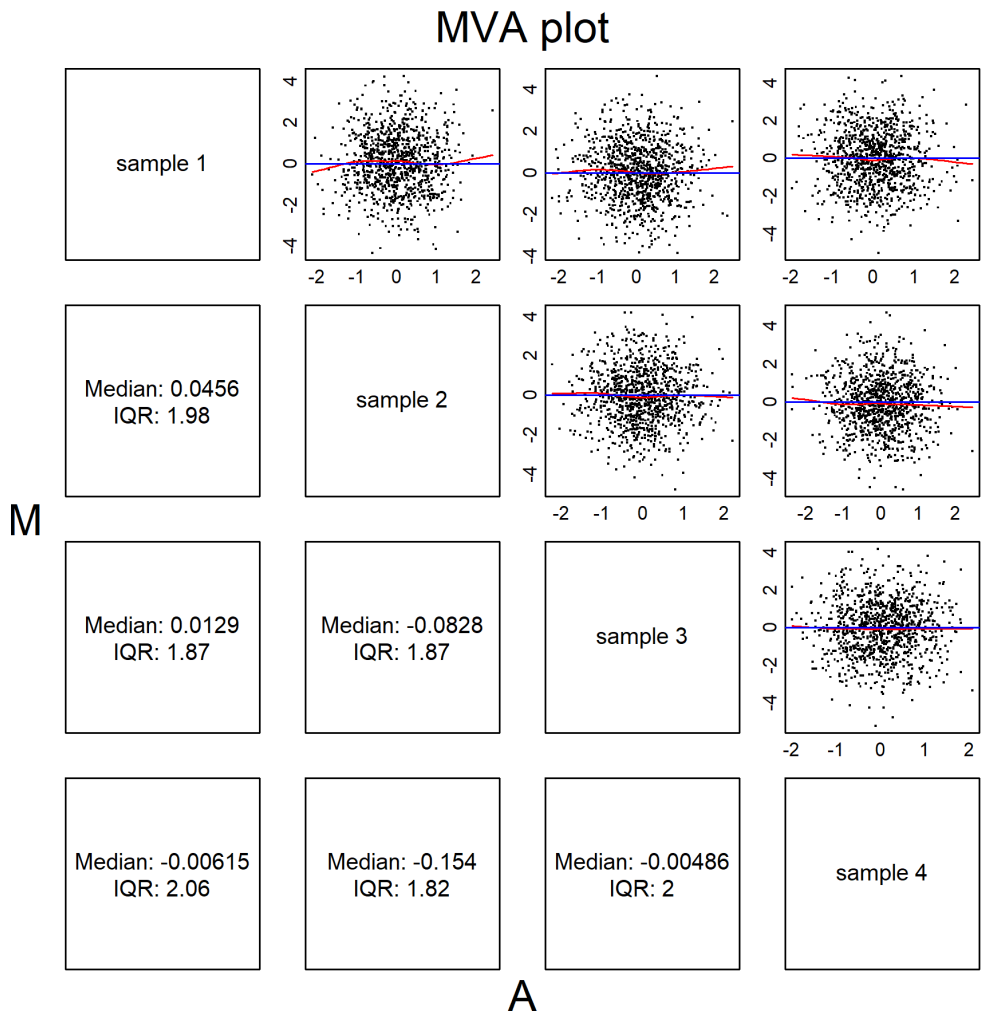
package	function	comment
`limma`	`plotMA`	one sample vs. average of all other samples
`affy`	`MAPlot`	data needs to be "AffyBatch" (read from .CEL file)
`affy`	`ma.plot`	M and A need to be pre-calculated
`affy`	`mva.pairs`	matrix of MA-plots, not suitable for many samples

for exercises: written function based on `ma.plot`

mva.pairs

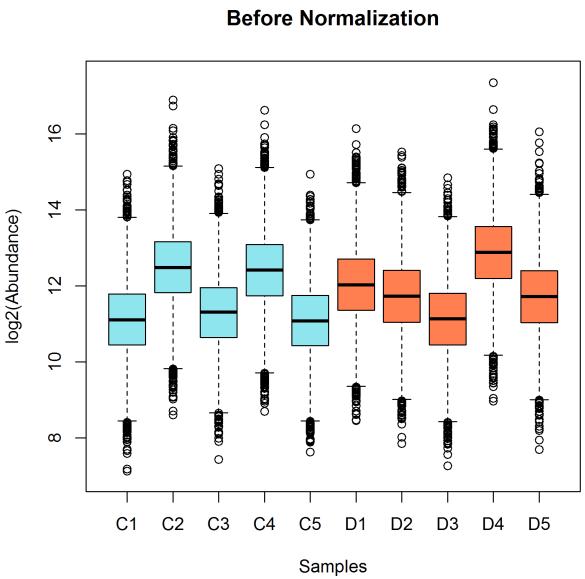
X: data.frame containing samples in columns

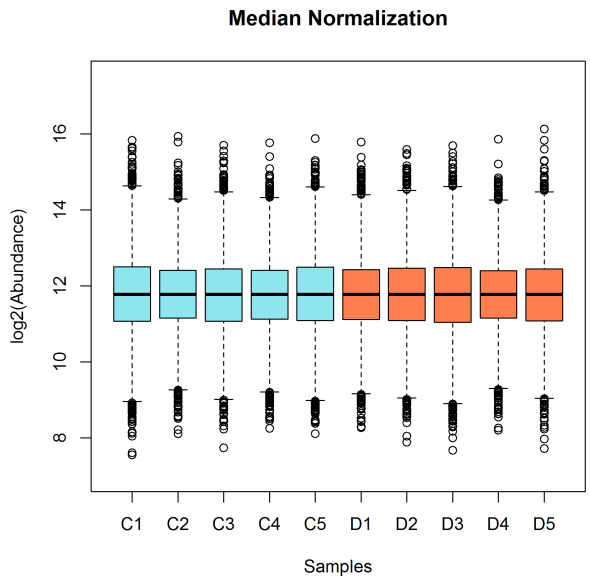
```
library(affy)
affy::mva.pairs(X, log.it = TRUE)
```

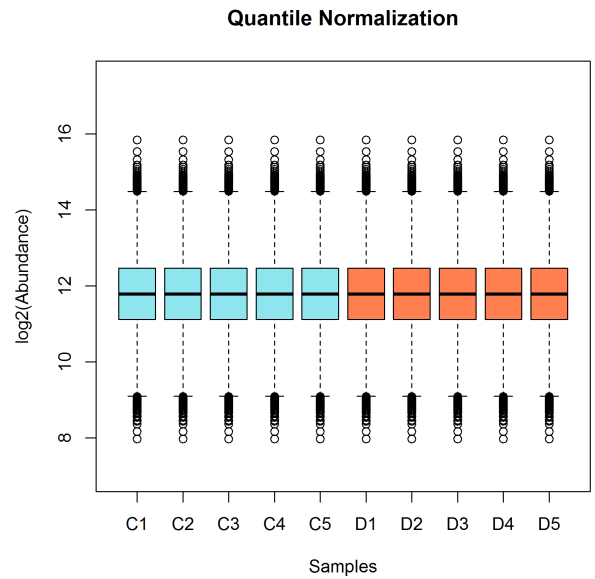


Normalization - Evaluation with Boxplots

- boxplots of can give a hint if normalization is needed
- differences in box size
- looking at boxplots alone is not enough to compare normalization methods!

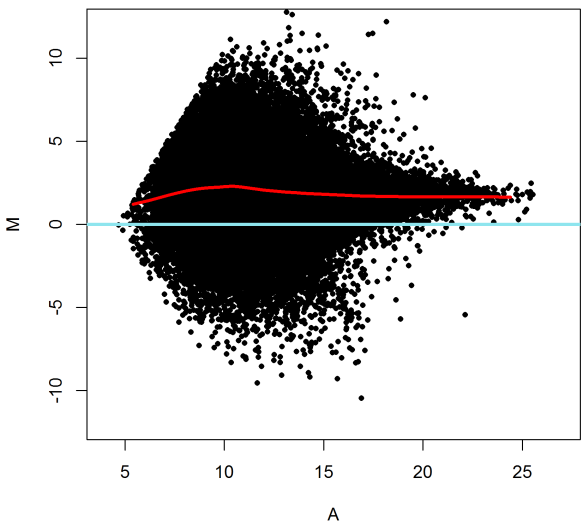


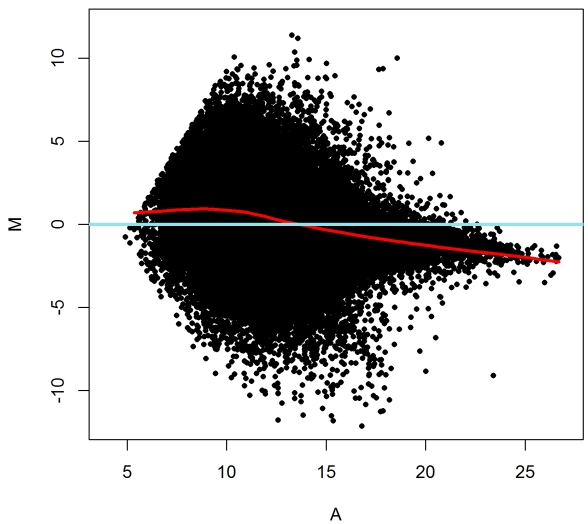


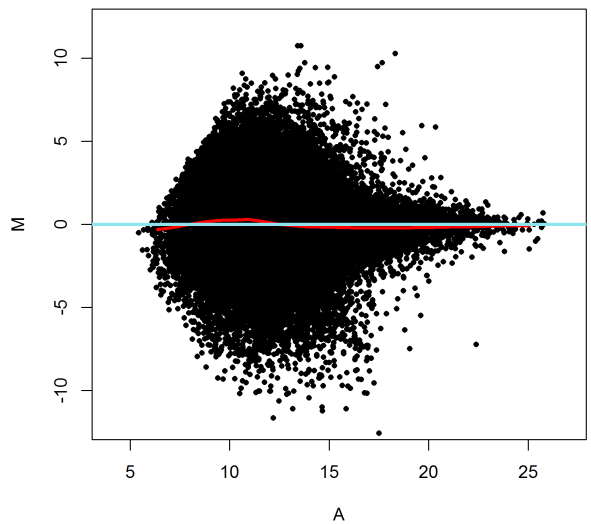


Normalization - Evaluation with MA-Plots

- look at all combinations of sample pairs
- left: unnormalized data $\rightarrow$ bias
- middle: unsuitable median normalization, regression line falls
- right: suitable LOESS normalization

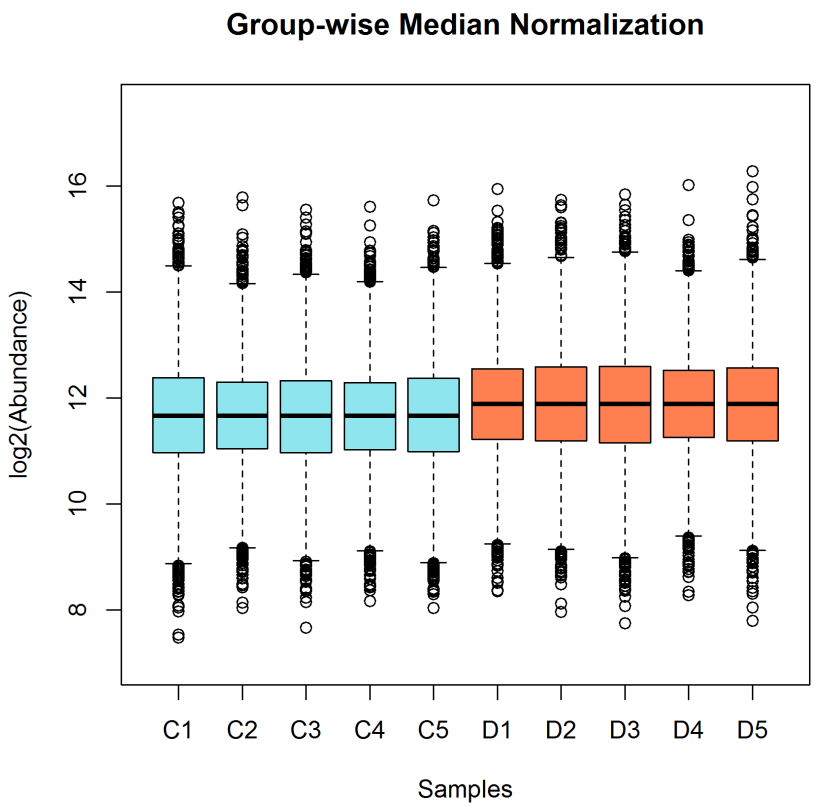


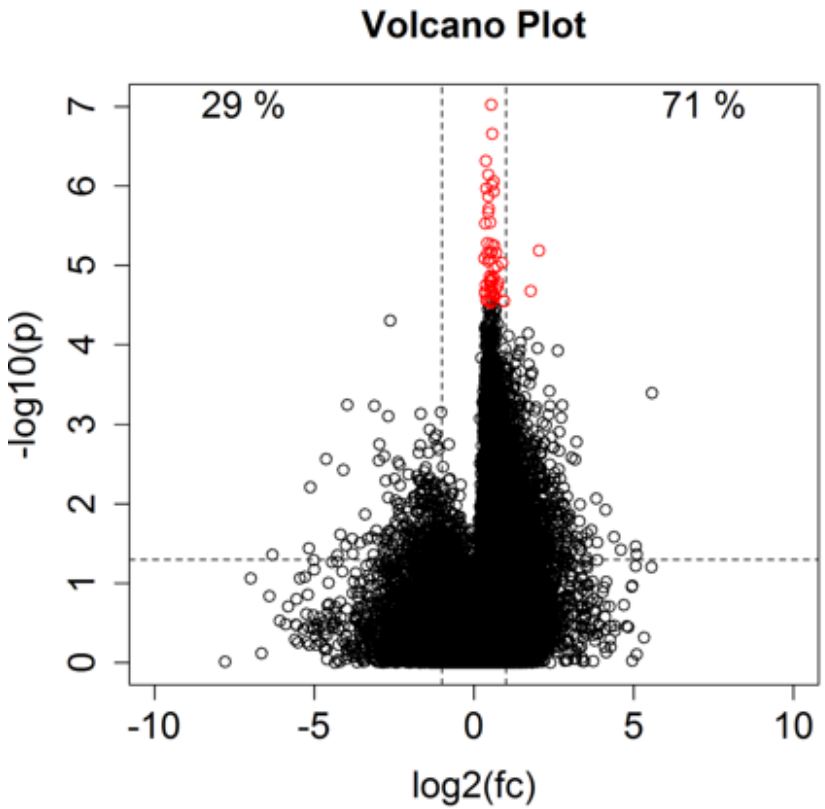




Normalization - group-wise normalization

- the assumption that most proteins do not change between groups may not hold in some situations
- avoid group-wise normalization (this can lead to artifical differences between groups)!
- use alternatives, e.g. LTS-normalization (least trimmed squares)



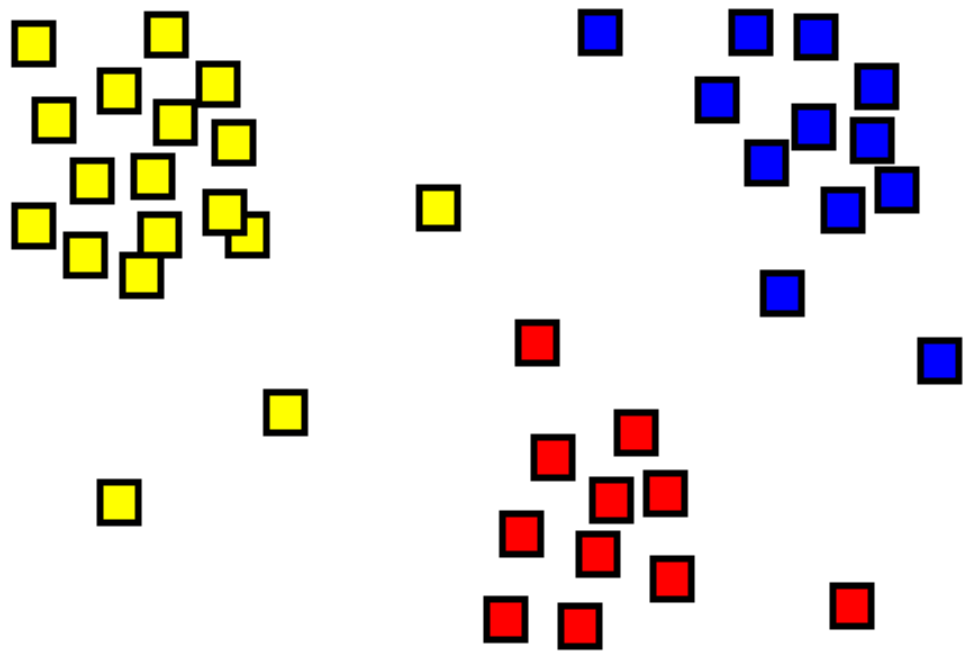


EXERCISE

Clustering

What is Clustering?

- Clustering is the grouping of objects into groups (= clusters)...
- in a way that all objects inside a specific cluster are more similar to each other than to objects in all other clusters.



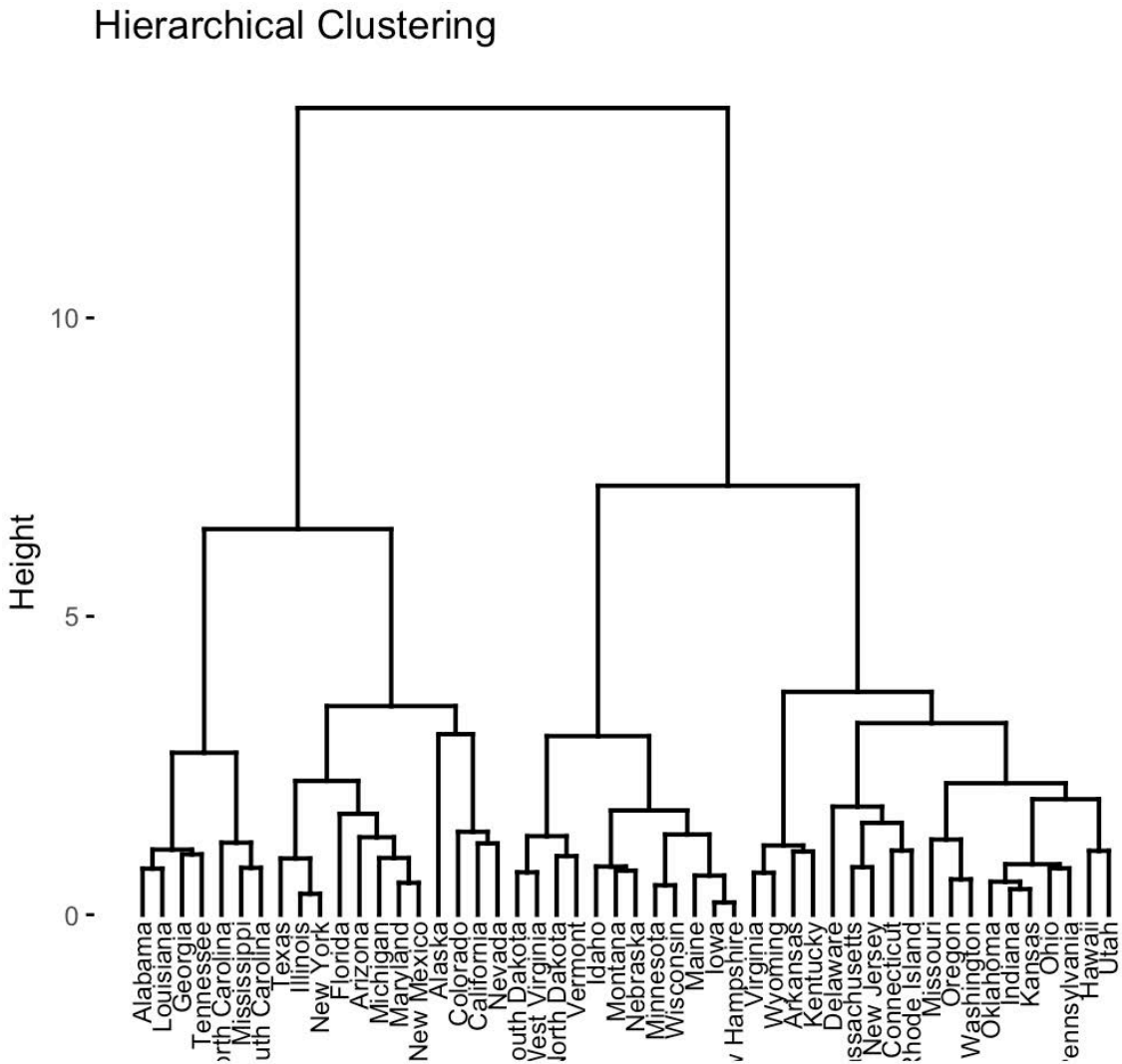
Clustering algorithms

- There are many published clustering algorithms & algorithm variants.
- They can be categorized into four major groups of algorithms:
  - Hierarchical clustering (e.g., single linkage, complete linkage, average linkage)
  - Centroid-based clustering (e.g., k-means)
  - Distribution-based clustering (e.g., EM clustering)
  - Density-based clustering (e.g., DB SCAN)
- In this course only hierarchical clustering will be discussed.

Hierarchical clustering: Main idea

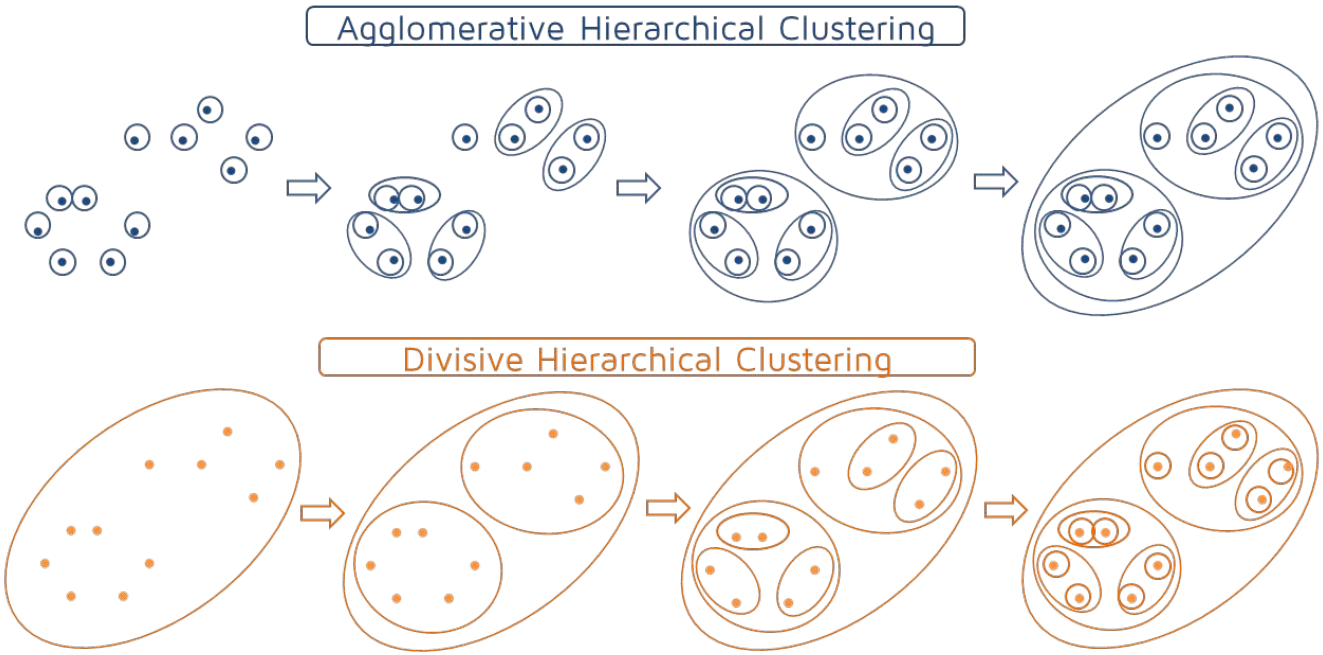
1. Each data point starts as its own cluster.
2. The distance between all pairs of clusters is computed.
3. Then the closest clusters are merged.
4. Steps 2. - 3. are repeated until all clusters are merged into a single cluster.

Hierarchical clustering: Dendrograms



Hierarchical clustering: Agglomerative vs. divisive

Note: In this course only agglomerative methods are discussed.

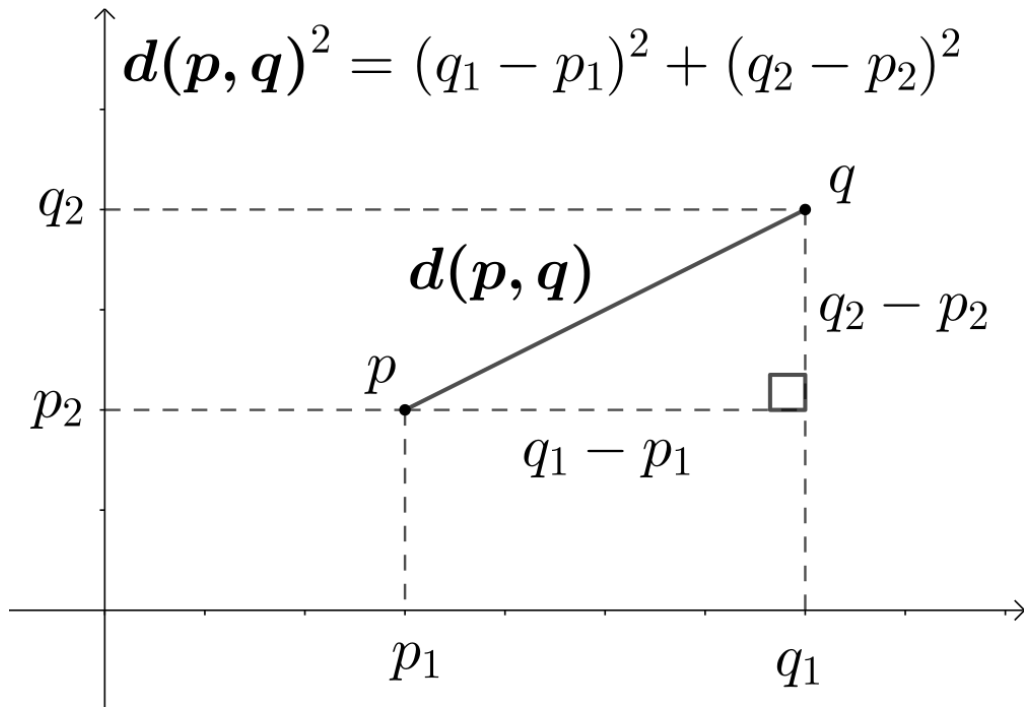


Hierarchical clustering: Distance functions

- Distance functions compute distances between pairs of vectors.
- Distances between pairs of vectors are necessary to obtain distances between clusters.
- In this course, we discuss euclidean, manhattan & correlation-based distances.

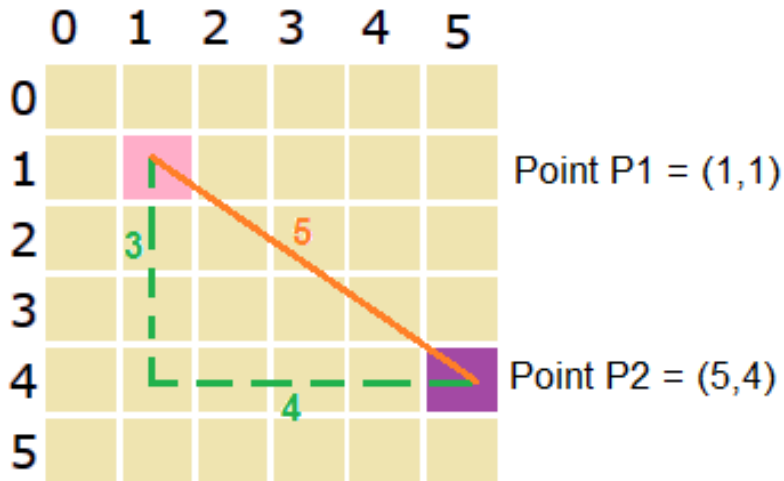
Hierarchical clustering: Euclidean distance

- The distance from classical geometry in school.
- $d(p, q) := \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$



Hierarchical clustering: Manhattan distance

- Also called taxicab distance or city block distance.
- $d(a, b) := \sum_{i=1}^{n} |a_i - b_i|$



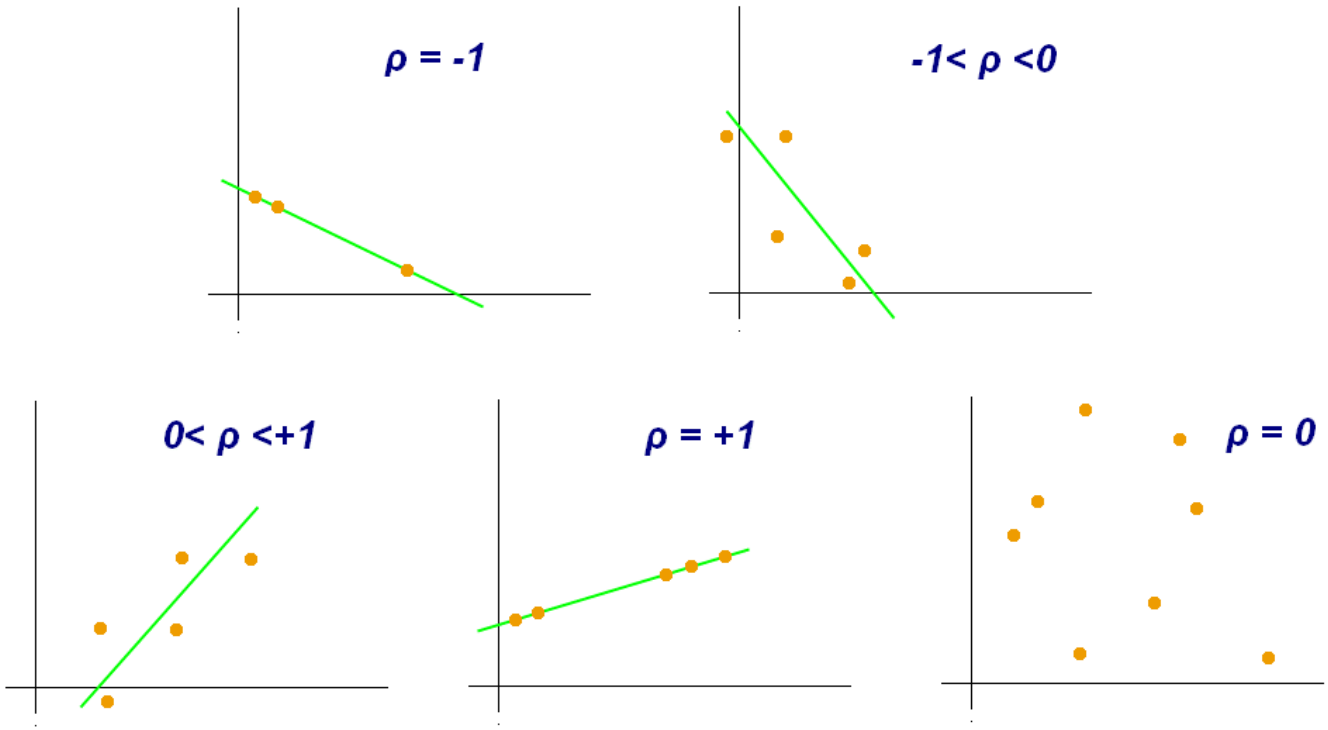
Euclidean distance = $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance = $|5-1| + |4-1| = 7$

Hierarchical clustering: Manhattan vs. euclidean

Hierarchical clustering: Correlation-based distance

- Using correlation as distance measure.
- $d(a, b) := (1 - cor(a, b))/2$, where $cor(a, b)$ is Pearson's correlation coefficient between vectors a and b.



Hierarchical clustering: dist() & as.dist

`dist()` computes & returns the distance matrix of distances between rows of a data matrix using the specified distance measure.

Most important arguments:

`x`	`numeric matrix, data frame or "dist" object.`
`method`	`the distance measure to be used, e.g. "euclidean" or "manhattan".`

`as.dist()` converts appropriate R objects to "dist" objects. E.g., `as.dist((1-cor(x))/2)`.

Hierarchical clustering: dist()

```
> example.dat <- matrix(c(1,1,5,4), ncol=2)
> example.dat
     [,1] [,2]
[1,]    1    5
[2,]    1    4
> dist(t(example.dat), method="euclidean")
  1
2 5
> dist(t(example.dat), method="manhattan")
  1
2 7
>
```

Hierarchical clustering: dist() & as.dist()

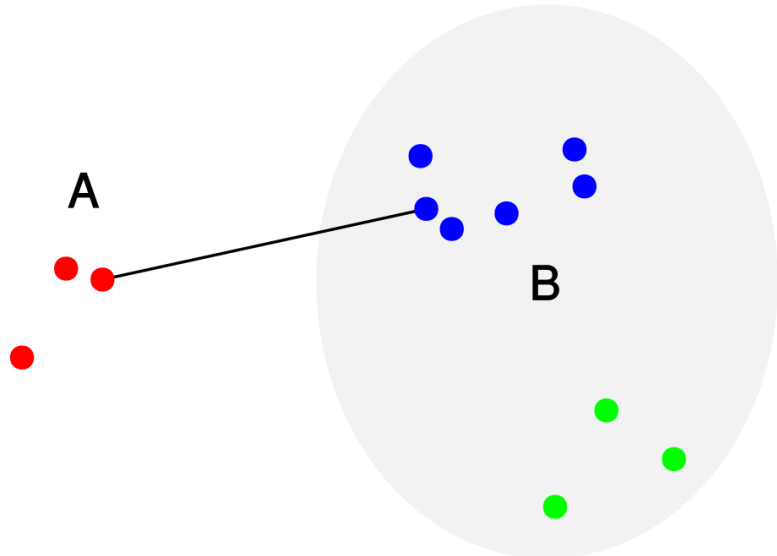
```
> example.dat2 <- matrix(c(11,9,12,111,108,113), ncol = 2)
> example.dat2
     [,1] [,2]
[1,]   11  111
[2,]    9  108
[3,]   12  113
> as.dist((1-cor(example.dat2))/2)
             1
2 0.001411768
> dist(t(example.dat2), method = "euclidean")
         1
2 173.2109
> dist(t(example.dat2), method = "manhattan")
    1
2 300
```

Hierarchical clustering: Linkage methods

- Linkage methods compute the distance between clusters.
- To this end, they use the distance methods between single cluster elements (e.g., euclidean).
- They are crucial for the decision which clusters should be merged.
- In this course we will discuss: Single, complete and average linkage.

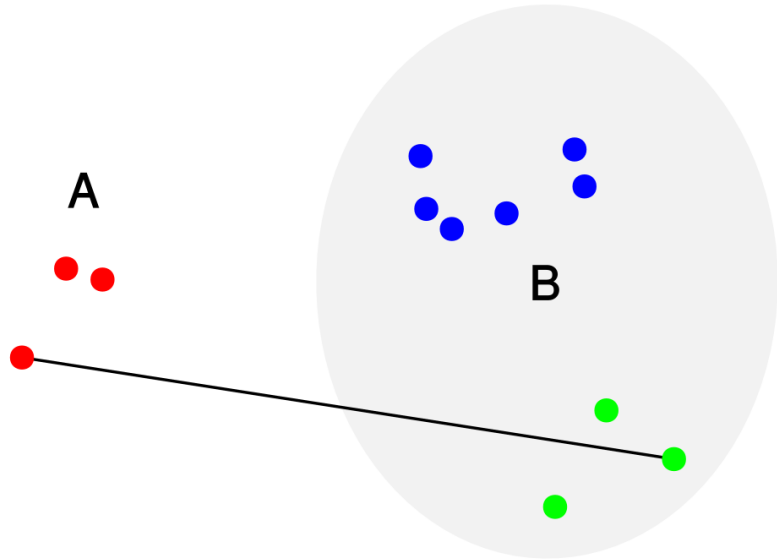
Hierarchical clustering: Single linkage

- Minimal distance between all mixed pairs from both clusters.
- $D_{single} := \min_{a \in A, b \in B} \{d(a, b)\}$.



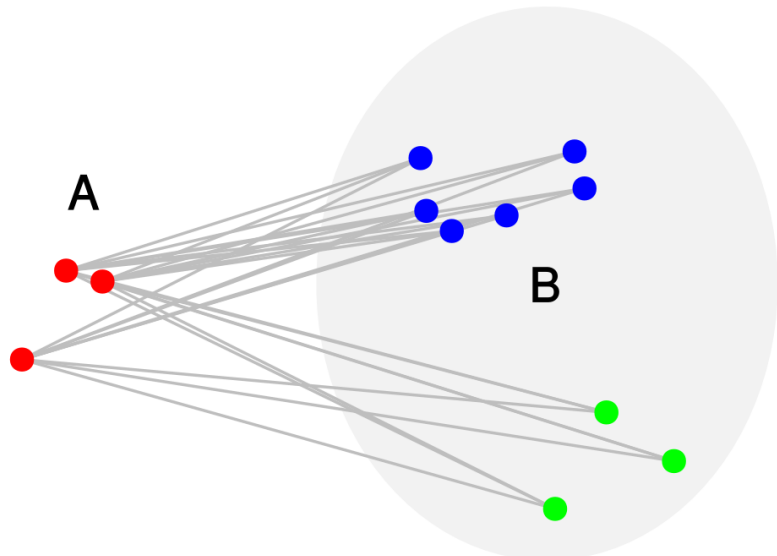
Hierarchical clustering: Complete linkage

- Maximal distance between all mixed pairs from both clusters.
- $D_{complete} := \max_{a \in A, b \in B} \{d(a, b)\}$.



Hierarchical clustering: Average linkage

- Unweighted pair group method with arithmetic mean (UPGMA).
- Average distance between all mixed pairs from both clusters.
- $D_{average} := \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$.



Hierarchical clustering: hclust()

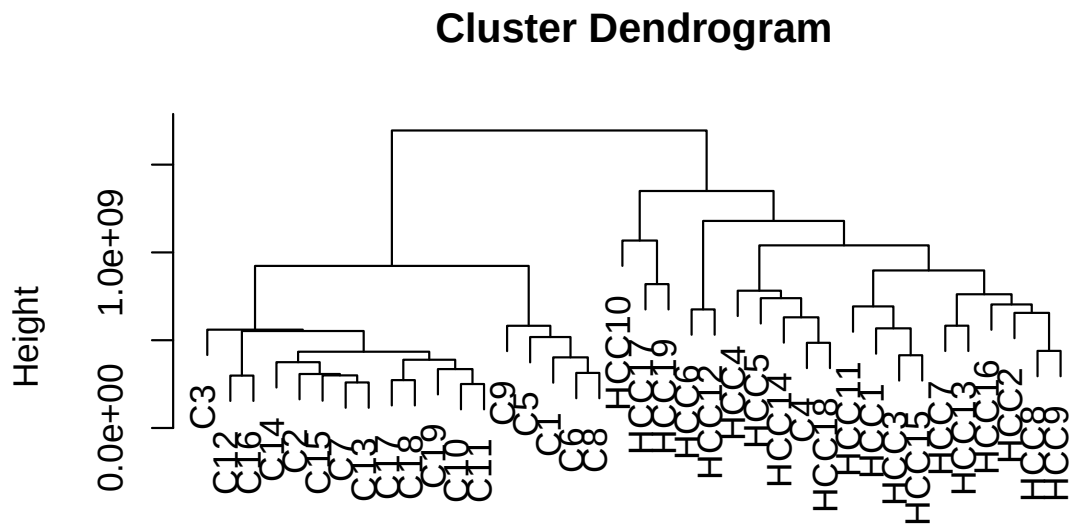
`hclust()` performs hierarchical clustering in R and returns objects describing a tree structure that can be used to plot dendrograms.

Most important arguments:

`d`	`a dissimilarity structure as produced by dist().`
`method`	`the linkage method to be used, e.g. "single", "complete" or "average".`

Hierarchical clustering: hclust()

```
> dat <- read.table(file="HCC_19vs19_normalized_abundances.txt", header=TRUE, sep="\t", quote = "\"")
> rownames(dat) <- dat[,"Accession"]
> dat <- as.matrix(dat[,10:47])
>
> hc <- hclust(d=dist(x=t(dat), method="manhattan"), method="complete")
> plot(hc)
>
```



Hierarchical clustering: heatmap()

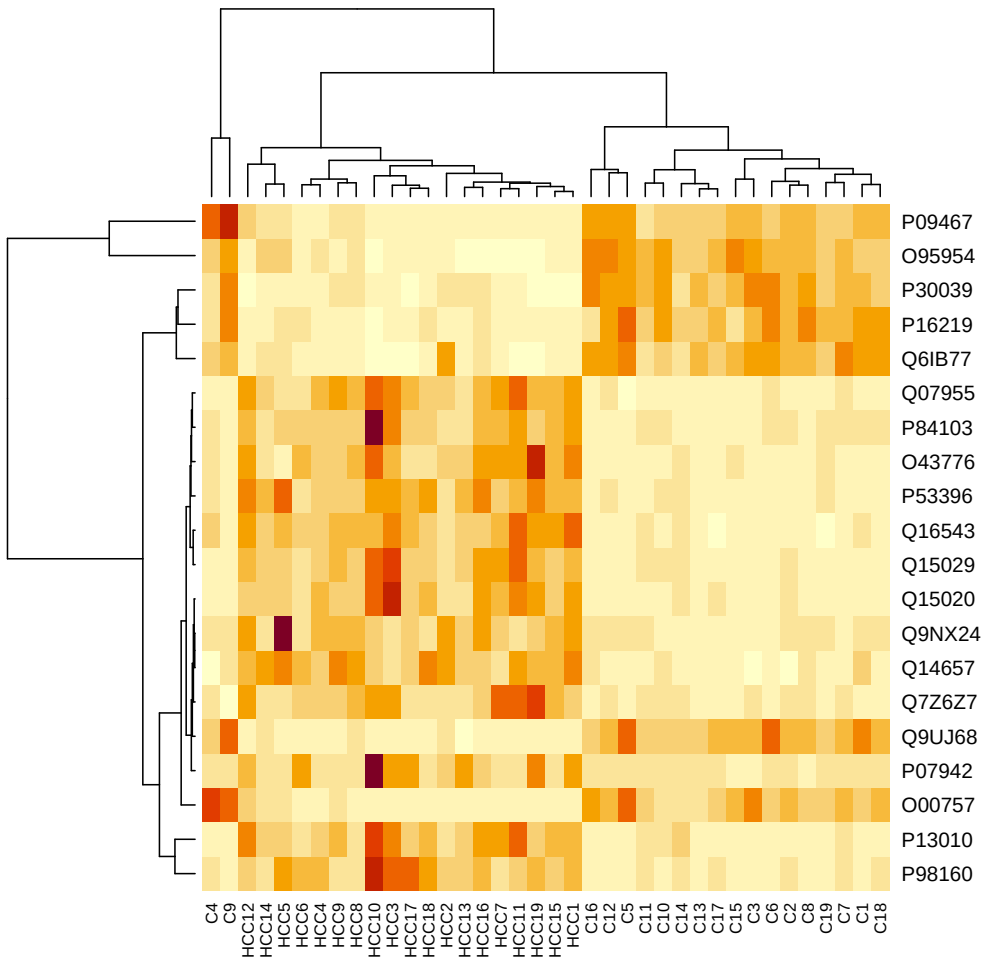
`heatmap()` draws heat maps, i.e. false color images with a dendrogram added to the left side and to the top. Rows and columns are reordered with respect to these dendrograms.

Most important arguments:

`x`	`a numeric matrix containing values to be plotted.`
`Rowv`	`NULL/NA whether row-wise dendrogram should be plotted.`
`Colv`	`NULL/NA whether column-wise dendrogram should be plotted.`
`scale`	`"row", "column" or "none"`

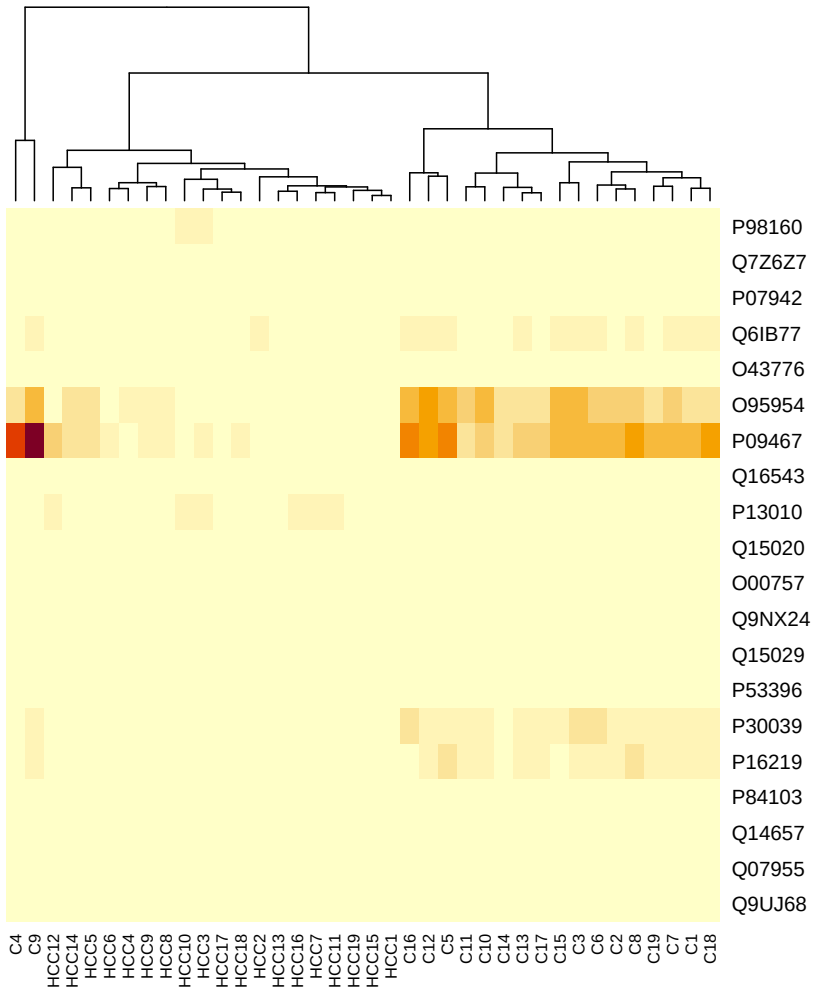
Hierarchical clustering: heatmap()

```
> heatmap(x=dat[1:20,], Rowv=NULL, Colv = NULL, scale="row")
> 
```



Hierarchical clustering: heatmap()

```
> heatmap(x=dat[1:20,], Rowv=NA, Colv = NULL, scale="none")
> 
```



Hierarchical clustering: heatmap.2()

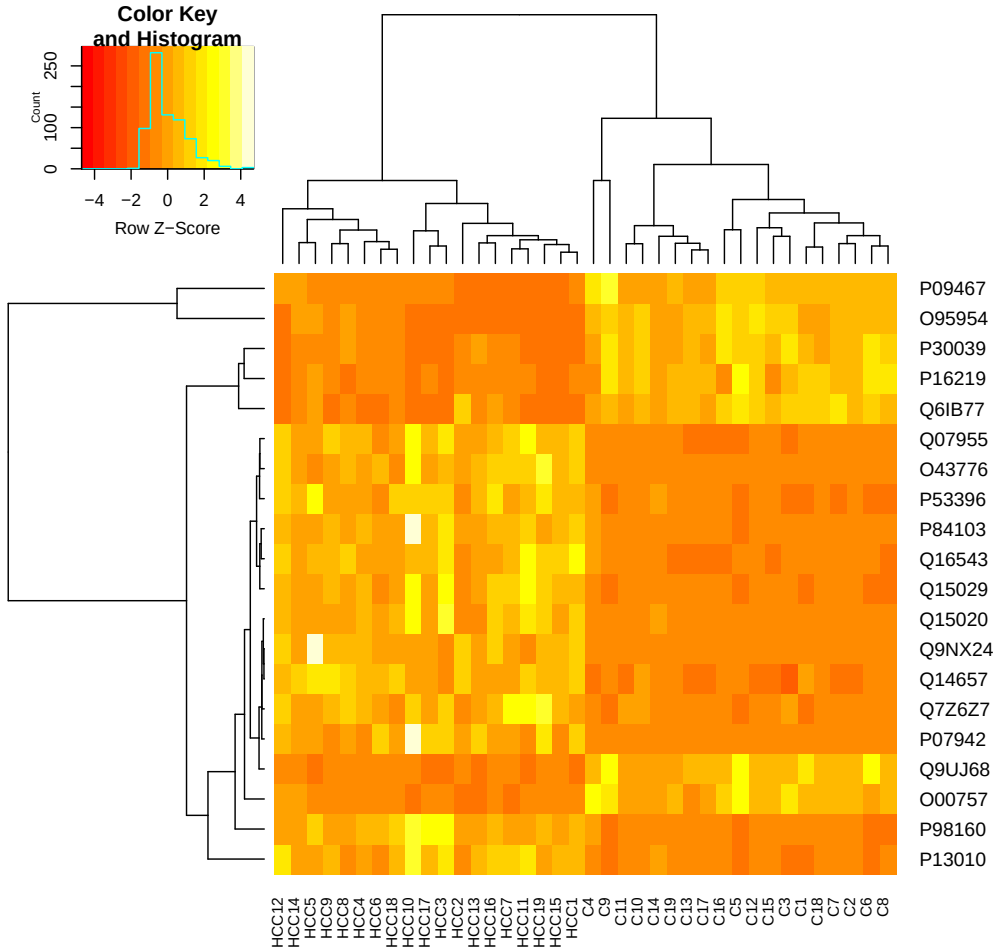
`heatmap.2()` from package `gplots` provides a number of extensions to the standard R `heatmap` function.

Most important arguments:

`x`	`a numeric matrix containing values to be plotted.`
`Rowv`	`TRUE/FALSE whether row-wise dendrogram should be plotted.`
`Colv`	`TRUE/FALSE whether column-wise dendrogram should be plotted.`
`distfun`	`dist.`
`scale`	`"row", "column" or "none"`
`trace`	`level trace: "column","row","both" or "none"`

Hierarchical clustering: heatmap.2()

```
> heatmap.2(x=dat[1:20,], distfun=function(x) {dist(x,method="manhattan")},
 Rowv=TRUE, Colv = TRUE, scale="row", trace="none")
> 
```



LUNCH BREAK

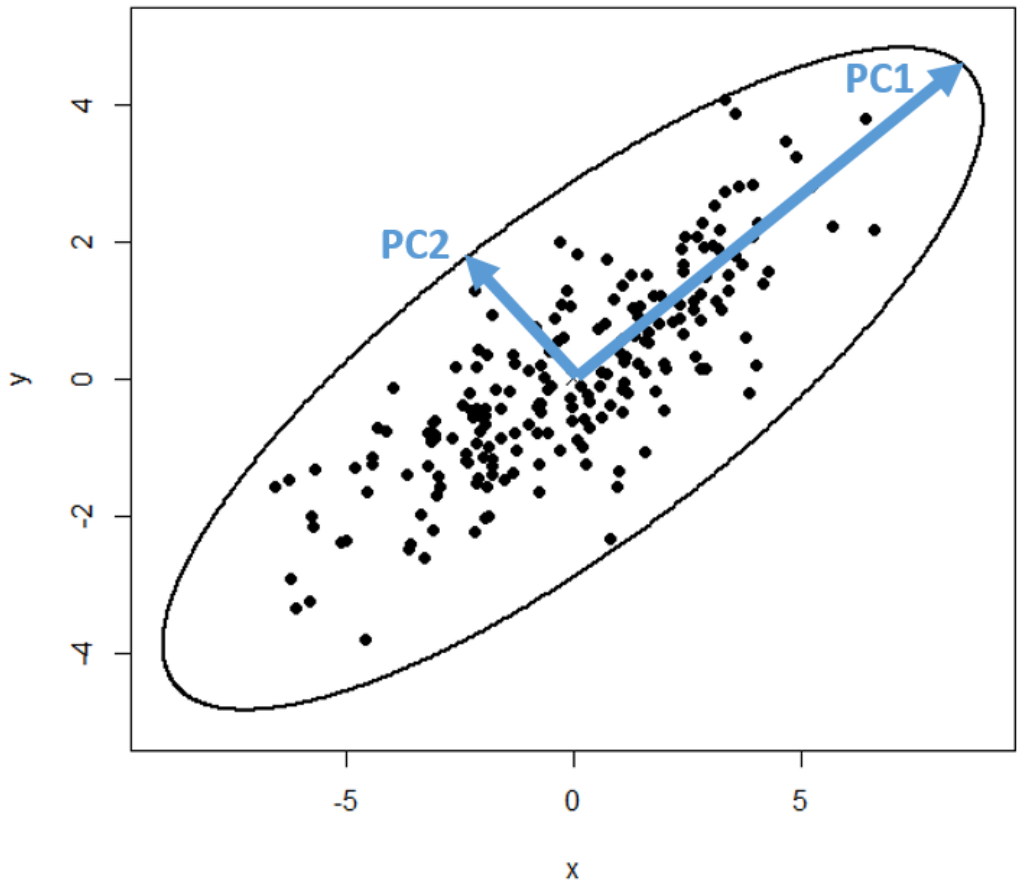
EXERCISE

Principal component analysis (PCA)

Principal Component Analysis

- transform n variables $(X_1, ..., X_n)$ into n Principal Components
- Principal Components (PCs) are linear combinations of the original variables: $PC = w_1X_1 + w_2X_2 + ... + w_nX_n$

- PC1 has the largest possible variance, PC2 the largest variance and of vectors orthogonal to PC1, ...
- 2-dimensional PCA-Plots as an overview over the whole data set



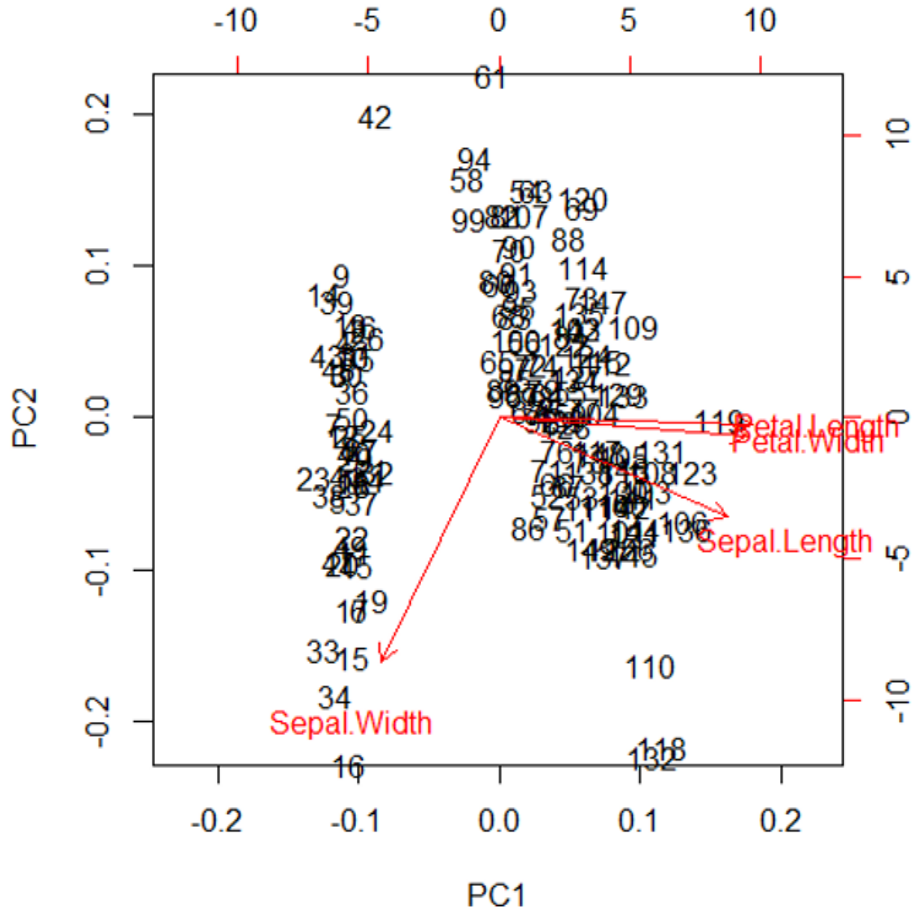
different functions for PCA in R

- `prcomp()` and `princomp()` in base R
- use different numerical methods for calculation
- differences in naming of arguments and output
- important advantage of `prcomp()`: it can handle datasets with more variables than observations (needed for omics data!)

PCA Biplot

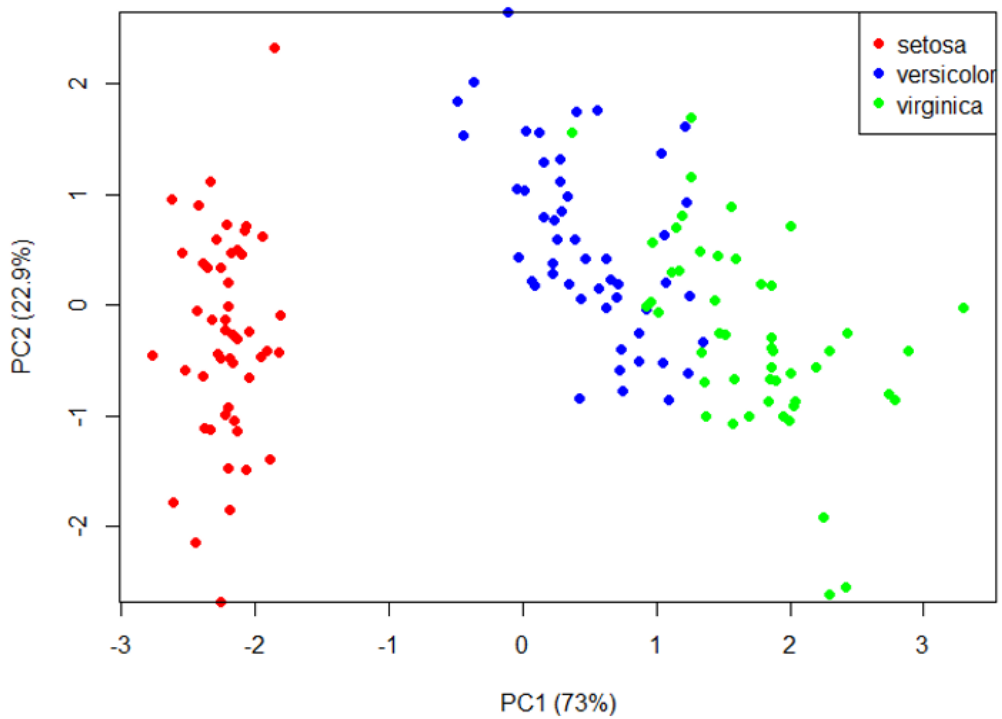
```
pca <- prcomp(iris[,1:4],
scale.  = TRUE)
biplot(pca)
```

- shows first and second PC of observations as a scatterplot
- direction of original axes (variables)
- not suitable for proteomics data (too many variables)



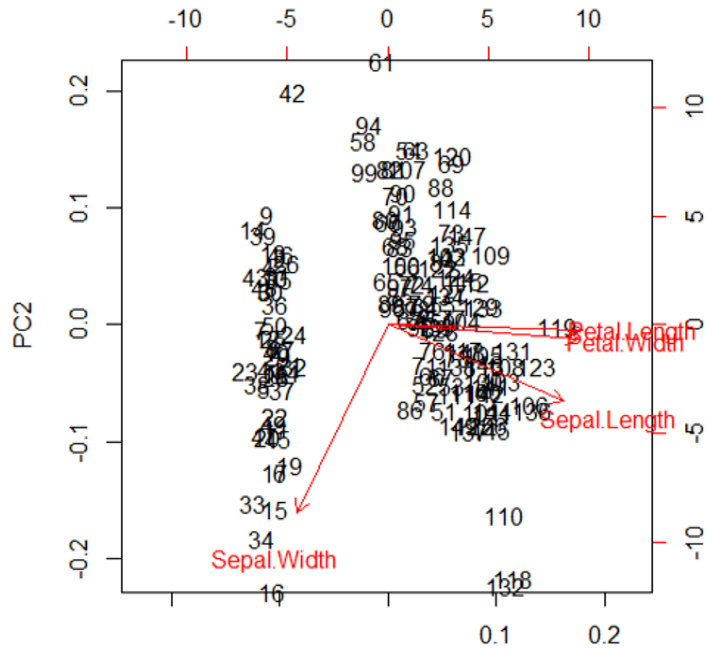
PCA Plot

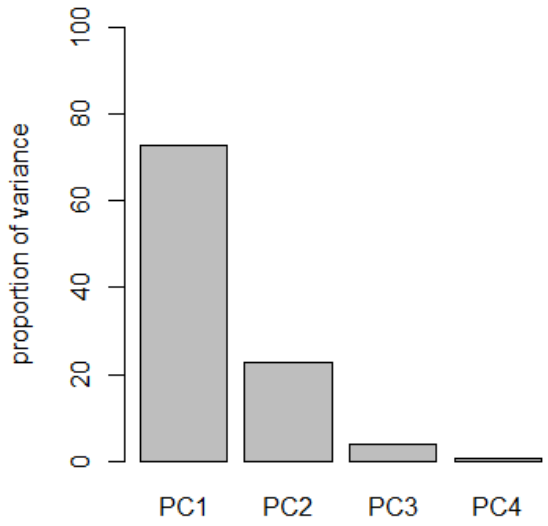
```
summ <- summary(pca)
plot(pca$x[,1], pca$x[,2], pch = 16,
  col = rep(c("red", "blue", "green"), each = 50),
  xlab = paste0("PC1 (", round(100*summ$importance[2,1], 1), "%)"),
  ylab = paste0("PC2 (", round(100*summ$importance[2,2], 1), "%)"))
legend("topright", col = c("red", "blue", "green"), pch = 16,
  legend = levels(iris$Species))
```

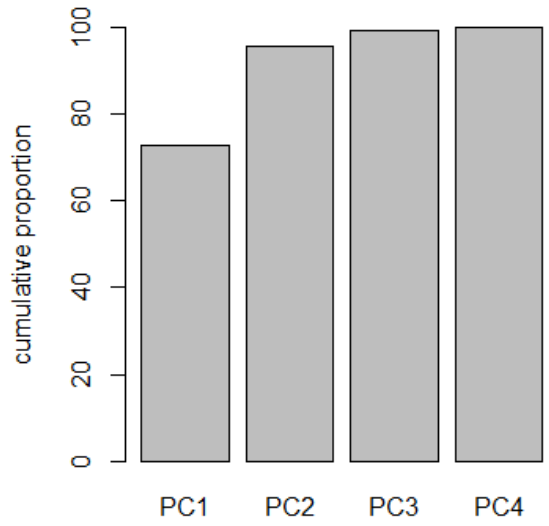


Explained variance

- each PC explains a portion of the total variance in the dataset (PC1 the most, PC2 the 2nd most, ...)
- the higher the variance in PC1 and PC2 combined, the more complete is the overview over the dataset in the 2D plot

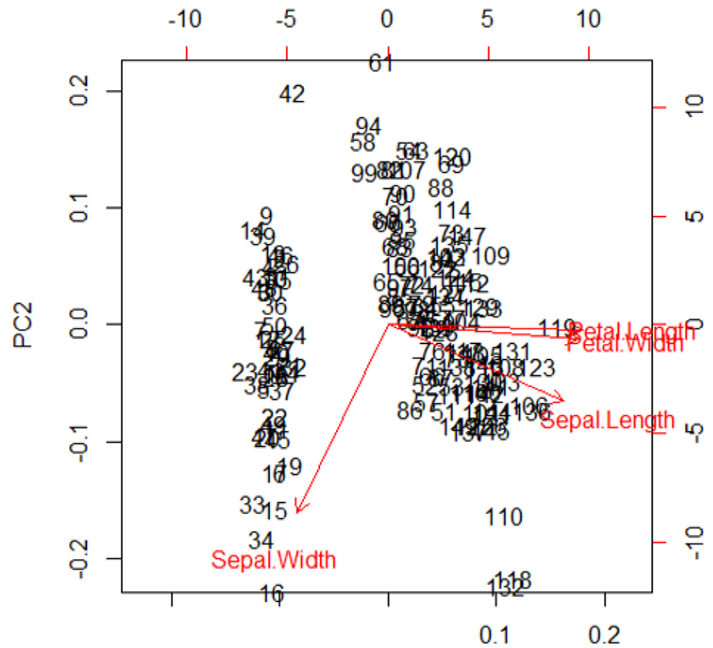


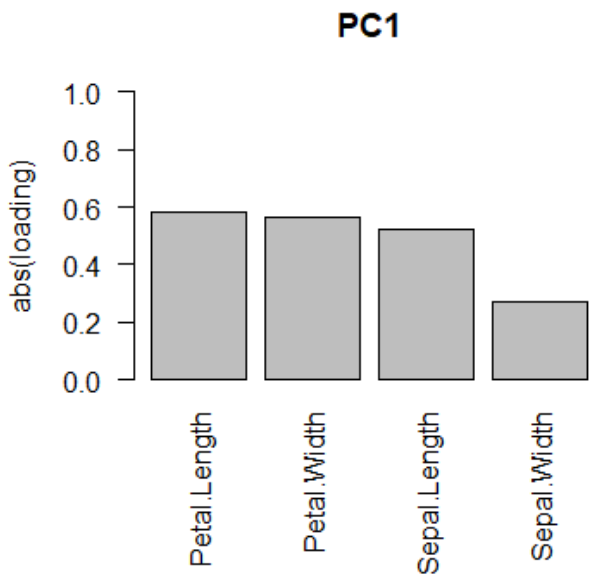


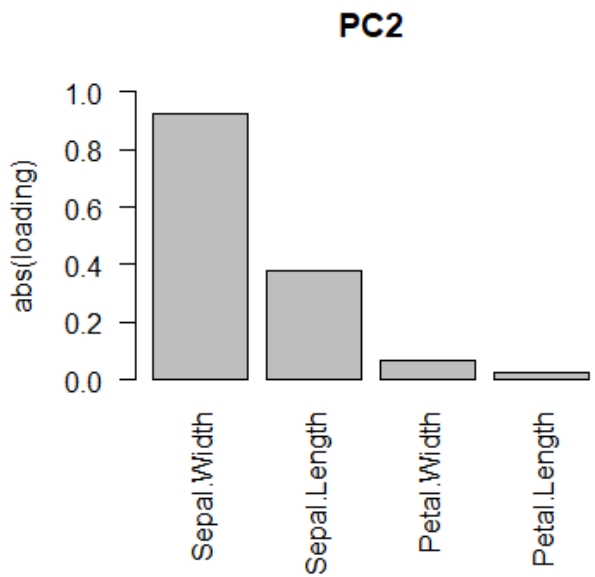


Loadings

- Loadings are the weights used in the linear combinations
- show how important a certain variable is for a certain PC
- called `rotation` by `prcomp`

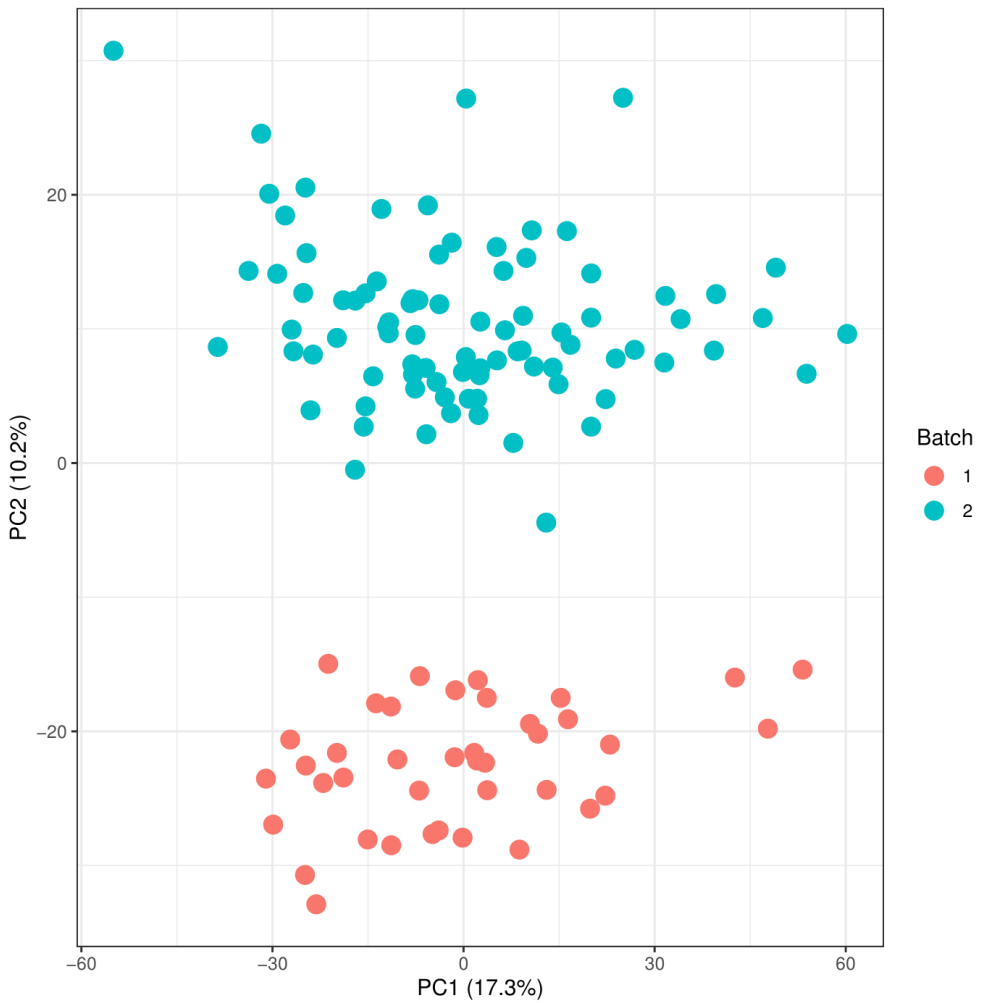






PCA Plots for Quality Control in Proteomics

- overview over whole data set
- detection of outlier points
- detection of batch effects



ROC Analysis

Diagnostic studies

- in diagnostic studies, biomarkers are searched that are able to reliably separate samples with and without a specific disease
- single biomarkers or biomarker panels
- for a certain cutoff you get a contingency table showing true/false positives and negatives

	patient	control	sum
test +	TP	FP	TP + FP
test -	FN	TN	FN+ TN
sum	TP + FN	FP+TN	n

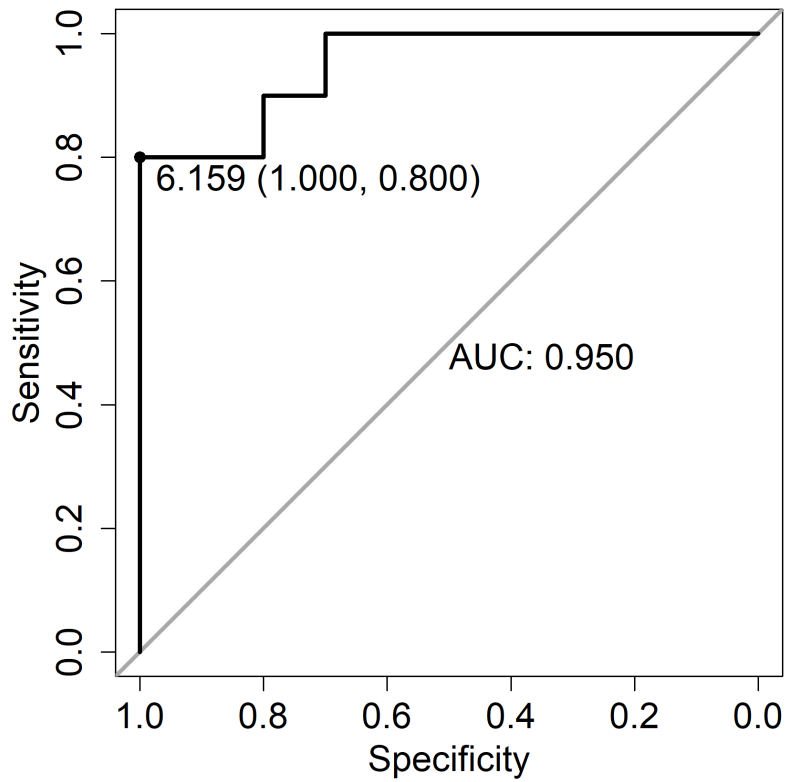
Accuracy, Sensitivity and Specificity

	patient	control	sum
test +	TP	FP	TP + FP
test -	FN	TN	FN + TN
sum	TP + FN	FP+TN	n

- Accuracy = proportion of correctly classified persons $\frac{TP+TN}{n}$
- Sensitivity = proportion of patients that are correctly classified as ill $\frac{TP}{TP+FN}$
- Specificity = proportion of controls that are correctly classified as healthy $\frac{TN}{FP+TN}$

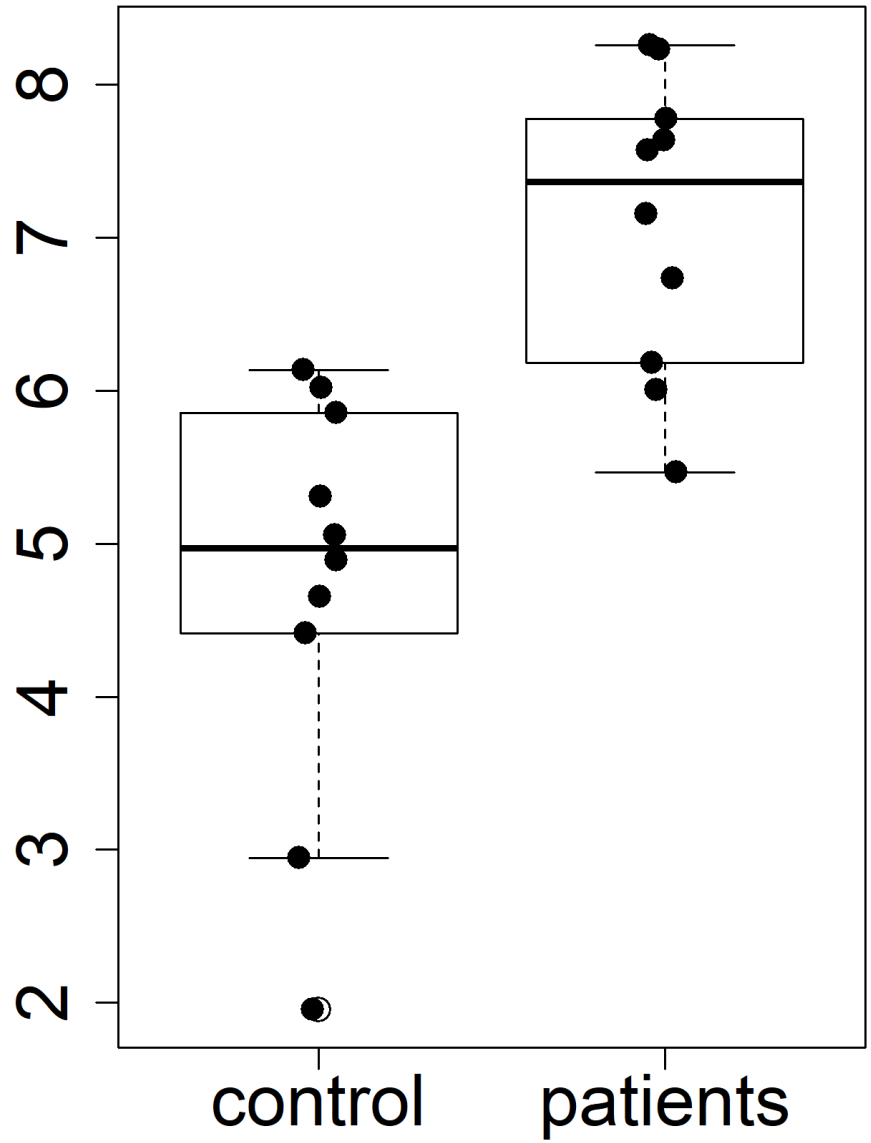
ROC curves

- ROC = Receiver operating characteristic
- shows the overall diagnostic ability of a biomarker or classifier
- plot sensitivity against specificity for all possible cutoffs



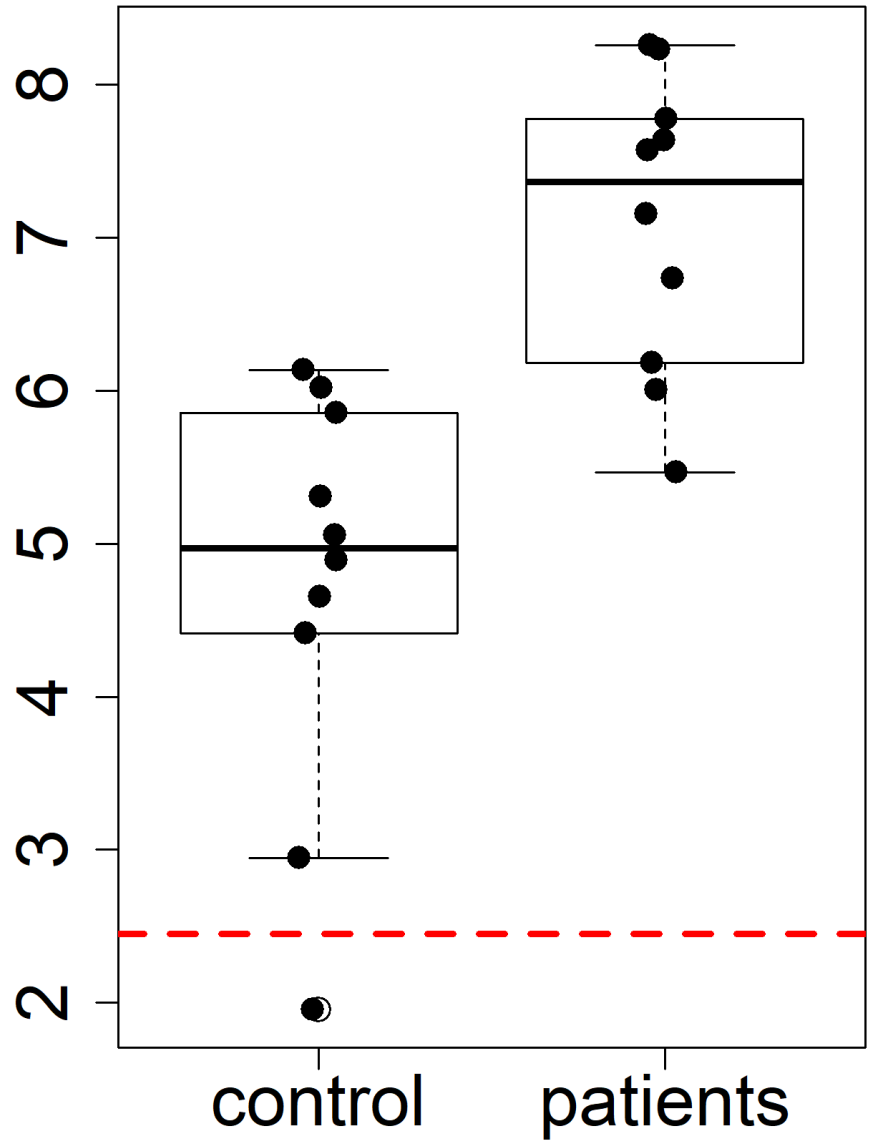
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0



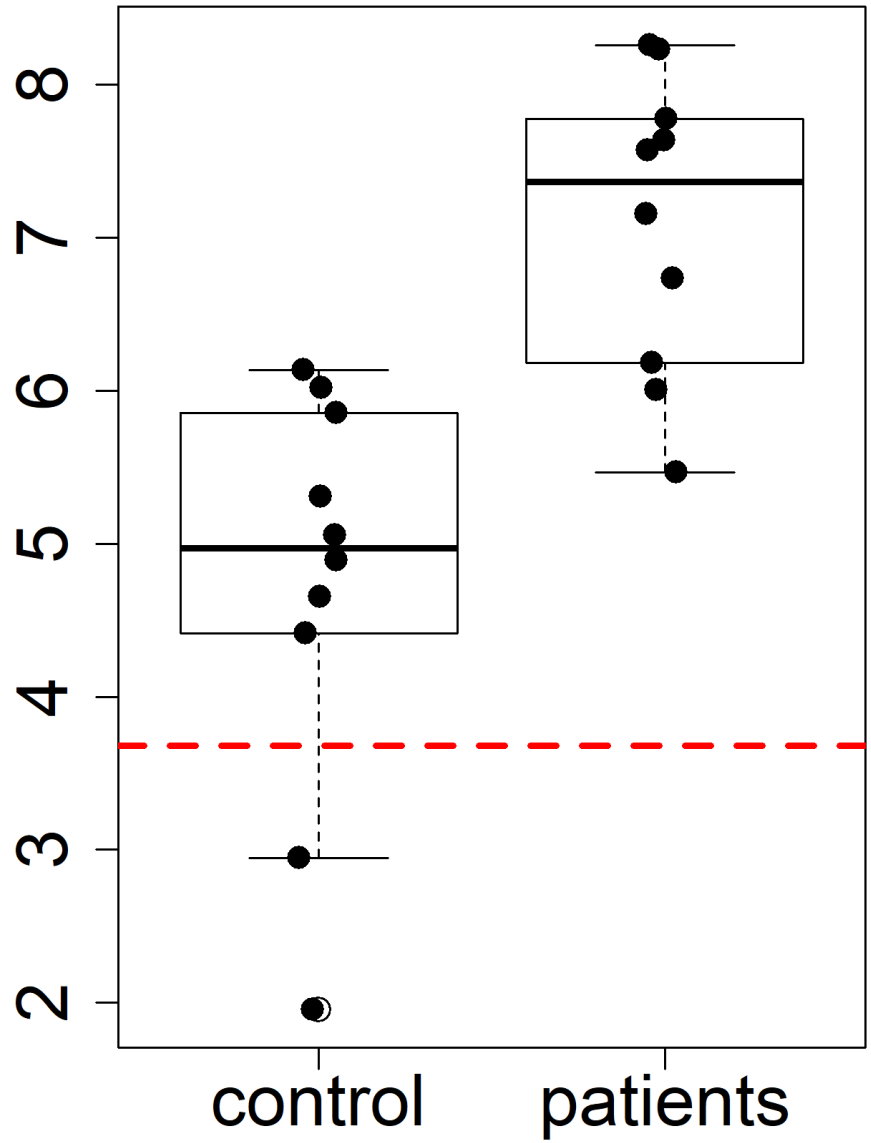
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1



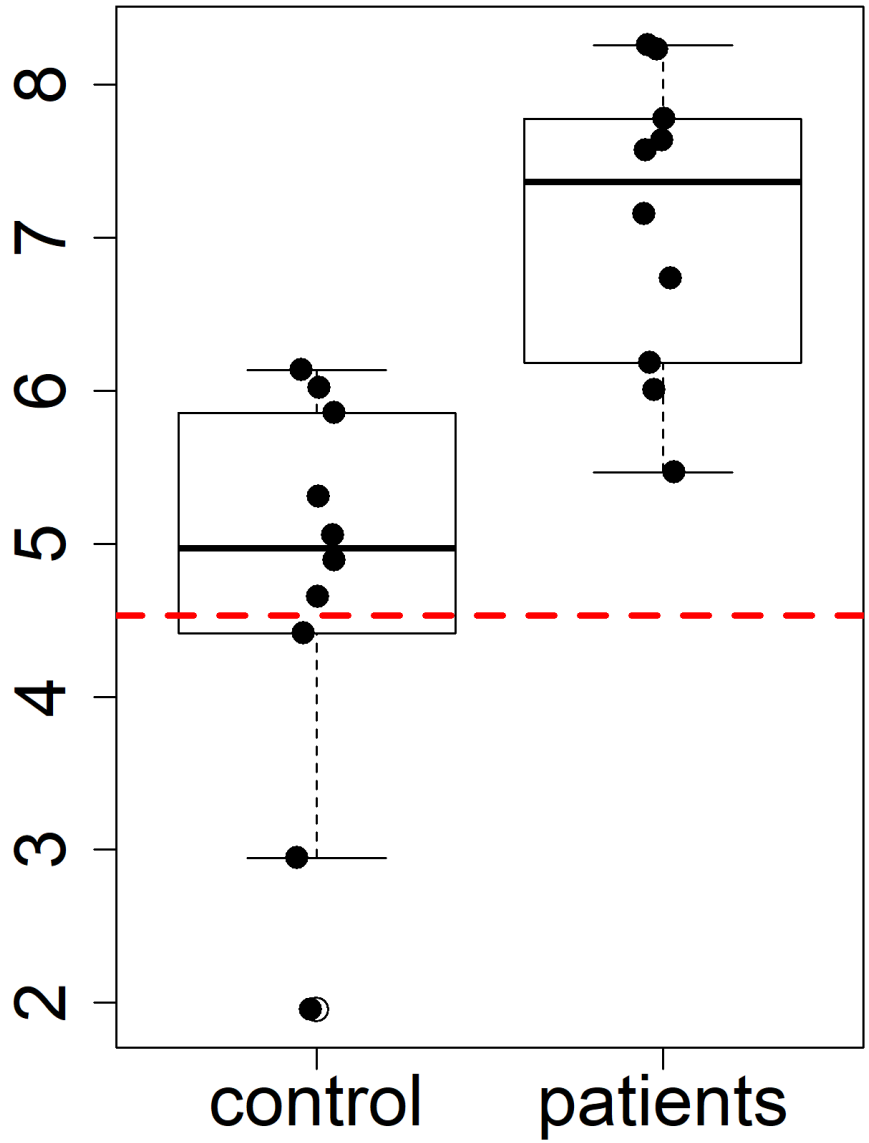
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2



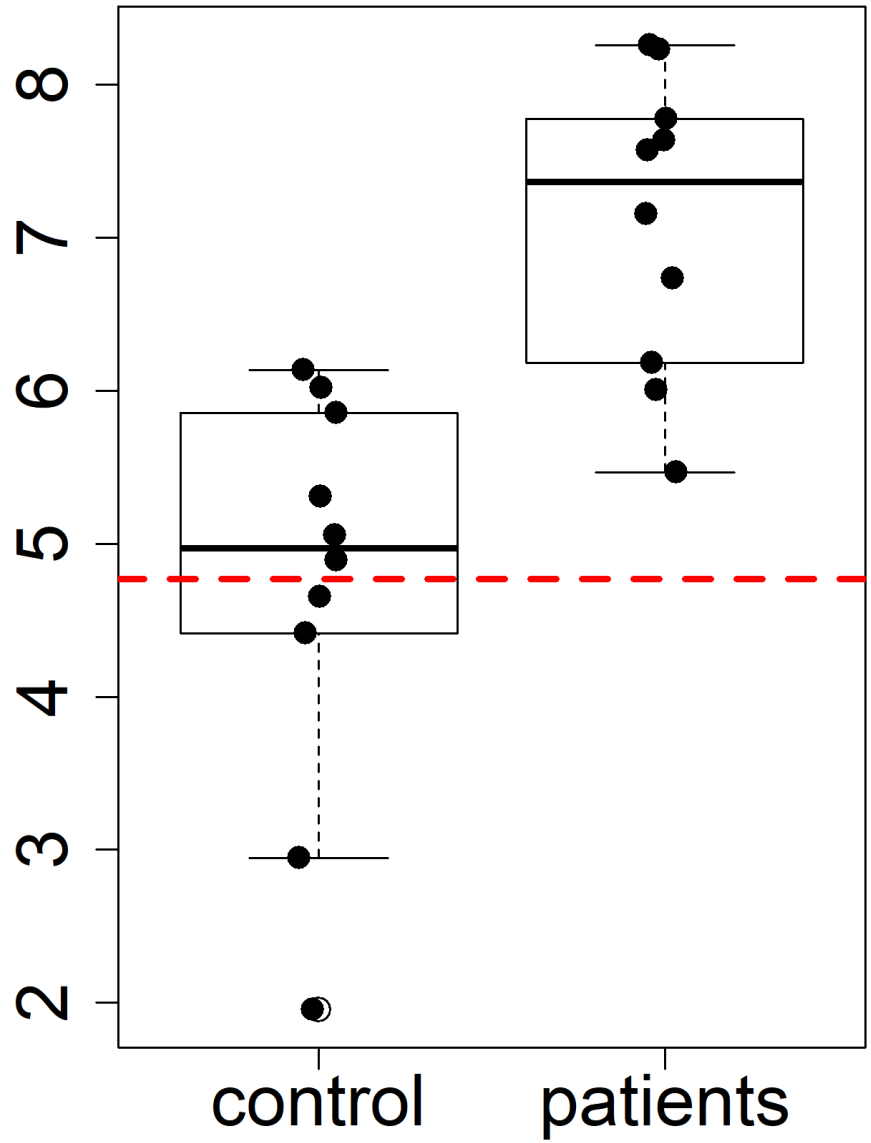
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...



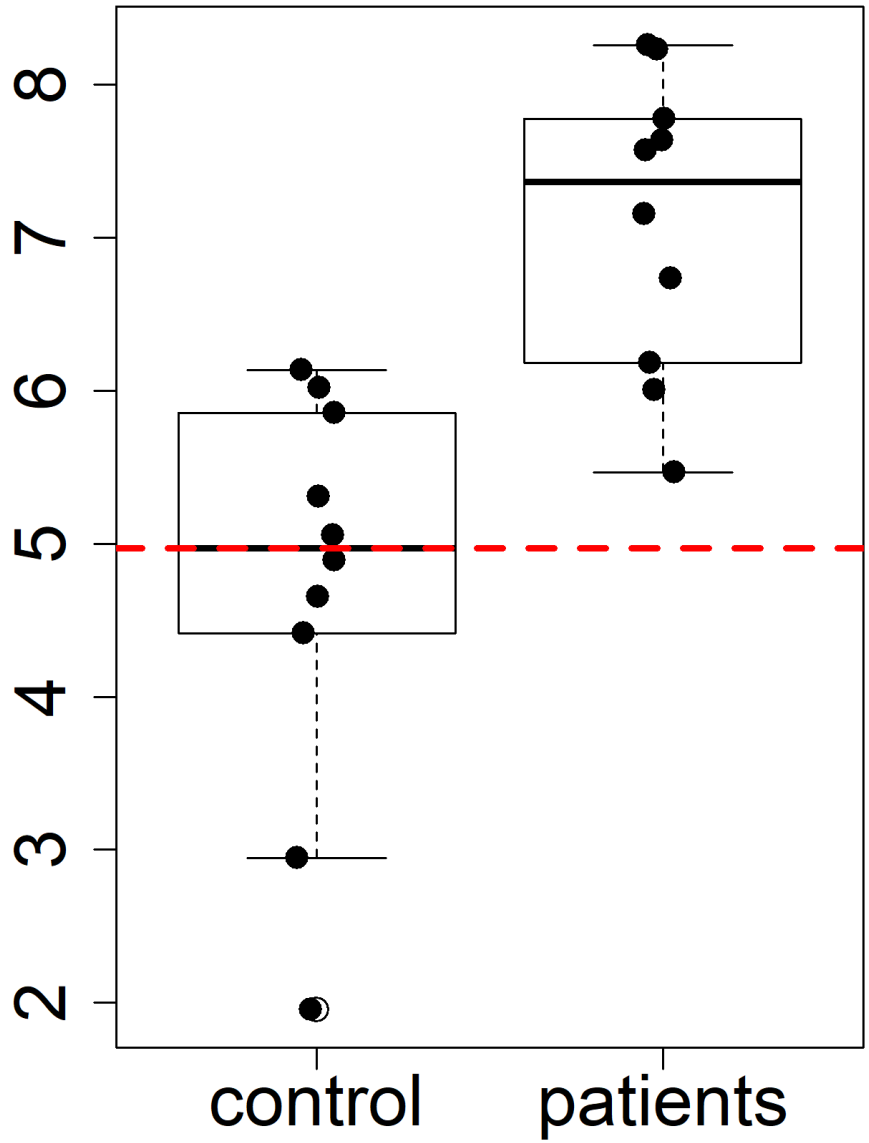
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...



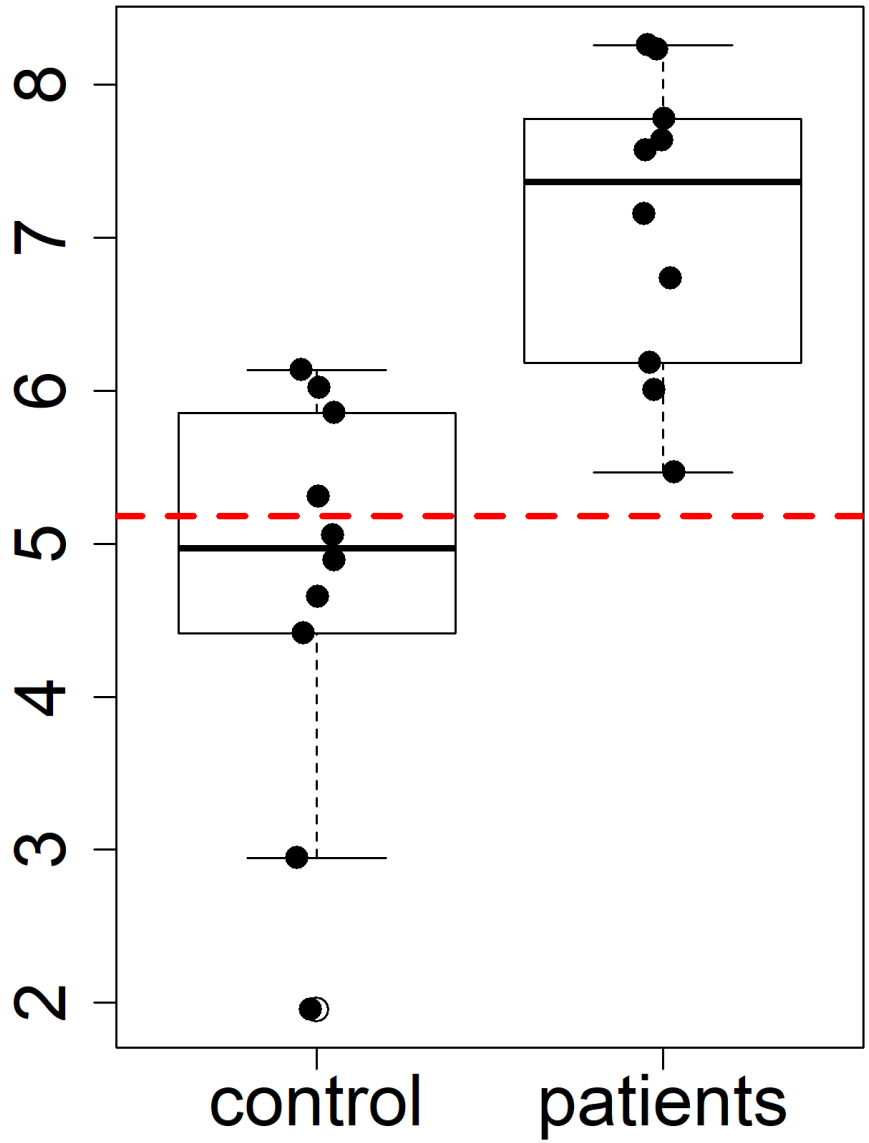
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...



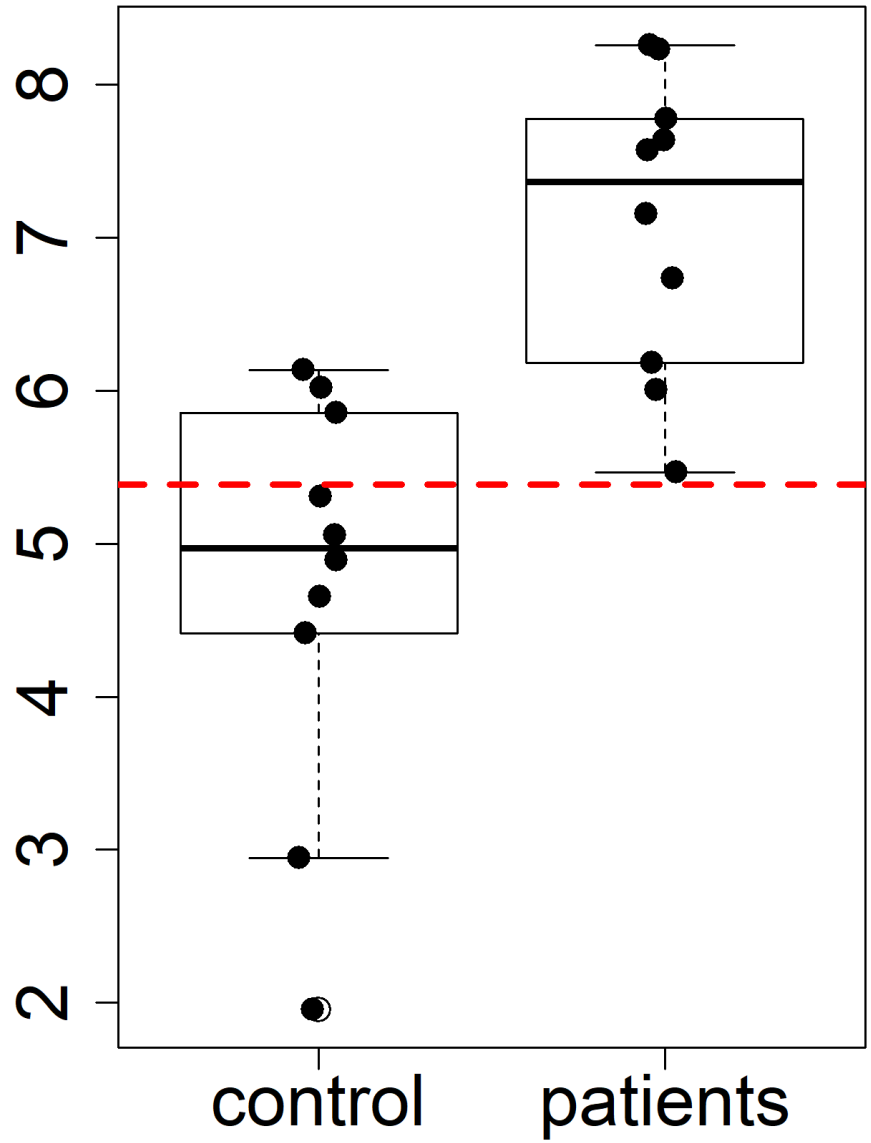
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...



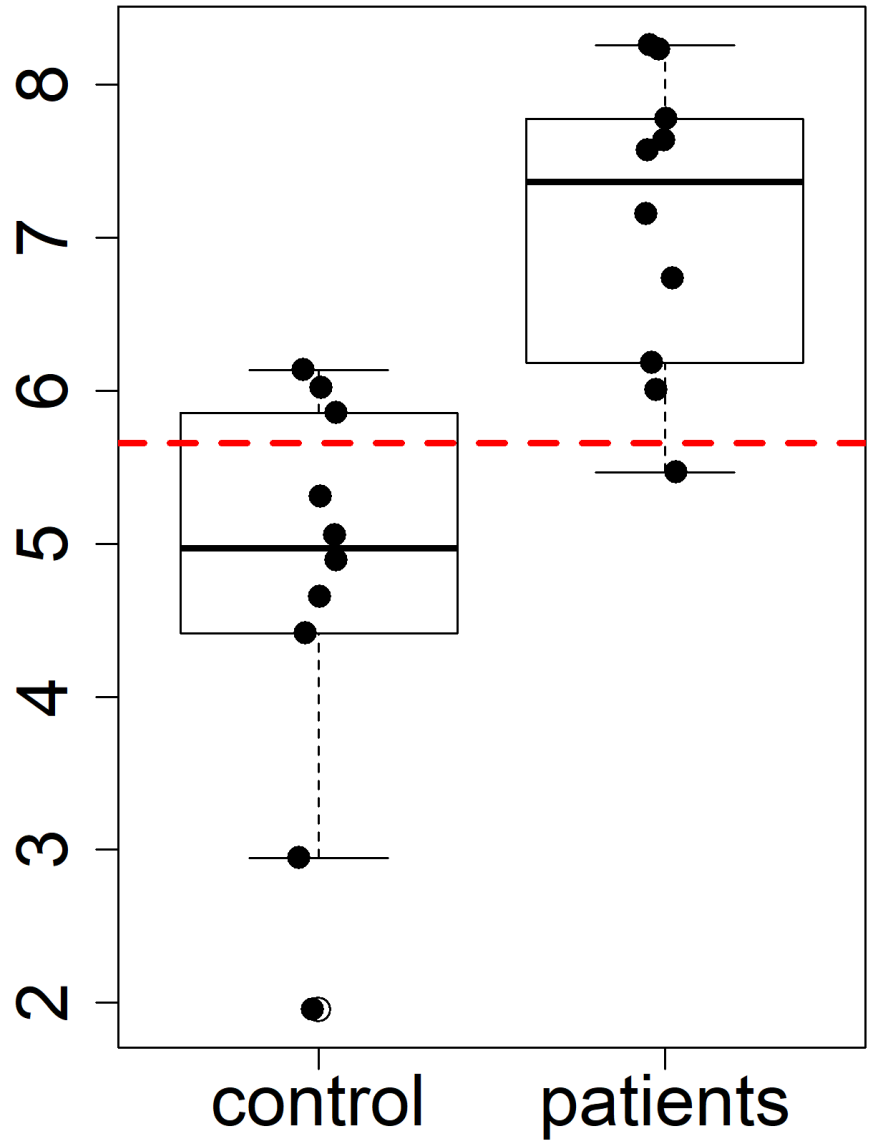
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...



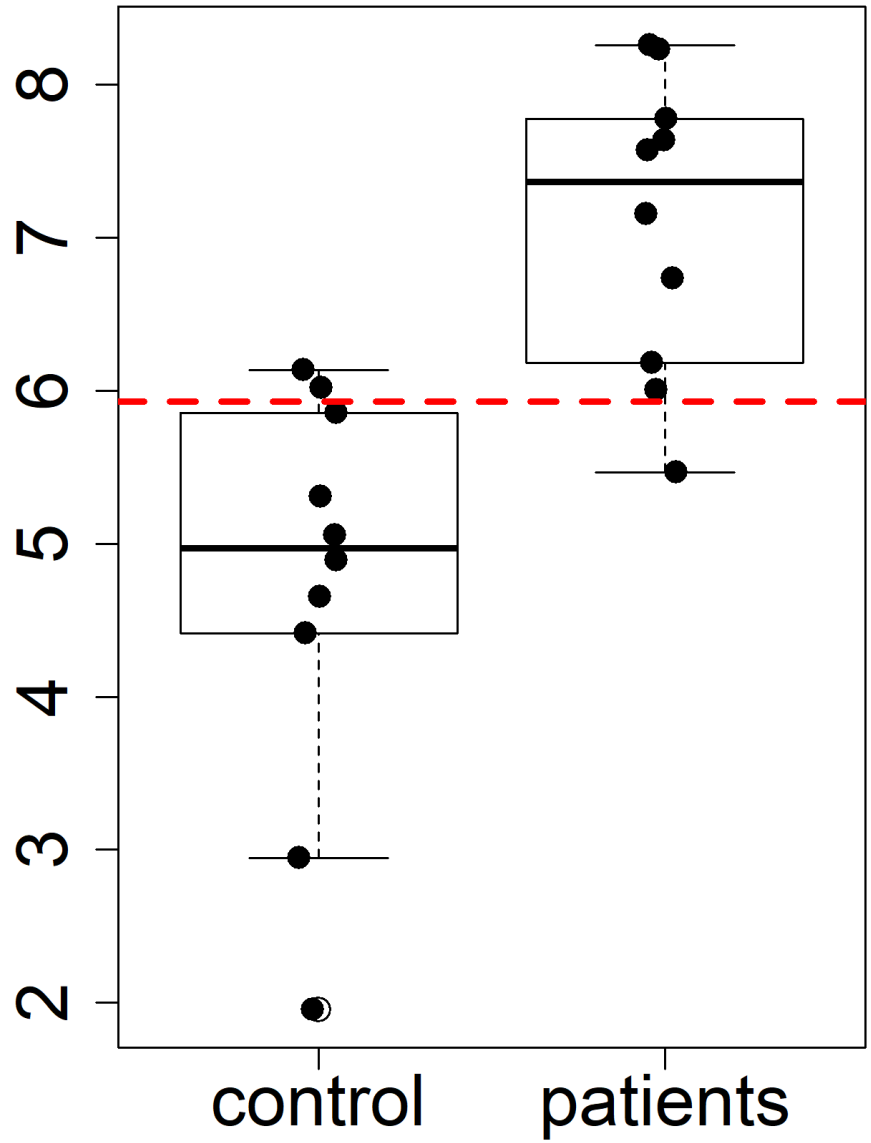
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7



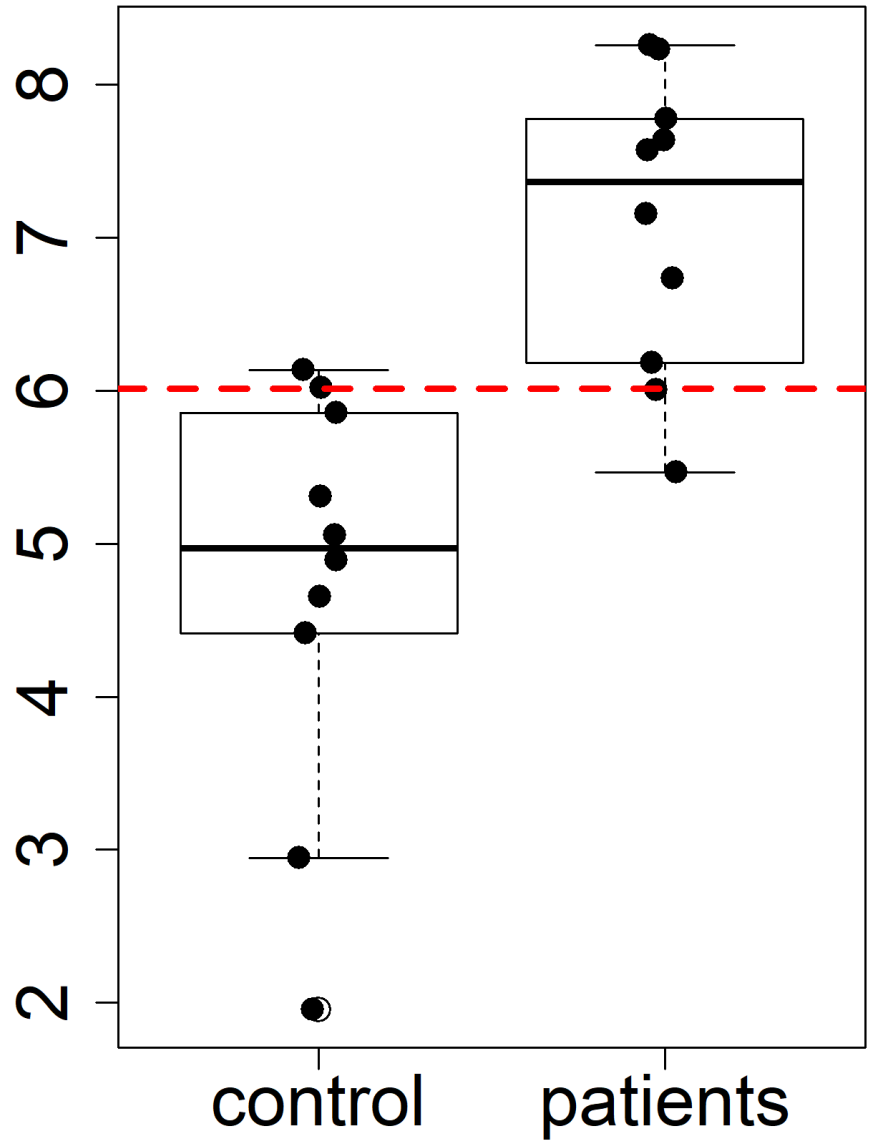
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8



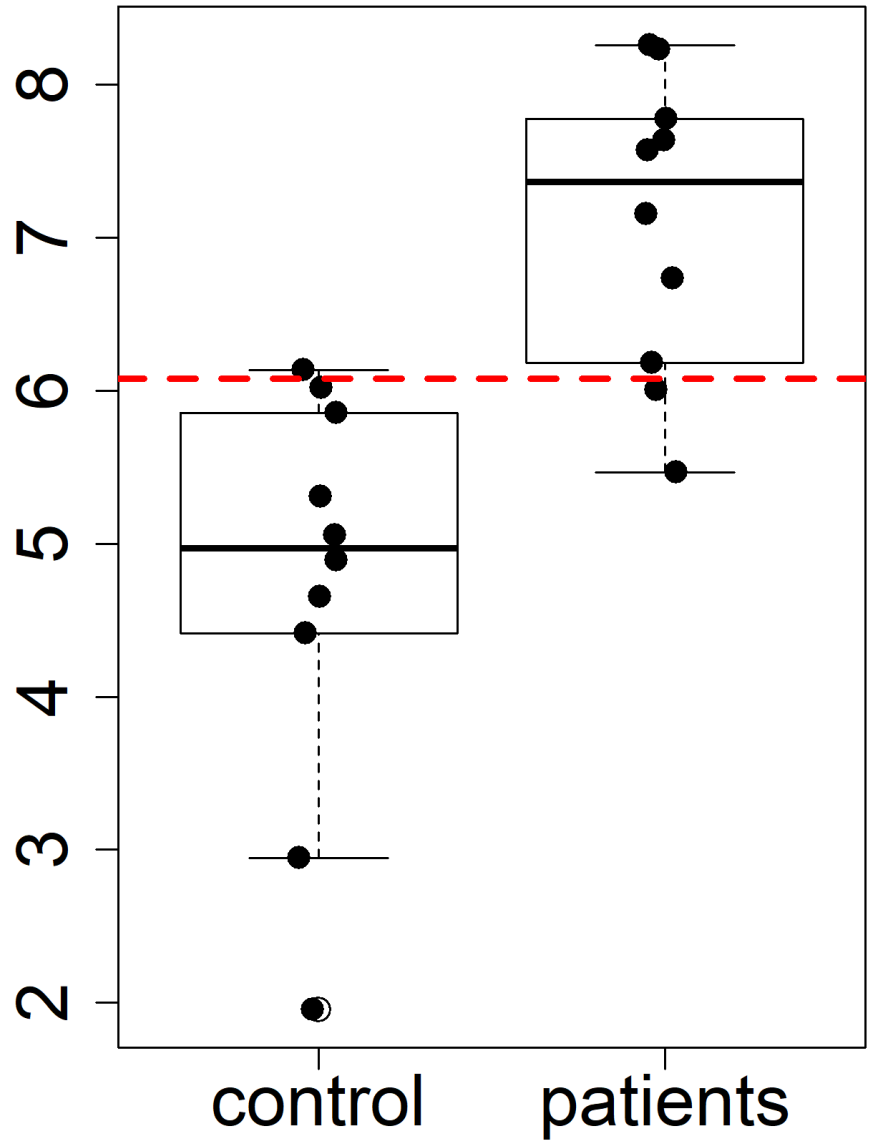
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...



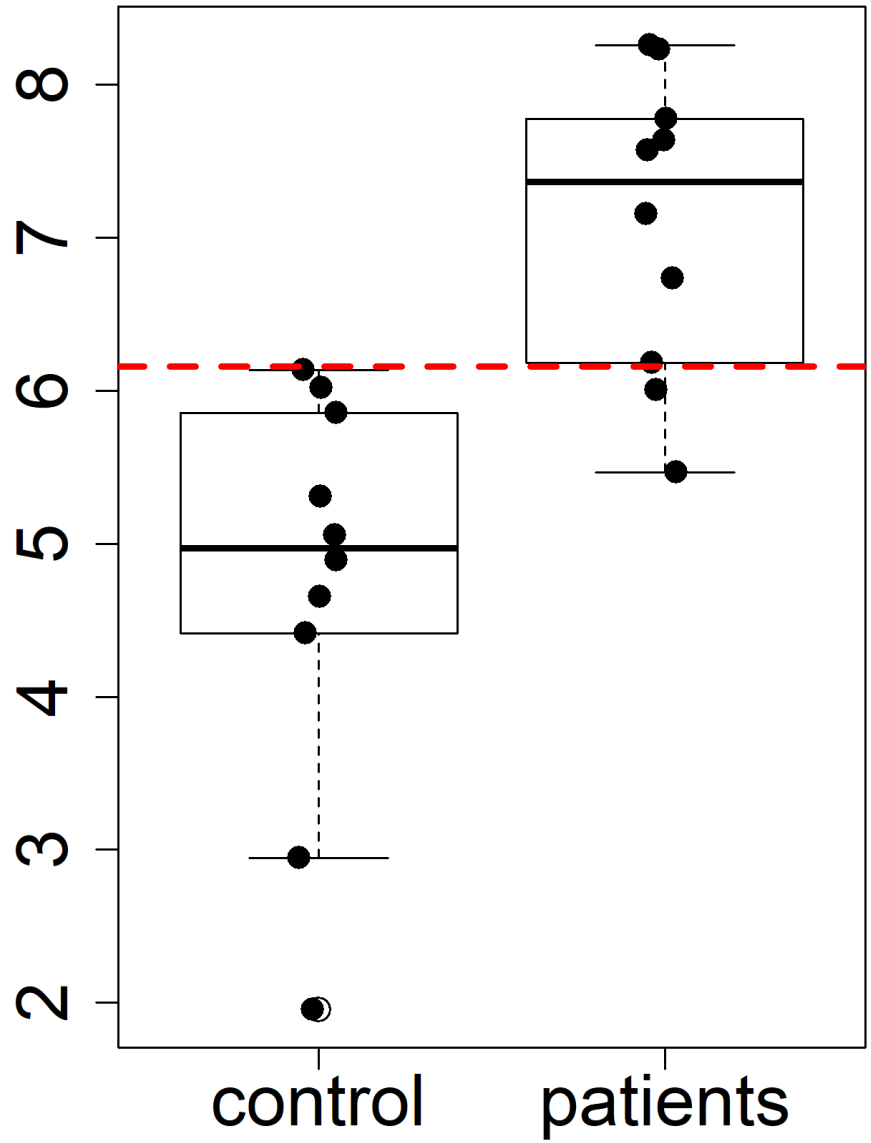
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...



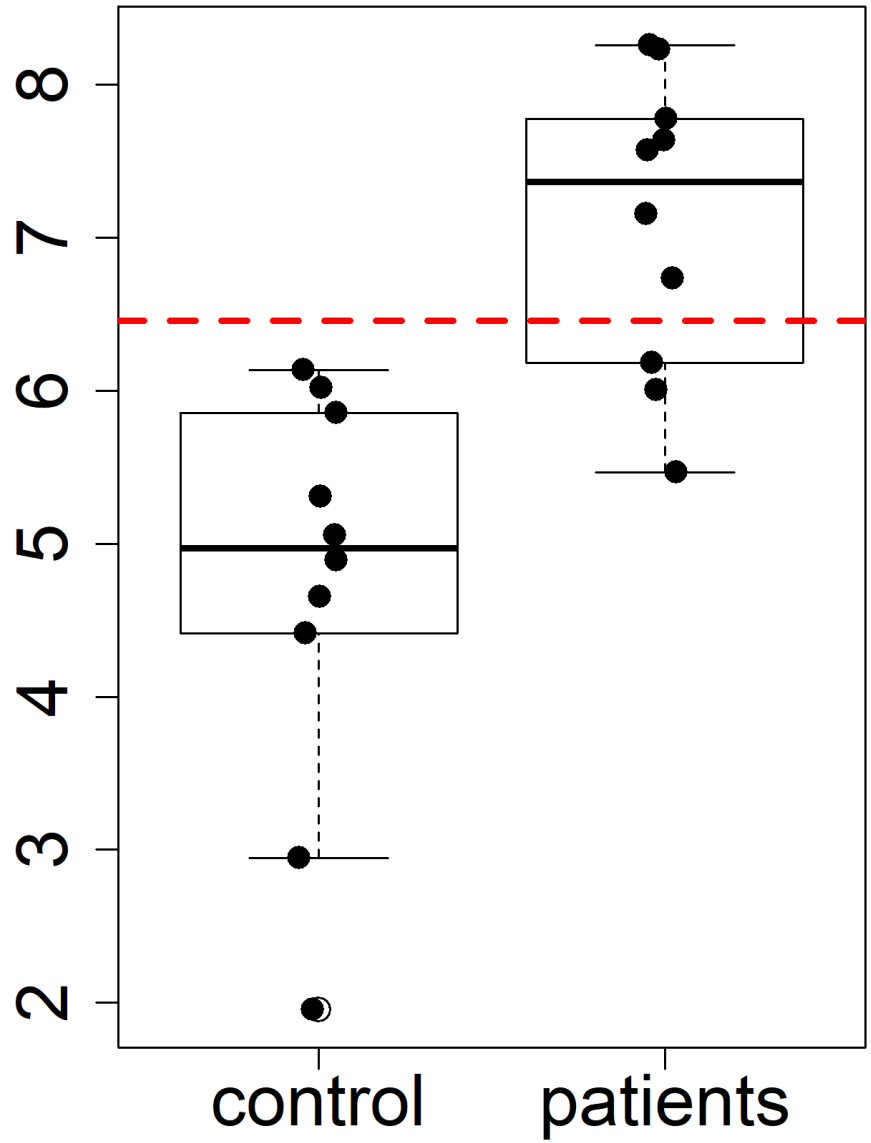
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0



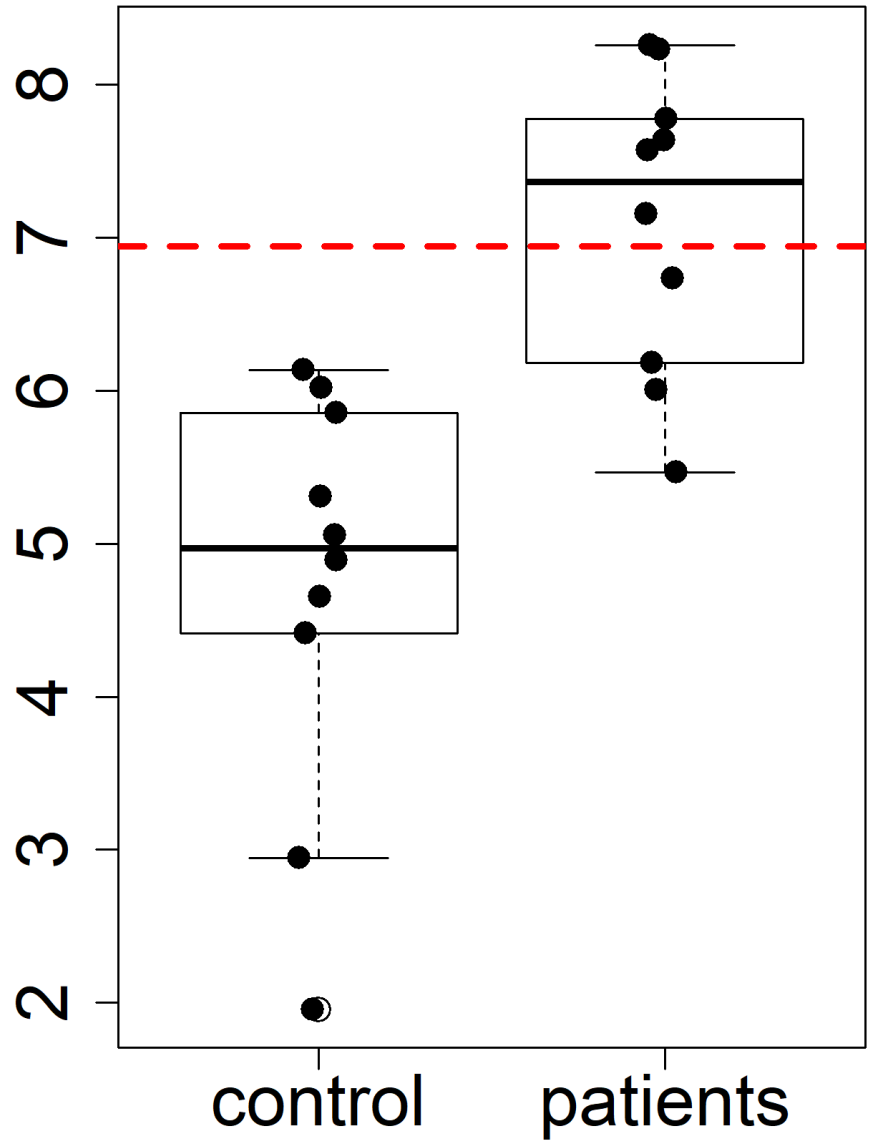
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0



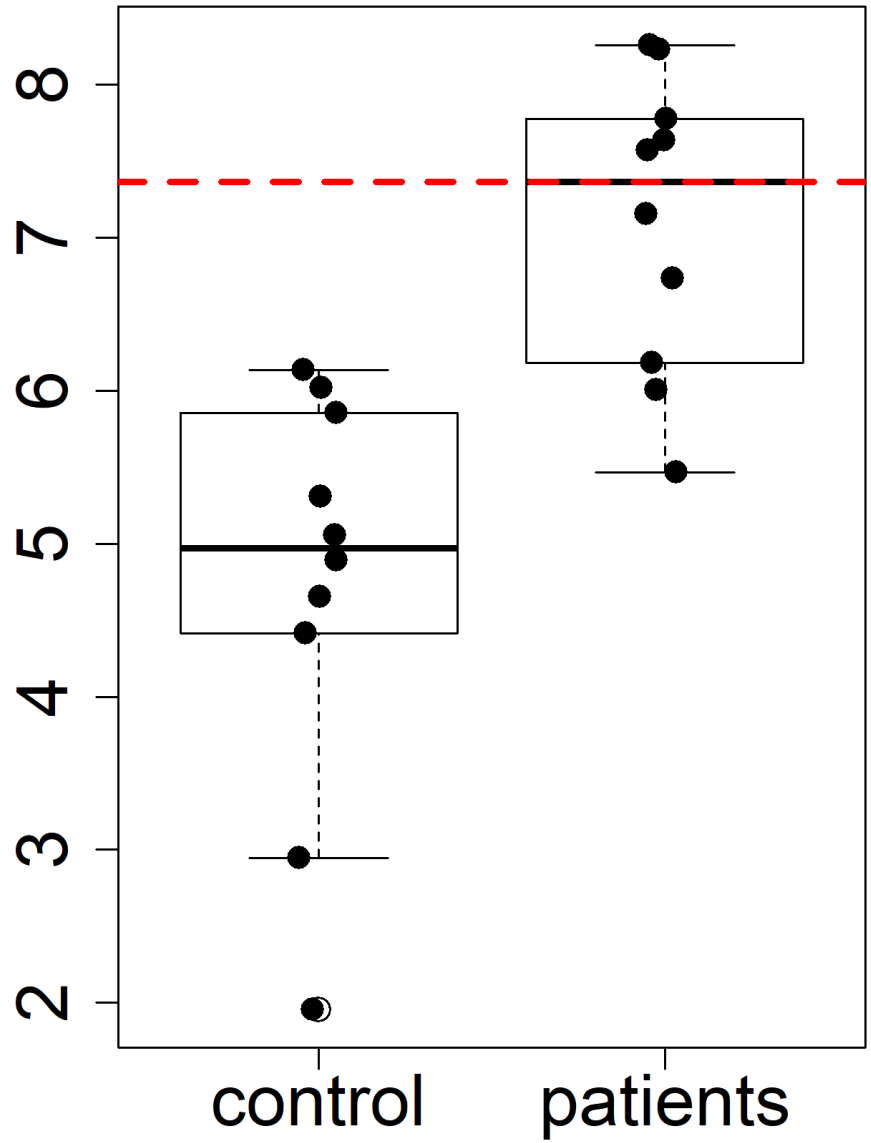
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...



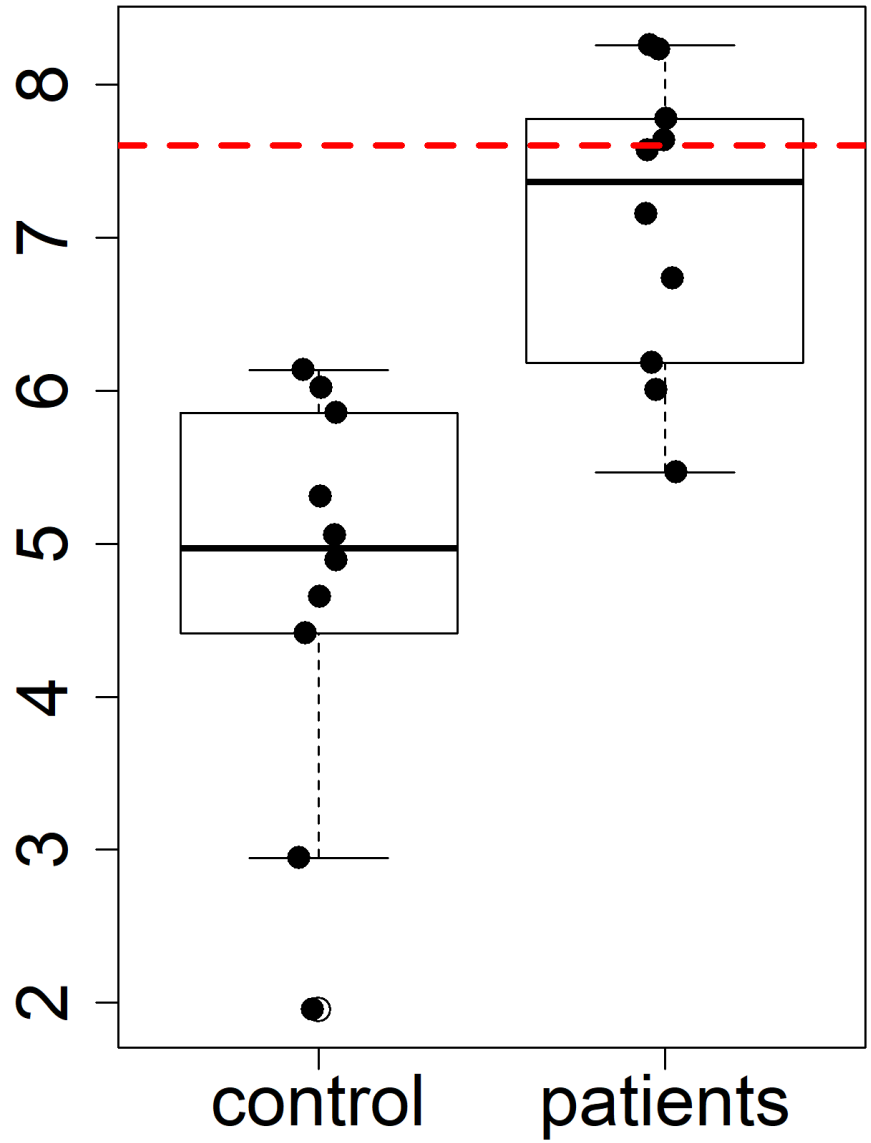
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...



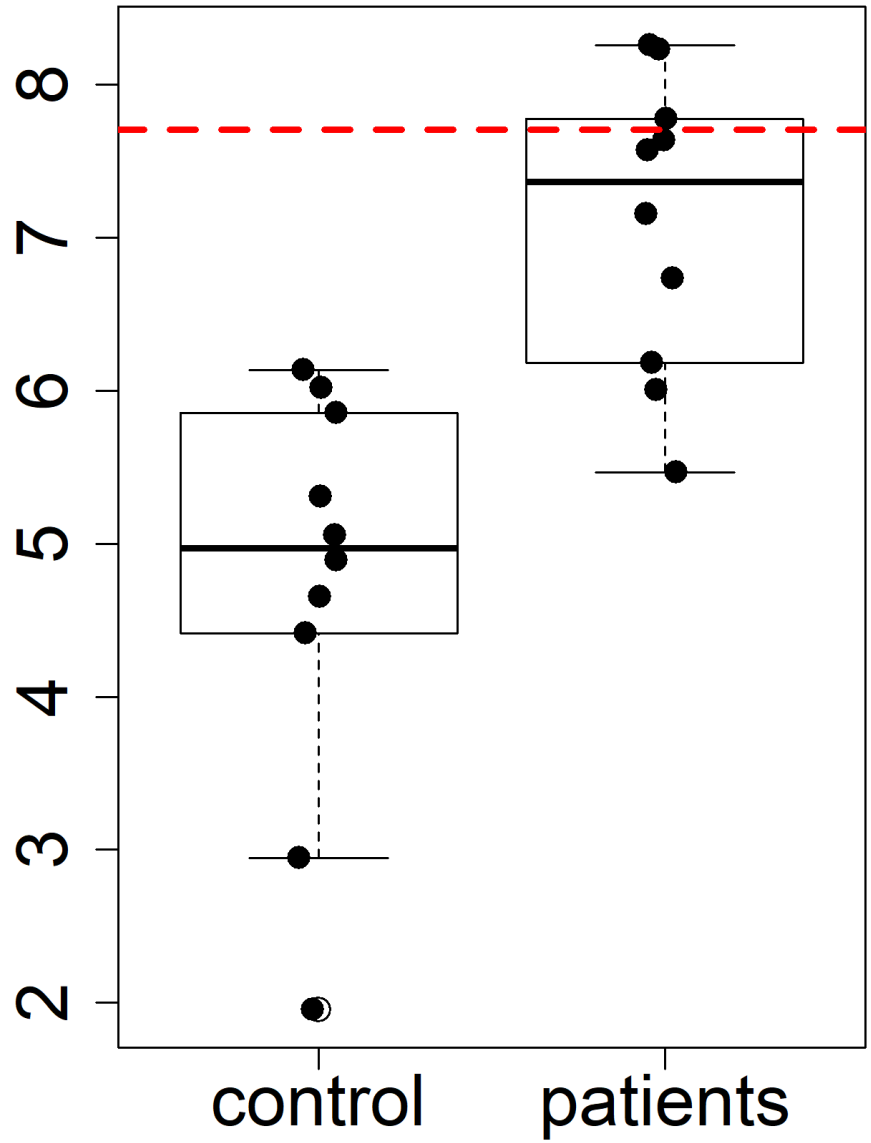
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...



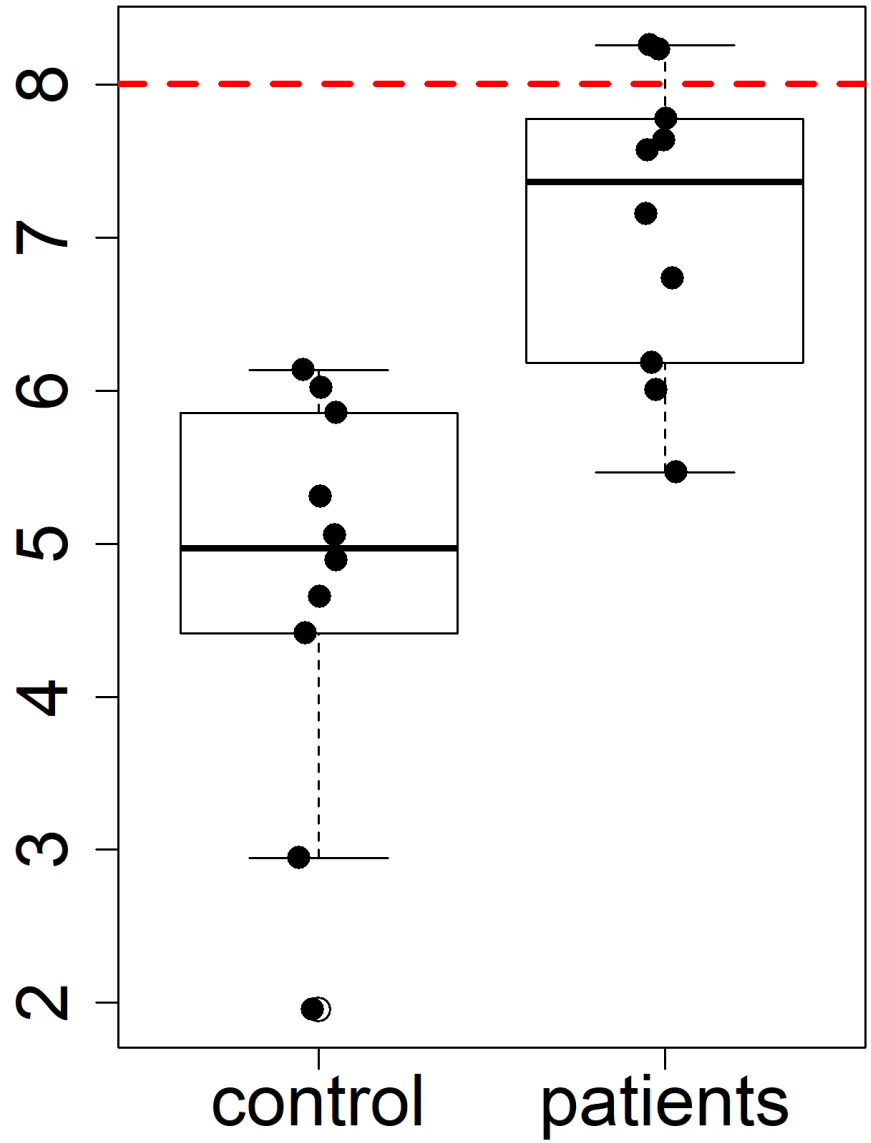
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...



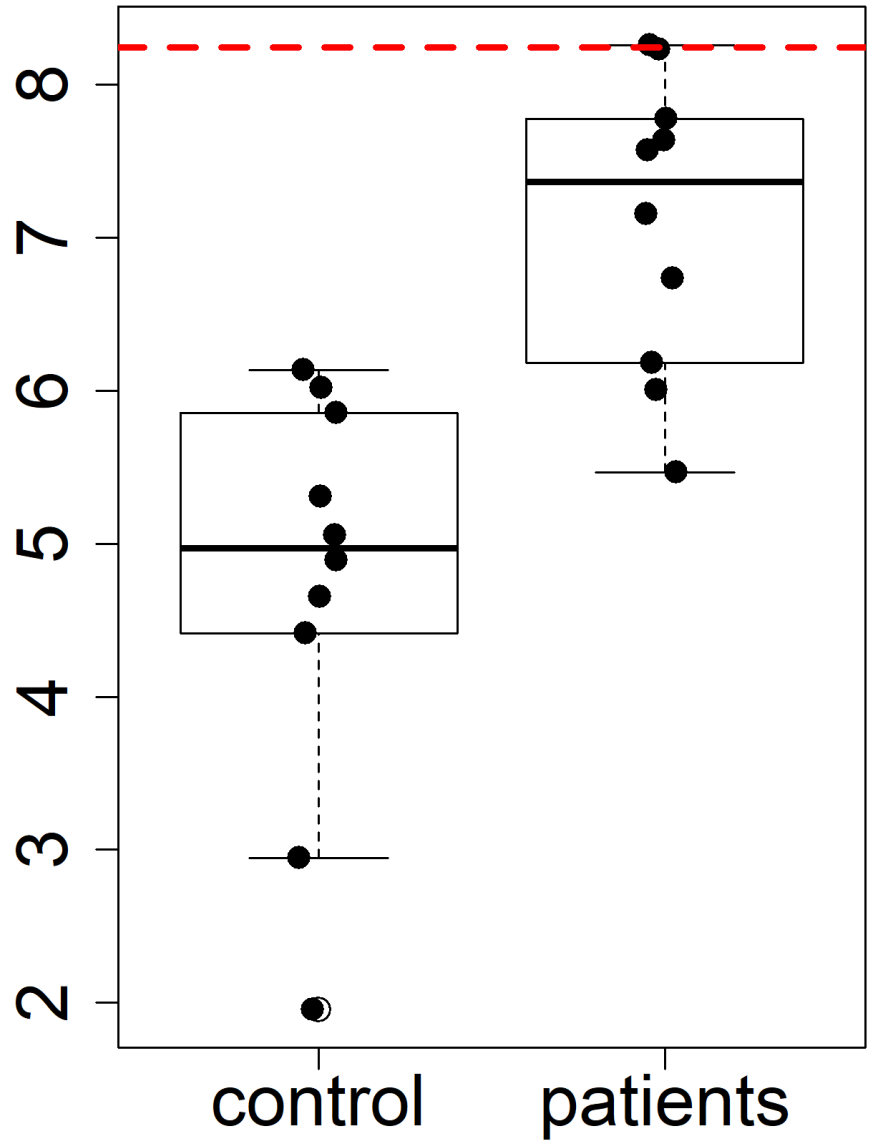
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...



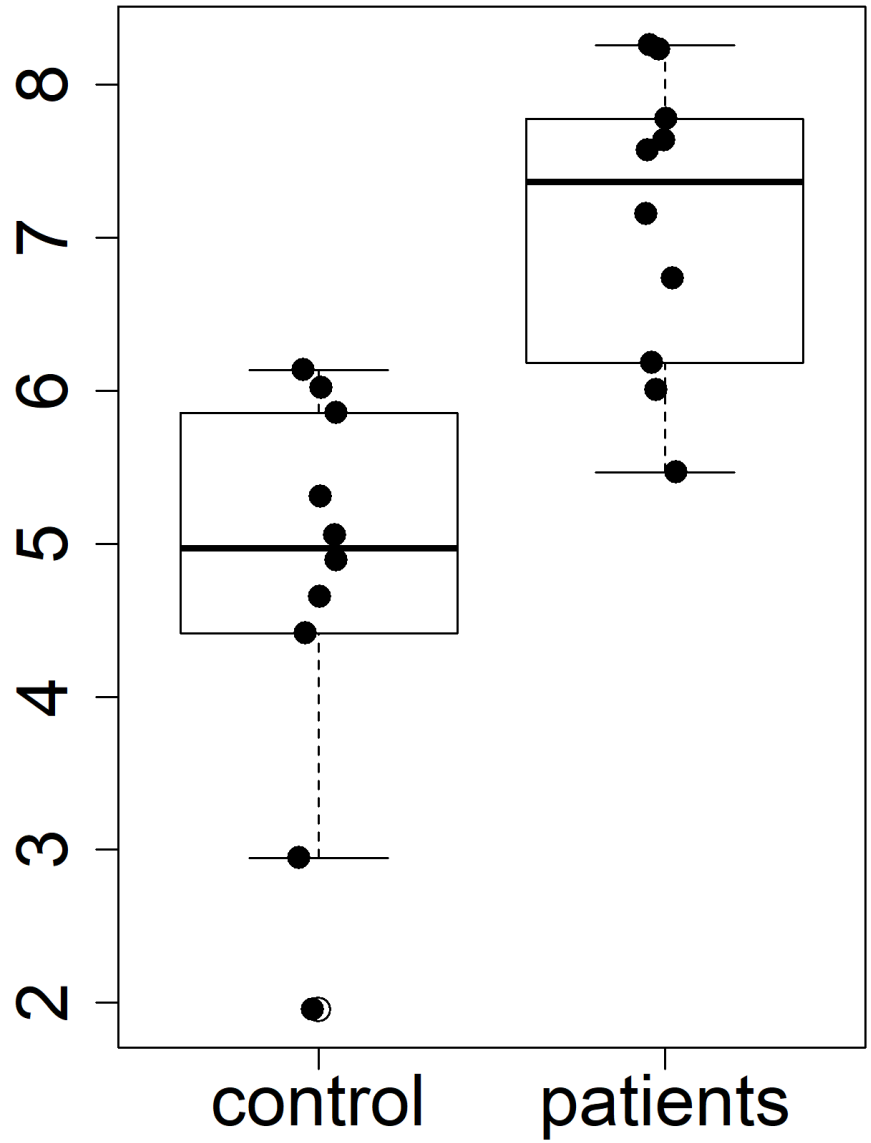
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...
19	8.25	0.1	1.0



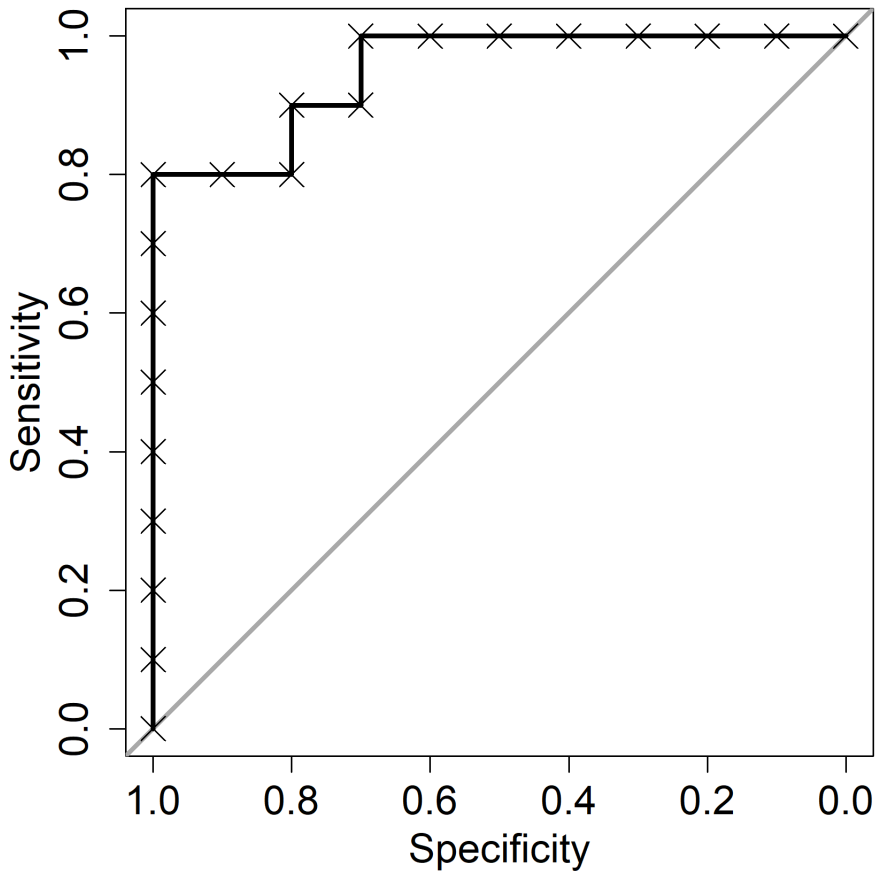
Construction of ROC curves

step	cutoff	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...
19	8.25	0.1	1.0
20	∞	0.0	1.0



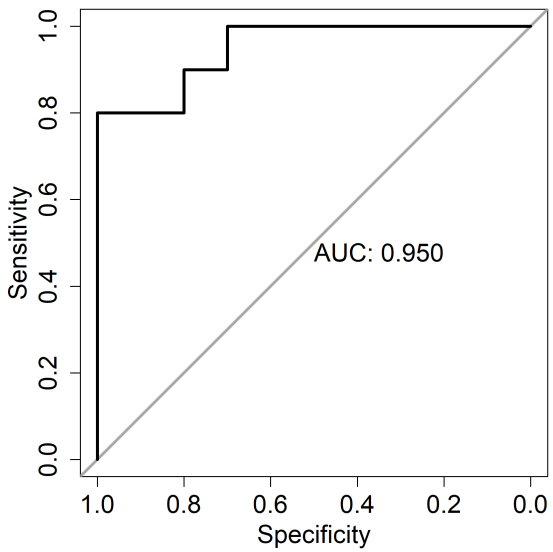
Construction of ROC curves

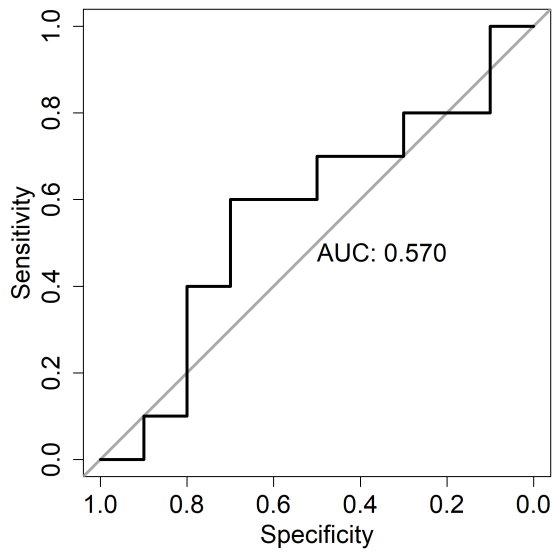
step	cut-off	sens	spec
0	$-\infty$	1.0	0.0
1	2.45	1.0	0.1
2	3.68	1.0	0.2
...	...	...	...
8	5.66	0.9	0.7
9	5.93	0.9	0.8
...	...	...	...
12	6.16	0.8	1.0
13	6.46	0.7	1.0
...	...	...	...
19	8.25	0.1	1.0
20	∞	0.0	1.0

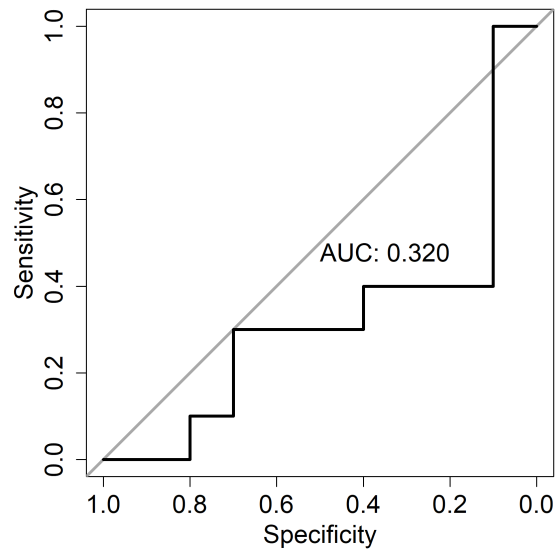


Area under the curve (AUC)

- area under the ROC curve ($\in [0, 1]$)
- measure of the overall ability of the biomarker to predict the class
- AUC $= 1 \rightarrow$ perfect separation of the two groups
- AUC $= 0.5 \rightarrow$ groups are not distinguishable
- AUC $< 0.5 \rightarrow$ direction was chosen wrong

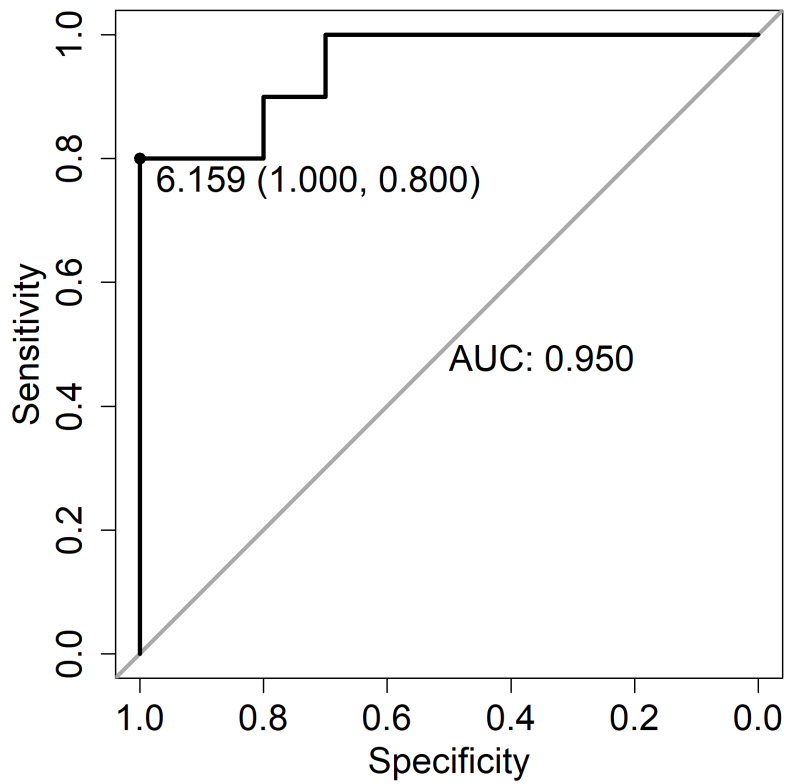






Choice of best cut-off

- in a ROC curve, sensitivity and specificity of all possible cut-offs are shown
- Youden criterion: choose threshold with highest sum of sensitivity and specificity
- calculated best threshold has to be tested/validated on a independent dataset

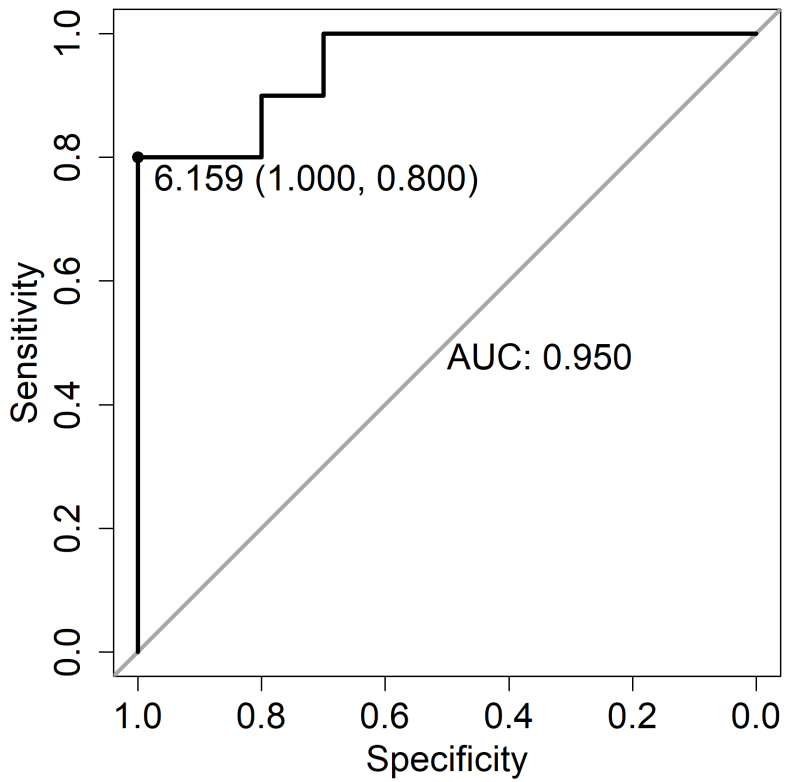


ROC curves with package pROC

$x1$: values of the control group

$x2$: values of the patient group

```
library(pROC)
ROC <- pROC::roc(controls = x1, cases = x2)
plot(ROC, print.thres = "best", print.auc = TRUE,
direction = "auto")
```



alternative R packages for ROC curves

- ROCR
  - supports more different perfomance measures (not only sens and spec)
  - handling is more technical / machine learning oriented
  - faster at calculating ROC curves and AUC for large data sets
- ROC (Bioconductor)
- plotROC
- ROCit

EXERCISE

Write own R functions

Why writing one's own functions

- A function is a designated section of code that performs specific task.
- Function code is written once & reused unlimitedly by calling its function name.
- **Advantages:** Code modularity, reusability, shorter code and fewer errors (paradigm: procedural programming).
- When called, functions get needed data & settings via arguments, i.e. variables for specific objects or values.
- Arguments are mandatory or optional i.e., with default value.
- Functions can return a result value or object.

Writing one's own functions: function()

function {base} R Documentation

Function Definition

Description

These functions provide the base mechanisms for defining new functions in the R language.

Usage

```
function( arglist ) expr
return(value)
```

Arguments

arglist	Empty or one or more name or name=expression terms.
expr	An expression.
value	An expression.

Writing one's own functions: Hello world!

Template: $name \leftarrow function(arg1 = x, arg2 = y, \dots)\{\dots\}$

```
> hello.world <- function() {print("Hello world!")}
> hello.world()
[1] "Hello world!"
>
```

For single-line definitions curly brackets are optional:

```
> hello.world <- function() print("Hello world!")
> hello.world()
[1] "Hello world!"
>
```

For multiline definitions they are mandatory:

```
> hello.world <- function(){
+   cat("Hello ")
+   cat("world!")
+ }
> hello.world()
Hello world!
>
```

Writing one's own functions

```
my.exp.2 <- function(base=2, power=NULL){

  result <- 1

  if(!is.element(power, 1:1000)){
    stop("Please define a power (integer) in the range 1 to 100!")
  }else{
    for(i in 1:power){
      result <- result*base
    }
  }

  return(result)
}
```

- Yes, due to R's "`x^y`" this function is obsolete :-)...
- `base` is optional (default value 2) and `power` is mandatory.
- `stop()` stops the execution of `my.exp.2()` with an error.
- `return()` returns the value of variable `result`.

Writing one's own functions

```
> my.exp.2(power=3)
[1] 8
> my.exp.2(base=3,power=2)
[1] 9
> my.exp.2(base=10,power=3)
[1] 1000
> my.exp.2(base=2,power=100)
[1] 1.267651e+30
> my.exp.2(base=2,power=1000)
[1] 1.071509e+301
> my.exp.2(base=2,power=10000)
Error in my.exp.2(base = 2, power = 10000) :
  Please define a power (integer) in the range 1 to 100!
> my.exp.2(base=2,power="bla")
Error in my.exp.2(base = 2, power = "bla") :
  Please define a power (integer) in the range 1 to 100!
```

EXERCISE