

# BST02: Using R for Statistics in Medical Research

## Part B: Basic use of R

Eleni-Rosalina Andrinopoulou

Department of Biostatistics, Erasmus Medical Center

✉ [e.andrinopoulou@erasmusmc.nl](mailto:e.andrinopoulou@erasmusmc.nl)

# Basic Use of R

## In this Section

---

- ▶ Using R
- ▶ Examples with Data
- ▶ Getting Familiar with R
- ▶ Importing data and saving your work
- ▶ A lot of practice

# Using R

---

- ▶ R is a command-based procedural language
  - ▶ write and execute commands
  - ▶ use and define functions
- ▶ You may write the commands in the R console (Windows) or in a shell (Linux)

**You will become more familiar with the syntax as you use it**

# Using R

---

- ▶ Strongly advisable to use a suitable text editor - Some available options:
  - ▶ RWinEdt (for Windows; you also need WinEdt installed)
  - ▶ Tinn-R (for Windows; <http://sciviews.org/Tinn-R/>)
  - ▶ Rkward (for Linux)
  - ▶ Emacs (w. ESS, all platforms)
  - ▶ Visual Studio (for Windows)
  - ▶ Rstudio (all major platforms; <http://www.rstudio.org/>)
  - ▶ for more check <https://r-dir.com/blog/2013/01/list-of-r-editors.html>

# Using R

---

- ▶ For this course: Rstudio (<http://www.rstudio.org/>)
  - ▶ free
  - ▶ works fine in Windows, MacOS and Linux
  - ▶ helpful with errors
  - ▶ alternative output options

# Using R

---

- ▶ Can I use R without Rstudio?
- ▶ Can I use Rstudio without R?

## Practical Examples

---

► **Package `survival` - pbc data set**

| id | time | status | trt | age      | sex | bili | chol |
|----|------|--------|-----|----------|-----|------|------|
| 1  | 400  | 2      | 1   | 58.76523 | f   | 14.5 | 261  |
| 2  | 4500 | 0      | 1   | 56.44627 | f   | 1.1  | 302  |
| 3  | 1012 | 2      | 1   | 70.07255 | m   | 1.4  | 176  |
| 4  | 1925 | 2      | 1   | 54.74059 | f   | 1.8  | 244  |
| 5  | 1504 | 1      | 2   | 38.10541 | f   | 3.4  | 279  |



## Practical Examples

---

- ▶ **id**: case number
- ▶ **time**: number of days between registration and the earlier of death, transplantation, or study analysis in July, 1986
- ▶ **status**: status at endpoint, 0/1/2 for censored, transplant, dead
- ▶ **trt**: 1/2/NA for D-penicillamine, placebo, not randomised
- ▶ **age**: in years
- ▶ **sex**: m/f
- ▶ **bili**: serum bilirunbin (mg/dl)
- ▶ **chol**: serum cholesterol (mg/dl)

More details:

<https://stat.ethz.ch/R-manual/R-devel/library/survival/html/pbc.html>

## Practical Examples

---

- What is a **scalar**/vector/matrix

| id | time | status | trt | age             | sex | bili | chol | dt |
|----|------|--------|-----|-----------------|-----|------|------|----|
| 1  | 400  | 2      | 1   | 58.76523        | f   | 14.5 | 261  | 1  |
| 2  | 4500 | 0      | 1   | 56.44627        | f   | 1.1  | 302  | 2  |
| 3  | 1012 | 2      | 1   | <b>70.07255</b> | m   | 1.4  | 176  | 3  |
| 4  | 1925 | 2      | 1   | 54.74059        | f   | 1.8  | 244  | 4  |
| 5  | 1504 | 1      | 2   | 38.10541        | f   | 3.4  | 279  | 5  |

## Practical Examples

---

- What is a scalar/**vector**/matrix

| id | time | status   | trt | age      | sex | bili | chol |
|----|------|----------|-----|----------|-----|------|------|
| 1  | 400  | <b>2</b> | 1   | 58.76523 | f   | 14.5 | 261  |
| 2  | 4500 | <b>0</b> | 1   | 56.44627 | f   | 1.1  | 302  |
| 3  | 1012 | <b>2</b> | 1   | 70.07255 | m   | 1.4  | 176  |
| 4  | 1925 | <b>2</b> | 1   | 54.74059 | f   | 1.8  | 244  |
| 5  | 1504 | <b>1</b> | 2   | 38.10541 | f   | 3.4  | 279  |

## Practical Examples

---

- What is a scalar/**vector**/matrix

| id | time        | status | trt | age      | sex | bili | chol |
|----|-------------|--------|-----|----------|-----|------|------|
| 1  | <b>400</b>  | 2      | 1   | 58.76523 | f   | 14.5 | 261  |
| 2  | <b>4500</b> | 0      | 1   | 56.44627 | f   | 1.1  | 302  |
| 3  | <b>1012</b> | 2      | 1   | 70.07255 | m   | 1.4  | 176  |
| 4  | <b>1925</b> | 2      | 1   | 54.74059 | f   | 1.8  | 244  |
| 5  | <b>1504</b> | 1      | 2   | 38.10541 | f   | 3.4  | 279  |

## Practical Examples

---

- What is a scalar/vector/**matrix**

| id | time        | status   | trt | age      | sex | bili | chol |
|----|-------------|----------|-----|----------|-----|------|------|
| 1  | <b>400</b>  | <b>2</b> | 1   | 58.76523 | f   | 14.5 | 261  |
| 2  | <b>4500</b> | <b>0</b> | 1   | 56.44627 | f   | 1.1  | 302  |
| 3  | <b>1012</b> | <b>2</b> | 1   | 70.07255 | m   | 1.4  | 176  |
| 4  | <b>1925</b> | <b>2</b> | 1   | 54.74059 | f   | 1.8  | 244  |
| 5  | <b>1504</b> | <b>1</b> | 2   | 38.10541 | f   | 3.4  | 279  |

## Practical Examples

---

- What is a scalar/vector/**matrix**

| id | time | status | trt | age      | sex | bili        | chol       |
|----|------|--------|-----|----------|-----|-------------|------------|
| 1  | 400  | 2      | 1   | 58.76523 | f   | <b>14.5</b> | <b>261</b> |
| 2  | 4500 | 0      | 1   | 56.44627 | f   | <b>1.1</b>  | <b>302</b> |
| 3  | 1012 | 2      | 1   | 70.07255 | m   | <b>1.4</b>  | <b>176</b> |
| 4  | 1925 | 2      | 1   | 54.74059 | f   | <b>1.8</b>  | <b>244</b> |
| 5  | 1504 | 1      | 2   | 38.10541 | f   | <b>3.4</b>  | <b>279</b> |

# Practical Examples

---

- ▶ Common questions
  - ▶ What is the average age?
  - ▶ What is the average serum bilirubin?
  - ▶ What is the average serum cholesterol?
  - ▶ What is the percentage of females?
  - ▶ How many missing values do we have for serum cholesterol?

**All these questions can be answered using R!**

# Getting Familiar with R

---

- ▶ Elementary commands: **expressions** and **assignments**
- ▶ An **expression** given as command is evaluated printed and discarded
- ▶ An **assignment** evaluates an expression and passes the value to a variable - the result is not automatically printed



## Getting Familiar with R

---

Expression is given as a command,

```
103473
```

```
[1] 103473
```

- ▶ However, it cannot be viewed again unless the command is rerun.

## Getting Familiar with R

---

Expression is given as a command,

```
103473
```

```
[1] 103473
```

- ▶ However, it cannot be viewed again unless the command is rerun.

In order to store information, the expression should assign the command

```
x <- 103473
```

```
x
```

```
[1] 103473
```

# Getting Familiar with R

---

## You can use R as a calculator!

- ▶ Basic arithmetics

`+, -, *, /, ^`

```
y <- 103473 + 100000  
y
```

```
[1] 203473
```

- ▶ Complicated arithmetics

# Getting Familiar with R

---

## Tips:

- ▶ R is case sensitive, e.g.,
  - ▶ **"sex"** is different than **"Sex"**
- ▶ Commands are separated by a semi-colon or by a newline
- ▶ Comments can be put anywhere, starting with a hashmark **#**: everything to the end of the line is a comment
- ▶ Assign a value to an object by **<-** or **=**
- ▶ Working directory: get with **getwd()** and set with **setwd()**

# Getting Familiar with R

---

- ▶ Missing values
  - ▶ are coded as **NA** (i.e., not available) **is.na()**
- ▶ Infinity
  - ▶ is coded as **Inf** (plus infinity) or **-Inf** (minus infinity) **is.finite()**
- ▶ The Null objects
  - ▶ are coded as **NULL** (undefined) **is.null()**
- ▶ Not a number
  - ▶ is coded as **NaN** (Not a Number). Example:

```
0/0
```

```
[1] NaN
```

# Importing Data

---

- ▶ function: **read.table()**, **read.csv()** and its variants
  - ▶ note: use forward slashes or double backward slashes in the file names, e.g.,  
"C:/Documents and Settings/User/Data/file.txt" or  
"C:\\Documents and Settings\\User\\Data\\file.txt"
- ▶ Specialized functions for importing data from other programs
  - ▶ package: **foreign**, function: **read.spss()**, **read.dta()**
  - ▶ package: **Hmisc**, function: **sas.get()**
  - ▶ package: **openxlsx**, function: **read.xlsx()**
  - ▶ package: **readxl**, function: **read\_excel()**
  - ▶ package: **haven**, function: **read\_spss()**
  - ▶ etc

# Exporting Data

---

- ▶ Specialized functions for exporting data to other programs
  - ▶ function: **write.table()**, **write.csv()**
  - ▶ package: **foreign**, function: **write.spss()**, **write.dta()**
  - ▶ package: **openxlsx**, function: **write.xlsx()**
  - ▶ etc

# Saving and Loading your Work

---

## *Multiple objects:*

- ▶ You can save your R objects using **save()**
  - ▶ be careful about overwriting
- ▶ You can load your saved R objects using **load()**

## *Single object:*

- ▶ Using **saveRDS()** you can save a single R object
- ▶ Using **readRDS()** you can load a single R object
  - ▶ we will need an assignment statement to store the results

**Save your code by using the tab File in Rstudio!**



# Saving and Loading your Work

---

## Tips:

- ▶ Short names are preferred over longer names
- ▶ Try to avoid using names that contain symbols
- ▶ Avoid spaces in names
- ▶ Remove any comments in your data set
- ▶ Make sure that any missing values in your data set are indicated with the same value (or no value)

# Summary

---

## Basic functions

- ▶ `getwd()`, `setwd()`,
- ▶ `is.na()`,  
`is.finite()`,  
`is.null()`

## Import/Export

- ▶ `read.csv()`, `write.csv()`
- ▶ `read.xlsx()`, `write.xlsx()`
- ▶ `read.table()`, `write.table()`

## Save/Load

- ▶ `save()`, `saveRDS()`
- ▶ `load()`, `readRDS()`

# Practice

---

## Demos

- ▶ Basic R [R](#) [html](#)
- ▶ Importing and Saving [R](#) [html](#)

## Practicals

- ▶ Importing and Saving [html](#)

# Common Objects in R

## In this Section

---

- ▶ Objects in R
- ▶ Data types
- ▶ Data structures
- ▶ A lot of practice

# Objects in R

---

- ▶ In R Everything (data, results, ...) is an object
- ▶ In order to list the created objects use the following functions

```
objects()  
ls()
```

# Objects in R

---

- ▶ In R Everything (data, results, ...) is an object
- ▶ In order to list the created objects use the following functions

```
objects()  
ls()
```

- ▶ In order to remove objects

```
rm()  
rm(list=ls(all=TRUE))
```

# Objects in R

---

- To investigate a specific object (e.g. pbc)

```
str(pbc[,c("id", "time", "status", "trt", "age", "sex", "bili", "chol")])
```

```
'data.frame':  418 obs. of  8 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ time    : int  400 4500 1012 1925 1504 2503 1832 2466 2400 51 ...
 $ status:  : int   2 0 2 2 1 2 0 2 2 2 ...
 $ trt     : int   1 1 1 1 2 2 2 2 1 2 ...
 $ age     : num   58.8 56.4 70.1 54.7 38.1 ...
 $ sex     : Factor w/ 2 levels "m","f": 2 2 1 2 2 2 2 2 2 2 ...
 $ bili    : num   14.5 1.1 1.4 1.8 3.4 0.8 1 0.3 3.2 12.6 ...
 $ chol    : int   261 302 176 244 279 248 322 280 562 200 ...
```



# Data Types

---

The simplest data types are:

- ▶ **numeric** : quantitative data
- ▶ **character** : qualitative data
- ▶ **integer** : whole numbers
- ▶ **logical** : TRUE or FALSE
- ▶ **factors** : qualitative data (levels)

## Data Types in R

---

To find out what type of object you have, you can use the following function

```
mode(pbc$age)
```

```
[1] "numeric"
```

```
str(pbc$age)
```

```
num [1:418] 58.8 56.4 70.1 54.7 38.1 ...
```

# Data Structures

---

The most important data structures are:

- ▶ **Scalar** a single element
- ▶ **Vectors** have the same type of elements
- ▶ **Matrices** have the same type of elements with the same length
- ▶ **Arrays** have the same type of elements with the same length but can store the data in more than two dimensions
- ▶ **Data frames** have elements of different type with the same length
- ▶ **Lists** have elements of different type and length

# Data Structures

---

**How do these data structures look like?**

# Data Structures

---

- Differences between **vector**, matrix, array, data.frame and list

```
pbcc[1:6, c("age")]
```

```
[1] 58.76523 56.44627 70.07255 54.74059 38.10541 66.25873
```

# Data Structures

---

- Differences between vector, **matrix**, array, data.frame and list

```
pbc[1:6, c("age", "bili", "chol")]
```

|   | age      | bili | chol |
|---|----------|------|------|
| 1 | 58.76523 | 14.5 | 261  |
| 2 | 56.44627 | 1.1  | 302  |
| 3 | 70.07255 | 1.4  | 176  |
| 4 | 54.74059 | 1.8  | 244  |
| 5 | 38.10541 | 3.4  | 279  |
| 6 | 66.25873 | 0.8  | 248  |

# Data Structures

- Differences between vector, matrix, **array**, data.frame and list

```
pbmc[1:3, c("age", "bili", "chol")]
```

|   | age      | bili | chol |
|---|----------|------|------|
| 1 | 58.76523 | 14.5 | 261  |
| 2 | 56.44627 | 1.1  | 302  |
| 3 | 70.07255 | 1.4  | 176  |

```
pbmc[4:6, c("age", "bili", "chol")]
```

|   | age      | bili | chol |
|---|----------|------|------|
| 4 | 54.74059 | 1.8  | 244  |
| 5 | 38.10541 | 3.4  | 279  |
| 6 | 66.25873 | 0.8  | 248  |

# Data Structures

---

- Differences between vector, matrix, **array**, data.frame and list

```
pbc[1:2, c("protime", "time")]
```

|   | protime | time |
|---|---------|------|
| 1 | 12.2    | 400  |
| 2 | 10.6    | 4500 |

```
pbc[3:4, c("protime", "time")]
```

|   | protime | time |
|---|---------|------|
| 3 | 12.0    | 1012 |
| 4 | 10.3    | 1925 |



# Data Structures

---

- Differences between vector, matrix, array, **data.frame** and list

```
pbmc[1:6, c("id", "sex", "bili", "chol")]
```

|   | id | sex | bili | chol |
|---|----|-----|------|------|
| 1 | 1  | f   | 14.5 | 261  |
| 2 | 2  | f   | 1.1  | 302  |
| 3 | 3  | m   | 1.4  | 176  |
| 4 | 4  | f   | 1.8  | 244  |
| 5 | 5  | f   | 3.4  | 279  |
| 6 | 6  | f   | 0.8  | 248  |

# Data Structures

- Differences between vector, matrix, array, data.frame and **list**

```
pbpc[1:6, c("sex")]
```

```
[1] f f m f f f  
Levels: m f
```

```
pbpc[1:2, c("sex", "bili")]
```

```
  sex bili  
1   f 14.5  
2   f  1.1
```

```
pbpc[1:4, c("age")]
```

```
[1] 58.76523 56.44627 70.07255 54.74059
```

# Data Structures in R

---

**Let's now create different data structure in R!**

# Data Structures in R

---

## Create a vector

```
vec <- c(1, 2, 3, 4, 5)  
vec
```

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

```
vec <- c(1:5)  
vec
```

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

# Data Structures in R

---

## Create a matrix

```
vec <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
mat <- matrix(data = vec,
              nrow = 3, ncol = 3)
mat
```

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

# Data Structures in R

## Create a matrix

```
vec <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
mat <- matrix(data = vec,
              nrow = 3, ncol = 3)
mat
```

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

```
vec <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
mat <- matrix(data = vec,
              nrow = 3, ncol = 3,
              byrow = TRUE)
mat
```

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

# Data Structures in R

## Create an array

```
ar <- array(data = c(1, 2, 3, 4, 5, 6, 7, 8), dim = c(2, 2, 2))  
ar
```

, , 1

|      | [,1] | [,2] |
|------|------|------|
| [1,] | 1    | 3    |
| [2,] | 2    | 4    |

, , 2

|      | [,1] | [,2] |
|------|------|------|
| [1,] | 5    | 7    |
| [2,] | 6    | 8    |

# Data Structures in R

---

## Create an array

```
ar <- array(data = c(1, 2, 3, 4), dim = c(2, 2, 1))  
ar
```

, , 1

|      | [,1] | [,2] |
|------|------|------|
| [1,] | 1    | 3    |
| [2,] | 2    | 4    |



# Data Structures in R

---

## Create a data frame

```
dtf <- data.frame(pbc[, "sex"],  
                  pbc[, "age"])  
dtf[1:3,]
```

|   | pbcc...sex.. | pbcc...age.. |
|---|--------------|--------------|
| 1 | f            | 58.76523     |
| 2 | f            | 56.44627     |
| 3 | m            | 70.07255     |

# Data Structures in R

---

## Create a data frame

```
dtf <- data.frame(pbc[, "sex"],  
                  pbc[, "age"])  
dtf[1:3,]
```

|   | pbc....sex.. | pbc....age.. |
|---|--------------|--------------|
| 1 | f            | 58.76523     |
| 2 | f            | 56.44627     |
| 3 | m            | 70.07255     |

```
dtf <- data.frame(Gender = pbc[, "sex"],  
                  Age = pbc[, "age"])  
dtf[1:3,]
```

|   | Gender | Age      |
|---|--------|----------|
| 1 | f      | 58.76523 |
| 2 | f      | 56.44627 |
| 3 | m      | 70.07255 |

# Data Structures in R

---

## Create a list

```
list1 <- list(vec = c(1:5), mat = pbc[1:2, c("age", "sex")])  
list1
```

\$vec

[1] 1 2 3 4 5

\$mat

|   | age      | sex |
|---|----------|-----|
| 1 | 58.76523 | f   |
| 2 | 56.44627 | f   |

# Summary

---

## Data types

- ▶ numeric
- ▶ character
- ▶ integer
- ▶ logical
- ▶ factors
- ▶ `str()`, `mode()`

## Data structures

- ▶ `c()`
- ▶ `matrix()`
- ▶ `array()`
- ▶ `data.frame()`
- ▶ `list()`

## Other

- ▶ `ls()`, `objects()`

# Practice

---

## Demos

- ▶ Common R Objects [R](#) [html](#)

## Practicals

- ▶ Common R Objects [html](#)

# Indexing and Subsetting

## In this Section

---

- ▶ Indexing
- ▶ Subsetting
- ▶ A lot of practice

# Indexing/Subsetting

---

- ▶ When transforming and analyzing data we often need to select specific observations or variables
  - ▶ Examples: Select ...
    - ▶ the 3rd element of vector age
    - ▶ the 3rd column of the pbc data set
    - ▶ the sex of the 10th patient
    - ▶ all information of the 5th patient
    - ▶ the serum cholesterol for all males
    - ▶ the age for male patients or patients that have serum bilirubin  $> 3$
    - ▶ the first measurement per patient



# Indexing/Subsetting

---

- ▶ This can be done using square bracket (**[ ]**) notation and indices.
- ▶ Three basic types
  - ▶ position indexing
  - ▶ logical indexing
  - ▶ name indexing

# Vectors

## Indexing with vector

- For position indexing, use a **positive** value to select an element

```
x <- c(6:17)
```

```
x
```

```
[1] 6 7 8 9 10 11 12 13 14 15 16 17
```

```
x[2]
```

```
[1] 7
```

- Use multiple positive values to select multiple elements

```
x[c(2,3,4)]
```

```
[1] 7 8 9
```

# Vectors

---

## Indexing with vector

- For position indexing, use duplicated **positive** values to select the same elements

```
x <- c(6:17)
```

```
x
```

```
[1] 6 7 8 9 10 11 12 13 14 15 16 17
```

```
x[c(2,2,2)]
```

```
[1] 7 7 7
```

# Vectors

## Indexing with vector

- For position indexing, use a **negative** value to remove an element

```
x <- c(6:17)
```

```
x
```

```
[1]  6  7  8  9 10 11 12 13 14 15 16 17
```

```
x[-5]
```

```
[1]  6  7  8  9 11 12 13 14 15 16 17
```

- **Positive and negative indices cannot be combined**

# Vectors

---

## Indexing with vector

- Use logical index of the same length to select elements where the value is **TRUE**

```
x <- c(6:10)
y <- c(TRUE, FALSE, FALSE, FALSE, FALSE)
x[y]
```

```
[1] 6
```

# Vectors

## Indexing with vector

- Use logical indexing in combination with conditions

```
x <- c(6:10)
x[x > 7]
```

```
[1] 8 9 10
```

```
x[x > 7 & x > 9]
```

```
[1] 10
```

```
x[x > 7 | x > 9]
```

```
[1] 8 9 10
```

# Vectors

---

## Indexing with vector

- For name/character indexing, use the name of the element

```
x <- c(foo=5, bar=4, one=7, two=12, three=2)
x[c('foo', 'one')]
```

```
foo one
  5   7
```

- Use the function `names` to obtain the names

# Matrices

---

## Indexing with `matrix`

- ▶ Indexing matrices is similar to indexing vectors but with double index
  - ▶ The first position denotes the rows `["index", ]`
  - ▶ The first position denotes the columns `[, "index"]`



# Matrices

## Indexing with `matrix`

- ▶ Indexing matrices is similar to indexing vectors but with double index
  - ▶ The first position denotes the rows `["index", ]`
  - ▶ The first position denotes the columns `[, "index"]`

```
mat <- matrix(data = 1:4,  
              nrow = 2, ncol = 2)
```

```
mat
```

|      | [,1] | [,2] |
|------|------|------|
| [1,] | 1    | 3    |
| [2,] | 2    | 4    |

- ▶ Use position indexing as:

```
mat <- matrix(data = 1:4,  
              nrow = 2, ncol = 2)
```

```
mat[2, 2]
```

```
[1] 4
```

# Matrices

## Indexing with `matrix`

- Be cautious, it also works with a single index. In this case, it selects the particular element of the vector that will be included in the matrix

```
mat <- matrix(data = 1:4,  
              nrow = 2, ncol = 2)  
mat[2]
```

```
[1] 2
```

```
mat[[2]]
```

```
[1] 2
```

# Matrices

## Indexing with `matrix`

- ▶ When we leave a position blank all elements are selected

```
mat <- matrix(data = 1:4,  
              nrow = 2, ncol = 2)
```

```
mat
```

|      | [,1] | [,2] |
|------|------|------|
| [1,] | 1    | 3    |
| [2,] | 2    | 4    |

```
mat[2, ]
```

```
[1] 2 4
```

# Arrays

## Indexing with array

```
ar <- array(data = 1:4,  
            dim = c(1,2,2))
```

```
ar
```

```
, , 1
```

|      | [,1] | [,2] |
|------|------|------|
| [1,] | 1    | 2    |

```
, , 2
```

|      | [,1] | [,2] |
|------|------|------|
| [1,] | 3    | 4    |

```
ar[1, 1, ]
```

```
[1] 1 3
```

# Data Frames

---

## Indexing with `data.frame`

- Works with single and double index

```
DF <- data.frame(x = 1:3,  
                 y = c("male", "male", "female"))
```

DF

|   | x | y      |
|---|---|--------|
| 1 | 1 | male   |
| 2 | 2 | male   |
| 3 | 3 | female |

# Data Frames

## Indexing with `data.frame`

- Works with single and double index

```
DF <- data.frame(x = 1:3,  
                 y = c("male", "male", "female"))
```

DF

|   | x | y      |
|---|---|--------|
| 1 | 1 | male   |
| 2 | 2 | male   |
| 3 | 3 | female |

- Use position single indexing

```
DF[2]
```

|   | y      |
|---|--------|
| 1 | male   |
| 2 | male   |
| 3 | female |

```
DF[[2]]
```

```
[1] "male" "male" "female"
```

# Data Frames

## Indexing with `data.frame`

- ▶ When using double index, indexing works like a matrix

```
DF <- data.frame(x = 1:3,  
                 y = c("male", "male", "female"))  
DF
```

|   | x | y      |
|---|---|--------|
| 1 | 1 | male   |
| 2 | 2 | male   |
| 3 | 3 | female |

- ▶ Use position indexing

```
DF[2, ]
```

|   | x | y    |
|---|---|------|
| 2 | 2 | male |

- ▶ Use logical indexing

```
DF[DF$x < 2, ]
```

|   | x | y    |
|---|---|------|
| 1 | 1 | male |

# Data Frames

---

## Indexing with `data.frame`

- ▶ `$` provides a convenient notation to extract an element by name

```
head(pbc$time)
```

```
[1] 400 4500 1012 1925 1504 2503
```

```
head(pbc[, "time"])
```

```
[1] 400 4500 1012 1925 1504 2503
```



# Data Frames

---

## Indexing with `data.frame`

- Combine logical and position indexing in data frame

```
head(pbc[pbc$sex == "m", 1:7])
```

|    | id | time | status | trt | age      | sex | ascites |
|----|----|------|--------|-----|----------|-----|---------|
| 3  | 3  | 1012 | 2      | 1   | 70.07255 | m   | 0       |
| 14 | 14 | 1217 | 2      | 2   | 56.22177 | m   | 1       |
| 21 | 21 | 3445 | 0      | 2   | 64.18891 | m   | 0       |
| 24 | 24 | 4079 | 2      | 1   | 44.52019 | m   | 0       |
| 48 | 48 | 4427 | 0      | 2   | 49.13621 | m   | 0       |
| 52 | 52 | 2386 | 2      | 1   | 50.54073 | m   | 0       |

# Data Frames

---

## Indexing with `data.frame`

- Combine logical and position indexing in data frame

```
head(pbc[pbc$age > 30 | pbc$sex == "f", 1:7])
```

|   | id | time | status | trt | age      | sex | ascites |
|---|----|------|--------|-----|----------|-----|---------|
| 1 | 1  | 400  | 2      | 1   | 58.76523 | f   | 1       |
| 2 | 2  | 4500 | 0      | 1   | 56.44627 | f   | 0       |
| 3 | 3  | 1012 | 2      | 1   | 70.07255 | m   | 0       |
| 4 | 4  | 1925 | 2      | 1   | 54.74059 | f   | 0       |
| 5 | 5  | 1504 | 1      | 2   | 38.10541 | f   | 0       |
| 6 | 6  | 2503 | 2      | 2   | 66.25873 | f   | 0       |

# Data Frames

---

## Indexing with `data.frame`

- Combine logical and position indexing in data frame

```
head(pbc[pbc$age > 30 & pbc$sex == "f", 1:7])
```

|   | id | time | status | trt | age      | sex | ascites |
|---|----|------|--------|-----|----------|-----|---------|
| 1 | 1  | 400  | 2      | 1   | 58.76523 | f   | 1       |
| 2 | 2  | 4500 | 0      | 1   | 56.44627 | f   | 0       |
| 4 | 4  | 1925 | 2      | 1   | 54.74059 | f   | 0       |
| 5 | 5  | 1504 | 1      | 2   | 38.10541 | f   | 0       |
| 6 | 6  | 2503 | 2      | 2   | 66.25873 | f   | 0       |
| 7 | 7  | 1832 | 0      | 2   | 55.53457 | f   | 0       |

# Lists

---

## Indexing with `list`

- ▶ Lists can be subsetted in the same way as vectors using single brackets - Note that the output is a list
- ▶ Use position indexing

```
mylist <- list(y = c(14, 45), z = c("m", "f", "f"))  
mylist[2]
```

```
$z  
[1] "m" "f" "f"
```

# Lists

---

## Indexing with `list`

- ▶ Double square brackets can be also used - Note that the output is a vector
- ▶ Use position indexing

```
mylist <- list(y = c(14, 45), z = c("m", "f", "f"))  
mylist[[2]]
```

```
[1] "m" "f" "f"
```

# Lists

---

## Indexing with `list`

- ▶ `$` provides a convenient notation to extract an element by name -  
Note that the output is a vector

```
mylist <- list(y = c(14, 45), z = c("m", "f", "f"))  
mylist
```

```
$y  
[1] 14 45
```

```
$z  
[1] "m" "f" "f"
```

```
mylist$y
```

```
[1] 14 45
```

# Summary

---

## Vectors

- ▶ `[]`
- ▶ `[""]` - for categorical variables

## Matrices

- ▶ `[,]`
- ▶ `[[]], []`

## Arrays

- ▶ `[ , , ]`

## Data frames

- ▶ `[,]`
- ▶ `[[]], []`
- ▶ `$`

## Lists


- ▶ `[]`
- ▶ `[[]]`
- ▶ `$`

## Practice

---

- ▶ Use the following webpage to further investigate indexing and subsetting  
<https://emcbiostatistics.shinyapps.io/indexing/>

### Demos

- ▶ Shiny app indexing subsetting 

In order to run the app you will need to install the packages:

- ▶ `survival`
- ▶ `shiny`



# Practice

---

## Demos

► Indexing/Subsetting **R** **html**

## Practicals

► Indexing/Subsetting **html**

# **Data Transformation Exploration Visualization**

## In this Section

---

- ▶ Data transformation
- ▶ Data exploration
- ▶ Data visualization
- ▶ A lot of practice

# Data Transformation

---

**You will never receive the perfect data set!**

- ▶ **Round** continuous variables
- ▶ Convert **numeric** variables to **factors**
- ▶ Compute **new variables**
  - ▶ transform variables
- ▶ **Sort** the data set
- ▶ Data sets of **wide**  $\iff$  **long** format

# Data Transformation

## ► Round continuous variables

```
pbpc[1:3, c("time", "age", "bili", "chol")]
```

|   | time | age      | bili | chol |
|---|------|----------|------|------|
| 1 | 400  | 58.76523 | 14.5 | 261  |
| 2 | 4500 | 56.44627 | 1.1  | 302  |
| 3 | 1012 | 70.07255 | 1.4  | 176  |

```
round(pbpc[1:3, c("time", "age", "bili", "chol")], digits = 2)
```

|   | time | age   | bili | chol |
|---|------|-------|------|------|
| 1 | 400  | 58.77 | 14.5 | 261  |
| 2 | 4500 | 56.45 | 1.1  | 302  |
| 3 | 1012 | 70.07 | 1.4  | 176  |

# Data Transformation

- Convert **numeric** variables to **factors**

```
DF <- pbc[,c("id", "time", "status", "trt", "age",  
            "sex", "bili", "chol")]  
head(DF)
```

|   | id | time | status | trt | age      | sex | bili | chol |
|---|----|------|--------|-----|----------|-----|------|------|
| 1 | 1  | 400  | 2      | 1   | 58.76523 | f   | 14.5 | 261  |
| 2 | 2  | 4500 | 0      | 1   | 56.44627 | f   | 1.1  | 302  |
| 3 | 3  | 1012 | 2      | 1   | 70.07255 | m   | 1.4  | 176  |
| 4 | 4  | 1925 | 2      | 1   | 54.74059 | f   | 1.8  | 244  |
| 5 | 5  | 1504 | 1      | 2   | 38.10541 | f   | 3.4  | 279  |
| 6 | 6  | 2503 | 2      | 2   | 66.25873 | f   | 0.8  | 248  |

# Data Transformation

- Convert **numeric** variables to **factors**

```
DF <- pbc[,c("id", "time", "status", "trt", "age",  
            "sex", "bili", "chol")]  
DF$trt <- factor(x = DF$trt, levels = c(1, 2),  
                labels = c("D-penicillmain", "placebo"))  
head(DF)
```

|   | id | time | status | trt            | age      | sex | bili | chol |
|---|----|------|--------|----------------|----------|-----|------|------|
| 1 | 1  | 400  | 2      | D-penicillmain | 58.76523 | f   | 14.5 | 261  |
| 2 | 2  | 4500 | 0      | D-penicillmain | 56.44627 | f   | 1.1  | 302  |
| 3 | 3  | 1012 | 2      | D-penicillmain | 70.07255 | m   | 1.4  | 176  |
| 4 | 4  | 1925 | 2      | D-penicillmain | 54.74059 | f   | 1.8  | 244  |
| 5 | 5  | 1504 | 1      | placebo        | 38.10541 | f   | 3.4  | 279  |
| 6 | 6  | 2503 | 2      | placebo        | 66.25873 | f   | 0.8  | 248  |

# Data Transformation

- Compute **new variables**
  - transform variables

```
DF <- pbc[,c("id", "time", "status", "trt", "age",  
            "sex", "bili", "chol")]  
head(DF)
```

|   | id | time | status | trt | age      | sex | bili | chol |
|---|----|------|--------|-----|----------|-----|------|------|
| 1 | 1  | 400  | 2      | 1   | 58.76523 | f   | 14.5 | 261  |
| 2 | 2  | 4500 | 0      | 1   | 56.44627 | f   | 1.1  | 302  |
| 3 | 3  | 1012 | 2      | 1   | 70.07255 | m   | 1.4  | 176  |
| 4 | 4  | 1925 | 2      | 1   | 54.74059 | f   | 1.8  | 244  |
| 5 | 5  | 1504 | 1      | 2   | 38.10541 | f   | 3.4  | 279  |
| 6 | 6  | 2503 | 2      | 2   | 66.25873 | f   | 0.8  | 248  |



# Data Transformation

- Compute **new variables**
  - transform variables

```
DF <- pbc[,c("id", "time", "status", "trt", "age",  
            "sex", "bili", "chol")]  
DF$time <- DF$time/30  
DF$time_years <- DF$time/12  
head(DF)
```

|   | id | time      | status | trt | age      | sex | bili | chol | time_years |
|---|----|-----------|--------|-----|----------|-----|------|------|------------|
| 1 | 1  | 13.33333  | 2      | 1   | 58.76523 | f   | 14.5 | 261  | 1.111111   |
| 2 | 2  | 150.00000 | 0      | 1   | 56.44627 | f   | 1.1  | 302  | 12.500000  |
| 3 | 3  | 33.73333  | 2      | 1   | 70.07255 | m   | 1.4  | 176  | 2.811111   |
| 4 | 4  | 64.16667  | 2      | 1   | 54.74059 | f   | 1.8  | 244  | 5.347222   |
| 5 | 5  | 50.13333  | 1      | 2   | 38.10541 | f   | 3.4  | 279  | 4.177778   |
| 6 | 6  | 83.43333  | 2      | 2   | 66.25873 | f   | 0.8  | 248  | 6.952778   |

# Data Transformation

---

- ▶ **Sort** the data set in either ascending or descending order
  - ▶ The variable by which we sort can be a numeric, string or factor

```
head(sort(pbc$bili))
```

```
[1] 0.3 0.3 0.3 0.4 0.4 0.4
```

# Data Transformation

- ▶ **Sort** the data set in either ascending or descending order
  - ▶ The variable by which we sort can be a numeric, string or factor

```
head(pbc[order(pbc$bili), ])
```

|     | id     | time | status   | trt     | age      | sex | ascites | hepato | spiders | edema | bili | chol | albumin | copper |
|-----|--------|------|----------|---------|----------|-----|---------|--------|---------|-------|------|------|---------|--------|
| 8   | 8      | 2466 | 2        | 2       | 53.05681 | f   | 0       | 0      | 0       | 0     | 0.3  | 280  | 4.00    | 52     |
| 36  | 36     | 3611 | 0        | 2       | 56.41068 | f   | 0       | 0      | 0       | 0     | 0.3  | 172  | 3.39    | 18     |
| 163 | 163    | 2055 | 2        | 1       | 53.49760 | f   | 0       | 0      | 0       | 0     | 0.3  | 233  | 4.08    | 20     |
| 84  | 84     | 4032 | 0        | 2       | 55.83025 | f   | 0       | 0      | 0       | 0     | 0.4  | 263  | 3.76    | 29     |
| 108 | 108    | 2583 | 2        | 1       | 50.35729 | f   | 0       | 0      | 0       | 0     | 0.4  | 127  | 3.50    | 14     |
| 135 | 135    | 3150 | 0        | 1       | 42.96783 | f   | 0       | 0      | 0       | 0     | 0.4  | 263  | 3.57    | 123    |
|     | ast    | trig | platelet | protime | stage    |     |         |        |         |       |      |      |         |        |
| 8   | 28.38  | 189  | 373      | 11.0    | 3        |     |         |        |         |       |      |      |         |        |
| 36  | 71.30  | 96   | 311      | 10.6    | 2        |     |         |        |         |       |      |      |         |        |
| 163 | 66.65  | 68   | 358      | 9.9     | 3        |     |         |        |         |       |      |      |         |        |
| 84  | 137.95 | 74   | 181      | 11.2    | 3        |     |         |        |         |       |      |      |         |        |
| 108 | 49.60  | 84   | 334      | 10.3    | 2        |     |         |        |         |       |      |      |         |        |
| 135 | 74.40  | 121  | 445      | 11.0    | 2        |     |         |        |         |       |      |      |         |        |

# Data Transformation

- ▶ **Sort** the data set in either ascending or descending order
  - ▶ The variable by which we sort can be a numeric, string or factor

```
head(pbc[order(pbc$bili, pbc$age), ])
```

|     | id  | time | status | trt | age      | sex | ascites | hepato | spiders | edema | bili | chol | albumin | copper |
|-----|-----|------|--------|-----|----------|-----|---------|--------|---------|-------|------|------|---------|--------|
| 8   | 8   | 2466 | 2      | 2   | 53.05681 | f   | 0       | 0      | 0       | 0.0   | 0.3  | 280  | 4.00    | 52     |
| 163 | 163 | 2055 | 2      | 1   | 53.49760 | f   | 0       | 0      | 0       | 0.0   | 0.3  | 233  | 4.08    | 20     |
| 36  | 36  | 3611 | 0      | 2   | 56.41068 | f   | 0       | 0      | 0       | 0.0   | 0.3  | 172  | 3.39    | 18     |
| 135 | 135 | 3150 | 0      | 1   | 42.96783 | f   | 0       | 0      | 0       | 0.0   | 0.4  | 263  | 3.57    | 123    |
| 320 | 320 | 2403 | 0      | NA  | 44.00000 | f   | NA      | NA     | NA      | 0.5   | 0.4  | NA   | 3.81    | NA     |
| 168 | 168 | 2713 | 0      | 2   | 47.75359 | f   | 0       | 1      | 0       | 0.0   | 0.4  | 257  | 3.80    | 44     |

|     | ast   | trig | platelet | protime | stage |
|-----|-------|------|----------|---------|-------|
| 8   | 28.38 | 189  | 373      | 11.0    | 3     |
| 163 | 66.65 | 68   | 358      | 9.9     | 3     |
| 36  | 71.30 | 96   | 311      | 10.6    | 2     |
| 135 | 74.40 | 121  | 445      | 11.0    | 2     |
| 320 | NA    | NA   | 226      | 10.5    | 3     |
| 168 | 97.65 | 110  | NA       | 9.2     | 2     |

# Data Transformation

- Data sets of **wide**  $\iff$  long format

```
head(pbc[,c("id", "time", "status", "trt", "age",  
            "sex", "bili", "chol")])
```

|   | id | time | status | trt | age      | sex | bili | chol |
|---|----|------|--------|-----|----------|-----|------|------|
| 1 | 1  | 400  | 2      | 1   | 58.76523 | f   | 14.5 | 261  |
| 2 | 2  | 4500 | 0      | 1   | 56.44627 | f   | 1.1  | 302  |
| 3 | 3  | 1012 | 2      | 1   | 70.07255 | m   | 1.4  | 176  |
| 4 | 4  | 1925 | 2      | 1   | 54.74059 | f   | 1.8  | 244  |
| 5 | 5  | 1504 | 1      | 2   | 38.10541 | f   | 3.4  | 279  |
| 6 | 6  | 2503 | 2      | 2   | 66.25873 | f   | 0.8  | 248  |

# Data Transformation

- Data sets of wide  $\iff$  **long** format

```
head(pbcseq[, c("id", "futime", "status", "trt", "age", "day",  
               "sex", "bili", "chol")])
```

|   | id | futime | status | trt | age      | day | sex | bili | chol |
|---|----|--------|--------|-----|----------|-----|-----|------|------|
| 1 | 1  | 400    | 2      | 1   | 58.76523 | 0   | f   | 14.5 | 261  |
| 2 | 1  | 400    | 2      | 1   | 58.76523 | 192 | f   | 21.3 | NA   |
| 3 | 2  | 5169   | 0      | 1   | 56.44627 | 0   | f   | 1.1  | 302  |
| 4 | 2  | 5169   | 0      | 1   | 56.44627 | 182 | f   | 0.8  | NA   |
| 5 | 2  | 5169   | 0      | 1   | 56.44627 | 365 | f   | 1.0  | NA   |
| 6 | 2  | 5169   | 0      | 1   | 56.44627 | 768 | f   | 1.9  | NA   |

# Data Transformation

---

- ▶ Data sets of **wide**  $\iff$  **long** format

?reshape

# Data Exploration

---

- ▶ Common questions for the pbc data set
  - ▶ What is the mean and standard deviation for age?
  - ▶ What is the mean and standard deviation for time?
  - ▶ What is the median and interquartile range for age?
  - ▶ What is the percentage of placebo patients?
  - ▶ What is the percentage of females?
  - ▶ What is the mean and standard deviation for age in males?
  - ▶ What is the mean and standard deviation for baseline serum bilirubin?
  - ▶ What is the percentage of missings in serum bilirubin?

**All these questions can be answered using R!**



# Data Exploration

---

- ▶ Hints

- ▶ Check functions: **mean(...)**, **sd(...)**, **percent(...)**, **median(...)**, **IQR(...)**, **table(...)**

# Data Exploration

---

## ► Hints

- Check functions: **mean(...)**, **sd(...)**, **percent(...)**, **median(...)**, **IQR(...)**, **table(...)**

What is the mean value for age?

```
mean(pbc$age)
```

```
[1] 50.74155
```

# Data Visualization

---

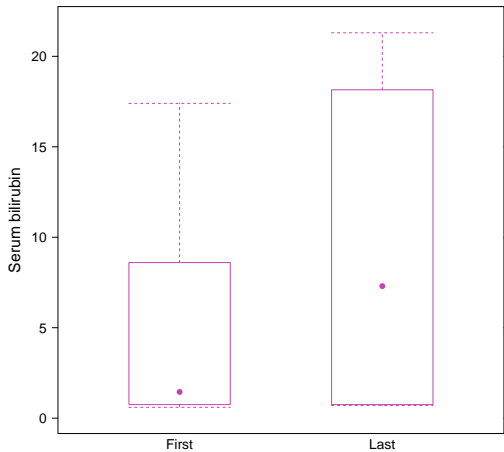
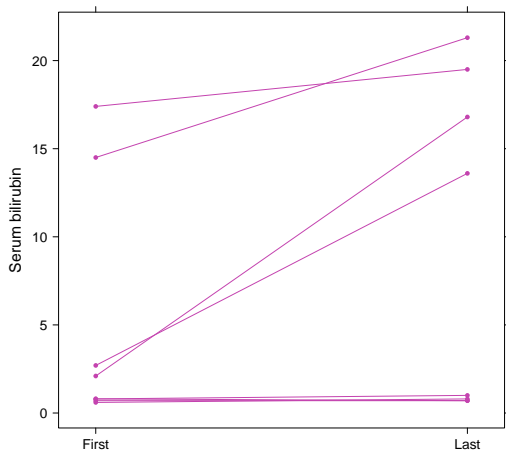
- ▶ It is important to investigate each variable in our data set using plots
  - ▶ Descriptive statistics for continuous and categorical variables
  - ▶ Distributions of variables
  - ▶ Distributions of variables per group
  - ▶ Extreme values
  - ▶ Linear/nonlinear evolutions

## Take care!



# Data Visualization

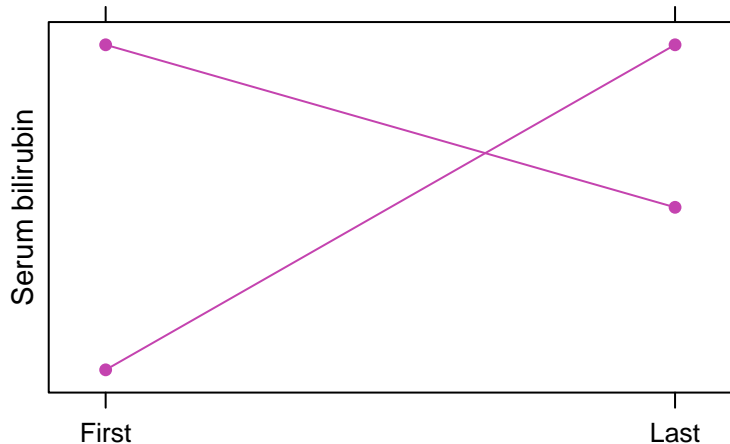
Take care!



# Data Visualization

---

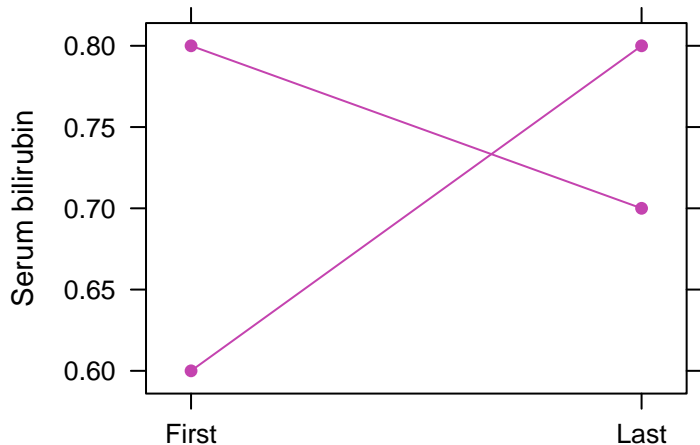
Take care!



# Data Visualization

---

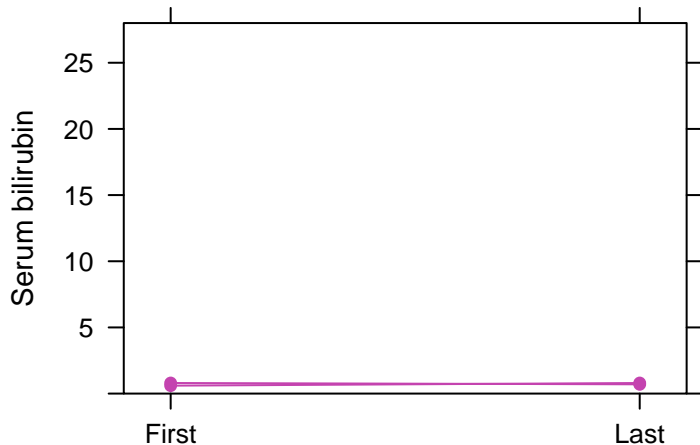
Take care!



# Data Visualization

---

Take care!





# Data Visualization

---

- ▶ R has very powerful graphics capabilities
- ▶ Some good references are
  - ▶ Murrell, P. (2005) R Graphics. Boca Raton: Chapman & Hall/CRC.
  - ▶ Sarkar, D. (2008) Lattice Multivariate Data Visualization with R. New York: Springer-Verlag.

# Data Visualization

---

- ▶ Traditional graphics system
  - ▶ package **graphics**
- ▶ Trellis graphics system
  - ▶ package **lattice** (which is based on package grid)
- ▶ Grammar of Graphics implementation (i.e., Wilkinson, L. (1999) The Grammar of Graphics. New York: Springer-Verlag)
  - ▶ packages **ggplot** & **ggplot2**

# Data Visualization

---

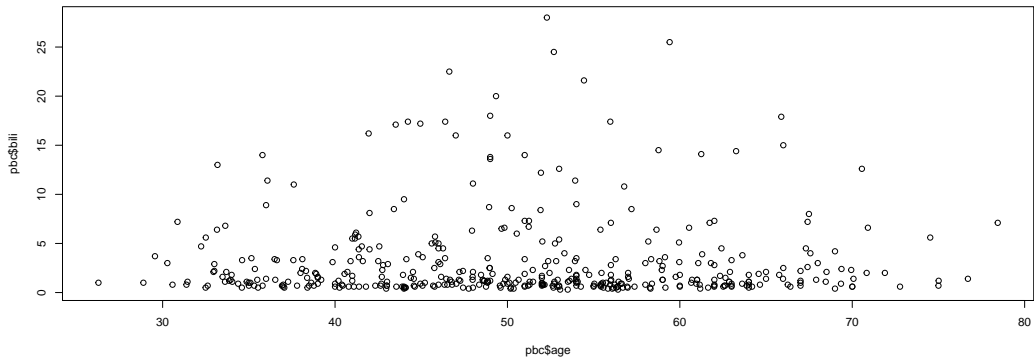
Important plotting basic functions

- ▶ **plot()**: scatter plot (and others)
- ▶ **barplot()**: bar plots
- ▶ **boxplot()**: box-and-whisker plots
- ▶ **hist()**: histograms
- ▶ **dotchart()**: dot plots
- ▶ **pie()**: pie charts
- ▶ **qqnorm()**, **qqline()**, **qqplot()**: distribution plots
- ▶ **pairs()**: for multivariate data

# Data Visualization

## Continuous variables

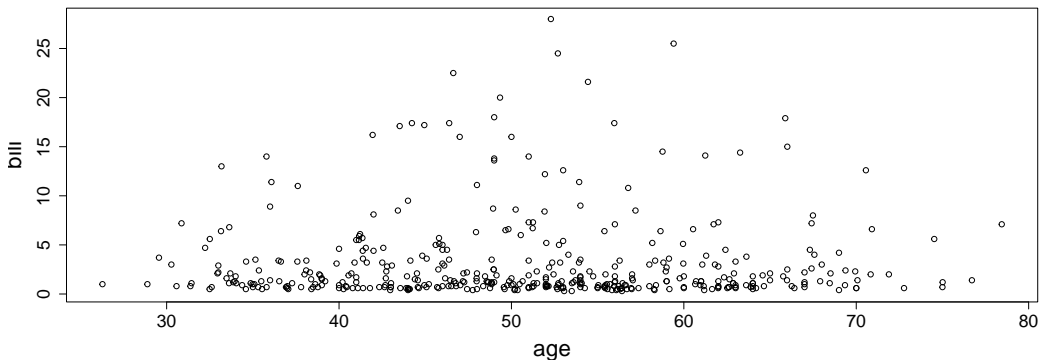
```
plot(x = pbc$age, y = pbc$bili)
```



# Data Visualization

## Continuous variables

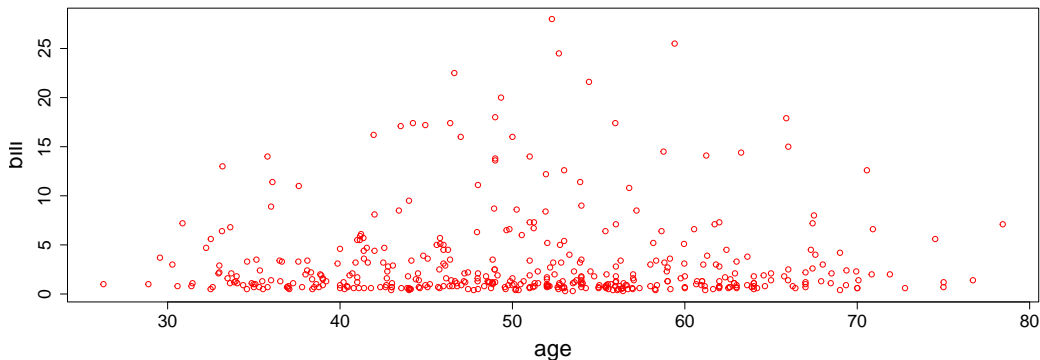
```
plot(x = pbc$age, y = pbc$bili, xlab = "age", ylab = "bili",  
     cex.lab = 1.9, cex.axis = 1.5)
```



# Data Visualization

## Continuous variables

```
plot(x = pbc$age, y = pbc$bili, xlab = "age", ylab = "bili",  
     cex.lab = 1.9, cex.axis = 1.5, col = "red")
```



# Data Visualization

---

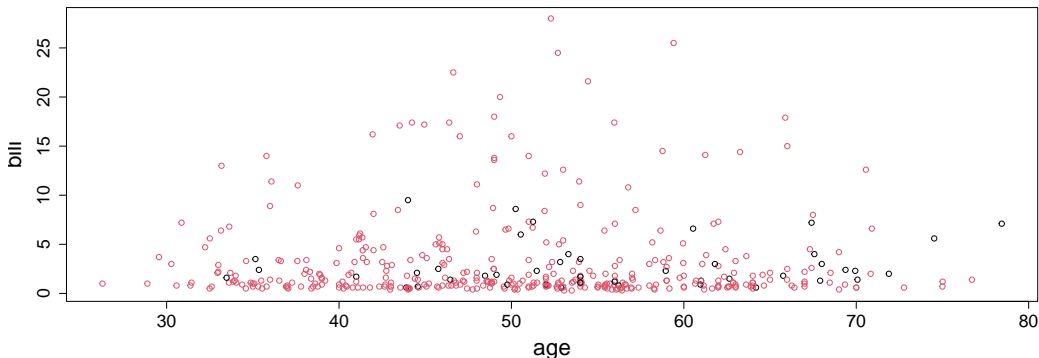
- ▶ For more options check

```
?plot
```

# Data Visualization

Continuous variables per group

```
plot(x = pbc$age, y = pbc$bili, xlab = "age", ylab = "bili",  
     cex.lab = 1.9, cex.axis = 1.5, col = pbc$sex)
```

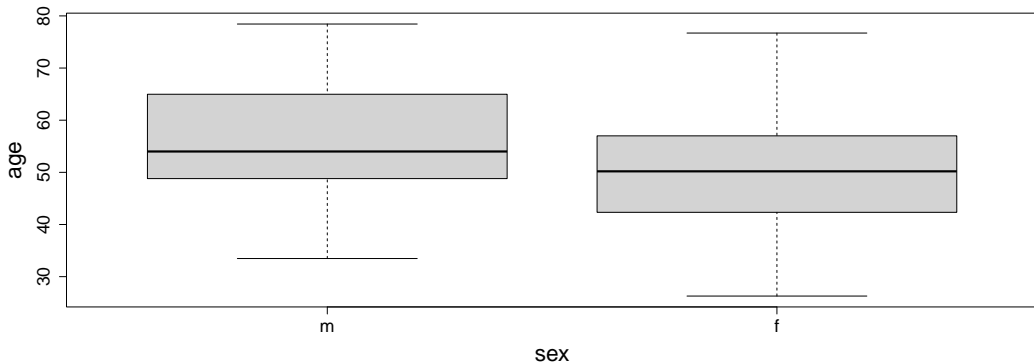




# Data Visualization

## Continuous variables per group

```
boxplot(formula = pbc$age ~ pbc$sex, xlab = "sex", ylab = "age",  
        cex.lab = 1.9, cex.axis = 1.5)
```



# Data Visualization

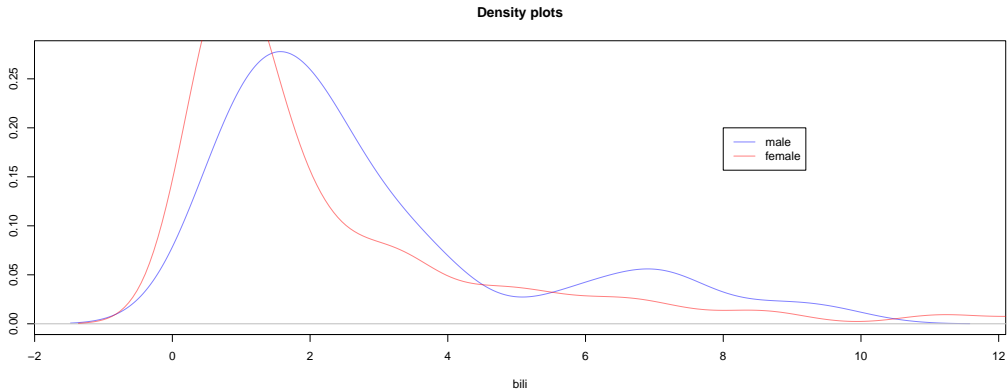
---

Continuous variables per group

```
pbm_male_bili <- pbm$bili[pbm$sex == "m"]
pbm_female_bili <- pbm$bili[pbm$sex == "f"]
plot(density(x = pbm_male_bili), col = rgb(0,0,1,0.5),
     main = "Density plots", xlab = "bili", ylab = "")
lines(density(x = pbm_female_bili), col = rgb(1,0,0,0.5))
legend(x = 8, y = 0.2, legend = c("male", "female"),
     col = c(rgb(0,0,1,0.5), rgb(1,0,0,0.5)), lty = 1)
```

# Data Visualization

## Continuous variables per group



# Data Visualization

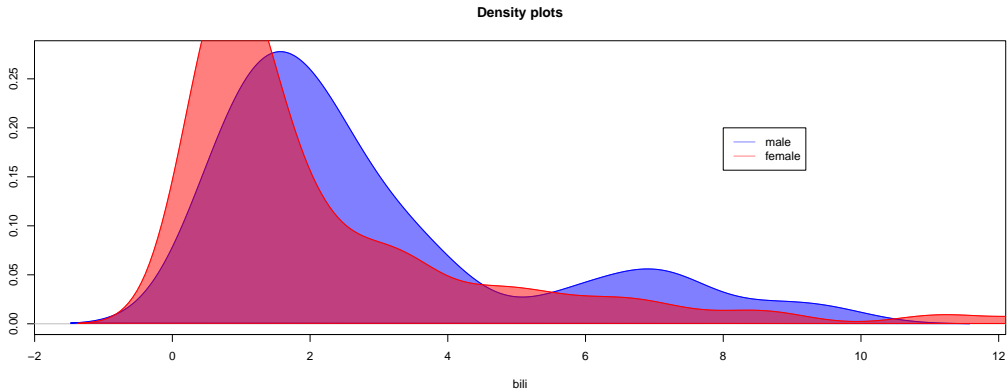
---

Continuous variables per group

```
pbm_male_bili <- pbm$bili[pbm$sex == "m"]
pbm_female_bili <- pbm$bili[pbm$sex == "f"]
plot(density(x = pbm_male_bili), col = rgb(0,0,1,0.5),
     main = "Density plots", xlab = "bili", ylab = "")
polygon(density(x = pbm_male_bili), col = rgb(0,0,1,0.5),
        border = "blue")
lines(density(x = pbm_female_bili), col = rgb(1,0,0,0.5))
polygon(density(x = pbm_female_bili), col = rgb(1,0,0,0.5),
        border = "red")
legend(x = 8, y = 0.2, legend = c("male", "female"),
      col = c(rgb(0,0,1,0.5), rgb(1,0,0,0.5)), lty = 1)
```

# Data Visualization

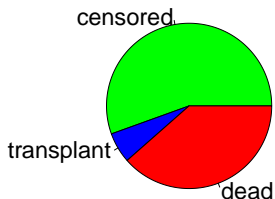
## Continuous variables per group



# Data Visualization

## Categorical variables

```
pbcs$status <- factor(x = pbcs$status, levels = c(0, 1, 2),  
                      labels = c("censored", "transplant", "dead"))  
pie(table(pbc$status), col = c("green", "blue", "red"), cex = 2)
```



# Summary

---

## Transformation

- ▶ `round()`
- ▶ `factor()`
- ▶ `order()`
- ▶ `reshape()`

## Exploration

- ▶ `mean(), sd()`
- ▶ `median(), IQR()`
- ▶ `table()`

## Visualization

- ▶ `plot(), legend()`
- ▶ `hist()`
- ▶ `barchart()`
- ▶ `boxplot()`
- ▶ `xyplot(), ggplot()`
- ▶ `par()`

# Practice

---

## Demos

- ▶ Data Transformation [R](#) [html](#)
- ▶ Data Exploration [R](#) [html](#)
- ▶ Data Visualization [R](#) [html](#)

## Practicals

- ▶ Data Manipulation [html](#)