Introduction to R for Basic Statistics

Alessandra Meddis
Section of Biostatistics, University of Copenhagen

Day 2: 29 August, 2024

Outline/ILOs

So far

- Use interface Rstudio
- Data structures in `R` with specific attention to `data.frame`
- Differences among types of Variables
- Manipulate `data.frame` to create/delete/transform a variable
- Descriptive analysis in `R` (mean,sd,tables,...)

Outline/ILOs

So far

- Use interface Rstudio
- Data structures in `R` with specific attention to `data.frame`
- Differences among types of Variables
- Manipulate `data.frame` to create/delete/transform a variable
- Descriptive analysis in `R` (mean,sd,tables,...)

Today

- Reshaping Data in `R`
  - Wide and Long format
  - Merge two data sets
- Create basic plots in `R`

Reshaping Data in R

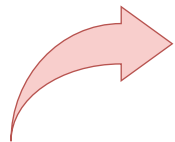
Wide and Long format

Same data can be represented in different ways.
In particular, when we have several measurements for the same individual at different times, places,...

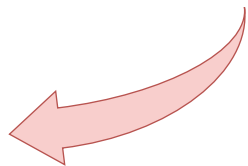
- Data in the wide format:
  - 1 row = 1 individual (level 1)
  - convenient when working with one or two time points
- Data in the long format:
  - 1 row = 1 measurement of 1 individual (level 0)
  - convenient when working over all time points

Wide and Long format

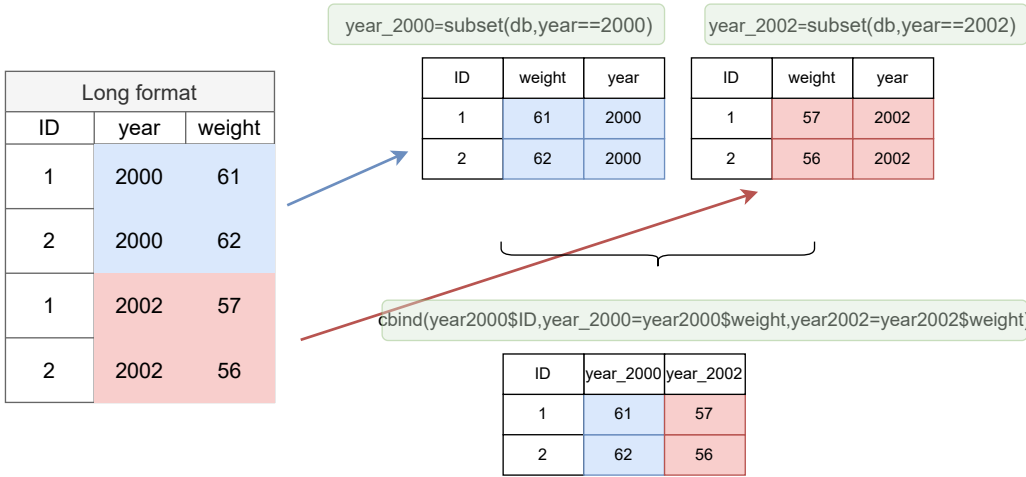
Long format		
ID	year	weight
1	2000	61
1	2002	57
2	2000	62
2	2002	56



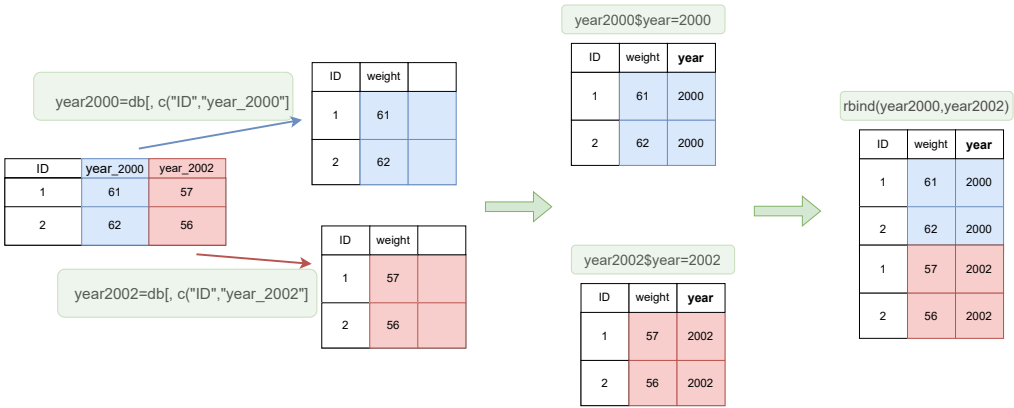
Wide format		
ID	year_2000	year_2002
1	61	57
2	62	56



Long to Wide


Long format
ID
year
weight
1
2000
61
2
2000
62
1
2002
57
2
2002
56
year_2000=subset(db,year==2000)
ID
weight
year
1
61
2000
2
62
2000
year_2002=subset(db,year==2002)
ID
weight
year
1
57
2002
2
56
2002
cbind(year2000$ID,year_2000=year2000$weight,year2002=year2002$weight)
ID
year_2000
year_2002
1
61
57
2
62
56

Wide to Long


year2000=db[, c("ID","year_2000"]
year2002=db[, c("ID","year_2002"]
ID
year_2000
year_2002
1
61
57
2
62
56
ID
weight
1
61
2
62
ID
weight
1
57
2
56
year2000$year=2000
ID
weight
year
1
61
2000
2
62
2000
year2002$year=2002
ID
weight
year
1
57
2002
2
56
2002
rbind(year2000,year2002)
ID
weight
year
1
61
2000
2
62
2000
1
57
2002
2
56
2002

Long to Wide by reshape

In R we can use the function reshape

```
> db_wide<-reshape(db1_long, direction="wide",
+                  idvar="ID",
+                  timevar="year")
> head(db_wide)
  ID weight.2000 weight.2002
1  1    57.48904    57.80775
2  2    56.39889    60.65766
3  3    59.60541    61.15472
4  4    54.21135    64.43392
5  5    60.58486    61.23538
6  6    59.54443    61.59315
```

Wide to Long by reshape

In R we can use the function reshape

```
db_long<-reshape(db, direction="long",
idvar="ID",
varying=c("year_2000","year_20002"),
v.names="weight",
timevar="year",
times=c("2000","2002"))
```

Merge two data sets

When information on patients are coming from different data sources, we have to *merge* several data sets together.

db1

ID	year	weight
1	2000	61
1	2002	57
2	2000	62
2	2002	56

+

db2

ID	Sex	Age	Diet
1	M	15	A
2	F	23	B

=

db1			db2		
ID	year	weight	sex	Age	Diet
1	2000	61	M	15	A
1	2002	57	M	15	A
2	2000	62	F	23	B
2	2002	56	F	23	B

Merge two data sets

In R we can use the function merge. It is important to identify the variable we want to merge *by* (the variable that connects the two data sets). In our example, it is the patient ID :

```
> head(db2_ex)
   ID sex    Age     Diet
1  1   M   20.39421    A
2  2   F   21.61736    B
3  3   F   17.90074    B
4  4   F   18.09905    A
5  5   M   22.74433    A
6  6   M   20.66363    A

> db_all<-merge(db1_long,db2_ex, by="ID")
```

Merge two data sets

```
> db_all<-merge(db1_long,db2_ex, by="ID")
  > head(db_all)
    ID year   weight sex      Age Diet
  1  1 2000 57.48904   M 20.39421    A
  2  1 2002 57.80775   M 20.39421    A
  3  2 2002 60.65766   F 21.61736    B
  4  2 2000 56.39889   F 21.61736    B
  5  3 2000 59.60541   F 17.90074    B
  6  3 2002 61.15472   F 17.90074    B
```

Merge two data sets

```
 # if the two data sets have different names
 # for the variable to merge by:
> names(db2_ex)
 [1] "id"   "sex"  "Age"  "Diet"

> db_all<-merge(db1_long,db2_ex, by.x="ID", by.y="id")
 > head(db_all)
   ID year   weight sex      Age Diet
1   1 2000 57.48904   M 20.39421    A
2   1 2002 57.80775   M 20.39421    A
3   2 2002 60.65766   F 21.61736    B
4   2 2000 56.39889   F 21.61736    B
5   3 2000 59.60541   F 17.90074    B
6   3 2002 61.15472   F 17.90074    B
```

Exercise II:

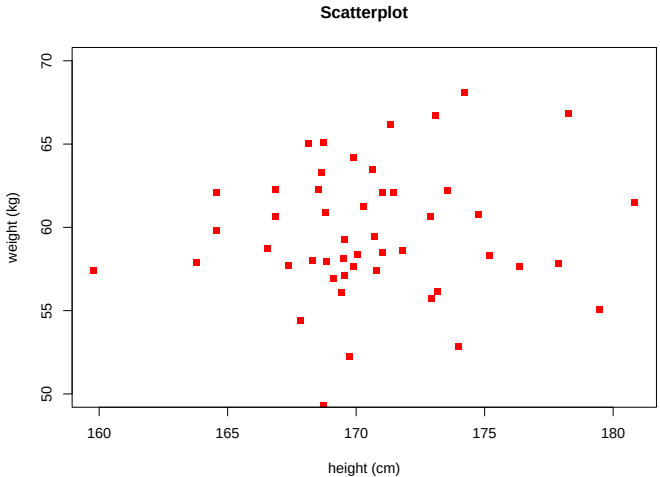
Exercise: A total of 20 individuals with cancer have been randomized to two groups of treatment **A** and **B** and a biomarker has been collected at baseline, 6 months and 12 months to see the effect on the progression of the disease. We have two data sets:

- dbex1: data for collection of biomarker at each visit
- dbex2: data for individual characteristics at baseline

1. Open the script *Exercise2.R* in the Exercise folder and run the code for creation of the data
2. Transform `dbex1` into the long format with variables:
   - biomarker for the values of the measurement
   - *visit* for the time of measurement with values: *0,6,12*
3. Merge the long version with `dbex2`

Basic Graphs with R

Scatterplot



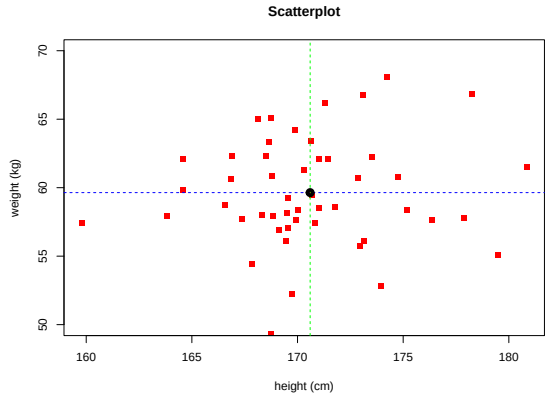
Scatterplot

```
plot(db1_ex$height,db1_ex$weight,
     ylab='weight', xlab='height', ylim=c(50,70),
     col='red', pch=15,
     main='Scatterplot')
```

- First two arguments are: x and y values (coordinates)
- `xlab,ylab`: name for the axes
- `xlim,ylim`: range of the axes to show
- `col`: colors of points
- `pch`: plot symbols (type of points)
- `main`: title of the plot (top)

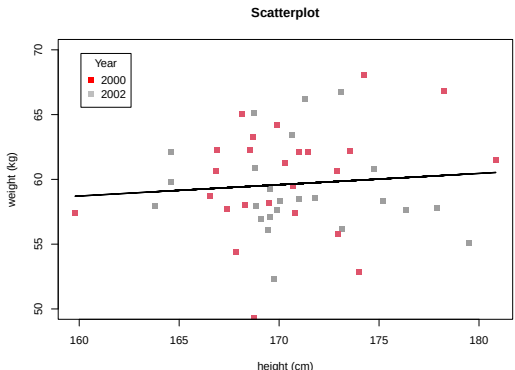
Add lines: `abline()`

```
abline(v=mean.height, col="green", lty=2)
abline(h=mean.weight, col="blue", lty=2)
points(mean.height,mean.weight, lwd=5)
```



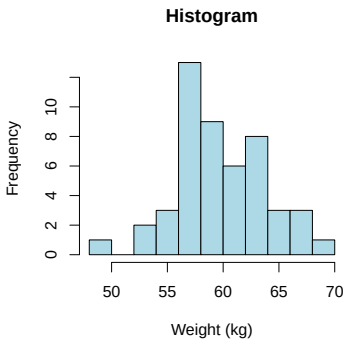
Add lines: `lines()`

```
plot(db1_ex$height,db1_ex$weight, ...,col=db1_ex$year)
lines(db1_ex$height,pred.w,lty=1, lwd=2)
legend( 'topleft', c('2000','2002'), pch=15, col=c('red','gray'),
     title='Year' )
```

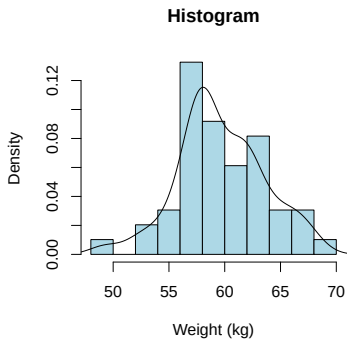


Histogram

```
hist( db1_ex$weight, col="lightblue",
xlab="Weight (kg)", main="Histogram")
```



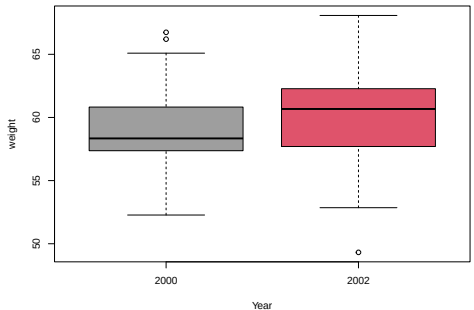

```
hist( ..., prob = TRUE)
lines(density(db1_ex$weight) )
```



Boxplot

A boxplot is used to illustrate key features of the distribution of a numerical variable for different groups.

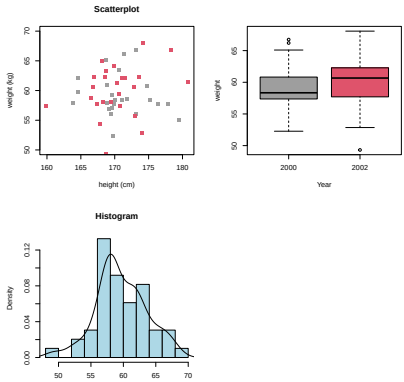
```
boxplot(weight~year, db1_ex,
xlab='Year',names=c('2000','2002'),col=c("gray","red"),
```



par()

The function `par()` is used to set several graphical parameters. This can be used to combine together several plots with the argument $= c(nrow, ncol)$

```
par(mfrow = c(2, 2))
# divide the space in 2 rows and 2 columns--> 4 plots max
```



General specification for plot

- `plot()`: Scatterplot or lines (check `type`)
- `points()`,`lines()`,`abline()`: to ADD points/lines to the plot
- For all graphics (plots, histograms,boxplots)
  - col: color(s) of what you are drawing
  - xlim,ylim: margins of the plot
  - xlab,ylab: labels of the axes
  - main: title of the plot
  - lty,pch: type of lines and points
  - lwd: line width

Exercise II: part 2

1. Create two data.frames:
   - one data.frame for group A
   - one data.frame for group B
2. use `tapply`/aggregate to calculate the mean of the biomarker by visit for each group
3. Save the results for both groups in the environment
4. Create a plot[1] for the mean in both groups with
   - points and lines for mean across visits
   - color red for group A and blue for group B
   - add the legend

[1] Careful, margins are automatically set from the plot function, you might want to define them with ylim=c()