Exploratory Analysis with R

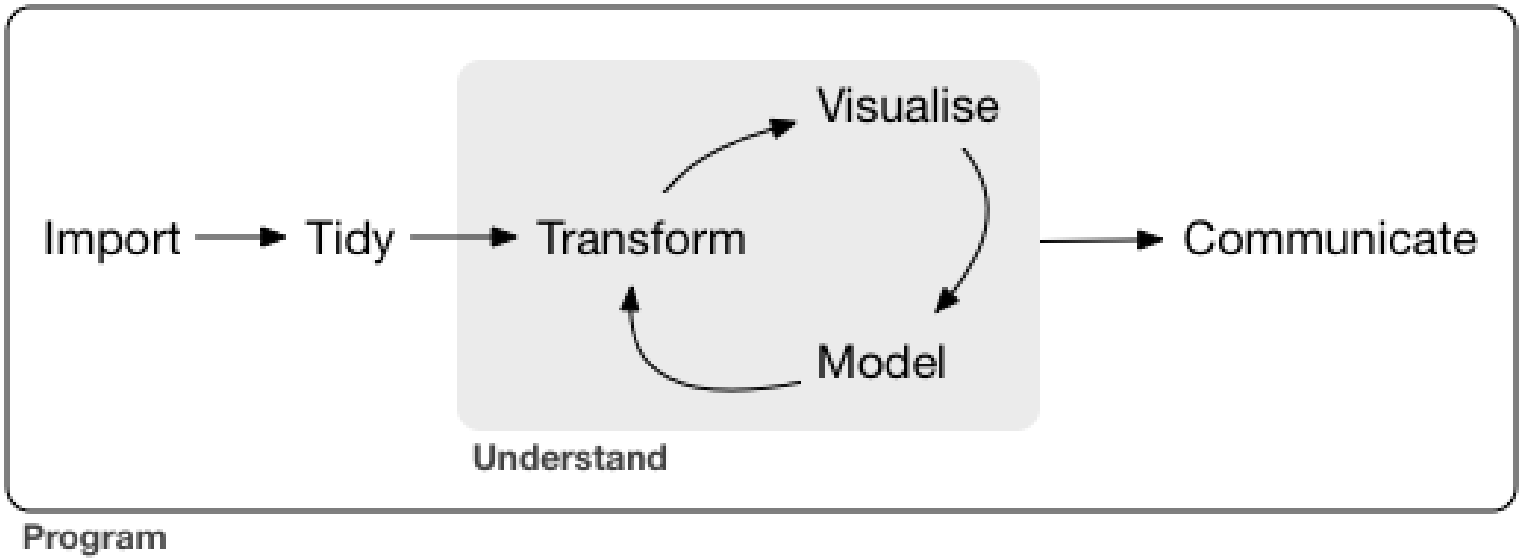
Alex Sanchez, Miriam Mota, Ricardo Gonzalo and Santiago Perez-Hoyos

Statistics and Bioinformatics Unit. Vall d'Hebron Institut de Recerca

Outline: Exploratory Analysis with R

- Getting and looking at datasets and data types
- Numerical summaries for exploring data
- Graphical summaries

The Data Science Approach in R



Section 1

Getting started

Getting started (I)

1. Load the dataset *diabetes*:

```
library(readxl)
diab <- read_excel("datasets/diabetes_mod.xls")
```

2. Check if we have loaded it correctly:

```
diab[1:4, 1:8]
```

```
## # A tibble: 4 x 8
##   numpacie mort  tempsviu  edat   bmi edatdiag tabac        sbp
##      <dbl> <chr>    <dbl> <dbl> <dbl>    <dbl> <chr>      <dbl>
## 1        1 Vivo      12.4    44  34.2       41 No fumador   132
## 2        2 Vivo      12.4    49  32.6       48 Fumador      130
## 3        3 Vivo       9.6    49  22         35 Fumador      108
## 4        4 Vivo       7.2    47  37.9       45 No fumador   128
```

Getting started (II): functions to check a dataframe:

- Content
  - `head()`: shows the first few rows
  - `tail()`: shows the last few rows
- Size
  - `dim()`: returns the number of rows and the number of columns
  - `nrow()`: returns the number of rows
  - `ncol()`: returns the number of columns
- Summary
  - `colnames()` or names(): returns the column names
  - `glimpse()`: returns a glimpse of your data: structure, class, length and content of each column

Getting started (III)

```
head(diab)
```

```
## # A tibble: 6 x 11
##   numpacie mort  tempsviu  edat   bmi edatdiag tabac      sbp   dbp ecg    chd
##      <dbl> <chr>    <dbl> <dbl> <dbl>    <dbl> <chr>    <dbl> <dbl> <chr>  <chr>
## 1        1 Vivo      12.4    44  34.2       41 No fuma~   132    96 Normal No
## 2        2 Vivo      12.4    49  32.6       48 Fumador    130    72 Normal No
## 3        3 Vivo       9.6    49  22         35 Fumador    108    58 Normal Si
## 4        4 Vivo       7.2    47  37.9       45 No fuma~   128    76 Front~ Si
## 5        5 Vivo      14.1    43  42.2       42 Fumador    142    80 Normal No
## 6        6 Vivo      14.1    47  33.1       44 No fuma~   156    94 Normal No
```

Getting started (IV)

```
dim(diab)
```

```
## [1] 149  11
```

```
nrow(diab)
```

```
## [1] 149
```

```
colnames(diab)
```

```
##  [1] "numpacie" "mort"     "tempsviu" "edat"     "bmi"      "edatdia
##  [7] "tabac"    "sbp"      "dbp"      "ecg"      "chd"
```

Getting started (IV)

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
glimpse(diab)
```

```
## Rows: 149
## Columns: 11
## $ numpacie <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18~
## $ mort     <chr> "Vivo", "Vivo", "Vivo", "Vivo", "Vivo", "Vivo", "Vivo", "Vivo~
## $ tempsviu <dbl> 12.4, 12.4, 9.6, 7.2, 14.1, 14.1, 12.4, 14.2, 12.4, 14.5, 12.~
## $ edat     <dbl> 44, 49, 49, 47, 43, 47, 50, 36, 50, 49, 50, 54, 42, 44, 40, 4~
## $ bmi      <dbl> 34.2, 32.6, 22.0, 37.9, 42.2, 33.1, 36.5, 38.5, 41.5, 34.1, 3~
## $ edatdiag <dbl> 41, 48, 35, 45, 42, 44, 48, NA, 47, 45, 48, 43, 36, 43, 26, 4~
## $ tabac    <chr> "No fumador", "Fumador", "Fumador", "No fumador", "Fumador", ~
## $ sbp      <dbl> 132, 130, 108, 128, 142, 156, 140, 144, 134, 102, 142, 128, 1~
## $ dbp      <dbl> 96, 72, 58, 76, 80, 94, 86, 88, 78, 68, 84, 74, 86, 58, 98, 6~
## $ ecg      <chr> "Normal", "Normal", "Normal", "Frontera", "Normal", "Normal",~
## $ chd      <chr> "No", "No", "Si", "Si", "No", "No", "Si", "No", "Si", "No", "~
```

Variables and data types

- Data managed in R . . .
  - is stored as *variables*
- Variables can be of distinct types
  - Numerical
    - numeric (13.7)
    - int (3)
  - Character
    - "R is cute"
  - Factors are used to represent *categorical data*
    - A,B,C,D
    - WT, Mut
  - Logical
    - TRUE/FALSE

Exercise I

- Load the osteoporosis dataset
- Proceed similarly as to what we have done above and obtain information on
  - How many variables and observations
  - How are them

More about factors

- Each data type is what it seems to be, but factors require more explanation.
- Factors are intended to describe categories such as “sex”, “blood group”, but also “risk” or “stage”.
- Factors are useful to describe groups without having to use numeric codes.
- Factors may be created while reading the file or later using the `factor` and `as.factor` commands.

Create factor while reading

- Factors may be created automatically from string data when ...
  - Files are read using the `read.delim`, `read.csv` or `read.table` functions of R base
  - Option 'stringAsFactors' is set to TRUE
- Reading files from Excel (`read_excel`) or using `tidyverse` functions such as `read_csv` does not allow to create factors automatically.

Read a file and create factors automatically

- Import the diabetes dataset from the `diabetes.csv` file using the Rstudio dialog.

```
diabetes <- read.csv("datasets/diabetes.csv", stringsAsFactors=TRUE)
glimpse(diabetes)
```

```
## Rows: 149
## Columns: 11
## $ numpacie <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
## $ mort     <fct> Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Viv
## $ tempsviu <dbl> 12.4, 12.4, 9.6, 7.2, 14.1, 14.1, 12.4, 14.2, 12.4,
## $ edat     <int> 44, 49, 49, 47, 43, 47, 50, 36, 50, 49, 50, 54, 42,
## $ bmi      <dbl> 34.2, 32.6, 22.0, 37.9, 42.2, 33.1, 36.5, 38.5, 41.
## $ edatdiag <int> 41, 48, 35, 45, 42, 44, 48, 33, 47, 45, 48, 43, 36,
## $ tabac    <fct> No fumador, Fumador, Fumador, No fumador, Fumador,
## $ sbp      <int> 132, 130, 108, 128, 142, 156, 140, 144, 134, 102, 1
## $ dbp      <int> 96, 72, 58, 76, 80, 94, 86, 88, 78, 68, 84, 74, 86,
## $ ecg      <fct> Normal, Normal, Normal, Frontera, Normal, Normal, F
## $ chd      <fct> No, No, Si, Si, No, No, Si, No, Si, No, No, No, No,
```

Read a file without creating factors automatically

- Re-read the file
  - from excel (`diabetes.xls`)
  - or without setting the "stringsAsFactors" to TRUE

```
diab <- read.csv("datasets/diabetes.csv", stringsAsFactors=FALSE)
glimpse(diab)
```

```
## Rows: 149
## Columns: 11
## $ numpacie <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
## $ mort     <chr> "Vivo", "Vivo", "Vivo", "Vivo", "Vivo", "Vivo", "Vi
## $ tempsviu <dbl> 12.4, 12.4, 9.6, 7.2, 14.1, 14.1, 12.4, 14.2, 12.4,
## $ edat     <int> 44, 49, 49, 47, 43, 47, 50, 36, 50, 49, 50, 54, 42,
## $ bmi      <dbl> 34.2, 32.6, 22.0, 37.9, 42.2, 33.1, 36.5, 38.5, 41.
## $ edatdiag <int> 41, 48, 35, 45, 42, 44, 48, 33, 47, 45, 48, 43, 36,
## $ tabac    <chr> "No fumador", "Fumador", "Fumador", "No fumador", "
## $ sbp      <int> 132, 130, 108, 128, 142, 156, 140, 144, 134, 102, 1
## $ dbp      <int> 96, 72, 58, 76, 80, 94, 86, 88, 78, 68, 84, 74, 86,
## $ ecg      <chr> "Normal", "Normal", "Normal", "Frontera", "Normal",
```

Creating factors directly

- Use `factor` or `as.factor`

```
diabetes <- read.csv("datasets/diabetes.csv", stringsAsFactors=FALSE)
class(diabetes$mort)
```

```
## [1] "character"
```

```
diabetes$mort <- as.factor(diabetes$mort)
class(diabetes$mort)
```

```
## [1] "factor"
```

```
levels(diabetes$mort)
```

```
## [1] "Muerto" "Vivo"
```

Warning! by default alphabetic order is used when creating factor levels.

```
vitalStatus <- factor(diabetes$mort, levels=c("Vivo", "Muerto"))
class(vitalStatus)
```

Change the levels of a factor

- When humans fill the database... many errors can happen :(
- An answer like "YES", could be entered like:

  "YES", "yes", "Yes", "Yeah"
- All this possible answers **would be differents levels for the same variable**
- This may be solved using `recode_factor`:

```
diab$mort <- recode_factor(diab$mort, "Muerto" = "muerto")
levels(diab$mort)
```

```
## [1] "muerto" "Vivo"
```

Changing *characters (chr)* to *factors (Factor)*

An alternative way to turn characters into factors is the `mutate_if` function:

```
library(dplyr)
diab <- diabetes %>% mutate_if(is.character, as.factor)
glimpse(diab)
```

```
## Rows: 149
## Columns: 11
## $ numpacie <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18~
## $ mort     <fct> Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, V~
## $ tempsviu <dbl> 12.4, 12.4, 9.6, 7.2, 14.1, 14.1, 12.4, 14.2, 12.4, 14.5, 12.~
## $ edat     <int> 44, 49, 49, 47, 43, 47, 50, 36, 50, 49, 50, 54, 42, 44, 40, 4~
## $ bmi      <dbl> 34.2, 32.6, 22.0, 37.9, 42.2, 33.1, 36.5, 38.5, 41.5, 34.1, 3~
## $ edatdiag <int> 41, 48, 35, 45, 42, 44, 48, 33, 47, 45, 48, 43, 36, 43, 26, 4~
## $ tabac    <fct> No fumador, Fumador, Fumador, No fumador, Fumador, No fumador~
## $ sbp      <int> 132, 130, 108, 128, 142, 156, 140, 144, 134, 102, 142, 128, 1~
## $ dbp      <int> 96, 72, 58, 76, 80, 94, 86, 88, 78, 68, 84, 74, 86, 58, 98, 6~
## $ ecg      <fct> Normal, Normal, Normal, Frontera, Normal, Normal, Frontera, N~
## $ chd      <fct> No, No, Si, Si, No, No, Si, No, Si, No, No, No, No, No, No, S~
```

Creating factors when categorizing numerical variables

- diab$edat is a numerical variable but we may want to categorize it that is creat a new factor variable whose categories are defined by nuemric intervals.

```
grupsEdat <- cut(diab$edat, breaks=c(0,30,50,75,Inf),
                 include.lowest=TRUE,
                 labels=c("<30","30-50","50-75",">75"))
library(tibble)
diab <- diab %>% add_column (grupsEdat, .after="edat")
glimpse(diab[,2:5])
```

```
## Rows: 149
## Columns: 4
## $ mort      <fct> Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vivo, Vi
## $ tempsviu  <dbl> 12.4, 12.4, 9.6, 7.2, 14.1, 14.1, 12.4, 14.2, 12.4
## $ edat      <int> 44, 49, 49, 47, 43, 47, 50, 36, 50, 49, 50, 54, 42
## $ grupsEdat <fct> 30-50, 30-50, 30-50, 30-50, 30-50, 30-50, 30-50, 3
```

Section 2

Descriptive Statistics: Numerical summaries

Numerical Summaries (I)

- There are many functions to provide numerical summaries

```
#Mean, median and rang
mean(diab$edat)
```

```
## [1] 52.16779
```

```
median(diab$edat)
```

```
## [1] 50
```

```
sd(diab$edat)
```

```
## [1] 11.77285
```

```
var(diab$edat)
```

```
## [1] 138.6
```

```
range(diab$edat)
```

```
## [1] 31 86
```

Numerical Summaries (II)

A general summary of all variables is provided by distinct functions

```
summary(diab[, 2:11])
```

```
##      mort        tempsviu          edat       grupsEdat       bmi
##  Muerto: 25   Min.   : 0.00   Min.   :31.00   <30  : 0   Min.   :18.20
##  Vivo  :124   1st Qu.: 7.30   1st Qu.:43.00   30-50:77   1st Qu.:26.60
##               Median :11.60   Median :50.00   50-75:66   Median :31.20
##               Mean   :10.52   Mean   :52.17   >75  : 6   Mean   :31.78
##               3rd Qu.:13.90   3rd Qu.:60.00              3rd Qu.:35.20
##               Max.   :16.90   Max.   :86.00              Max.   :59.70
##     edatdiag            tabac         sbp             dbp
##  Min.   :26.00   Ex fumador:41   Min.   : 98.0   Min.   : 58.00
##  1st Qu.:38.00   Fumador   :51   1st Qu.:124.0   1st Qu.: 74.00
##  Median :45.00   No fumador:57   Median :138.0   Median : 80.00
##  Mean   :45.99                   Mean   :139.1   Mean   : 90.04
##  3rd Qu.:53.00                   3rd Qu.:152.0   3rd Qu.: 88.00
##  Max.   :81.00                   Max.   :222.0   Max.   :862.00
##        ecg
##  Anormal : 11
##  Frontera: 27
##  Normal  :111
##
##
##
```

Numerical SUmmaries (III)

If categorical variables are not adequately represented summaries will be less informative.

Remember that we read created `diabetes` without turning variables into factors.

```
diabetes <- read.csv("datasets/diabetes.csv", stringsAsFactors=FALSE)
summary(diabetes)
```

```
##     numpacie          mort               tempsviu          edat
##  Min.   :  1.00   Length:149         Min.   : 0.00   Min.   :31.00
##  1st Qu.: 38.00   Class :character   1st Qu.: 7.30   1st Qu.:43.00
##  Median : 75.00   Mode  :character   Median :11.60   Median :50.00
##  Mean   : 75.01                      Mean   :10.52   Mean   :52.17
##  3rd Qu.:112.00                      3rd Qu.:13.90   3rd Qu.:60.00
##  Max.   :149.00                      Max.   :16.90   Max.   :86.00
##       bmi            edatdiag        tabac                sbp
##  Min.   :18.20   Min.   :26.00   Length:149         Min.   : 98.0
##  1st Qu.:26.60   1st Qu.:38.00   Class :character   1st Qu.:124.0
##  Median :31.20   Median :45.00   Mode  :character   Median :138.0
```

Improving the `summary` function

- There are many packages to do descriptive statistics.
- See Dabbling with data
- Give a try, for instance to the `skimr` or `summarytools` packages.

More complete descriptions (I)

```
library(summarytools)
```

```
## Registered S3 method overwritten by 'pryr':
##   method      from
##   print.bytes Rcpp
```

```
## For best results, restart R session and update pander using devtools
```

```
##
## Attaching package: 'summarytools'
```

```
## The following object is masked from 'package:tibble':
##
##     view
```

```
summarytools::dfSummary(diabetes)
```

```
## Data Frame Summary
## diabetes
## Dimensions: 149 x 11
## Duplicates: 1
##
```

Grouped summaries

If we want to group the descriptive summaries by other variables we can use group_by function:

```
diab %>%
  group_by(tabac, ecg) %>%
  summarize(mean(edat))
```

```
## `summarise()` has grouped output by 'tabac'. You can override using the `.groups` argument.

## # A tibble: 9 x 3
## # Groups:   tabac [3]
##   tabac       ecg      `mean(edat)`
##   <fct>       <fct>           <dbl>
## 1 Ex fumador Anormal          68.5
## 2 Ex fumador Frontera         59.8
## 3 Ex fumador Normal           51.1
## 4 Fumador    Anormal          58
## 5 Fumador    Frontera         44.8
## 6 Fumador    Normal           44.7
## 7 No fumador Anormal          66.5
## 8 No fumador Frontera         53.8
## 9 No fumador Normal           56.0
```

Handling missing data

- What happens if we have missing data in our dataset?
- The file `diabetes_mod.xls` contains some missings

```
diabetes_mod <- read_excel("datasets/diabetes_mod.xls")
diab <- diabetes_mod %>% mutate_if(is.character, as.factor)
mean(diab$sbp)
```

```
## [1] NA
```

NA indicates *missing data* in the variable

Let's look the `sbp` variable:

```
diab$sbp
```

```
##   [1] 132 130 108 128 142 156 140 144 134 102 142 128 156 102 146 120 142 144
##  [19]  NA 134 130 122 132 150 134 142 124 102 134 118 192 122 122 112 142 152
##  [37] 112 118 152 136 134 130 108 126 132 144 126 128  NA 128 142 132 148 170
##  [55] 140 138 112 140 138 130 178 158 168 146 128 132 154 154 122 144 178 162
##  [73] 142 120 124 174 142 160 122 162 132 116 152 144  98 138 138 184 158 176
##  [91] 118 172 182 144 142 154 122 222 150 142 128 122 162 172 132 112 138 128
## [109] 132 120 140 140 172 136 152 126 104 142 128 122 122 122 122 168 162  NA
## [127] 126 180 132 150 106 154 122 120 120 144 134 148 170 160 154 124 130 156
## [145] 162 132 120 160 146
```

How to work with *missing data*:

```
?mean
```

```
## Help on topic 'mean' was found in the following packages:
##
##   Package               Library
##   rapportools           /home/alex/R/x86_64-pc-linux-gnu-library/4.1
##   base                  /usr/lib/R/library
##
##
## Using the first match ...
```

```
mean(diab$sbp, na.rm = TRUE)
```

```
## [1] 139.2603
```

```
is.na(diab$sbp)
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE
```

How to work with *missing data*:

- Count missings

```
sum(is.na(diab$sbp))
```

```
## [1] 3
```

```
sum(is.na(diab$dbp))
```

```
## [1] 0
```

- Remove all rows with missing

```
diab_noNAS <-na.omit(diab)
dim(diab)
```

```
## [1] 149  11
```

```
dim(diab_noNAS)
```

```
## [1] 141  11
```

EXERCISE II

With the `diab` dataset

- Show only the rows from 35 to 98 and columns 5, 7, and from 9 to 11
- Change the level of the variable *tabac*, from **No Fumador** to **No_Fumador**
- Recode the BMI variable into a factor defined by
  - Thin: < 20
  - Normal [20-25)
  - Overweight [25-30)
  - Obese $>= 30$
- Display the mean of *edatdiag*, grouped by *ecg*

Exercise III

- Read the `osteoporosis.csv` data set into R so that character variables are automatically converted into factors.
- Make a summary of the dataset using standard and non-standard summary functions.
- Display the range of `bua`, grouped by `clasific`

Section 3

Descriptive Statistics: Graphical summaries

Exploratory Data Analysis (EDA)

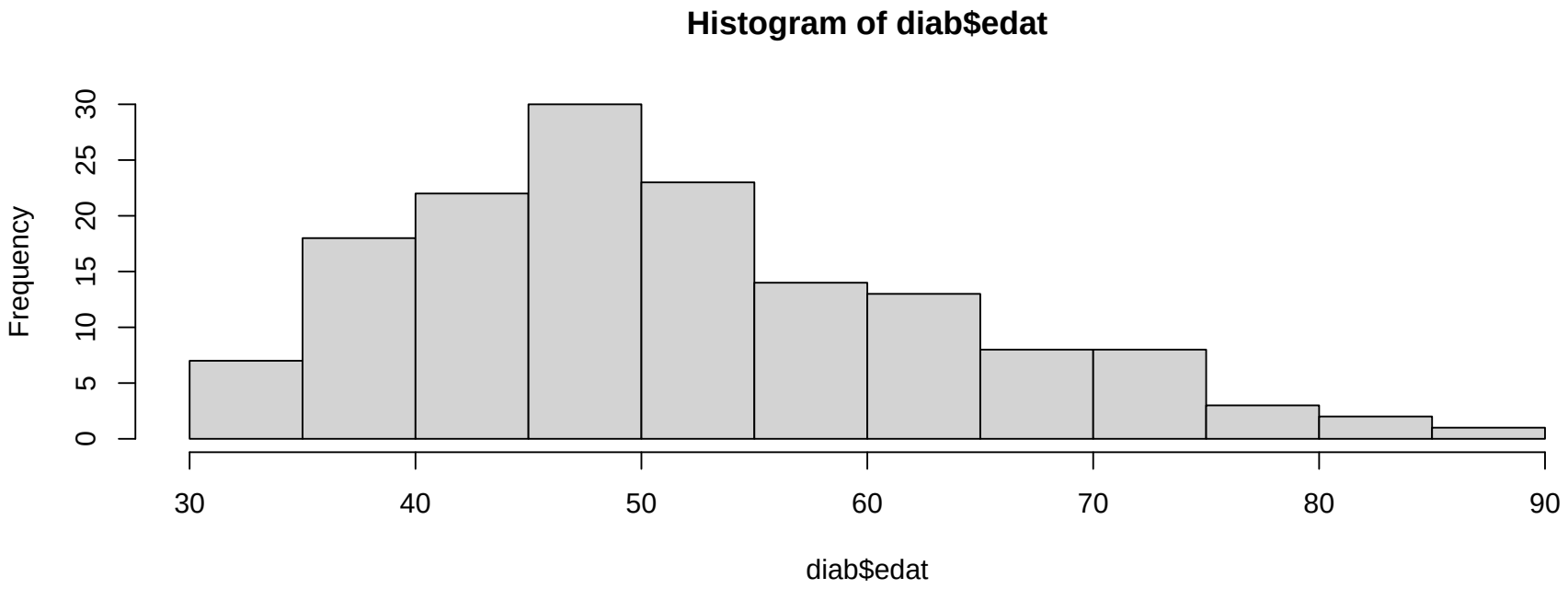
We could dedicate half of the course only to EDA. Here we will only see the most common approaches to visualize data:

- `Histograms`
- `Scatterplots`
- `Boxplots`

Histograms

We will use histograms to plot the frequencies of each level of variables. This is the way to see the data distribution of particulars variables.

```
hist(diab$edat)
```



Tuning plots with graphic parameters

The aspect of plots can be improved using "graphical parameters"

Some parameters are passed as arguments to the plotting function

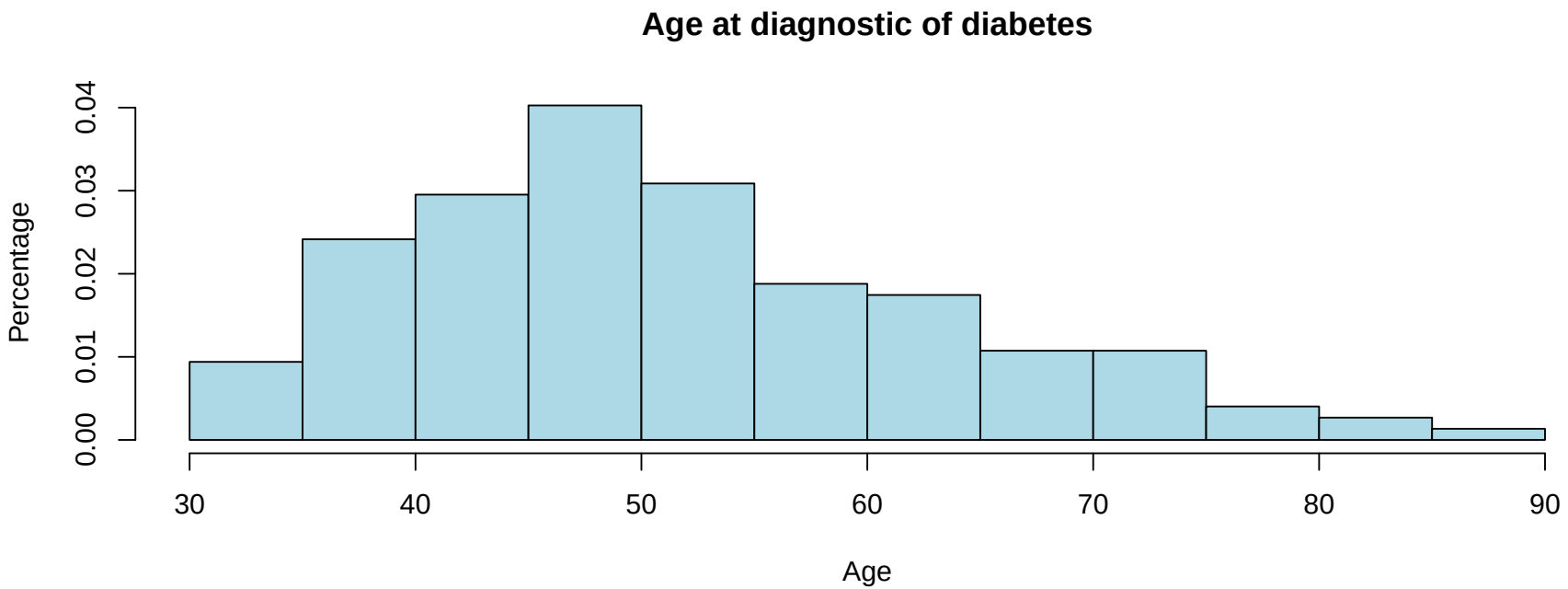
Others are set globally.

Setting graphical parameters has become so sophisticated that a new approach to graphics -in principle more intuitive- has emerged and come to change the way that graphics are created.

- The "traditional" approach relies on the `plot` function available in Rbase
- The "modern" approch is based on the `ggplots2` package and its "language" of graphics.

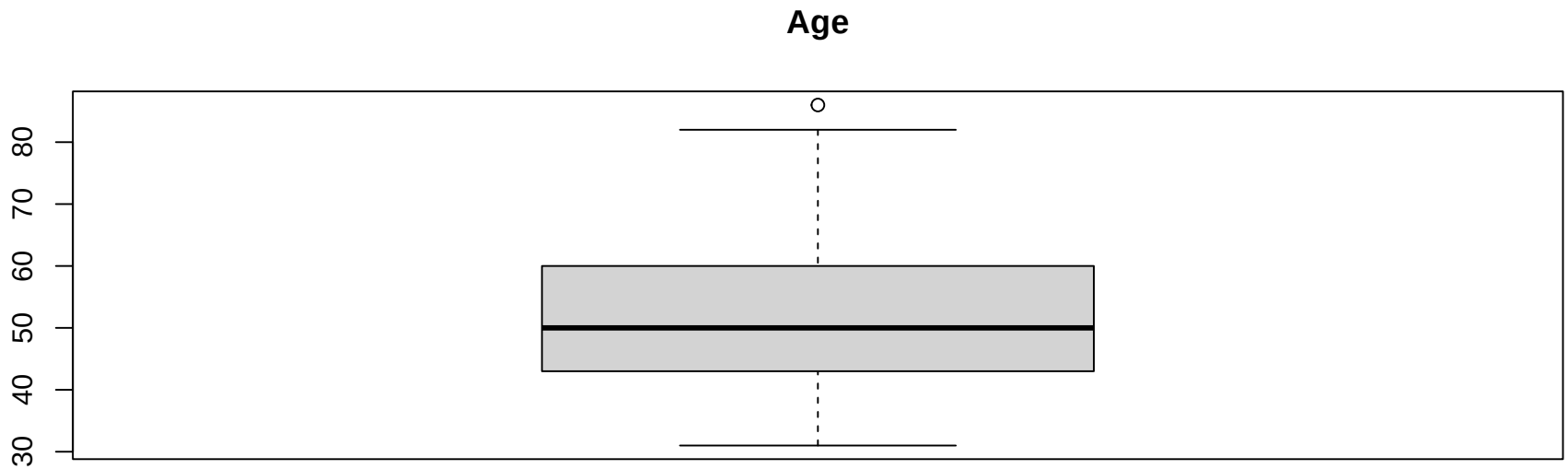
An improved histogram

```
hist(diab$edat, main="Age at diagnostic of diabetes",
     probability = TRUE, xlab="Age", ylab="Percentage",
     col="lightblue", breaks = 10)
```



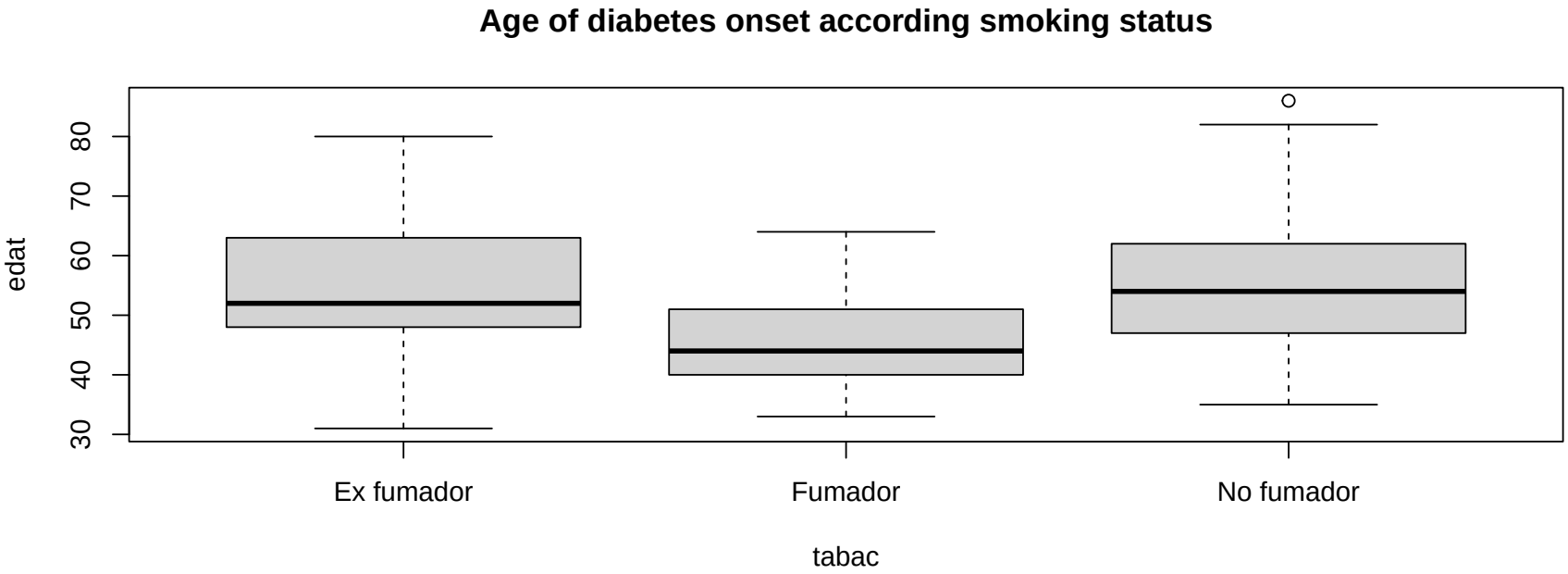
Boxplot. A unidimensional histogram

```
boxplot (diab$edat, main ="Age")
```



Boxplot: Decomposing plots by groups

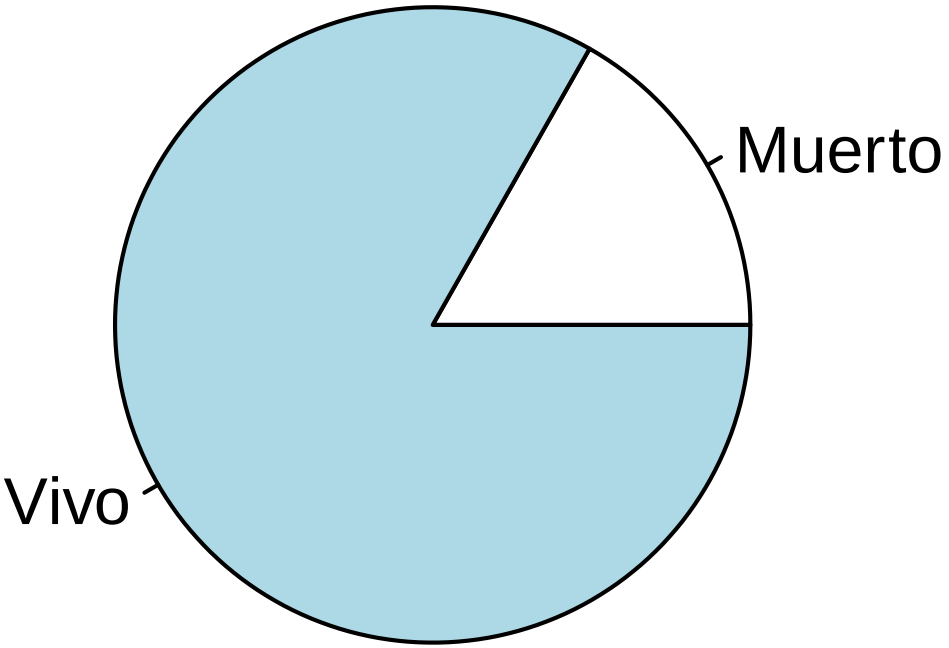
```
boxplot(edat~tabac, data=diab, main="Age of diabetes onset according sm
```



Plots for categorical variables

- Some simple principles
  - Use pie charts only with categorical variables in nominal scale
  - Use barplots for any categorical variable
  - Never use 3D-plots

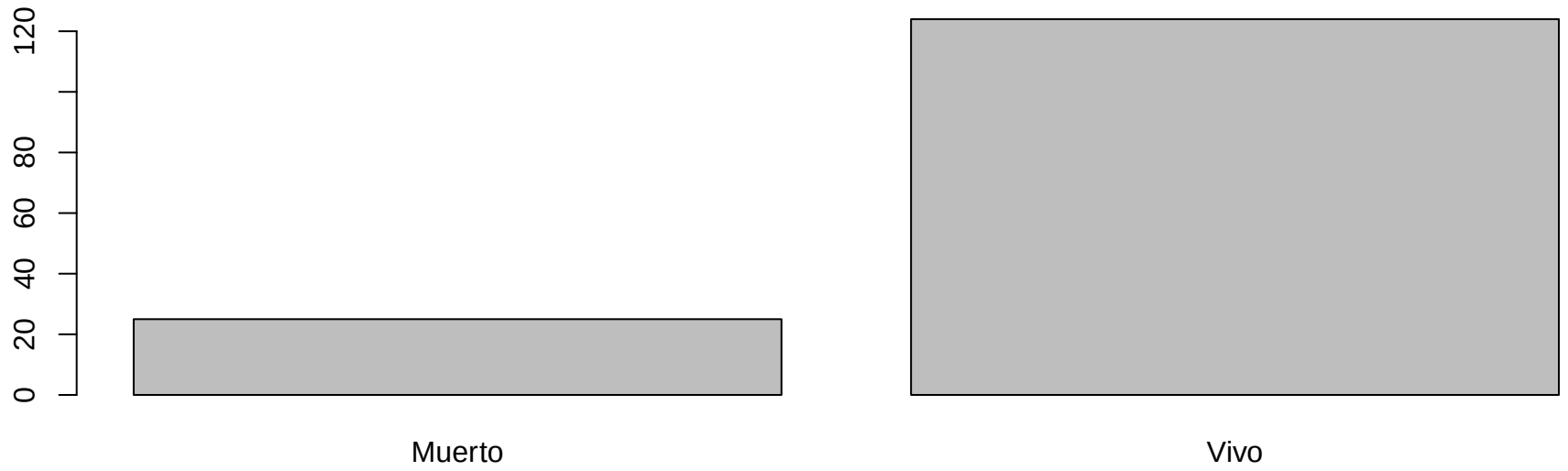
```
t <- table(diab$mort)
pie(t)
```



Barplots

- Similar to pie charts but, implicitly, suggest ordering

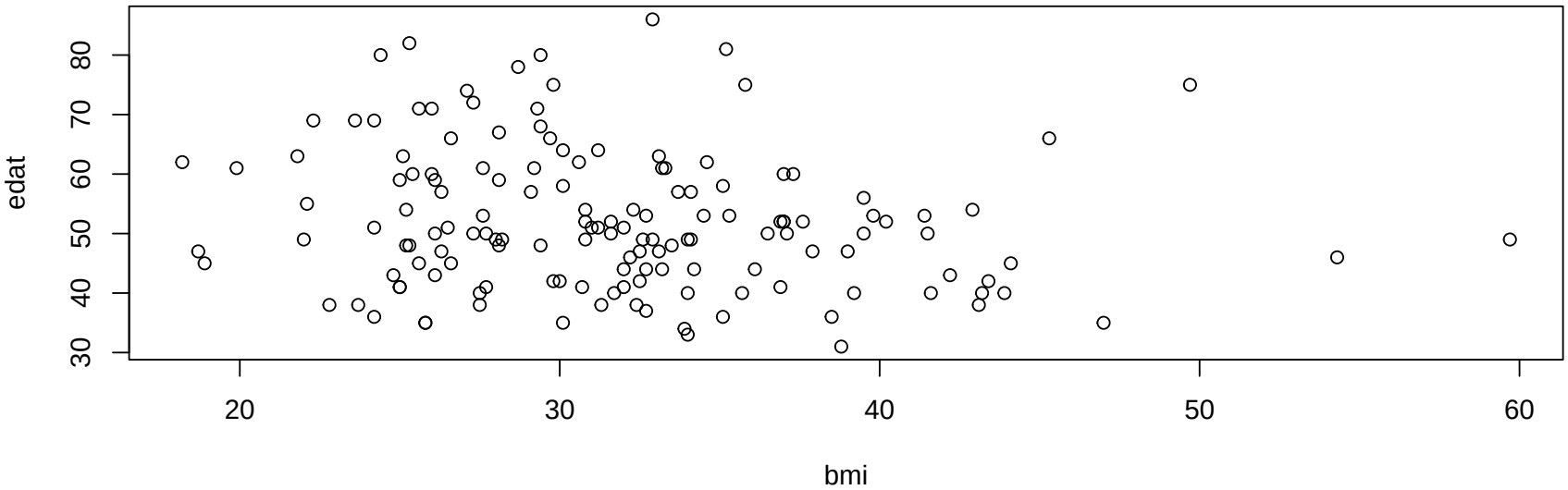
```
barplot(t)
```



Scatterplots

- For two variables, simply use `plot`

```
plot(edat~bmi, data=diab)
```



Exercise IV

- With the variables in the osteoporosis dataset
- Try to represent the different variables using *the most appropriate plot* for each of them.

EXERCISE

2. With the `diab` dataset
   - Use the best graphic type to plot the relation between *sbp* and *dbp*
   - Show graphically the relation between *edat* and *ecg*
   - Plot the *sbp* frequencies

EXERCISE

4. Using the *osteoporosis.csv* dataset
   - Load the dataset and check if it is correctly loaded
   - Calculate the mean and standard deviation of imc grouped by clasific
   - Plot the distribution of edat
   - Plot the relationship between talla and peso