

# Introduction to R

## MedILS School in Bioinformatics

Nikolina Pleić

MSc Mathematics  
University of Split, School of Medicine  
Department of Biology and Human Genetics

Monday 21<sup>st</sup> August, 2023

# Installing R

R is maintained by an international team of developers who make the language available through the web page of The Comprehensive R Archive Network (CRAN).

The web page provides three links for downloading R.

Follow the link that describes your operating system: Windows, Mac, or Linux.

# Starting R console

- Usually, what you would first do, is to create a folder to store your working data
- here we will work within the 'workdir' folder (you can create one on your Desktop)
- Search for R in your Windows app searcher
- From the console go to File → Change dir → set to 'workdir' folder or alternatively type into the console the following:  
`setwd("path-to-the-folder")`

# Using R console as a calculator

## Operators:

- Addition: +
- Subtraction: -
- Multiplication: \*
- Division: /
- etc.

You can type R commands in the R console which will be processed and their results output:

```
2+2
```

```
## [1] 4
```

```
14/7
```

```
## [1] 2
```

# Adding comments to the code

- tell R to ignore a part of your program
- R interprets anything after a `#` as a comment
- commenting your code can help other people read your program
- it can help you understand the code when you go back to an old script

```
a<-25  
b<-sqrt(a)  
b
```

```
## [1] 5
```

```
#b is the square root of a
```

# Data types

In programming and in R, data types are the classifications we give to different kinds of information. We will explore the following R data types:

- Numeric: any number (with or without a decimal point)
- Character (also known as string): any sequence of characters from your keyboard surrounded by quotes
- Logical: only two values: TRUE or FALSE. You can think of them as "yes" or "no" answers to a question
- Vectors: a list of data of the same type
- NA: missing value

# Example data types

```
class(6)
```

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

```
class('6')
```

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

```
class('Bioinfo')
```

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

```
class(TRUE)
```

```
## [1] "logical"
```

```
class(NA)
```

```
## [1] "logical"
```

# Assigning variables

The most basic concept in (statistical) programming is called a variable. A variable lets you store a value or a function which you can later easily access by typing the variable name.

*#We use the assignment operator, an arrow sign (<-) made  
#with a carat and a dash*

```
var1<-100
```

```
var1
```

```
## [1] 100
```

```
var2<- 'brain'
```

```
var2
```

```
## [1] "brain"
```



# Important things to note

Variable names can't have:

- spaces
- symbols (other than underscore \_)
- can't begin with numbers, but can have number after the first letter

*#Example*

```
1-my_variable6<-6
```

```
## Error in 1 - my_variable6 <- 6: target of assignment  
expands to non-language object
```

```
my_variable6<-6
```

```
my_variable6
```

```
## [1] 6
```

# Important things to note

Each reassignment updates the variable value:

```
message<-'Hello world!'
print(message)

## [1] "Hello world!"

message<-'Hello Split!'
print(message)

## [1] "Hello Split!"

#If we don't want to overwrite our variable:
message<-'Hello world!'
message2<-'Hello Split!'
print(message)

## [1] "Hello world!"
```

# Vectors

A list-like structure containing items of the same data type

```
croatian_cities<-c('Zagreb','Split','Rijeka')
croatian_cities

## [1] "Zagreb" "Split"  "Rijeka"
```

Basically, we list the strings and wrap them with `c()`.

Try to create a character vector that contains your address, for example:

```
address<-c('Soltanska 2','21000 Split', 'Croatia')
address

## [1] "Soltanska 2" "21000 Split" "Croatia"
```

# Vectors

Operations with vectors:

```
typeof(address)
```

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

*#If we want to get the postal code and the city:*

```
address[2]
```

```
## [1] "21000 Split"
```

```
length(address)
```

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

# For-loop

Operations with vectors:

```
new_vector<-c(1,2,3,4,5,6,7,8,9,10)
```

Let's say we want to increase each value of the vector by 10. In order to do this, we need to use the 'for-loop':

```
for (i in 1:length(new_vector)){  
  new_vector[i]<-new_vector[i]+10  
}  
new_vector  
  
##    [1] 11 12 13 14 15 16 17 18 19 20
```

## If-else statement loop

```
new_vector<-c(1,2,3,4,5,6,7,8,9,10)
```

Let's say we want to increase the values of the vector by 10, but only for values that are larger than 5. In order to do this, we need to use the 'if-else statement':

```
for (i in 1:length(new_vector)){  
  if(new_vector[i]>5){  
    new_vector[i]<-new_vector[i]+10  
  } else {new_vector[i]<-new_vector[i]}  
}  
new_vector
```

```
##      [1]  1  2  3  4  5 16 17 18 19 20
```

## If-else statement loop

Note that the else statement in the previous example could have been omitted because the values stay the same. Now let's say we want to increase the values of the vector by 10, but only for values that are larger than 5, and for the values smaller than 5, we want to multiply them by 10. In order to do this, we need to use the 'if-else statement':

```
new_vector<-c(1,2,3,4,5,6,7,8,9,10)
```

```
for (i in 1:length(new_vector)){  
  if(new_vector[i]>5){  
    new_vector[i]<-new_vector[i]+10  
  } else {new_vector[i]<-new_vector[i]*10}  
}  
new_vector  
  
##    [1] 10 20 30 40 50 16 17 18 19 20
```

# Importing packages

Even though base R is very powerful, R packages can make your life easier. A package is a bundle of code that makes performing tasks easier.

Let's import one of the most popular packages: dplyr.

Dplyr is a package for cleaning, processing, and organizing data.

1. To install (you do this only once):

```
install.packages('dplyr')
```

In case of an lifecycle error: `install.packages('lifecycle', type='binary')`

```
#2. To import:
```

```
library(dplyr)
```

```
##
```

```
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
## filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
## intersect, setdiff, setequal, union
```



# Data frames

- A data frame in R is an object that stores data in a tabular form, with rows and columns
- Can be created in R, but is most often imported from an Excel spreadsheet
- each column has a name and stores values for one variable
- tibble = data frame

# Data frames

We will now read an excel spreadsheet of biomedical data into R. The data consist of 19 variables on 403 subjects from 1046 subjects who were interviewed in a study to understand the prevalence of obesity, diabetes, and other cardiovascular risk factors in central Virginia for African Americans.

Source: <https://hbiostat.org/data/>

You can download the data into your created folder from the workshop's repository at: <https://github.com/npleic> → Bioinfo2023

# Data frames

First, we need a new package: `install.packages("readxl")`

Or alternatively, package 'xlsx'

```
#Since we have set the working directoy to be  
#our newly created folder,  
#we can load the data downloaded into it.
```

```
library(readxl)  
data<-read_excel("diabetes_data.xlsx")  
#alternatively, library('xlsx')  
#data<-read.xlsx("diabetes_data.xlsx",1)
```

# Data frames

*#You want to get an understanding of what the data looks like*

*#Get a preview of rows and columns*

```
str(data)
```

```
## tibble [403 x 12] (S3: tbl_df/tbl/data.frame)
```

```
## $ ID           : num [1:403] 1000 1001 1002 1003 1005 ...
## $ age          : num [1:403] 46 29 58 67 64 34 30 37 45 55 ...
## $ gender       : chr [1:403] "female" "female" "female" "male" ...
## $ height       : num [1:403] 62 64 61 67 68 71 69 59 69 63 ...
## $ weight       : num [1:403] 121 218 256 119 183 190 191 170 166 202
## $ waist        : num [1:403] 29 46 49 33 44 36 46 34 34 45 ...
## $ hip          : num [1:403] 38 48 57 38 41 42 49 39 40 50 ...
## $ location     : chr [1:403] "Buckingham" "Buckingham" "Buckingham" "
## $ cholesterol  : num [1:403] 203 165 228 78 249 248 195 227 177 263 .
## $ glucose      : num [1:403] 82 97 92 93 90 94 92 75 87 89 ...
## $ HDL          : num [1:403] 56 24 37 12 28 69 41 44 49 40 ...
## $ glycosylated_hemoglobin: num [1:403] 4.31 4.44 4.64 4.63 7.72 ...
```

# Data frames

*#The head() command shows the first 6 rows of the data frame*

```
head(data)
```

```
## # A tibble: 6 x 12
```

```
##       ID    age gender height weight waist  hip location  cholesterol glucose
##   <dbl> <dbl> <chr>   <dbl>  <dbl> <dbl> <dbl> <chr>          <dbl>    <dbl>
## 1  1000    46 female    62    121   29   38 Buckingham    203      82
## 2  1001    29 female    64    218   46   48 Buckingham    165      97
## 3  1002    58 female    61    256   49   57 Buckingham    228      92
## 4  1003    67 male      67    119   33   38 Buckingham     78      93
## 5  1005    64 male      68    183   44   41 Buckingham    249      90
## 6  1008    34 male      71    190   36   42 Buckingham    248      94
## # ... with 2 more variables: HDL <dbl>, glycosylated_hemoglobin <dbl>
```

*#If you want to see n rows, use head(data,n)*

# Data frames

```
#Get summary statistics for numeric columns
#Get class and length info for non-numeric columns
summary(data)
```

```
##      ID      age      gender      height
## Min.   : 1000   Min.   :19.00   Length:403   Min.   :52.00
## 1st Qu.: 4792   1st Qu.:34.00   Class :character 1st Qu.:63.00
## Median :15766   Median :45.00   Mode  :character Median :66.00
## Mean   :15978   Mean   :46.85                Mean   :66.02
## 3rd Qu.:20336   3rd Qu.:60.00                3rd Qu.:69.00
## Max.   :41756   Max.   :92.00                Max.   :76.00
##                                     NA's   :5
##      weight      waist      hip      location
## Min.   : 99.0   Min.   :26.0   Min.   :30.00   Length:403
## 1st Qu.:151.0   1st Qu.:33.0   1st Qu.:39.00   Class :character
## Median :172.5   Median :37.0   Median :42.00   Mode  :character
## Mean   :177.6   Mean   :37.9   Mean   :43.04
## 3rd Qu.:200.0   3rd Qu.:41.0   3rd Qu.:46.00
## Max.   :325.0   Max.   :56.0   Max.   :64.00
## NA's   :1       NA's   :2       NA's   :2
##      cholesterol      glucose      HDL      glycosylated_hemoglobin
## Min.   : 78.0   Min.   : 48.0   Min.   : 12.00   Min.   : 2.68
## 1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00   1st Qu.: 4.38
## Median :204.0   Median : 89.0   Median : 46.00   Median : 4.84
## Mean   :207.8   Mean   :106.7   Mean   : 50.45   Mean   : 5.59
## 3rd Qu.:230.0   3rd Qu.:106.0   3rd Qu.: 59.00   3rd Qu.: 5.60
## Max.   :443.0   Max.   :385.0   Max.   :120.00   Max.   :16.11
## NA's   :1       NA's   :1       NA's   :13
```

# Selecting columns

- `select()` returns a new data frame containing only the desired columns

*#Let's say we only want to keep age and gender*

*#It's recommended to keep the ID column*

```
data_age_gender<-select(data, ID, age, gender)
```

```
str(data_age_gender)
```

```
## tibble [403 x 3] (S3: tbl_df/tbl/data.frame)
```

```
##   $ ID      : num [1:403] 1000 1001 1002 1003 1005 ...
```

```
##   $ age     : num [1:403] 46 29 58 67 64 34 30 37 45 55 ...
```

```
##   $ gender: chr [1:403] "female" "female" "female" "male" ...
```

# Excluding columns

- `select()` also lets us exclude specific columns

```
#Let's say we don't need the variable location
data_without_location<-select(data, -location)
str(data_without_location)

## tibble [403 x 11] (S3: tbl_df/tbl/data.frame)
##  $ ID                : num [1:403] 1000 1001 1002 1003 1005 ...
##  $ age                : num [1:403] 46 29 58 67 64 34 30 37 45 55 ...
##  $ gender              : chr [1:403] "female" "female" "female" "male" ...
##  $ height              : num [1:403] 62 64 61 67 68 71 69 59 69 63 ...
##  $ weight              : num [1:403] 121 218 256 119 183 190 191 170 166 202
##  $ waist               : num [1:403] 29 46 49 33 44 36 46 34 34 45 ...
##  $ hip                 : num [1:403] 38 48 57 38 41 42 49 39 40 50 ...
##  $ cholesterol         : num [1:403] 203 165 228 78 249 248 195 227 177 263 .
##  $ glucose              : num [1:403] 82 97 92 93 90 94 92 75 87 89 ...
##  $ HDL                 : num [1:403] 56 24 37 12 28 69 41 44 49 40 ...
##  $ glycosylated_hemoglobin: num [1:403] 4.31 4.44 4.64 4.63 7.72 ...
```



# Excluding columns

- excluding multiple columns

```
#Let's say we don't need the variables location, waist and hip
```

```
data_without_cols<-select(data, -c(location,waist,hip))
```

```
str(data_without_cols)
```

```
## tibble [403 x 9] (S3: tbl_df/tbl/data.frame)
```

```
## $ ID : num [1:403] 1000 1001 1002 1003 1005 ...
```

```
## $ age : num [1:403] 46 29 58 67 64 34 30 37 45 55 ...
```

```
## $ gender : chr [1:403] "female" "female" "female" "male" ...
```

```
## $ height : num [1:403] 62 64 61 67 68 71 69 59 69 63 ...
```

```
## $ weight : num [1:403] 121 218 256 119 183 190 191 170 166 202
```

```
## $ cholesterol : num [1:403] 203 165 228 78 249 248 195 227 177 263 .
```

```
## $ glucose : num [1:403] 82 97 92 93 90 94 92 75 87 89 ...
```

```
## $ HDL : num [1:403] 56 24 37 12 28 69 41 44 49 40 ...
```

```
## $ glycosylated_hemoglobin: num [1:403] 4.31 4.44 4.64 4.63 7.72 ...
```

# Descriptive statistics

```
#Let's say we want to know the number of male/female participants  
table(data$gender)
```

```
##  
## female    male  
##      234     169
```

```
#Further, we want to know the average age  
summary(data$age) #we're looking at the mean value
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.  
##      19.00   34.00   45.00   46.85   60.00   92.00
```

```
#Last, we want to know the median height  
summary(data$height)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's  
##      52.00   63.00   66.00   66.02   69.00   76.00         5
```

# Descriptive statistics

```
#Let's inspect variable cholesterol  
#We want to know the average cholesterol level  
summary(data$cholesterol)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's  
##      78.0   179.0   204.0   207.8   230.0   443.0        1
```

```
#Let's say we want to inspect cholesterol levels stratified by gender  
by(data$cholesterol, data$gender, summary)
```

```
## data$gender: female
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.  
##      118.0   179.0   204.5   208.4   230.5   443.0
```

```
## -----
```

```
## data$gender: male
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's  
##       78     179     204     207     230     404        1
```

## Exercise - Descriptive statistics

You will now inspect the famous *mtcars* dataset. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Data can be used by calling 'mtcars' in the console. Perform the following tasks:

- 1 Explore the dataset. How many rows and columns are there?
- 2 Get the summary statistics of cars' weights (variable 'wt').
- 3 Inspect the fuel consumption (Miles/(US) gallon 'mpg') stratified by the number of cylinders 'cyl'.
- 4 From the mtcars data, create a new data frame without the horsepower (hp) and number of gears (gear).
- 5 Create a new variable cyl2 which will be the original variable multiplied by 100

# Exercise - Descriptive statistics

## Solution:

```
#1.  
str(mtcars) #there are 32 rows and 11 columns  
  
## 'data.frame': 32 obs. of 11 variables:  
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
## $ disp: num 160 160 108 258 360 ...  
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
## $ qsec: num 16.5 17 18.6 19.4 17 ...  
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

```
#2.  
summary(mtcars$wt)  
  
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
##    1.513   2.581   3.325   3.217   3.610   5.424
```

# Exercise - Descriptive statistics

## Solution:

```
#3.  
by(mtcars$mpg,mtcars$cyl,summary)
```

## mtcars\$cyl: 4
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
##    21.40   22.80   26.00   26.66   30.40   33.90
## -----

## mtcars\$cyl: 6
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
##    17.80   18.65   19.70   19.74   21.00   21.40
## -----

## mtcars\$cyl: 8
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
##    10.40   14.40   15.20   15.10   16.25   19.20

# Exercise - Descriptive statistics

## Solution:

```
#4.
library('dplyr')
data_new<-select(mtcars, -c(hp,gear))
str(data_new)

## 'data.frame': 32 obs. of 9 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
## $ disp: num 160 160 108 258 360 ...
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
## $ qsec: num 16.5 17 18.6 19.4 17 ...
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

# Exercise - Descriptive statistics

## Solution:

```
#5.  
mtcars$cyl2<-mtcars$cyl*100  
str(mtcars)  
  
## 'data.frame': 32 obs. of 12 variables:  
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
## $ disp: num 160 160 108 258 360 ...  
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
## $ qsec: num 16.5 17 18.6 19.4 17 ...  
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...  
## $ cyl2: num 600 600 400 600 800 600 800 400 400 600 ...
```



```
print('The End')
```

```
## [1] "The End"
```