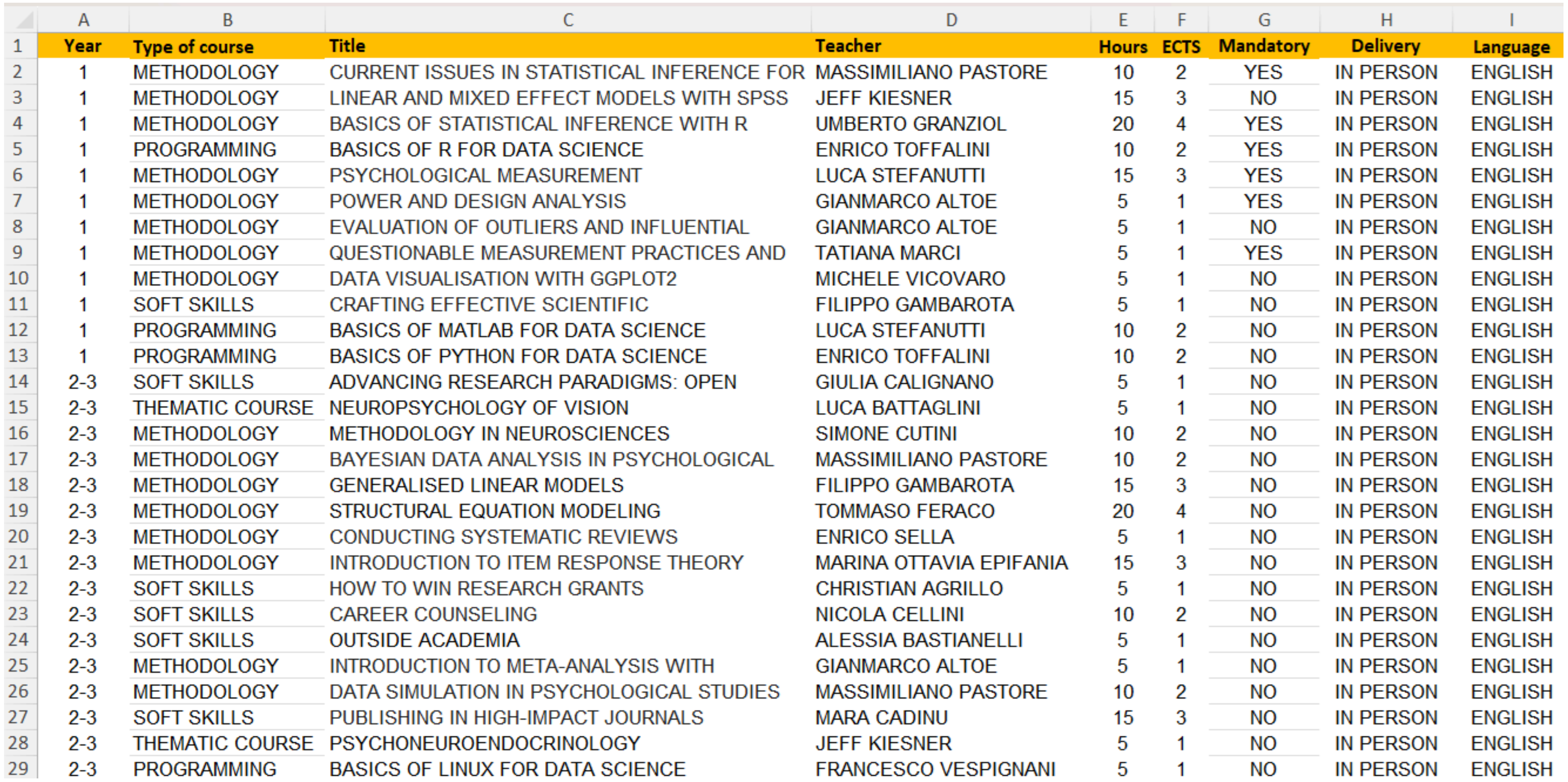
# Data Structures: Data Frames

Enrico Toffalini

# A type of data structure you are already familiar with



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Year | Type of course | Title | Teacher | Hours | ECTS | Mandatory | Delivery | Language |
| 2 | 1 | METHODOLOGY | CURRENT ISSUES IN STATISTICAL INFERENCE FOR | MASSIMILIANO PASTORE | 10 | 2 | YES | IN PERSON | ENGLISH |
| 3 | 1 | METHODOLOGY | LINEAR AND MIXED EFFECT MODELS WITH SPSS | JEFF KIESNER | 15 | 3 | NO | IN PERSON | ENGLISH |
| 4 | 1 | METHODOLOGY | BASICS OF STATISTICAL INFERENCE WITH R | UMBERTO GRANZIOL | 20 | 4 | YES | IN PERSON | ENGLISH |
| 5 | 1 | PROGRAMMING | BASICS OF R FOR DATA SCIENCE | ENRICO TOFFALINI | 10 | 2 | YES | IN PERSON | ENGLISH |
| 6 | 1 | METHODOLOGY | PSYCHOLOGICAL MEASUREMENT | LUCA STEFANUTTI | 15 | 3 | YES | IN PERSON | ENGLISH |
| 7 | 1 | METHODOLOGY | POWER AND DESIGN ANALYSIS | GIANMARCO ALTOE | 5 | 1 | YES | IN PERSON | ENGLISH |
| 8 | 1 | METHODOLOGY | EVALUATION OF OUTLIERS AND INFLUENTIAL | GIANMARCO ALTOE | 5 | 1 | NO | IN PERSON | ENGLISH |
| 9 | 1 | METHODOLOGY | QUESTIONABLE MEASUREMENT PRACTICES AND | TATIANA MARCI | 5 | 1 | YES | IN PERSON | ENGLISH |
| 10 | 1 | METHODOLOGY | DATA VISUALISATION WITH GGPLOT2 | MICHELE VICOVARO | 5 | 1 | NO | IN PERSON | ENGLISH |
| 11 | 1 | SOFT SKILLS | CRAFTING EFFECTIVE SCIENTIFIC | FILIPPO GAMBAROTA | 5 | 1 | NO | IN PERSON | ENGLISH |
| 12 | 1 | PROGRAMMING | BASICS OF MATLAB FOR DATA SCIENCE | LUCA STEFANUTTI | 10 | 2 | NO | IN PERSON | ENGLISH |
| 13 | 1 | PROGRAMMING | BASICS OF PYTHON FOR DATA SCIENCE | ENRICO TOFFALINI | 10 | 2 | NO | IN PERSON | ENGLISH |
| 14 | 2-3 | SOFT SKILLS | ADVANCING RESEARCH PARADIGMS: OPEN | GIULIA CALIGNANO | 5 | 1 | NO | IN PERSON | ENGLISH |
| 15 | 2-3 | THEMATIC COURSE | NEUROPSYCHOLOGY OF VISION | LUCA BATTAGLINI | 5 | 1 | NO | IN PERSON | ENGLISH |
| 16 | 2-3 | METHODOLOGY | METHODOLOGY IN NEUROSCIENCES | SIMONE CUTINI | 10 | 2 | NO | IN PERSON | ENGLISH |
| 17 | 2-3 | METHODOLOGY | BAYESIAN DATA ANALYSIS IN PSYCHOLOGICAL | MASSIMILIANO PASTORE | 10 | 2 | NO | IN PERSON | ENGLISH |
| 18 | 2-3 | METHODOLOGY | GENERALISED LINEAR MODELS | FILIPPO GAMBAROTA | 15 | 3 | NO | IN PERSON | ENGLISH |
| 19 | 2-3 | METHODOLOGY | STRUCTURAL EQUATION MODELING | TOMMASO FERACO | 20 | 4 | NO | IN PERSON | ENGLISH |
| 20 | 2-3 | METHODOLOGY | CONDUCTING SYSTEMATIC REVIEWS | ENRICO SELLA | 5 | 1 | NO | IN PERSON | ENGLISH |
| 21 | 2-3 | METHODOLOGY | INTRODUCTION TO ITEM RESPONSE THEORY | MARINA OTTAVIA EPIFANIA | 15 | 3 | NO | IN PERSON | ENGLISH |
| 22 | 2-3 | SOFT SKILLS | HOW TO WIN RESEARCH GRANTS | CHRISTIAN AGRILLO | 5 | 1 | NO | IN PERSON | ENGLISH |
| 23 | 2-3 | SOFT SKILLS | CAREER COUNSELING | NICOLA CELLINI | 10 | 2 | NO | IN PERSON | ENGLISH |
| 24 | 2-3 | SOFT SKILLS | OUTSIDE ACADEMIA | ALESSIA BASTIANELLI | 5 | 1 | NO | IN PERSON | ENGLISH |
| 25 | 2-3 | METHODOLOGY | INTRODUCTION TO META-ANALYSIS WITH | GIANMARCO ALTOE | 5 | 1 | NO | IN PERSON | ENGLISH |
| 26 | 2-3 | METHODOLOGY | DATA SIMULATION IN PSYCHOLOGICAL STUDIES | MASSIMILIANO PASTORE | 10 | 2 | NO | IN PERSON | ENGLISH |
| 27 | 2-3 | SOFT SKILLS | PUBLISHING IN HIGH-IMPACT JOURNALS | MARA CADINU | 15 | 3 | NO | IN PERSON | ENGLISH |
| 28 | 2-3 | THEMATIC COURSE | PSYCHONEUROENDOCRINOLOGY | JEFF KIESNER | 5 | 1 | NO | IN PERSON | ENGLISH |
| 29 | 2-3 | PROGRAMMING | BASICS OF LINUX FOR DATA SCIENCE | FRANCESCO VESPIGNANI | 5 | 1 | NO | IN PERSON | ENGLISH |

## A type of data structure you are already familiar with

here is how I would import it in R (download here), and display the first few rows:

```r
library(readxl)
df = data.frame(read_excel("data/Courses40Cycle.xlsx"))
head(df)
```

```
  Year TypeOfCourse                                                Title
1    1  METHODOLOGY CURRENT ISSUES IN STATISTICAL INFERENCE FOR PSYCHOLOGY
2    1  METHODOLOGY               LINEAR AND MIXED EFFECT MODELS WITH SPSS
3    1  METHODOLOGY                 BASICS OF STATISTICAL INFERENCE WITH R
4    1  PROGRAMMING                           BASICS OF R FOR DATA SCIENCE
5    1  METHODOLOGY                              PSYCHOLOGICAL MEASUREMENT
6    1  METHODOLOGY                              POWER AND DESIGN ANALYSIS
               Teacher Hours ECTS MandatoryAttendance DeliveryMethod Language
1 MASSIMILIANO PASTORE    10    2                 YES      IN PERSON  ENGLISH
2         JEFF KIESNER    15    3                  NO      IN PERSON  ENGLISH
3     UMBERTO GRANZIOL    20    4                 YES      IN PERSON  ENGLISH
4     ENRICO TOFFALINI    10    2                 YES      IN PERSON  ENGLISH
5      LUCA STEFANUTTI    15    3                 YES      IN PERSON  ENGLISH
6      GIANMARCO ALTOE     5    1                 YES      IN PERSON  ENGLISH
```

## Dataframes as collections of vectors

In fact, **dataframes** are just collections (lists) of **vectors of different types**, all with the same length. Each column in a dataframe is a vector (a variable):

```
df$Teacher
```

```
 [1] "MASSIMILIANO PASTORE"    "JEFF KIESNER"
 [3] "UMBERTO GRANZIOL"        "ENRICO TOFFALINI"
 [5] "LUCA STEFANUTTI"         "GIANMARCO ALTOE"
 [7] "GIANMARCO ALTOE"         "TATIANA MARCI"
 [9] "MICHELE VICOVARO"        "FILIPPO GAMBAROTA"
[11] "LUCA STEFANUTTI"         "ENRICO TOFFALINI"
[13] "GIULIA CALIGNANO"        "LUCA BATTAGLINI"
[15] "SIMONE CUTINI"           "MASSIMILIANO PASTORE"
[17] "FILIPPO GAMBAROTA"       "TOMMASO FERACO"
[19] "ENRICO SELLA"            "MARINA OTTAVIA EPIFANIA"
[21] "CHRISTIAN AGRILLO"       "NICOLA CELLINI"
[23] "ALESSIA BASTIANELLI"     "GIANMARCO ALTOE"
[25] "MASSIMILIANO PASTORE"    "MARA CADINU"
[27] "JEFF KIESNER"            "FRANCESCO VESPIGNANI"
```

```
df$Hours
```

```
 [1] 10 15 20 10 15  5  5  5  5  5 10 10  5  5 10 10 15 20  5 15  5 10  5  5
10
[26] 15  5  5
```

## Basic Functions on Dataframes

To know the names of all variables in a dataframe, use the **`names()`** function:

```
names(df)
```

```
[1] "Year"                  "TypeOfCourse"          "Title"
[4] "Teacher"               "Hours"                 "ECTS"
[7] "MandatoryAttendance" "DeliveryMethod"          "Language"
```

Use the **`dim()`** function to view the dimensions of a dataframe:

```
# first value is number of rows, second is number of columns (variable
dim(df)
```

```
[1] 28  9
```

Alternatively, you can use **`nrow()`** and **`ncol()`**:

```
nrow(df) # number of rows
```

```
[1] 28
```

```
ncol(df) # number of columns
```

```
[1] 9
```

## Basic Functions on Dataframes

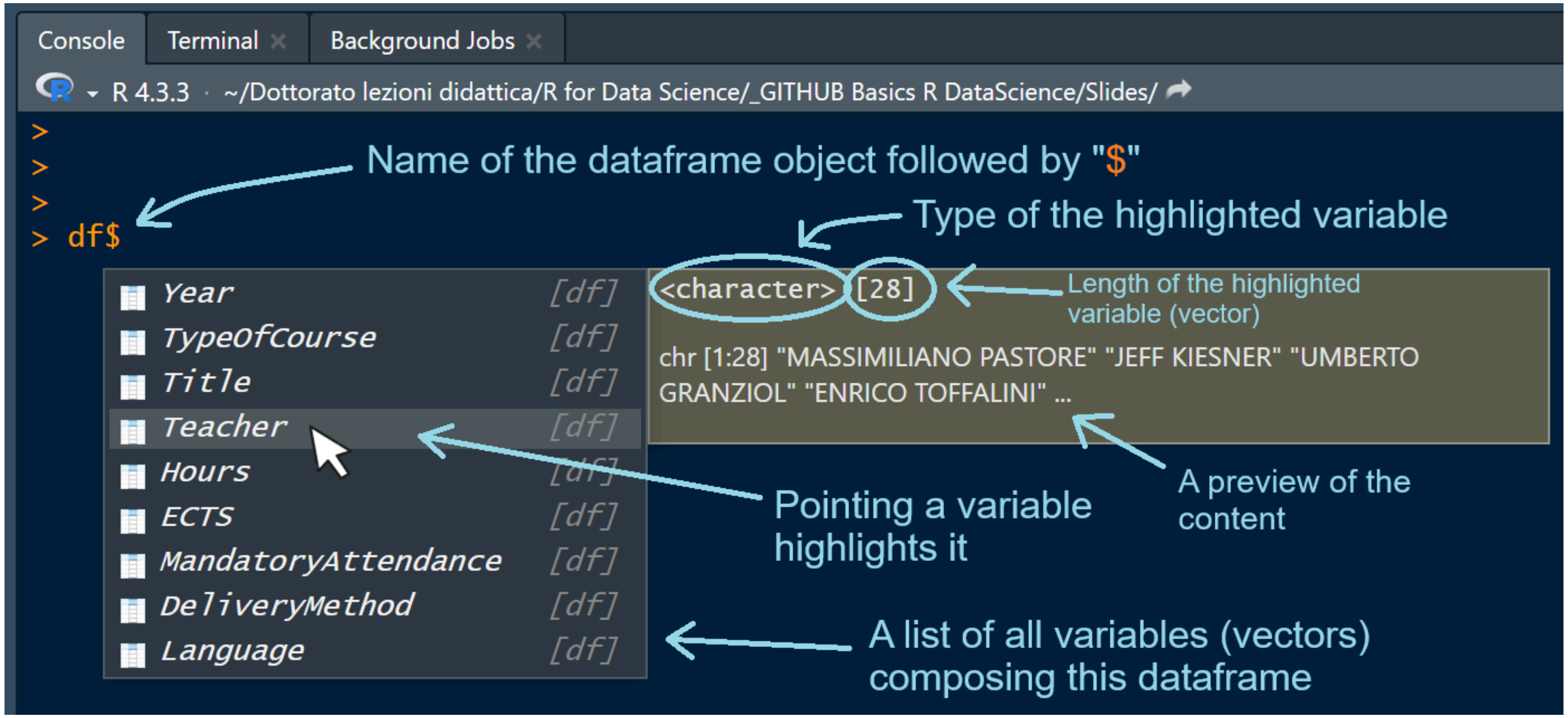
The `str()` function provides a quick overview of the structure of a dataframe, including its dimensions, variables, their data types, and first few observations:

```r
str(df)
```

```
'data.frame':	28 obs. of  9 variables:
 $ Year               : chr  "1" "1" "1" "1" ...
 $ TypeOfCourse       : chr  "METHODOLOGY" "METHODOLOGY" "METHODOLOGY"
"PROGRAMMING" ...
 $ Title              : chr  "CURRENT ISSUES IN STATISTICAL INFERENCE FOR
PSYCHOLOGY" "LINEAR AND MIXED EFFECT MODELS WITH SPSS" "BASICS OF STATISTICAL
INFERENCE WITH R" "BASICS OF R FOR DATA SCIENCE" ...
 $ Teacher            : chr  "MASSIMILIANO PASTORE" "JEFF KIESNER" "UMBERTO
GRANZIOL" "ENRICO TOFFALINI" ...
 $ Hours              : num  10 15 20 10 15 5 5 5 5 5 ...
 $ ECTS               : num  2 3 4 2 3 1 1 1 1 1 ...
 $ MandatoryAttendance: chr  "YES" "NO" "YES" "YES" ...
 $ DeliveryMethod     : chr  "IN PERSON" "IN PERSON" "IN PERSON" "IN PERSON"
...
 $ Language           : chr  "ENGLISH" "ENGLISH" "ENGLISH" "ENGLISH" ...
```

## Accessing Elements in Dataframes

The “$” (dollar) operator is essential to access variables in a dataframe:


Console
Terminal
Background Jobs
R 4.3.3 · ~/Dottorato lezioni didattica/R for Data Science/_GITHUB Basics R DataScience/Slides/
Name of the dataframe object followed by "$"
Type of the highlighted variable
Length of the highlighted variable (vector)
Pointing a variable highlights it
A preview of the content
A list of all variables (vectors) composing this dataframe

```
>
>
>
> df$
```

```
Year                  [df]
TypeOfCourse          [df]
Title                 [df]
Teacher               [df]
Hours                 [df]
ECTS                  [df]
MandatoryAttendance   [df]
DeliveryMethod        [df]
Language              [df]
```

```
<character> [28]

chr [1:28] "MASSIMILIANO PASTORE" "JEFF KIESNER" "UMBERTO
GRANZIOL" "ENRICO TOFFALINI" ...
```

## Accessing and Working with Elements in Dataframes

As an exercise, let's check whether `ECTS` is actually always `Hours*5`

We may use many different, increasingly sophisticated, strategies:

```
df$Hours / df$ECTS
```

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

```
(df$Hours / df$ECTS) == 5
```

```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
TRUE
```

```
sum((df$Hours / df$ECTS) == 5)
```

```
[1] 28
```

```
sum((df$Hours / df$ECTS) == 5) == nrow(df)
```

```
[1] TRUE
```

```
sum((df$Hours / df$ECTS) != 5)
```

```
[1] 0
```

## Accessing and Working with Elements in Dataframes

Variables in a dataframe can be manipulated like any other vector:

```r
log(df$Hours)
```

```
 [1] 2.302585 2.708050 2.995732 2.302585 2.708050 1.609438 1.609438 1.609438
 [9] 1.609438 1.609438 2.302585 2.302585 1.609438 1.609438 2.302585 2.302585
[17] 2.708050 2.995732 1.609438 2.708050 1.609438 2.302585 1.609438 1.609438
[25] 2.302585 2.708050 1.609438 1.609438
```

Also, new variables can easily be created and added at any time:

```r
df$newVar = log(df$Hours)
names(df)
```

```
 [1] "Year"                "TypeOfCourse"        "Title"
 [4] "Teacher"             "Hours"               "ECTS"
 [7] "MandatoryAttendance" "DeliveryMethod"      "Language"
[10] "newVar"
```

```r
df$newVar
```

```
 [1] 2.302585 2.708050 2.995732 2.302585 2.708050 1.609438 1.609438 1.609438
 [9] 1.609438 1.609438 2.302585 2.302585 1.609438 1.609438 2.302585 2.302585
[17] 2.708050 2.995732 1.609438 2.708050 1.609438 2.302585 1.609438 1.609438
[25] 2.302585 2.708050 1.609438 1.609438
```

## Indexing Elements in a Dataframe

In addition to using the "`$`" (dollar) operator, you can directly access a variable of a dataframe using **indexing** with **square brackets** `[ ]`:

```
        df[ , "Hours"]
```

```
 [1] 10 15 20 10 15  5  5  5  5  5 10 10  5  5 10 10 15 20  5 15  5 10  5  5
10
[26] 15  5  5
```

Notice the comma "`,`" above.

Unlike vectors, dataframes must be indexed by **both row and column**. In the example above, we're specifying only the desired column (`"Hours"`), leaving the row index blank before the comma `,` This selects all rows for the column named `"Hours"`. Remember that blank index means "all". Importantly, the "`,`" must always be there when indexing dataframes!

# Indexing Elements in a Dataframe - Examples

```r
df[ 1 , "Hours"]
```

```
[1] 10
```

```r
df[ 1:5 , "Hours"]
```

```
[1] 10 15 20 10 15
```

```r
df[ 1 , c("Teacher","Hours","TypeOfCourse")]
```

```
               Teacher Hours TypeOfCourse
1 MASSIMILIANO PASTORE    10  METHODOLOGY
```

```r
df[ 1:5 , c("Teacher","Hours","TypeOfCourse")]
```

```
               Teacher Hours TypeOfCourse
1 MASSIMILIANO PASTORE    10  METHODOLOGY
2         JEFF KIESNER    15  METHODOLOGY
3     UMBERTO GRANZIOL    20  METHODOLOGY
4     ENRICO TOFFALINI    10  PROGRAMMING
5      LUCA STEFANUTTI    15  METHODOLOGY
```

```r
df[ 1 , c(4, 5, 2)]
```

```
               Teacher Hours TypeOfCourse
1 MASSIMILIANO PASTORE    10  METHODOLOGY
```

# Indexing and Modifying Elements in a Dataframe

Editing/modifying elements in a dataframe is similar to what you do in vectors

```r
dx = data.frame(name = letters[1:10],
                score = rnorm(10))
dx
```

```
   name       score
1     a  1.262954285
2     b -0.326233361
3     c  1.329799263
4     d  1.272429321
5     e  0.414641434
6     f -1.539950042
7     g -0.928567035
8     h -0.294720447
9     i -0.005767173
10    j  2.404653389
```

```r
dx[3:5, "score"] = NA

dx
```

```
   name       score
1     a  1.262954285
2     b -0.326233361
3     c           NA
4     d           NA
5     e           NA
6     f -1.539950042
7     g -0.928567035
8     h -0.294720447
9     i -0.005767173
10    j  2.404653389
```

## Indexing and Modifying Elements in a Dataframe

Editing/modifying elements in a dataframe is similar to what you do in vectors

```
dx = data.frame(name = letters[1:10],
                score = rnorm(10))
dx
```

```
   name        score
1     a  1.262954285
2     b -0.326233361
3     c  1.329799263
4     d  1.272429321
5     e  0.414641434
6     f -1.539950042
7     g -0.928567035
8     h -0.294720447
9     i -0.005767173
10    j  2.404653389
```

```
dx$score = dx$score * 10

dx
```

```
   name        score
1     a  12.62954285
2     b  -3.26233361
3     c  13.29799263
4     d  12.72429321
5     e   4.14641434
6     f -15.39950042
7     g  -9.28567035
8     h  -2.94720447
9     i  -0.05767173
10    j  24.04653389
```

## Indexing and Modifying Elements in a Dataframe

Editing/modifying elements in a dataframe is similar to what you do in vectors

```r
dx = data.frame(name = letters[1:10],
                score = rnorm(10))
dx
```

```
   name        score
1     a  1.262954285
2     b -0.326233361
3     c  1.329799263
4     d  1.272429321
5     e  0.414641434
6     f -1.539950042
7     g -0.928567035
8     h -0.294720447
9     i -0.005767173
10    j  2.404653389
```

```r
dx$score = round( 100 + d

dx
```

```
   name score
1     a   119
2     b    95
3     c   120
4     d   119
5     e   106
6     f    77
7     g    86
8     h    96
9     i   100
10    j   136
```

## Indexing and Modifying Elements in a Dataframe

**Assigning a single value** to a column replaces the value for all rows (this is unlike vectors)

```r
dx = data.frame(name = letters[1:10],
                score = rnorm(10))
dx
```

```
   name        score
1     a  1.262954285
2     b -0.326233361
3     c  1.329799263
4     d  1.272429321
5     e  0.414641434
6     f -1.539950042
7     g -0.928567035
8     h -0.294720447
9     i -0.005767173
10    j  2.404653389
```

```r
dx$score = 3

dx
```

```
   name score
1     a     3
2     b     3
3     c     3
4     d     3
5     e     3
6     f     3
7     g     3
8     h     3
9     i     3
10    j     3
```

## Indexing Dataframes by Logical Conditions

Just like for vectors, you can use **logical conditions** for indexing a dataframe.

How do you extract **only the rows** related to `"MASSIMILIANO PASTORE`“?

```r
head(df)
```

```
  Year TypeOfCourse                                                   Title
1    1  METHODOLOGY CURRENT ISSUES IN STATISTICAL INFERENCE FOR PSYCHOLOGY
2    1  METHODOLOGY              LINEAR AND MIXED EFFECT MODELS WITH SPSS
3    1  METHODOLOGY                BASICS OF STATISTICAL INFERENCE WITH R
4    1  PROGRAMMING                          BASICS OF R FOR DATA SCIENCE
5    1  METHODOLOGY                             PSYCHOLOGICAL MEASUREMENT
6    1  METHODOLOGY                             POWER AND DESIGN ANALYSIS
              Teacher Hours ECTS MandatoryAttendance DeliveryMethod Language
1 MASSIMILIANO PASTORE    10    2                 YES      IN PERSON  ENGLISH
2         JEFF KIESNER    15    3                  NO      IN PERSON  ENGLISH
3     UMBERTO GRANZIOL    20    4                 YES      IN PERSON  ENGLISH
4     ENRICO TOFFALINI    10    2                 YES      IN PERSON  ENGLISH
5      LUCA STEFANUTTI    15    3                 YES      IN PERSON  ENGLISH
6      GIANMARCO ALTOE     5    1                 YES      IN PERSON  ENGLISH
    newVar
1 2.302585
```

## Indexing Dataframes by Logical Conditions

Just like for vectors, you can use **logical conditions** for indexing a dataframe.

Let’s consider this logical condition:

```
df$Teacher == "MASSIMILIANO PASTORE"
```

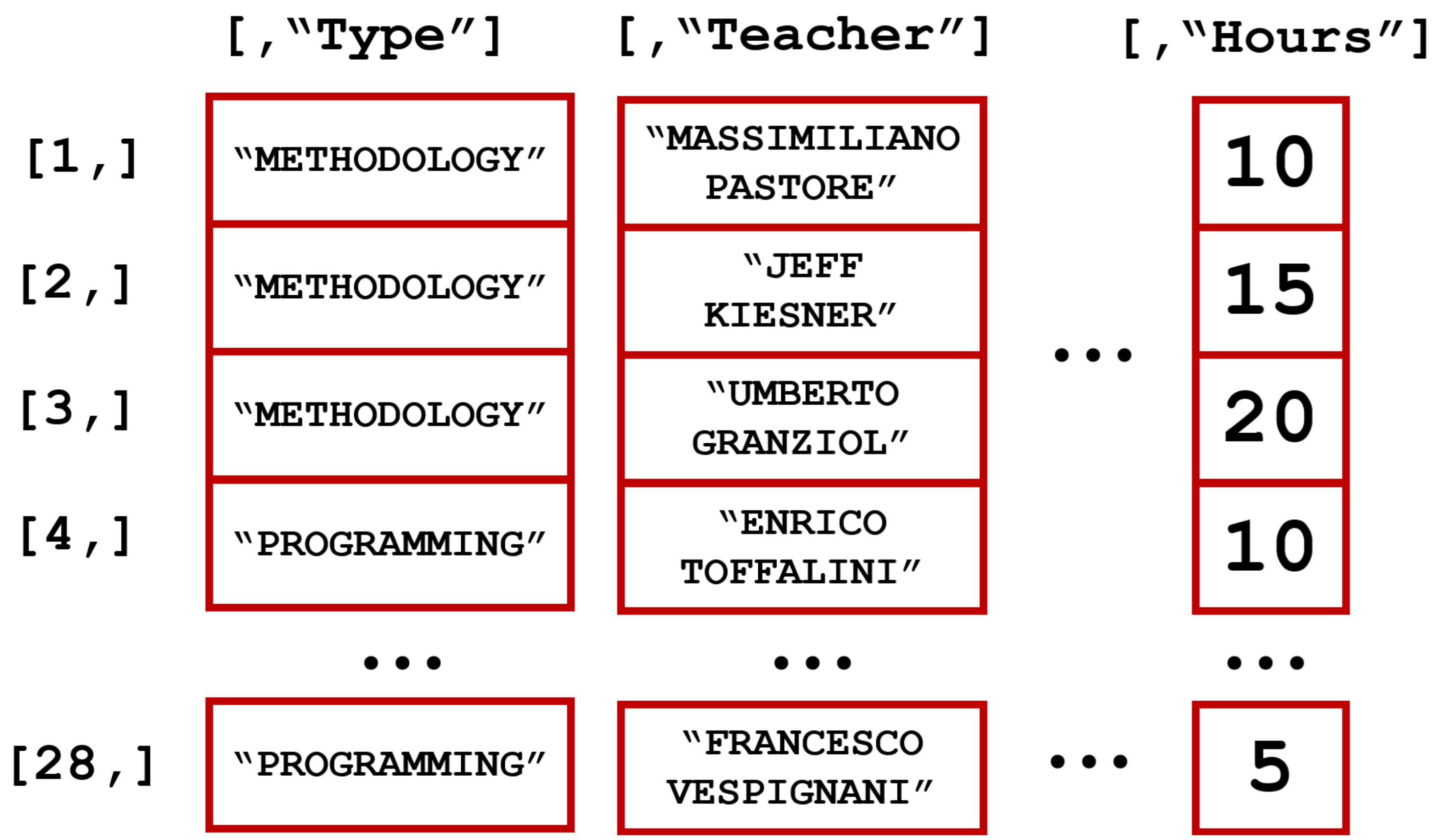
```
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25]  TRUE FALSE FALSE FALSE
```

Let’s use it to extract some dataframe rows:

```
df[df$Teacher == "MASSIMILIANO PASTORE" , ] # some rows by condition ,
```

```
   Year TypeOfCourse                                               Title
1     1  METHODOLOGY CURRENT ISSUES IN STATISTICAL INFERENCE FOR PSYCHOLOGY
16  2-3  METHODOLOGY       BAYESIAN DATA ANALYSIS IN PSYCHOLOGICAL RESEARCH
25  2-3  METHODOLOGY               DATA SIMULATION IN PSYCHOLOGICAL STUDIES
                Teacher Hours ECTS MandatoryAttendance DeliveryMethod Language
1  MASSIMILIANO PASTORE    10    2                 YES      IN PERSON  ENGLISH
16 MASSIMILIANO PASTORE    10    2                  NO      IN PERSON  ENGLISH
25 MASSIMILIANO PASTORE    10    2                  NO      IN PERSON  ENGLISH
     newVar
1  2.302585
16 2.302585
25 2.302585
```

## Indexing Elements in a Dataframe - A Summary



## Subset

Base function `subset()` can also be used as an alternative to indexing

```
subset(df, Teacher == "MASSIMILIANO PASTORE", select=c("Teacher","Hours
```

```
                Teacher Hours TypeOfCourse
1  MASSIMILIANO PASTORE    10  METHODOLOGY
16 MASSIMILIANO PASTORE    10  METHODOLOGY
25 MASSIMILIANO PASTORE    10  METHODOLOGY
```

```
df[df$Teacher == "MASSIMILIANO PASTORE" , c("Teacher","Hours","TypeOfC
```

```
                Teacher Hours TypeOfCourse
1  MASSIMILIANO PASTORE    10  METHODOLOGY
16 MASSIMILIANO PASTORE    10  METHODOLOGY
25 MASSIMILIANO PASTORE    10  METHODOLOGY
```

However, indexing with `[]` is more “computationally focused”, computationally faster (especially if working with large datasets), and more similar to programming in other languages (e.g., **Python**), so should probably be favoured by data scientists!

## Combine Two Dataframes Using `rbind()`

Imagine you have two datasets collected by two students, each including different participants:

```
df1
```

```
  subjName age accuracy
1    Julie  12     0.92
2    Tommy  10     0.78
3     Phil  10     0.85
```

```
df2
```

```
  subjName age accuracy
1    Amber   9     0.87
2      Max  13     0.90
```

Our goal is to get one single dataset including all participant’s data for the final analysis. Of course, you could combined these files manually outside R (e.g., in Excel). However, it’s simpler and more efficient to do this directly in R using `rbind()`

## Combine Two Dataframes Using `rbind()`

```
dfTotal = rbind(df1, df2)

dfTotal
```

```
  subjName age accuracy
1    Julie  12     0.92
2    Tommy  10     0.78
3     Phil  10     0.85
4    Amber   9     0.87
5      Max  13     0.90
```

**Important**: for `rbind()` to work, the two to-be-combined dataframes must:

- have the **exact same number of columns**;
- the **column names must be identical** (remember that R is case-sensitive).

## Merge Two Dataframes Using `merge()`

Another frequent case is having data collected from the **same participants** across **different dataframes**, and having to analyze all information together:

```
        df1
```

```
  subjName age
1    Julie  12
2    Amber   9
3    Tommy  10
4     Phil  10
```

```
        df2
```

```
  subjName accuracy time
1    Julie     0.92 1203
2    Tommy     0.78 3302
3     Phil     0.85  994
4    Amber     0.87 1163
```

## Merge Two Dataframes Using `merge()`

You can merge the two dataframes into a single, comprehensive dataframe:

```
dfTotal = merge(df1, df2, by="subjName")

dfTotal
```

```
  subjName age accuracy time
1    Amber   9     0.87 1163
2    Julie  12     0.92 1203
3     Phil  10     0.85  994
4    Tommy  10     0.78 3302
```

**Important:** `merge()` will work even if some or even all values that should be used for merging do not match… but in that case part of or all data will be lost

## Contingency Tables

The `table()` function, which counts frequencies, can also be used on dataframes. Importantly, it can also create **contingency tables** when applied to multiple variables at once

```
table(df$Hours) # just counts frequencies
```

```
 5 10 15 20 
13  8  5  2 
```

```
table(df$TypeOfCourse, df$Hours) # creates contingency tab
```

```
                  
                    5 10 15 20
  METHODOLOGY       6  4  4  2
  PROGRAMMING       1  3  0  0
  SOFT SKILLS       4  1  1  0
  THEMATIC COURSE   2  0  0  0
```