Introduction to R

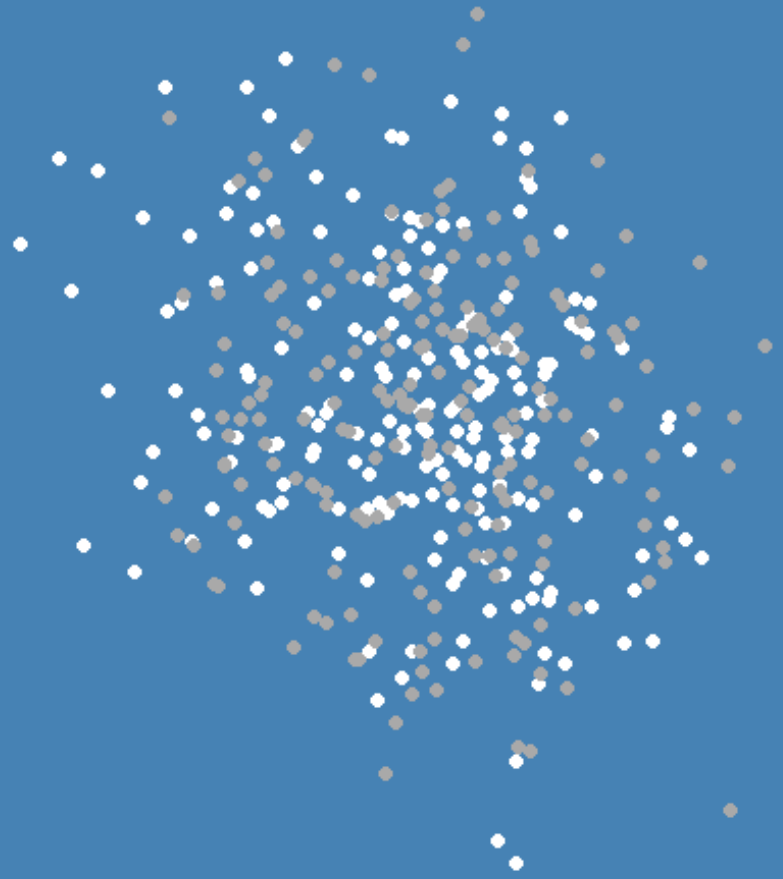
1.4 Object Classes

Vectors, Matrices, Data Frames, Lists

Lion Behrens, M.Sc.

UNIVERSITY OF MANNHEIM

University of Mannheim
Chair of Social Data Science and Methodology
Chair of Quantitative Methods in the Social Sciences



Basic R Functionality: Philosophy of R

"To understand computations in R, two slogans are helpful:

- *Everything that exists is an object.*
- *Everything that happens is a function call."*

---John Chambers, developer of R.

R Objects - Central Distinction

The central workhorse in R are objects which, in their most general sense, simply store information.

Data types

(how your data is stored)

- Numeric values (integers & doubles)
- Character strings
- Boolean values
- Factors
- ...

Data formats

(where your data is stored)

- Vectors
- Matrices
- Data frames/Tibbles
- Lists
- Arrays
- ...

Data Types

Numeric Values

There are two types of numerical values in R.

Integers are numbers without a decimal value. To explicitly denote that an object is an integer, you need to place an L behind the actual number.

```
x1 <- 5L
typeof(x1)
```

```
## [1] "integer"
```

Doubles are numbers with (a) decimal value(s).

```
x2 <- 5.25
typeof(x2)
```

```
## [1] "double"
```

Both x1 and x2 are numeric objects.

```
is.numeric(x1)
```

```
## [1] TRUE
```

Numeric Values

There are two types of numerical values in R.

Integers are numbers without a decimal value. To explicitly denote that an object is an integer, you need to place an L behind the actual number.

```
x1 <- 5L
typeof(x1)
```

```
## [1] "integer"
```

Doubles are numbers with (a) decimal value(s).

```
x2 <- 5.25
typeof(x2)
```

```
## [1] "double"
```

Both x1 and x2 are numeric objects.

```
is.numeric(x2)
```

```
## [1] TRUE
```

Character Strings

A string in R simply denotes a series of characters. Importantly, numbers (or any kind of symbol) can also be part of a string. To denote a character string in R, it is simply wrapped in quotation marks.

```
string1 <- "University of Mannheim" # example of a string
class(string1)
```

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

```
string2 <- "5" # still a string, not an integer as we use quotation marks
class(string2)
```

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

```
string3 <- "TRUE" # still a string
class(string3)
```

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

Factors

Factors are data types that consist of a pre-defined set of distinct categories, which are called levels in R.

- other than numeric objects, factors do not assume their values are continuous (e.g. ordinal, nominal)
- other than character strings, each value of a factor has to correspond to one of the pre-defined levels, while character strings can hold virtually any information

This is an example of a valid factor operation:

```
parties <- factor(levels=c("SPD", "CDU", "Greens", "FDP", "AfD", "Left")) # create an empty factor
parties[1:3] <- c("SPD", "Greens", "FDP") # fill with three values
print(parties)
```

```
## [1] SPD    Greens FDP
## Levels: SPD CDU Greens FDP AfD Left
```

This is an example of an invalid factor operation:

```
parties[4] <- "CSU" # add invalid value
```

```
## Warning in `[<-.factor`(`*tmp*`, 4, value = "CSU"): ungültiges
## Faktorniveau, NA erzeugt
```

Boolean Values

Boolean (or logical) values are either *TRUE* or *FALSE* values. You can produce these values by making logical requests on your data.

```
(7 - 3) > 1
```

```
## [1] TRUE
```

Of course, logical values can be stored in objects as well.

```
bool <- 2 > 1
print(bool)
```

```
## [1] TRUE
```

Operators in R

There are quite a few operators that you can use.

Arithmetic Operators

- + Addition
- - Subtraction
- * Multiplication
- / Division
- ^ Exponentiation
- %% Modulus (remainder from division)

Relational Operators

- < Less than
- > Greater than
- <= Less than or equal to
- >= Greater than or equal to
- == Equal to
- != Not equal to

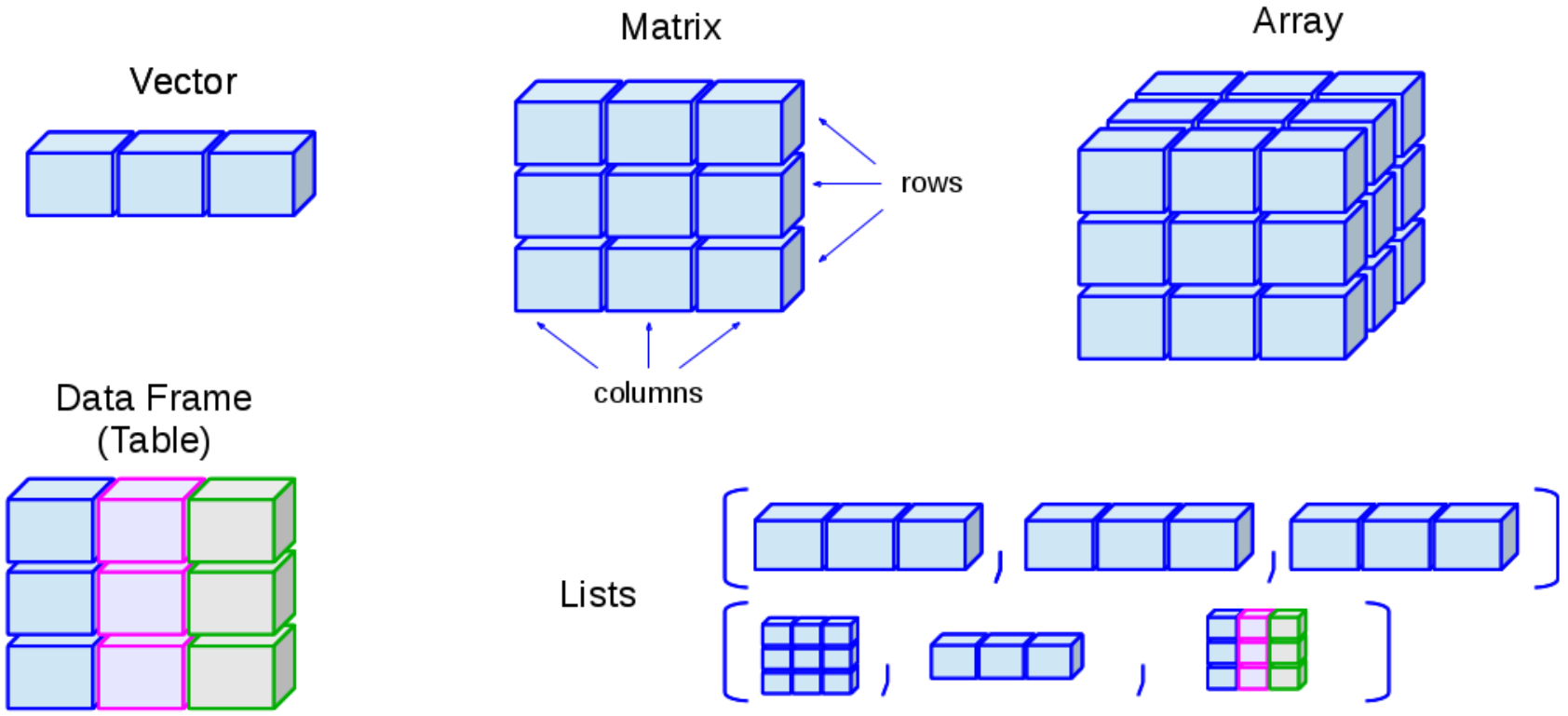
Logical Operators

- & And
- | Or

Data Formats

Data Formats

Individual objects can be stored in different data types, or containers.



Source: https://devopedia.org/r-data-structures.

Data Format: Vector

Vectors are built by combining (or "concatenating") individual objects with c().

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

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

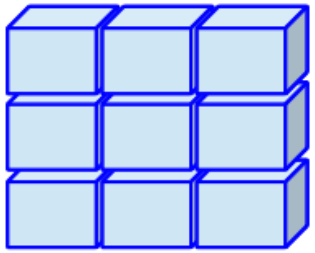
```
character_vector <- c("Austria", "England", "Brazil", "Germany")
print(character_vector)
```

```
## [1] "Austria" "England" "Brazil"  "Germany"
```

```
logical_vector <- c(TRUE, FALSE, FALSE, TRUE)
print(logical_vector)
```

```
## [1]  TRUE FALSE FALSE  TRUE
```

Data Format: Matrix

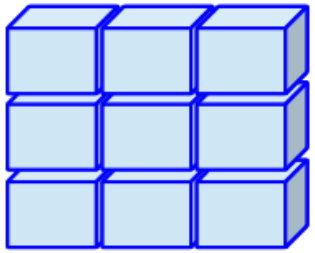


- Matrices are the most genuine rectangular data format in R.
- A matrix consists of n rows and k columns and hence has the dimensionality n x k .

```
matrix_example <- matrix(1:20, nrow = 4, ncol = 5) # create numeric matrix
print(matrix_example)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
```

Data Format: Matrix



- Matrices are the most genuine rectangular data format in R.
- A matrix consists of n rows and k columns and hence has the dimensionality n x k .

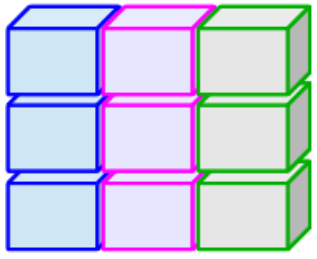
Let's check the dimensionality of this matrix.

```
dim(matrix_example)
```

```
## [1] 4 5
```

- *Note:* You cannot store multiple data types (e.g. character strings and numeric values in the same matrix or vector.)
- If you attempt this, all of your entries will be converted to character values.

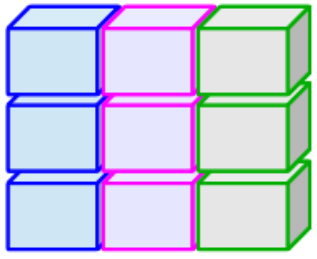
Data Format: Data Frame



- Data frames are R's equivalent of data sets as you know them from Stata, SPSS or other programs
- Data frames also are rectangular of dimensionality n rows x k columns
- *Note:* Data frames store variables of different classes.

```
df_example <-
  data.frame(
    country = c("Austria", "England", "Brazil", "Germany"),
    capital = c("Vienna", "London", "Brasília", "Berlin"),
    elo = c(1761, 1938, 2166, 1988)
  )
```

Data Format: Data Frame

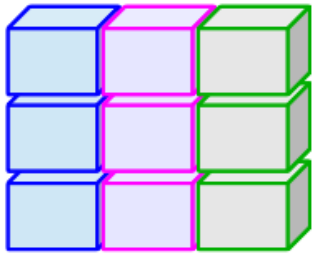


- Data frames are R's equivalent of data sets as you know them from Stata, SPSS or other programs
- Data frames also are rectangular of dimensionality n rows x k columns
- *Note:* Data frames store variables of different classes.

```
print(df_example)
```

```
##   country  capital  elo
## 1 Austria   Vienna 1761
## 2 England   London 1938
## 3  Brazil Brasília 2166
## 4 Germany   Berlin 1988
```

Data Format: Data Frame

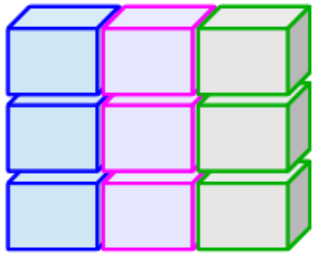


- Data frames are R's equivalent of data sets as you know them from Stata, SPSS or other programs
- Data frames also are rectangular of dimensionality n rows x k columns
- *Note:* Data frames store variables of different classes.

```
dim(df_example) # dimensions of the data frame
```

```
## [1] 4 3
```

Data Format: Data Frame

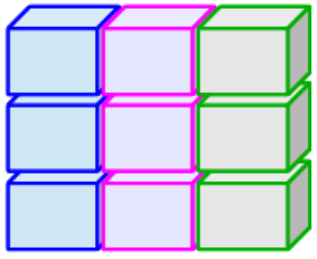


- Data frames are R's equivalent of data sets as you know them from Stata, SPSS or other programs
- Data frames also are rectangular of dimensionality n rows x k columns
- *Note:* Data frames store variables of different classes.

```
nrow(df_example) # number of observations
```

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

Data Format: Data Frame



- Data frames are R's equivalent of data sets as you know them from Stata, SPSS or other programs
- Data frames also are rectangular of dimensionality n rows x k columns
- *Note:* Data frames store variables of different classes.

```
ncol(df_example) # number of variables
```

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

Data Format: List

Essentially, lists just bind any number of objects together, even if these are of different classes.

```
list_example <-
  list(numeric_vector,
       character_vector,
       logical_vector,
       matrix_example,
       df_example)
```

Data Format: List

```
print(list_example)
```

```
## [[1]]
## [1] 1 2 3 4 5
##
## [[2]]
## [1] "Austria" "England" "Brazil"  "Germany"
##
## [[3]]
## [1]  TRUE FALSE FALSE  TRUE
##
## [[4]]
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    5    9   13   17
## [2,]    2    6   10   14   18
## [3,]    3    7   11   15   19
## [4,]    4    8   12   16   20
##
## [[5]]
##   country  capital  elo
## 1 Austria   Vienna 1761
## 2 England   London 1938
## 3  Brazil Brasília 2166
## 4 Germany   Berlin 1988
```

References

Parts of this course are inspired by the following resources:

- Wickham, Hadley and Garrett Grolemund, 2017. *R for Data Science - Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly.
- Bahnsen, Oke and Guido Ropers, 2022. *Introduction to R for Quantitative Social Science*. Course held as part of the GESIS Workshop Series.
- Breuer, Johannes and Stefan Jünger, 2021. *Introduction to R for Data Analysis*. Course held as part of the GESIS Summer School in Survey Methodology.
- Teaching material developed by Verena Kunz, David Weyrauch, Oliver Rittmann and Viktoriia Semenova.