Introduction to R

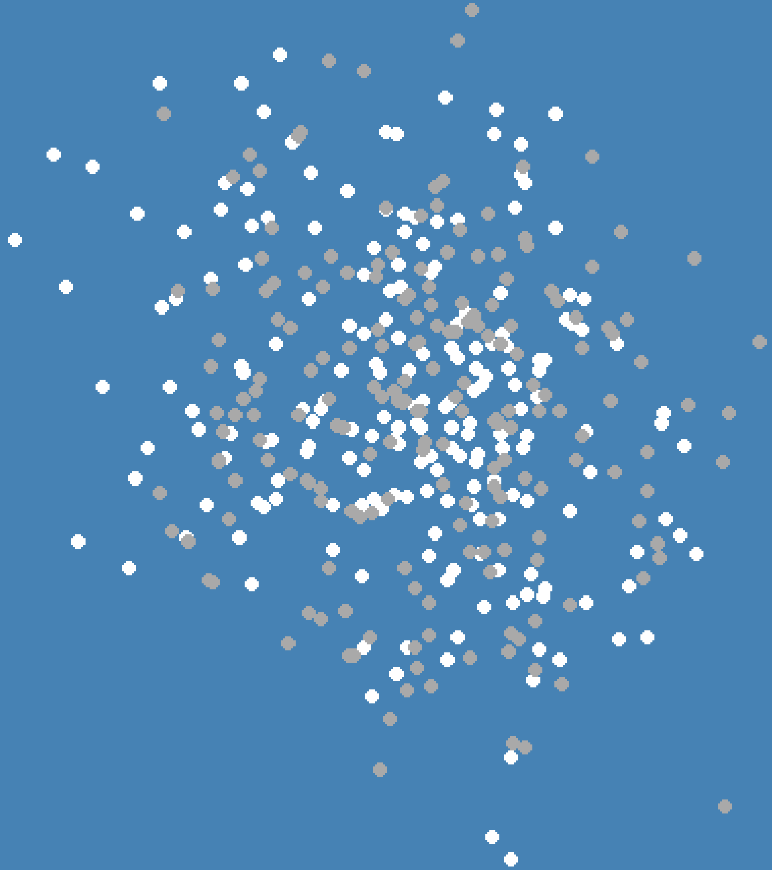
2.1 Data Wrangling - Tidyverse Philosophy

What is dplyr? Structure of a Function Call, Pipe Operator %>%

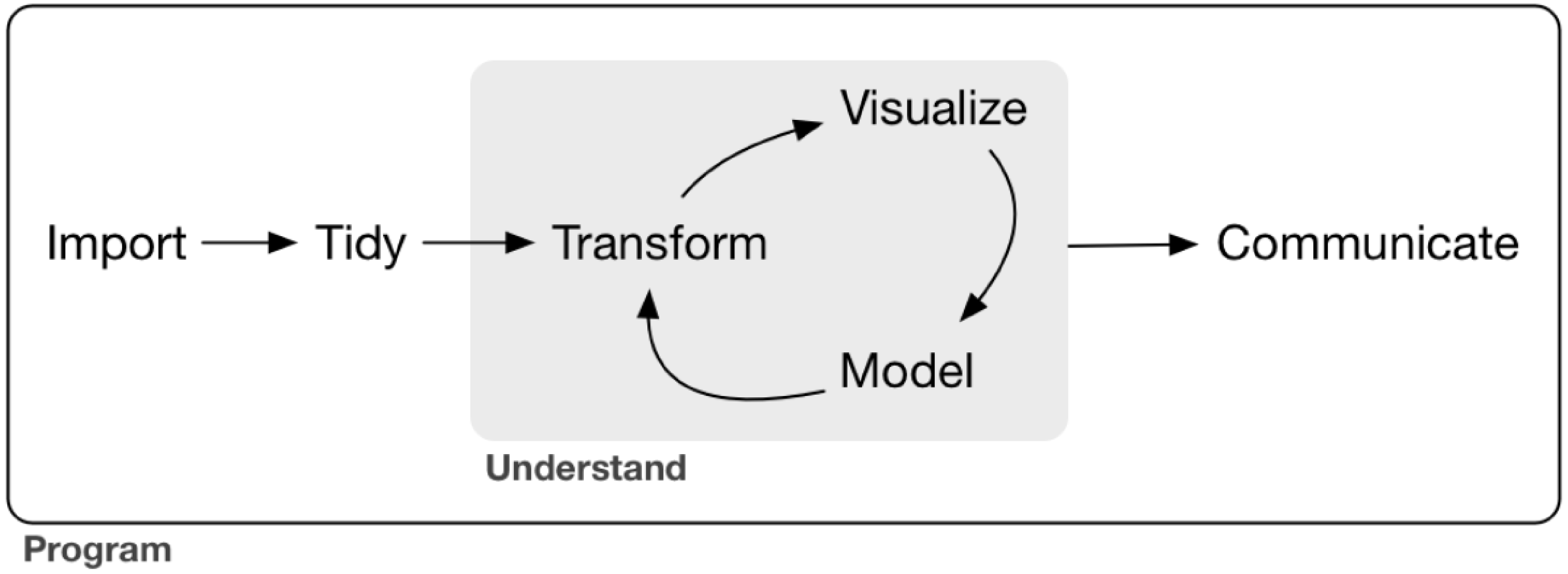
Lion Behrens, M.Sc.

UNIVERSITY OF MANNHEIM

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

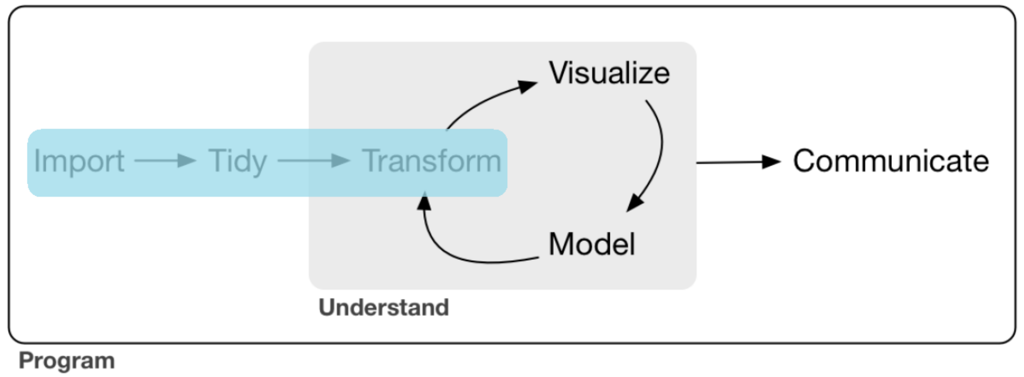


Data Project Flow



Note: Figure from Wickham and Grolemund, 2017. R For Data Science. Sebastopol: O' REILLY.

What is Data Wrangling?



Note: Figure adapted from Wickham and Grolemund, 2017. R For Data Science. Sebastopol: O' REILLY.

Data wrangling is the process of getting your data into shape so that you can it analyze, visualize and model it. The most common steps are:

- importing data of various formats into R
- renaming variables
- selecting a subset of variables or filtering out a subset of cases
- recoding variables (defining missing values)
- creating new variables as transformations/mutations of existing variables

Data wrangling (or feature engineering) is one of the most important and time-consuming tasks in social science research.

Tidyverse

R packages for data science

The tidyverse is an opinionated collection of R packages designed for data science.

All packages share an underlying

- design philosophy
- grammar
- data structures

Install the complete tidyverse with

```r
install.packages("tidyverse")
```

Dplyr - A Language for Data Manipulation

A collection of key function calls provides the verbs for dplyr's language of data manipulation.

- select() subsets variables based on their names
- filter() subsets observations (rows) based on their values
- relocate() and arrange() re-order variables and observations
- mutate() generates new variables from existing variables
- summarize() reduces multiple values to single summaries
- group_by() performs operations by group
- left_join(), right_join(), full_join() (etc.) merge several data sets

... and there are many more!

dplyr - Structur of a Function Call

Example Data

```
library(readstata13)
ess10 <- read.dta13("dat/ess10.dta")
```

```
glimpse(ess10)
```

```
## Rows: 18,060
## Columns: 6
## $ name      [3m [38;5;246m<chr> [39m [23m "ESS10e01_2", "ESS10e01_2", "ESS10e01_2", "ESS10e0…
## $ essround  [3m [38;5;246m<dbl> [39m [23m 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10…
## $ edition   [3m [38;5;246m<chr> [39m [23m "1.2", "1.2", "1.2", "1.2", "1.2", "1.2", "1.2", "…
## $ proddate  [3m [38;5;246m<chr> [39m [23m "28.06.2022", "28.06.2022", "28.06.2022", "28.06.2…
## $ idno      [3m [38;5;246m<dbl> [39m [23m 10002, 10006, 10009, 10024, 10027, 10048, 10053, 1…
## $ cntry     [3m [38;5;246m<chr> [39m [23m "BG", "BG", "BG", "BG", "BG", "BG", "BG", "BG", "B…
```

dplyr - Structur of a Function Call

Let's say we only want to extract respondents from the country of Hungary.

```
filter(
  ess10,
  cntry == "HU"
)
```

- start with stating the verb (function): filter()

dplyr - Structur of a Function Call

Let's say we only want to extract respondents from the country of Hungary.

```
filter(
  ess10,
  cntry == "HU"
)
```

- start with stating the verb (function): filter()
- the first argument is the data frame that we are working with: ess10

dplyr - Structur of a Function Call

Let's say we only want to extract respondents from the country of Hungary.

```r
filter(
  ess10,
  cntry == "HU"
)
```

- start with stating the verb (function): filter()
- the first argument is the data frame that we are working with: ess10
- the second argument is the observations that we want to select: cntry == "HU"

dplyr - Structur of a Function Call

Let's say we only want to extract respondents from the country of Hungary.

```
filter(
  ess10,
  cntry == "HU"
)
```

- start with stating the verb (function): filter()
- the first argument is the data frame that we are working with: ess10
- the second argument is the observations that we want to select: cntry == "HU"
- we now have reduced our data frame to the $n = 1,849$ respondents from Hungary

```
## Rows: 1,849
## Columns: 6
## $ name      [3m [38;5;246m<chr> [39m [23m "ESS10e01_2", "ESS10e01_2", "ESS10e01_2", "ESS10e0…
## $ essround  [3m [38;5;246m<dbl> [39m [23m 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10…
## $ edition   [3m [38;5;246m<chr> [39m [23m "1.2", "1.2", "1.2", "1.2", "1.2", "1.2", "1.2", "…
## $ proddate  [3m [38;5;246m<chr> [39m [23m "28.06.2022", "28.06.2022", "28.06.2022", "28.06.2…
## $ idno      [3m [38;5;246m<dbl> [39m [23m 10002, 10006, 10017, 10018, 10027, 10043, 10045, 1…
## $ cntry     [3m [38;5;246m<chr> [39m [23m "HU", "HU", "HU", "HU", "HU", "HU", "HU", "HU", "H…
```

Tidyverse - Pipe Operator %>%

What if we want information on respondents from Hungary but are only interested in extracting the design weight (dweight) and population weight (pweight) that is attached to each respondent?

- This means we want to conduct more than one operation on a data set
- Enter: The pipe operator %>%

```
ess10 %>%
  filter(cntry == "HU") %>%
  select(dweight, pweight)
```

- start with the data frame that we are working with

Tidyverse - Pipe Operator %>%

What if we want information on respondents from Hungary but are only interested in extracting the design weight (dweight) and population weight (pweight) that is attached to each respondent?

- This means we want to conduct more than one operation on a data set
- Enter: The pipe operator %>%

```r
ess10 %>%
  filter(cntry == "HU") %>%
  select(dweight, pweight)
```

- start with the data frame that we are working with
- the second line filters for the observations that we want to select: cntry == "HU"

Tidyverse - Pipe Operator %>%

What if we want information on respondents from Hungary but are only interested in extracting the design weight (dweight) and population weight (pweight) that is attached to each respondent?

- This means we want to conduct more than one operation on a data set
- Enter: The pipe operator %>%

```
ess10 %>%
  filter(cntry == "HU") %>%
  select(dweight, pweight)
```

- start with the data frame that we are working with
- the second line filters for the observations that we want to select: cntry == "HU"
- the third line selects the variables we are interested in: dweight, pweight

Note: The pipe operator %>% is used for passing information from one process to the other.

Tidyverse - Piping vs. Assigning

The pipe operator itself will not store the output in a new object.

```
# print the output, store nothing
ess10 %>%
  filter(cntry == "HU") %>%
  select(dweight, pweight)
```

```
# store the output, print nothing
weights_hu <- ess10 %>%
  filter(cntry == "HU") %>%
  select(dweight, pweight)
```

Tidyverse Data - Tibble

There is basically a new object type that exists in the tidyverse! Tibbles!

Tibble vs. Data Frames

- Tibbles are modernized versions of data frames
- Tibbles are designed to make working with data frames smarter
- print(df) only prints out first ten observations: output is tidier
- Additional metadata is displayed for each variable: output is more informative

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.