



Visualizing morphological data

Stefan Hartmann
HHU Düsseldorf



Visualizing (morphological) data

Stefan Hartmann
HHU Düsseldorf

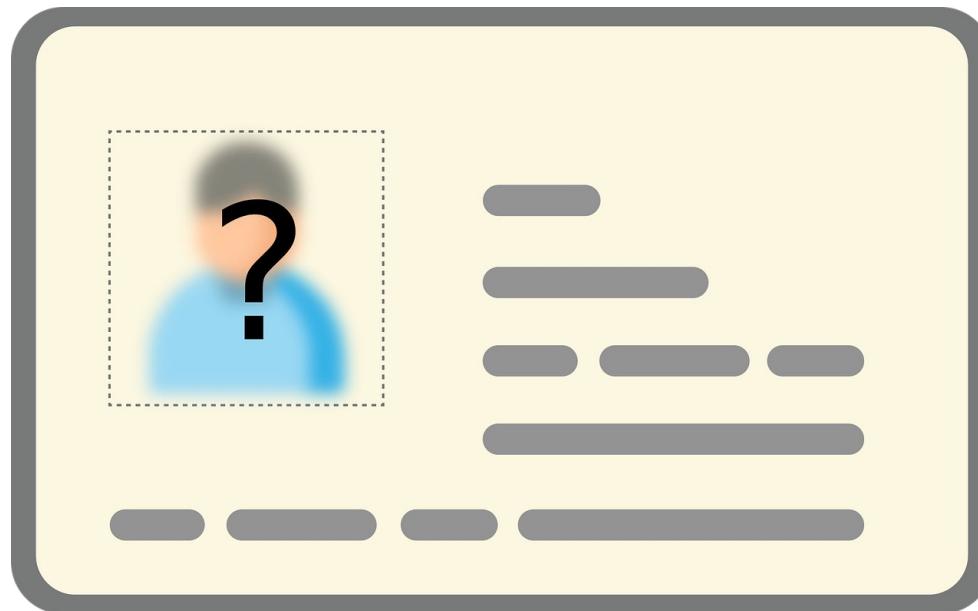
Plan for today

- Introduction
- Basics of data visualization
- Data wrangling in R
- Basics of ggplot2
- A first example graph
- Break
- Case studies and hands-on exercises
- Some (fairly random but perhaps useful) tips and tricks



Introduction

Who am I? Who are you?





Principles of data visualization

Why visualize?

- **For yourself**
 - Exploring your data
 - detecting outliers
 - checking assumptions of statistical tests or models (e.g. are the data normally distributed?)
 - etc.
- **For others**
 - Showing your findings in a clear and efficient way
 - Graphs tend to be more reader-friendly than tables...
 - and *much* more reader-friendly than long inline lists!

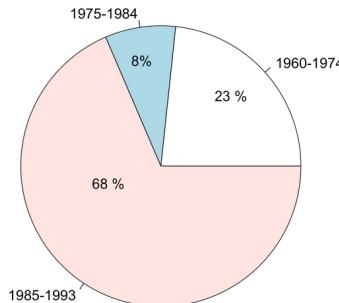
Choosing the "right" plot

- What kind of data are you dealing with?
- What is your research question?

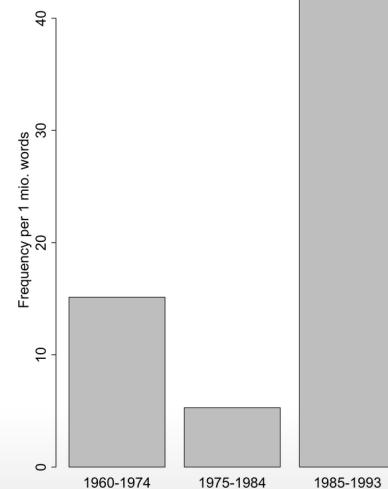
Data visualization

- What kind of variable are we dealing with here?
- Which visualization seems most appropriate to you?

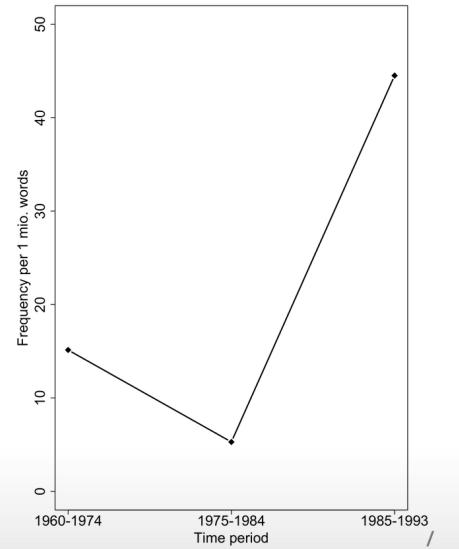
Frequency of the f word in the BNC



Frequency of the f-word in the BNC

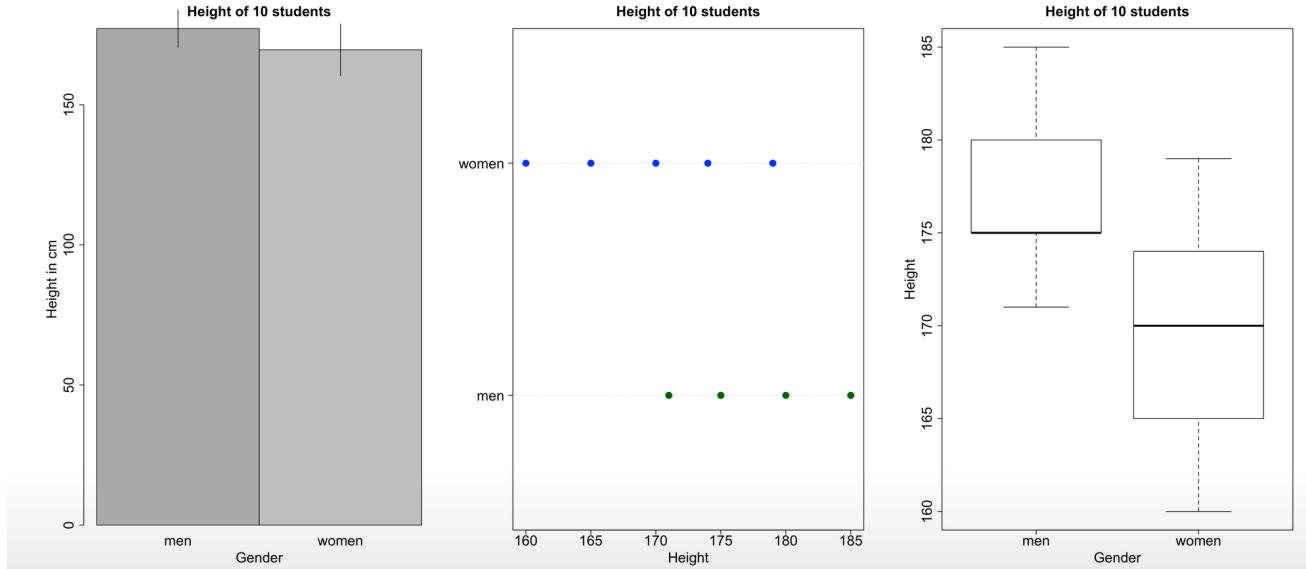


Frequency of the f-word in the BNC



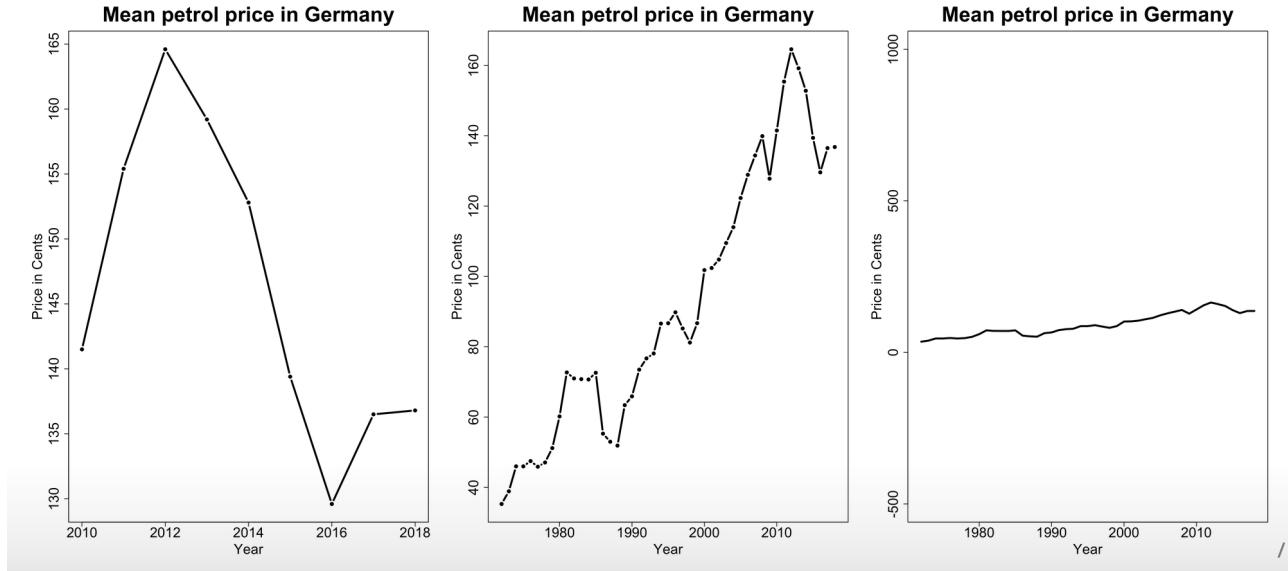
Data visualization

- What kind of variable are we dealing with here?
- Which visualization seems most appropriate to you?



Data visualization

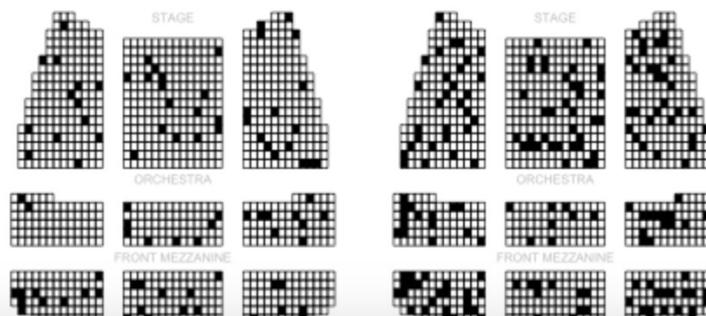
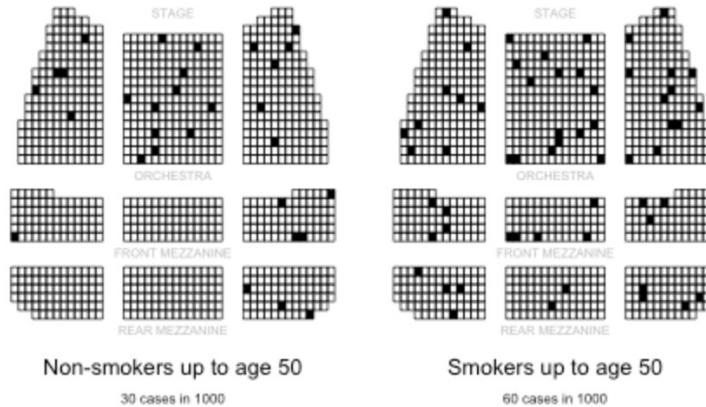
- What kind of variable are we dealing with here?
- Which visualization seems most appropriate to you?



Alternative visualizations

- from
<http://www.stubbornmule.net/2010/10/visualizing-smoking-risk/>
- "Risk Characterization Theatre"
from Rifkin & Bouwer (2007)

stubbornmule.net



The plot as a metaphor

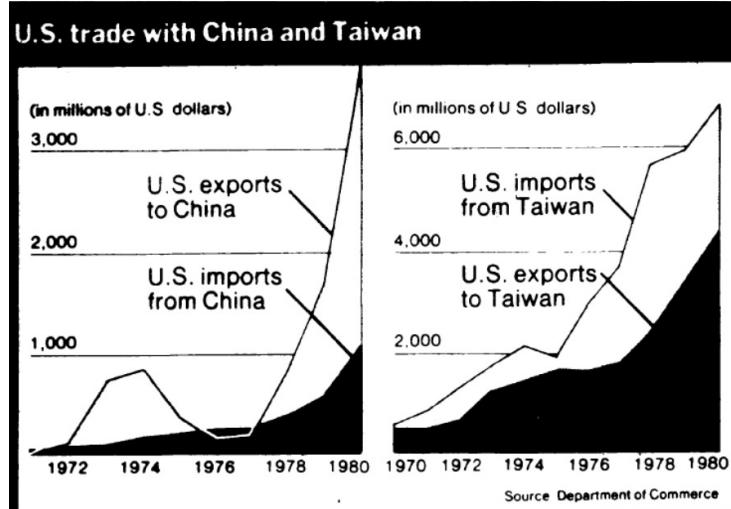
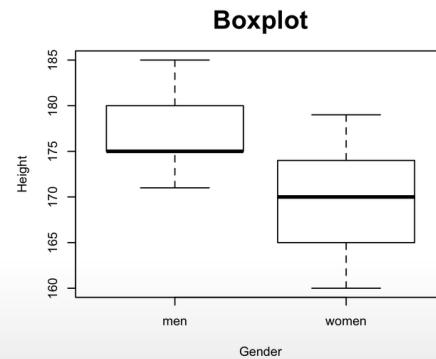
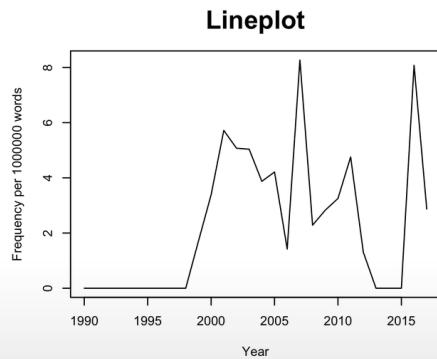
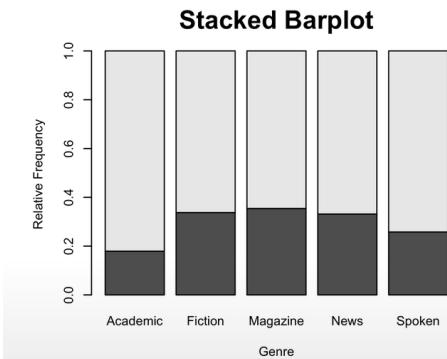
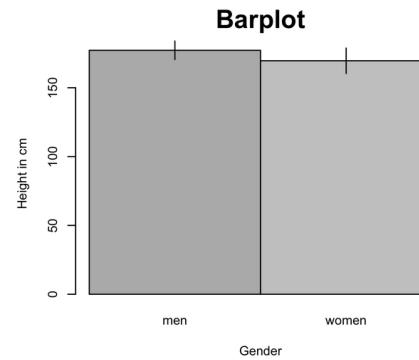
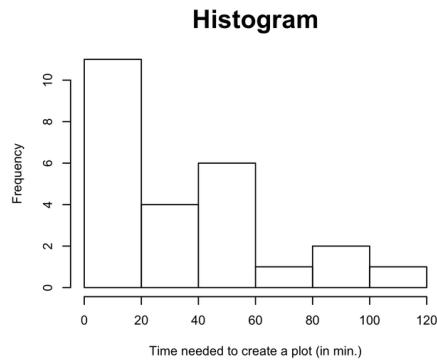
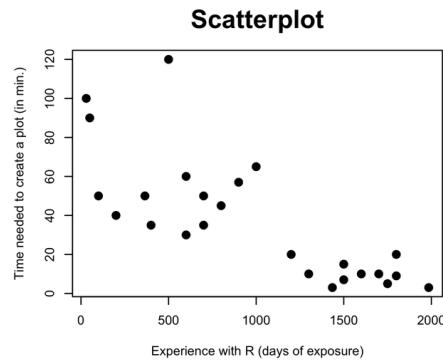


Figure 7. Reversing the metaphor in mid-graph while changing scales on both axes (© June 14, 1981, The New York Times).

"The essence of a graphic display is that a set of numbers having both magnitudes and an order are represented by an appropriate visual metaphor - the magnitude and order of the metaphorical representation match the numbers." (Wainer 1984: 139)

Plot types



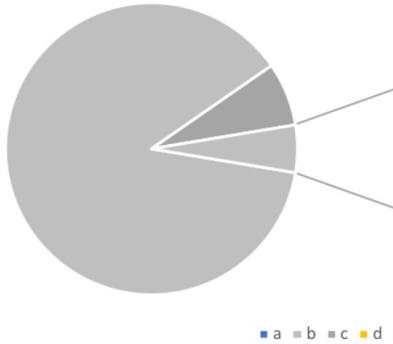
Best practice for reporting & displaying data

- Most importantly: **Know your data!**
- When reporting percentages, also report the denominator (i.e. the size of your sample)
- Example: "50% of academics are alcoholics" - it makes a difference whether your sample size is 2 or 2,000.
- When reporting comparisons of absolute frequencies, double-check if your samples are comparable.
- Example: "255 women agree that cats are adorable, but only 5 men." - it makes a difference whether your sample consists of 300 women and 300 men or of 300 women and 10 men.
- When reporting means, also report standard deviations.

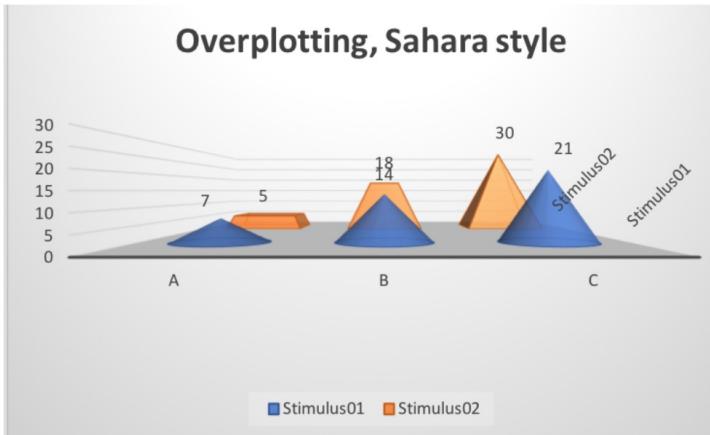
- Show the data
- Avoid distorting the data
- Keep "Ink-to-data ratio" as low as possible
- Use meaningful x and y labels
- Avoid overplotting (e.g. 3-dimensional plots when only 2 dimensions are displayed)

Beware of overplotting

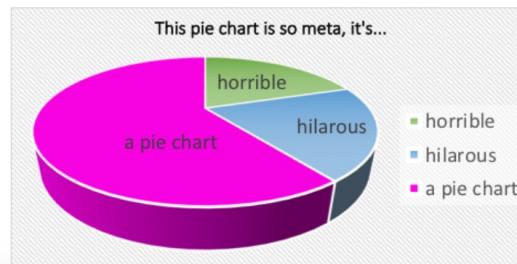
Overplotting, Star Trek style



Overplotting, Sahara style



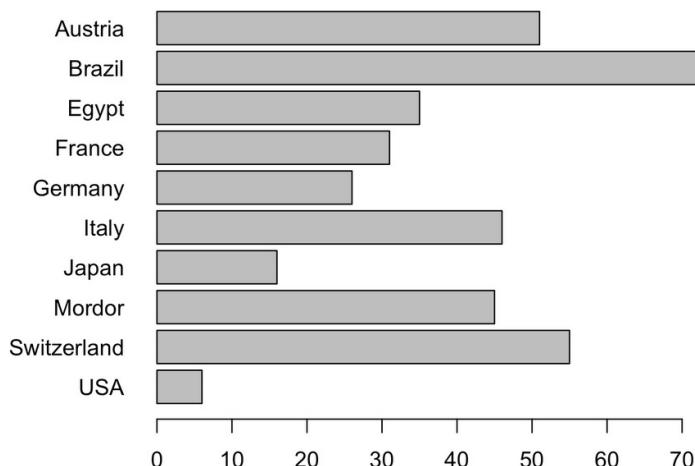
This pie chart is so meta, it's...



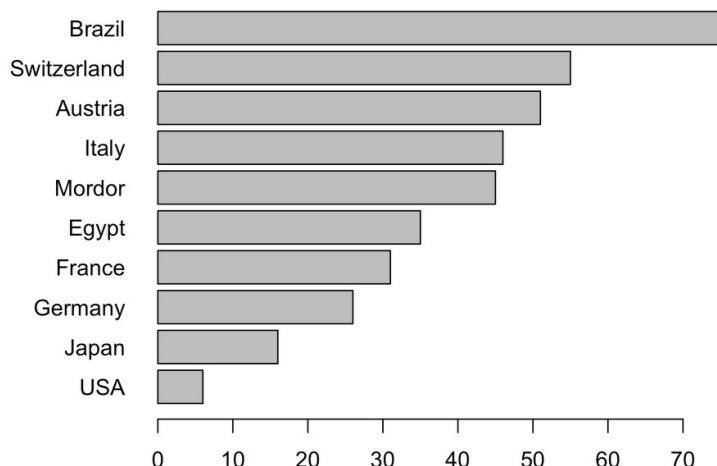
Further tips

- If there is no natural order to your data, order them by value

Some random stuff

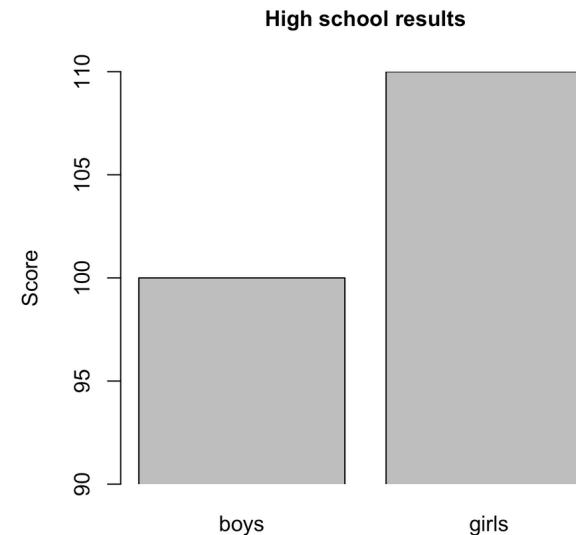
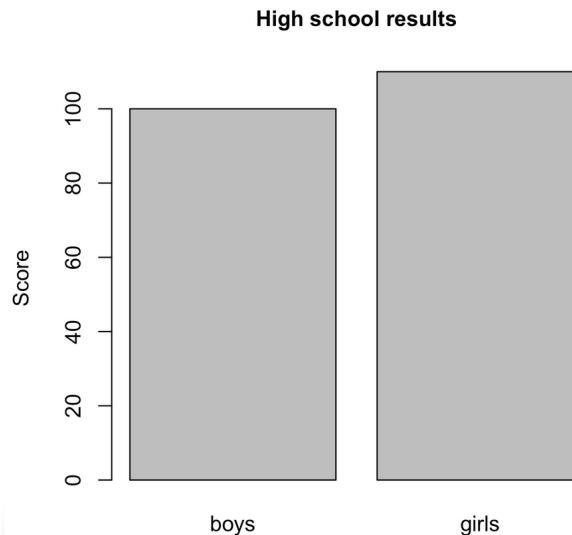


Some random stuff



Further tips

- Don't cut the y axis unless there are good conceptual reasons to do so.





Basics of R



programming language



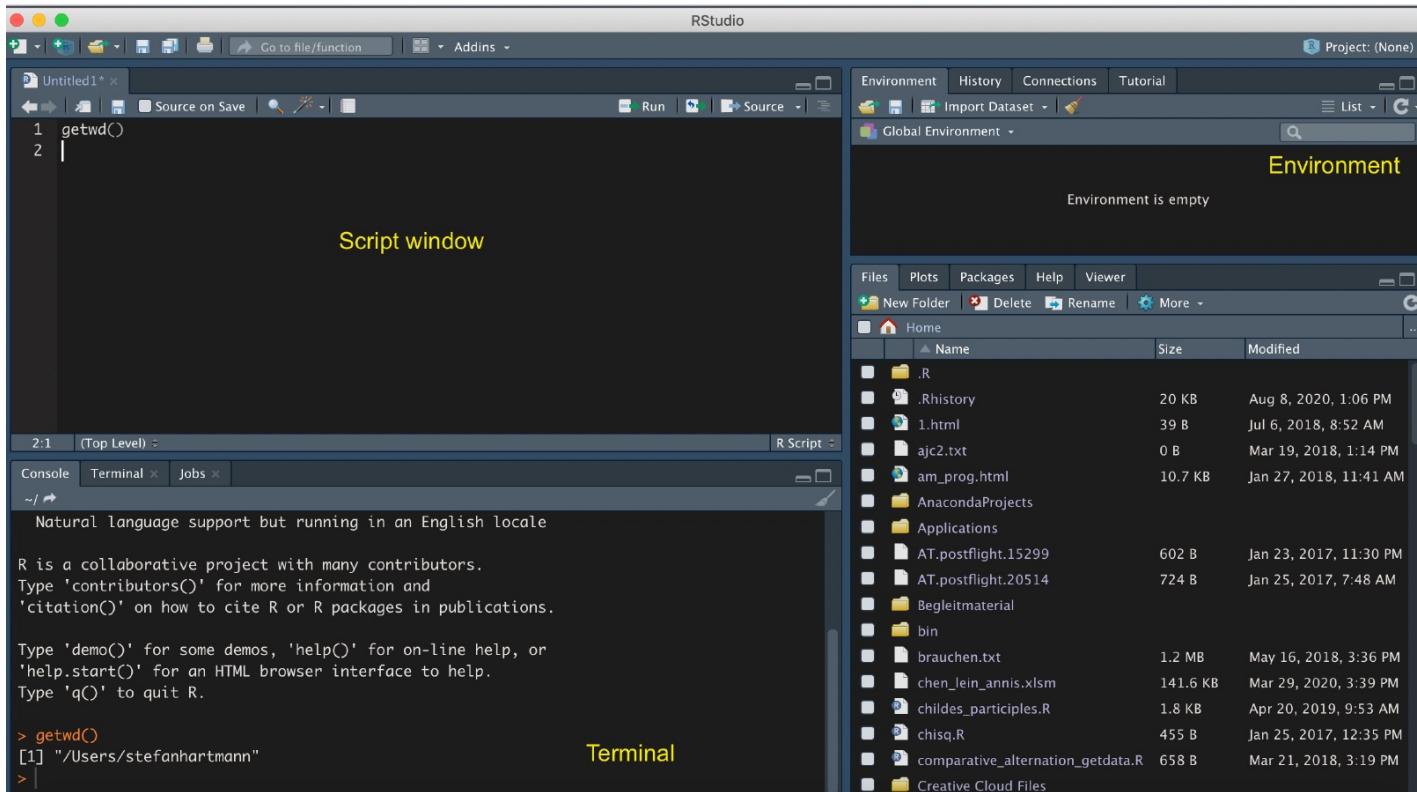
Integrated Developing Environment

- R: <https://cran.r-project.org/>
- RStudio: <https://posit.co/download/rstudio-desktop/>

Things to know about R

- R is, first and foremost, a **programming language**
- RStudio is an **Integrated Developer Environment (IDE)** that provides a user-friendly "window" to R
- There are two "dialects" of R:
 - base R,
 - the "**tidyverse**" syntax

The RStudio Interface



R as a calculator

- Please try:

2 + 2

sqrt(25)

5 * 6

Important preliminaries

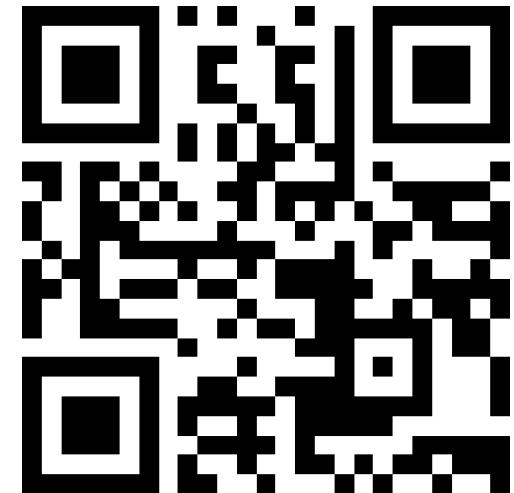
- Ground rule: R is "stupid".
- Unlike Google or ChatGPT, R doesn't try to correct our errors.
- Among other things, this means that...
 - it is case sensitive
 - in some contexts, it makes a difference if you use scarequotes or not
 - When it has assigned a data type to an object, it will stick to it: if it thinks that "26" is a set of strings (characters) and not a number, then it won't do any calculations with it, unless you coerce it to treat it as a number using a command like `as.numeric()`.
- See <https://www.burns-stat.com/documents/books/the-r-inferno/> for entertaining examples of R users' common errors.



Data wrangling in R

- Data available at
<https://tinyurl.com/evalmogithub>
- If you want you can clone the entire repository by entering in your terminal:

```
git clone https://github.com/empirical-  
linguistics/dataviz-evalmo
```



Types of data in R

- **character**: "a", "swc"
- **double**: 2, 15.5
- **integer**: 2L (the L tells R R to store this as an integer)
- **logical**: TRUE, FALSE
- **complex**: 1+4i (complex numbers with real and imaginary parts)

What is the Tidyverse?



- family of packages developed by RStudio/Posit
- implement an own "dialect" of R
- still fully compatible with base R, but adding more syntax possibilities

Pros and cons of the Tidyverse

■ Pro:

- syntax is arguably more intuitive and cleaner once you get used to it (especially "piping"!)
- offers really neat functions for data wrangling
- ggplot2 offers great possibilities for visualization
- it is very widely used and many replies in forums rely on it

■ Cons:

- it is under very active development, some functions become deprecated or change their names → problems for reproducibility
- can be a bit patronizing (although this can sometimes be a good thing)
- some things are ridiculously counterintuitive

Steps

- Structuring
- Cleaning
- Validating
- Handling missing data (→ imputation)
- Merging data from different sources
- Transforming variables (can also be seen as already part of the statistical analysis*)

*In fact, all these steps are, strictly speaking, part of the analysis!

Reading in dataframes

- Commands for reading in spreadsheets:

CSV:

```
readr::read_csv()
```

TSV etc.:

```
readr::read_delim(delim="\t", ...)
```

Excel:

```
readxl::read_xlsx()
```

plain text:

```
readr::read_lines()
```

large files:

```
vroom::vroom or datatable::data.table()
```

the stuff before :: refers to the **packages** to which these functions belong and can be omitted if you have loaded the packages via `library()` or `require()`. `readr()` is part of the tidyverse and is automatically loaded when you run `library(tidyverse)`

"Tidy data"

- Use "tidy data": One variable per column
- One observation per row

Hi there this is my cool fancy Excel spreadsheet with the results of my psycholinguistic experiment!!!			
Subject ID	190808		
	Stimulus 1	response time	980
		answer	yes
	Stimulus 2	response time	1080
			no
		metadata	age 28
			gender female

Subject ID	Stimulus	Response	Answer	Age	gender
190808	1	980	yes	28	female
190808	2	1080	no	28	female
809653	1	830	no	45	male
809653	2	420	yes	45	male
207436	1	320	yes	25	male
207436	2	954	no	25	male
185848	1	430	yes	30	female
185848	2	850	no	30	female
947379	1	530	yes	84	female
947379	2	1045	no	84	female
374957	1	890	yes	18	female
374957	2	1150	no	18	female

"Long" vs. "wide" format

Subject ID	Time_Stim	Time_Stimulus	Answer_S_1	Answer_S_2	Age	gender	Subject ID	Stimulus	Response	Answer	Age	gender
190808	980	1080	yes	no	28	female	190808	1	980	yes	28	female
809653	830	420	no	yes	45	male	190808	2	1080	no	28	female
207436	320	954	yes	no	25	male	809653	1	830	no	45	male
185848	430	850	yes	no	30	female	809653	2	420	yes	45	male
947379	530	1045	yes	no	84	female	207436	1	320	yes	25	male
374957	890	1150	yes	no	18	female	207436	2	954	no	25	male
							185848	1	430	yes	30	female
							185848	2	850	no	30	female
							947379	1	530	yes	84	female
							947379	2	1045	no	84	female
							374957	1	890	yes	18	female
							374957	2	1150	no	18	female

wide

long

From long to wide format

- If necessary, data can be converted from "long" to "wide" format using the `pivot_wider()` and `pivot_longer()` functions from the `tidyverse` family of packages.
- (They may not be 100% intuitive but with a bit of trial and error they work really well!)

"Long" vs. "wide" format

Hands-on example...

- Read the dataset fake_RT_data.xlsx using `readxl::read_xlsx`
- Use the `pivot_longer()` command to transform the dataset in a longer format:

	Participant	Condition	RT
	<i><chr></i>	<i><chr></i>	<i><dbl></i>
1	A	A	6.25
2	A	B	9.14
3	B	A	7.34
4	B	B	10.1
5	C	A	8.54
6	C	B	11.1

Merging multiple dataframes

- In many cases we want to **combine** dataframes, e.g. because one dataframe contains metadata pertaining to the other dataframe
- Example: We have one file with corpus data, with one column specifying the author, and one file with birth and death dates of each author crawled from the German National Library and/or Wikipedia
- We can combine the dataframes using dplyr's *join()* commands.

Dealing with dataframes

Hands-on task

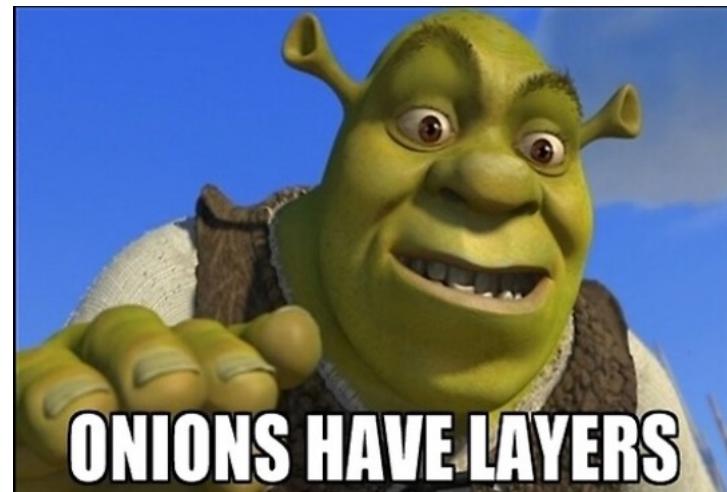
- ...



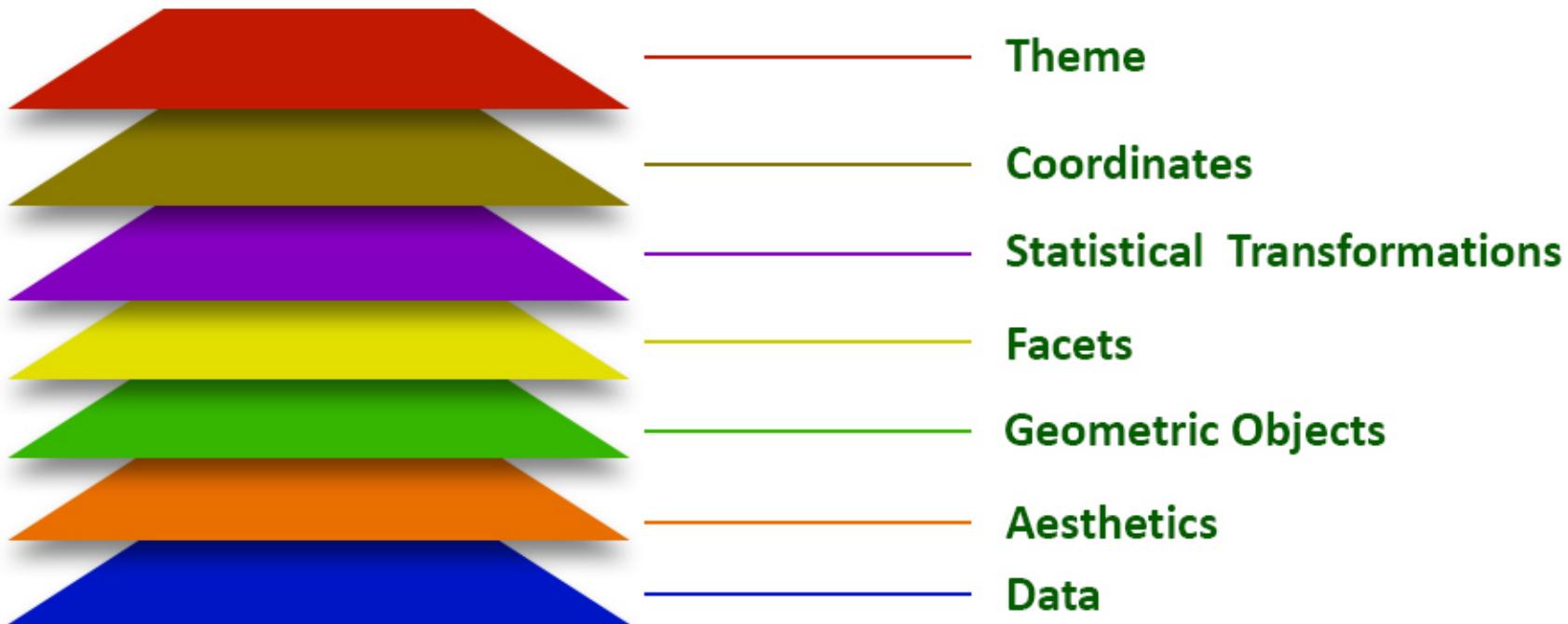
Basics of ggplot

The syntax of ggplot

- A ggplot consists of three components
 - the **data**,
 - a set of **aesthetic mappings**,
 - at least one **layer** (usually created with the `geom` function) describing how to render each observation.



Main Components of the Grammar of Graphics



Creating a ggplot

- A ggplot is built **layer by layer**
- We start out with the **data** and the **aesthetic mappings**
- Basic syntax:

```
p <- ggplot(data, aes(x = ..., y = ..., group = ...))
```

- We specify the **geometric objects** to plot, e.g.

```
p <- p + geom_line() # lineplot
```

- Optional: We customize the **scales** (position, color, size) and/or change the **theme** of the plot

```
p <- p + scale_color_grey() + theme_minimal()
```

A basic ggplot

■ First step: generating fake data

Try to create a dataframe with two columns x and y, with x containing the numbers from 1 to 100 and y 100 normally-distributed random numbers (`rnorm(100)`).

■ Second step: visualizing the data

Plot x against y using base R first and then using ggplot.

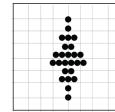
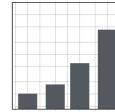
■ Third step: customizing the plot

Play around with different scale configurations and themes.

Different Geoms (Plot Type) in ggplot2

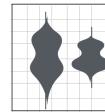
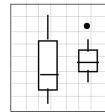
Two Variables (X, Y)

- Discrete X, continuous Y
- Visualise distribution of Y with respect to X



geom_col()
- heights of bars represent values

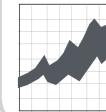
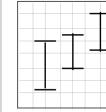
geom_jitter()
- adds jitter to prevent overplotting



geom_boxplot()
- summarise distribution using median, hinges and whiskers

geom_violin()
- mirrored density plot (smoothed distribution)

Visualising Errors and Uncertainties

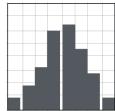


geom_errorbar()
- uncertainty in continuous Y against discrete X

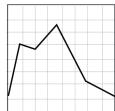
geom_ribbon()
- uncertainty in continuous Y against continuous X

One Variable (X)

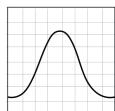
- Continuous X
- Visualise distribution of X



geom_histogram()
- divide X into bins and count no. observation



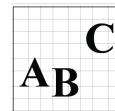
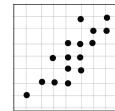
geom_freqpoly()
- display counts with lines
- able to overlay multiple distributions



geom_density()
- smoothed version of the histogram

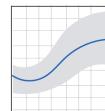
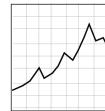
Two Variables (X, Y)

- Continuous X, continuous Y
- Visualise relationship between X and Y



geom_point()
- scatterplot of X vs Y

geom_text()
- labelling data points

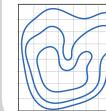


geom_line()
- connect data points, ordered by X
- alt: geom_path()

geom_smooth()
- add smoothed curve
- helps to see trends

Contour Plots

- Representing a third dimension using contours



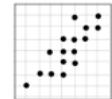
geom_density2d()
- contour represents 2D density of data points

geom_contour()
- contour represents z-axis value / height

Which plot types for which purpose?

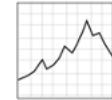
■ Scatterplots

- show / explore correlations between two variables
- metric data on both the x- and the y-axis



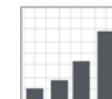
■ Line plots

- Lineplots are useful for showing e.g. change over time
- Count variable on y-axis, (at least) ordinal variable on x-axis



■ Barplots

- useful to show counts of categorical variables (e.g. number of men vs. number of women in parliament)...
- summary statistics (usually: means) of metric variables across different categories (e.g. mean height of humans vs. Klingons)



Beeswarm plots

- Packages *beeswarm* and *ggbeeswarm*
- can be combined with violin or boxplots

Interactive plots

- e.g. *(gg)plotly* package
- and *shinyplots* (shinyplots.io)

Some tips and tricks

- Cheat sheets
- Code snippets



**Hands-on
examples...**

ung-nominalizations

- The file *ungbaby.csv* contains a concordance of *ung*-nominalizations from a subset of the German Text Archive
- The file *dtababy_with_author_data.csv* contains metadata about the authors.

Find out how the two files can be merged in such a way that the metadata about author and texttype are added to each row, based on the document ID.

ung-nominalizations – Morphological productivity

- Use the resulting dataframe to calculate the **potential productivity** (e.g. Baayen 2009) of the pattern for each text type and for each decade.

- To do so,
 - add a decade column based on the existing column "Year" – replace the last digit by 5 using `gsub()` (hint: regular expression for the last character is `. $`);
 - use the **tidyverse** `summarise()` command.

Task 1c

ung-nominalizations – Visualizing productivity

- Now **visualize** the productivity development. Select the plot time that seems most appropriate to you and plot one panel for each of the three text types.

Task 2

-ish in the BNC

- Load the dataset *ish_BNC.csv.gz*
- Plot the development of the pattern's type-token ratio and potential productivity over time.