University of Zurich[UZH]

Department of Geography

GEO 812
Getting started with R for Spatial Analysis

Session 1: Data exploration

Peter Ranacher, Nico Neureiter

September 2023

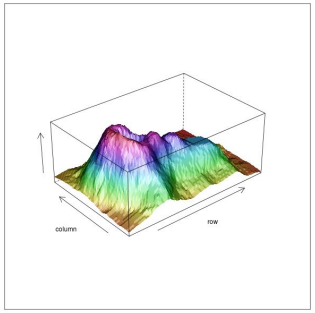
Learning objectives

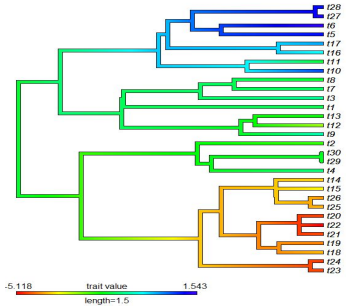
You are able to

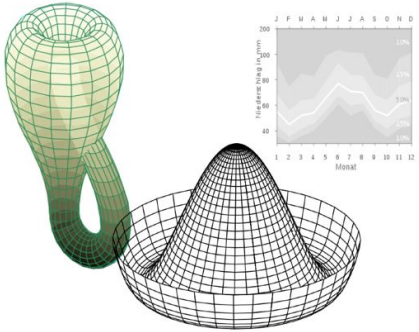
- perform basic operations in R
- transform and explore data in the `tidyverse`
- visualize data with `ggplot`

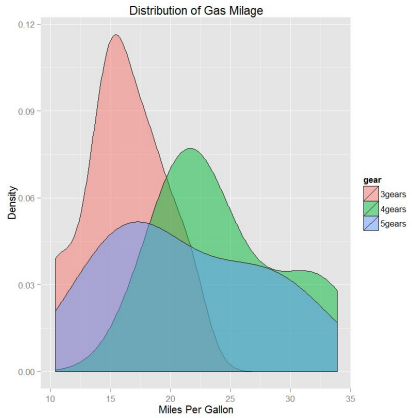
What is R?

R is a programming language and software environment for statistical analysis. R is widely used in science, since it is open, independent and **free.**

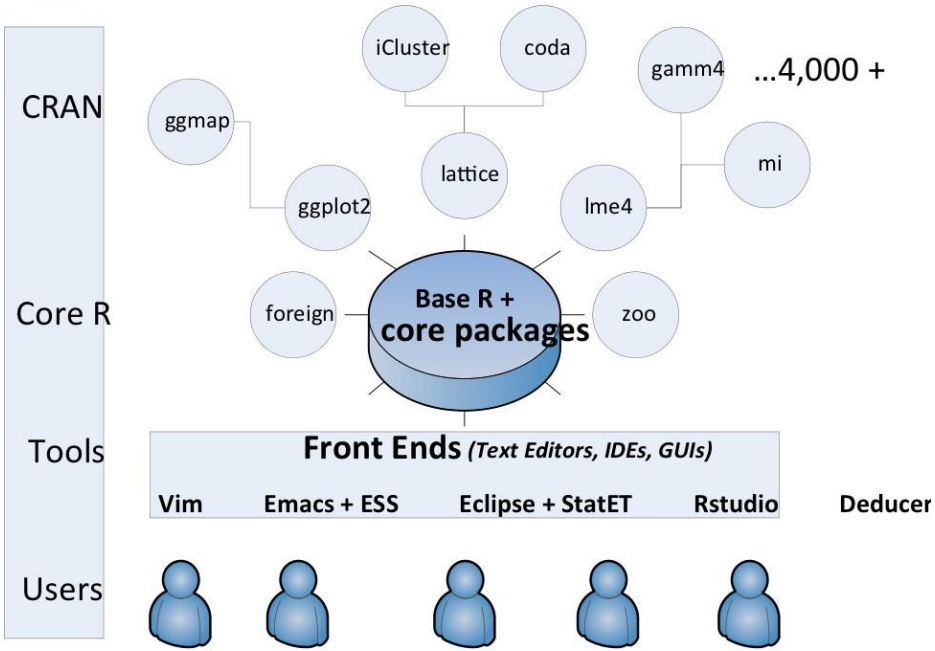


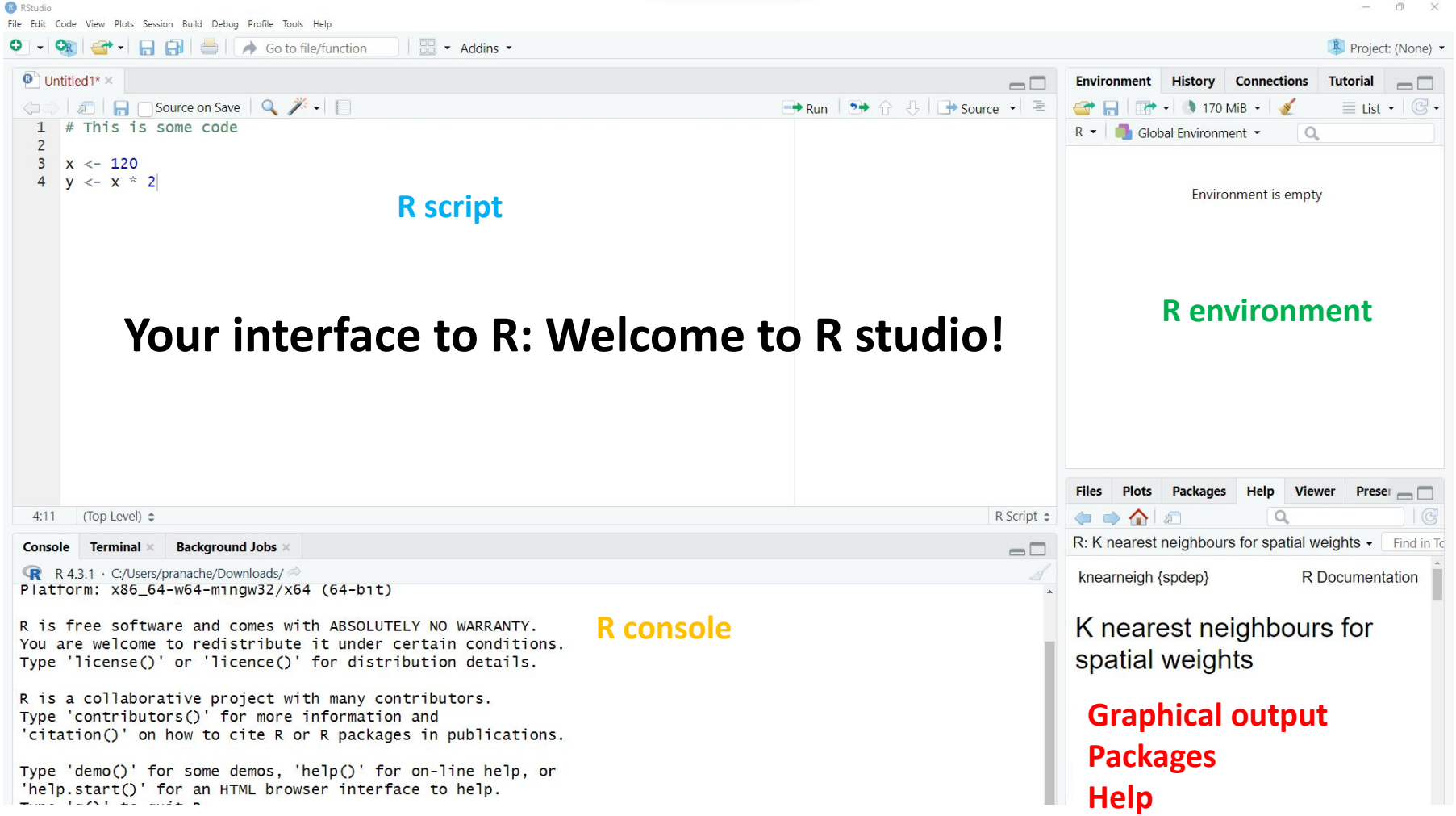






The architecture of R

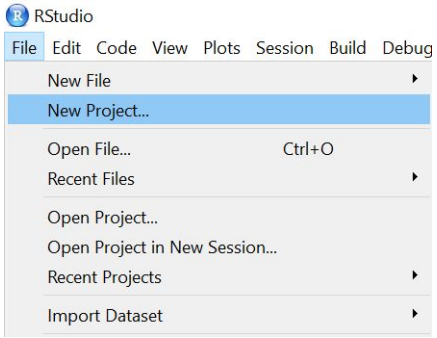



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
# This is some code
x <- 120
y <- x * 2
R script
Your interface to R: Welcome to R studio!
4:11
(Top Level)
R Script
Environment
History
Connections
Tutorial
170 MiB
List
R
Global Environment
Environment is empty
R environment
Console
Terminal
Background Jobs
R 4.3.1 · C:/Users/pranache/Downloads/
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
R console
Files
Plots
Packages
Help
Viewer
R: K nearest neighbours for spatial weights
Find in To
knearneigh {spdep}
R Documentation
K nearest neighbours for spatial weights
Graphical output
Packages
Help

RStudio projects

RStudio projects provide a **compartmentalized**, simple **framework** for analyses, making it straightforward to divide your work into multiple contexts, each with their own working directory, workspace, history, and source documents.

So before we start, create a new project.



R coding basics: assign values to objects

named object

assignment operator (y “gets” value 20)

```
y <- 20
```

value

R coding basics: call a function

function name

arguments

```
x <- seq(1, 10)
```

Specify arguments by name

```
seq(from = 1, to = 10)
```

Specify arguments by position

```
seq(1, 10)
```

Naming things in R

- Names consist of a combination of letters, numbers and _ and .
- Must start with a letter.
- Are case sensitive.

`some_name ≠ Some_Name ≠ SOME_NAME`

The tidyverse

Collection of R packages for data science

Common design philosophy, grammar, and data structure

Install and load packages

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


Find out more: Tidyverse

Data frames (tibbles)

Observations of of a phenomenon with several variables

columns are variables

survived	sex	age	passengerClass
yes	female	29	1st
yes	male	1	1st
no	female	2	1st
no	male	30	1st
no	female	25	1st

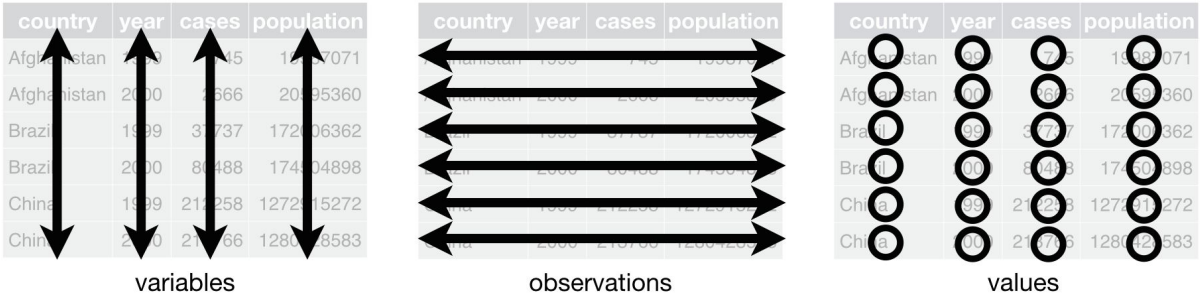
rows are observations

phenomenon

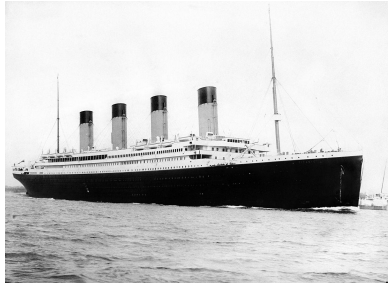
here: survival status of the passengers of the Titanic

Keep your data frames tidy!

1. Each variable has its own column.
2. Each observation has its own row.
3. Each value must has its own cell.



Get some data



survival status of the passengers of the Titanic

```
install.packages("carData")
library(carData)
titanic_survival <- as_tibble(TitanicSurvival)
```

sleep times and weights for a set of mammals

```
msleep
```

Transforming data

- `filter()` observations by their values
- `arrange()` the rows and order the observations
- `select()` variables by their names
- `mutate()` the data and create new variables from existing ones
- collapse many values down to a single value and `summarise()` the data

We can do the above on the entire dataset or `group_by()` a group.

filter()

Filter all passengers on the Titanic older than 25.

```
filter(titanic_survival, age > 25)
```

survived	sex	age	passengerClass
yes	female	29	1st
yes	male	1	1st
no	female	2	1st
no	male	30	1st
no	female	25	1st

survived	sex	age	passengerClass
yes	female	29	1st
no	male	30	1st

Find out more: Keep rows that match a condition — filter • dplyr

Operators for comparison

Filter all passengers who are exactly 25 years old.

```
filter(titanic_survival, age == 25)
```

Filter all passengers who are younger than or exactly 25 years old.

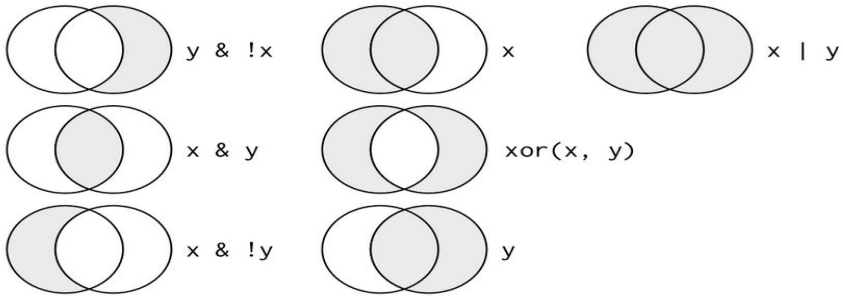
```
filter(titanic_survival, age <= 25)
```

Operator	description
==	equal
!=	not equal
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal

Find out more: R: Relational Operators

Logical operators

Logical operators	description
&	logical AND
|	logical OR
!	logical NOT



Combining filters and logical operators

Filter all carnivores or omnivores.

```
filter(msleep, vore == "carni" | vore == "omni")
```

Filter all carnivores that sleep longer than 11 hours.

```
filter(msleep, vore == "carni" &  sleep_total > 11)
```

Filter all rows with valid sleep values.

```
filter(msleep, !is.na(sleep_total))
```

select()

Select only data about survival and age.

```
select(titanic_survival, survived, age)
```

survived	sex	age	passengerClass
yes	female	29	1st
yes	male	1	1st
no	female	2	1st
no	male	30	1st
no	female	25	1st

survived	age
yes	29
yes	1
no	2
no	30
no	25

Find out more: Keep or drop columns using their names and types — select • dplyr

arrange()

Order the titanic survival data by age in ascending order.

```
arrange(titanic_survival, age)
```

Order the titanic survival data by age in descending order.

```
arrange(titanic_survival, desc(age))
```

survived	sex	age	passengerClass
yes	female	29	1st
yes	male	1	1st
no	female	2	1st
no	male	30	1st
no	female	25	1st

survived	sex	age	passengerClass
yes	male	1	1st
no	female	2	1st
no	female	25	1st
yes	female	29	1st
no	male	30	1st

Find out more: Order rows using column values — arrange • dplyr

mutate()

Compute the ratio between REM sleep and total sleep in the mammal sleep data and assign it to a new variable.

```
mutate(msleep, rem_ratio = sleep_rem / sleep_total)
```

name	....	sleep_total	sleep_rem
Owl monkey	…	17	1.8
Cow	…	4	0.7
Dog	…	10.1	2.9

name	....	sleep_total	sleep_rem	rem_ratio
Owl monkey	…	17	1.8	0.106
Cow	…	4	0.7	0.175
Dog	…	10.1	2.9	0.287

Find out more: Create, modify, and delete columns — mutate • dplyr

summarise()

Compute the mean age of passengers on the Titanic removing NA values.

```
summarise(titanic_survival, mean_age = mean(age, na.rm = TRUE))
```

survived	sex	age	passengerClass
yes	female	29	1st
yes	male	1	1st
no	female	2	1st
no	male	30	1st
no	female	25	1st

mean_age
29.9

Find out more: Summarise each group down to one row

`group_by()` and `summarise()`

Group the data by passenger class, and compute the mean age of passengers per class.

```
group_by(titanic_survival, passengerClass) %>%
  summarise(mean_age = mean(age, na.rm = TRUE))
```

survived	sex	age	passengerClass
yes	female	29	1st
yes	male	1	1st
no	female	2	1st
no	male	30	1st
no	female	25	1st

passengerClass	mean_age
1st	39.2
2nd	29.5
3rd	24.8

Find out more: Group by one or more variables — group_by • dplyr

Useful summary functions

Measures of location

```
mean(x), median(x)
```

Measures of spread

```
sd(x), IQR(x), mad(x)
```

Measures of rank

```
min(x), quantile(x, 0.95), max(x)
```

Measures of position

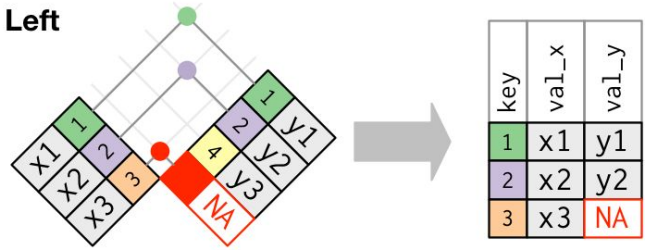
```
first(x), nth(x, 2), last(x)
```

Counts and proportions of logical values

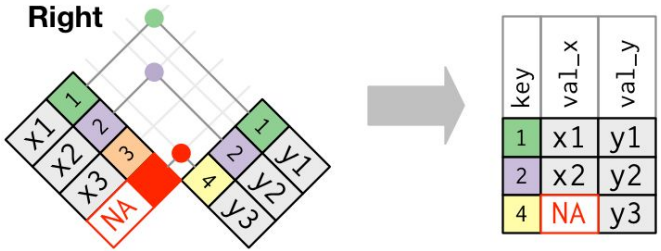
```
n(), n_distinct(x), sum(x>10)
```

Joining data

- Merge data from two or more data frames into one.
- A unique key specifies which rows are joined.



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA



key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

Find out more: mutate-joins • dplyr

Taking apart and joining back

Add the row number as a unique key to the Titanic data.

```
titanic_survival <- mutate(titanic_survival, id = row_number())
```

Split the data in two data frames.

```
a <- select(titanic_survival, age, sex, id)
b <- select(titanic_survival, survived, passengerClass, id)
```

Then put them back together again.

```
left_join(a, b, by = "id")
```

Combining multiple operations with pipes (%>%)

```
data %>%
    function1(column1) %>%
    function2(column2, column3)
```

take the value on the left

%>%

pass it to the right as an argument

Find out more: Pipe — %>% • magrittr

Transform and plot in one go using pipes

Group the data, compute the mean sleep per group and plot.

```
group_by(msleep, order) %>%
  summarise(mean_sleep = mean(sleep_total, na.rm = TRUE),
            mean_bodywt = mean(bodywt, na.rm = TRUE)) %>%
  ggplot() +
  geom_point(mapping = aes(x = mean_sleep, y = mean_bodywt))
```

transform the data

plot

Plotting data with `ggplot2`

creates a plot object

adds components to the plot

```
ggplot(data) +
  geom_point(mapping = aes(x, y))
```

creates a scatter plot

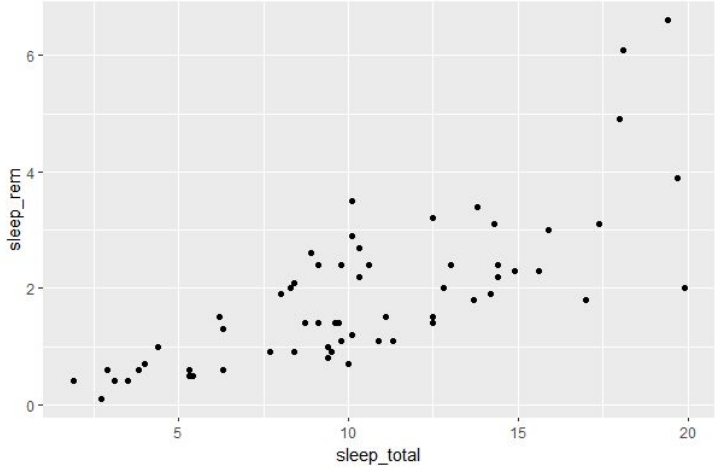
how are variables mapped to visuals?

Find out more: ggplot2

Scatter plots

Plot the relationship between total and REM sleep

```
ggplot(data = msleep) +
  geom_point(mapping = aes(x = sleep_total, y = sleep_rem))
```



Find out more: Points — geom_point • ggplot2

Changing the aesthetics

Color code different vores.

color

```
ggplot(data = msleep) +
  geom_point(mapping = aes(sleep_total, sleep_rem, color = vore))
```

Change the size of the dots proportional to the body weight.

size

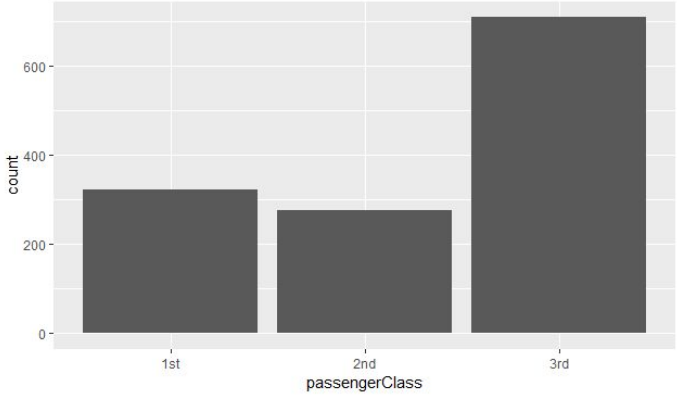
```
ggplot(data = msleep) +
  geom_point(mapping = aes(sleep_total, sleep_rem, size = bodywt))
```


Find out more: https://ggplot2.tidyverse.org/articles/ggplot2-specs.html

Bar plots

Plot the number of passengers per class on the Titanic.

```
ggplot(data = titanic_survival) +
  geom_bar(aes(x = passengerClass))
```



Under the hood, the function

- groups the data per passenger class
- counts the number of entries per class
- plots the count

Find out more: Bar charts — geom_bar • ggplot2

Changing the aesthetics of bar plots

Color code the bars with passenger class.

```
ggplot(data = titanic_survival) +
  geom_bar(mapping = aes(passengerClass, fill = passengerClass))
```

Color code the bars with survival status.

```
ggplot(data = titanic_survival) +
  geom_bar(mapping = aes(passengerClass, fill = survived))
```

Cheat sheets

Data transformation with dplyr :: Cheatsheet

Data visualization with ggplot2 :: Cheat Sheet

Learning objectives revisited

You are able to

- perform basic operations in R
- transform and explore data in the `tidyverse`
- visualize data with `ggplot2`

Exercises

1. Load the `gapminder` package, which provides values for life expectancy, GDP per capita, and population.
2. The data set contains data from multiple decades. Extract the data from the most recent year available.
3. Which 5 countries have the highest life-expectancy (in the most recent year)?
4. Which continent has the highest absolute GDP? Which has the highest GDP per capita? Beware that different countries have different population size.
5. Create a plot that shows GDP and life expectancy in different countries.
6. Add another visual variable (color, size, shape,...) to the plot (decide yourself what attribute of the data you want to show).