

[figure source](#)

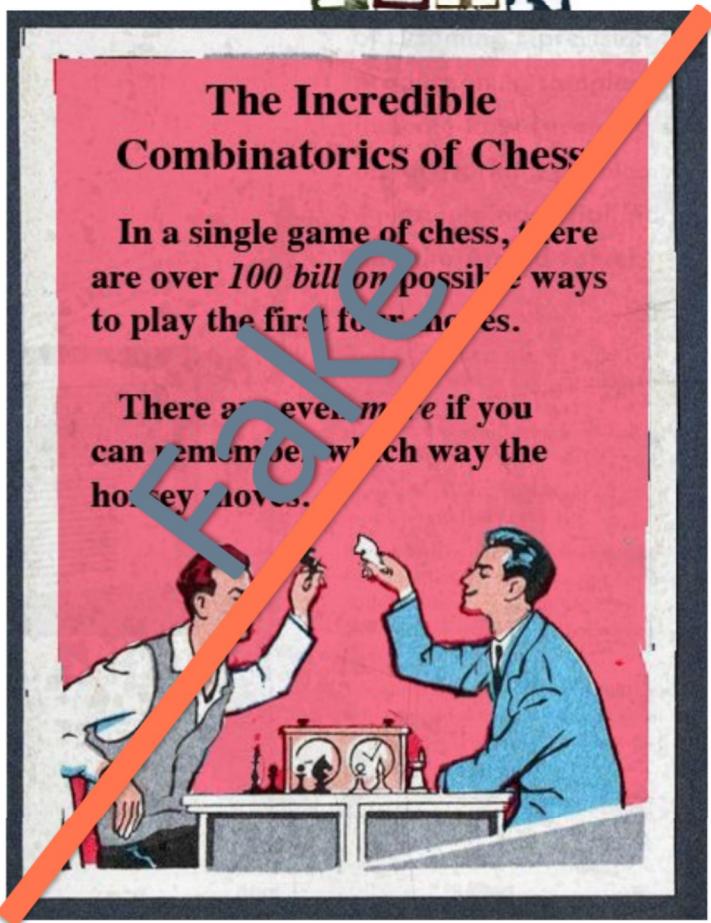
Statistics for WING

Vidushi Bigler, FS 2020, Classes W1 & W2

► Technik und Informatik

PLAN

- ▶ Combinatorics – the exquisite art of counting
- ▶ Some basic information
- ▶ For classes W2a & W2r



[figure source](#)

Combinatorics – the exquisite art of counting

Es gibt zwei grundlegende Prinzipien

- ▶ das Summenprinzip und
- ▶ das Produktprinzip

Combinatorics – the exquisite art of counting

Das Summenprinzip

Wenn man ein Task auf n verschiedene Arten erledigen kann **oder** auf m verschiedenen Arten, dann gibt es insgesamt $n + m$ mögliche Wege diesen Task zu erledigen.

Wichtig: $n + m$ stimmt nur, wenn der Task **nicht gleichzeitig** auf beide verschiedene Arten erledigt werden kann.

Wichtig: wenn nicht gesagt wird, ist in der Mathematik mit **oder** immer **einschliessendes-oder** gemeint und nie *entweder-oder* (also kein **xor**).
Bsp: Ich gehe heute Abend auswärts essen oder lese ein Buch zu Hause.
Heisst – ich tue das eine oder das andere, aber vielleicht auch beides!
Heisst nicht – ich gehe *entweder* essen **oder** ich lese ein Buch.

Combinatorics – the exquisite art of counting

Das Produktprinzip

Angenommen eine Aufgabe lässt sich in zwei Tasks zerlegen, die hintereinander ausgeführt werden. Wenn man Task 1 auf n verschiedene Arten erledigen kann **und** Task 2 für jede Möglichkeit des ersten Tasks auf m verschiedenen Arten ausgeführt werden kann, dann kann die Aufgabe auf insgesamt $n \times m$ Arten erledigt werden.

Wichtig: $n \times m$ stimmt nur, wenn die Tasks **unabhängig** von einander erledigt werden können.

Permutations, Variations and Combinations without Repetition

Combinatorics – the exquisite art of counting

Permutationen: es gibt $n!$ (gelesen n-Fakultät) mögliche Anordnungen von n Objekten, wenn wir uns für die Reihenfolge dieser Objekten untereinander interessieren.

Variationen: aus n Objekten werden k ausgewählt. In diesem Fall gibt es $\frac{n!}{(n-k)!}$ mögliche Anordnungen, wenn wir uns für die Reihenfolge dieser k Objekten untereinander interessieren.

Kombinationen: aus n Objekten werden k ausgewählt (wobei $n=k$ auch möglich). **Wichtig** – die Reihenfolge dieser Objekte untereinander interessiert uns nicht mehr. In diesem Fall gibt es $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ mögliche Anordnungen.

A couple of examples

Permutations, Variations and Combinations without Repetition

Combinatorics – the exquisite art of counting

1.2 Permutationen

Beispiel 8 5 Läufer A, B, C D und E machen einen Wettkampf. Wie viele verschiedenen Ranglisten gibt es?

Lösung: Wir zählen alle Möglichkeiten auf:

ABCDE	ABCED	ABDCE	ABDEC	ABECD	ABEDC
ACBDE	ACBED	ACDBE	ACDEB	ACEBD	ACEDB
ADBCE	ADBEC	ADCBE	ADCEB	ADEBC	ADECB
AEBCD	AEBDC	AECBD	AECDB	AEDBC	AEDCB
BACDE	BACED	BADCE	BADEC	BAECD	BAEDC
BCADE	BCAED	BCDAE	BCDEA	BCEAD	BCEDA
BDACE	BDAEC	BDCAE	BDCEA	BDEAC	BDECA
BEACD	BEADC	BECAD	BECDA	BEDAC	BEDCA
CABDE	CABED	CADBE	CADEB	CAEBD	CAEDB
CBADE	CBAED	CBDAA	CBDEA	CBEAD	CBEDA
CDABE	CDAEB	CDBAE	CDBEA	CDEAB	CDEBA
CEABD	CEADB	CEBAD	CEBDA	CEDAB	CEDBA
DABCE	DABEC	DACBE	DACEB	DAEBC	DAECB
DBACE	DBAEC	DBC AE	DBCEA	DBEAC	DBECA
DCABE	DCAEB	DCBAE	DCBEA	DCEAB	DCEBA
DEABC	DEACB	DEBAC	DEBCA	DECAB	DECBA
EABCD	EABDC	EACBD	EACDB	EADBC	EADCB
EBACD	EBADC	EBCAD	EBCDA	EBDAC	EBDCA
ECABD	ECADB	ECBAD	ECBDA	ECDAB	ECDBA
EDABC	EDACB	EDBAC	EDBCA	EDCAB	EDCBA

Es gibt also 120 verschiedene Ranglisten.



Combinatorics – the exquisite art of counting

1.3 Variationen ohne Wiederholung

Beispiel 12 Wir betrachten wiederum das Problem der 5 Läufer. Diesmal interessieren wir uns aber bloss für die Klassierung der ersten 3 Läufer. Wie viele verschiedene Ranglisten gibt es?

Lösung: Wir schreiben alle Möglichkeiten auf:

ABC	BAC	CAB	DAB	EAB
ABD	BAD	CAD	DAC	EAC
ABE	BAE	CAE	DAE	EAD
ACB	BCA	CBA	DBA	EBA
ACD	BCD	CBD	DBC	EBC
ACE	BCE	CBE	DBE	EBD
ADB	BDA	CDA	DCA	ECA
ADC	BDC	CDB	DCB	ECB
ADE	BDE	CDE	DCE	ECD
AEB	BEA	CEA	DEA	EDA
AEC	BEC	CEB	DEB	EDB
AED	BED	CED	DEC	EDC

Es gibt also 60 verschiedene Ranglisten. \diamond

Combinatorics – the exquisite art of counting

1.4 Kombinationen

Beispiel 16 Wir betrachten wiederum das Beispiel der 5 Läufer. Diesmal interessieren wir uns nur für die 3 Medaillengewinner. Wie viele mögliche Gruppen von Medaillengewinnern gibt es (die Reihenfolge innerhalb der ersten 3 interessiert uns nicht mehr)?

Lösung: Im Beispiel 12 hatten wir $5 \cdot 4 \cdot 3 = 60$ Möglichkeiten für die ersten 3 Ränge aus den 5 Athleten berechnet. Darunter waren z.B. die Anordnungen:

$$ABC \ ACB \ BAC \ BCA \ CAB \ CBA . \quad (1)$$

Im jetzigen Problem, wo es nicht mehr auf die Reihenfolge ankommt, zählen diese 6 Anordnungen nur noch als eine einzige Möglichkeit, nämlich

A, B, C sind die Medaillengewinner.

Die Zeile (1) enthält die $3! = 6$ Permutationen der 3 Elemente A, B und C .

Wir können diese Überlegung für jede Dreiergruppe von Athleten (z.B. A, B, D oder C, D, E usw.) wiederholen und stellen jedesmal fest, dass aus $3! = 6$ Anordnungen eine einzige Möglichkeit wird.

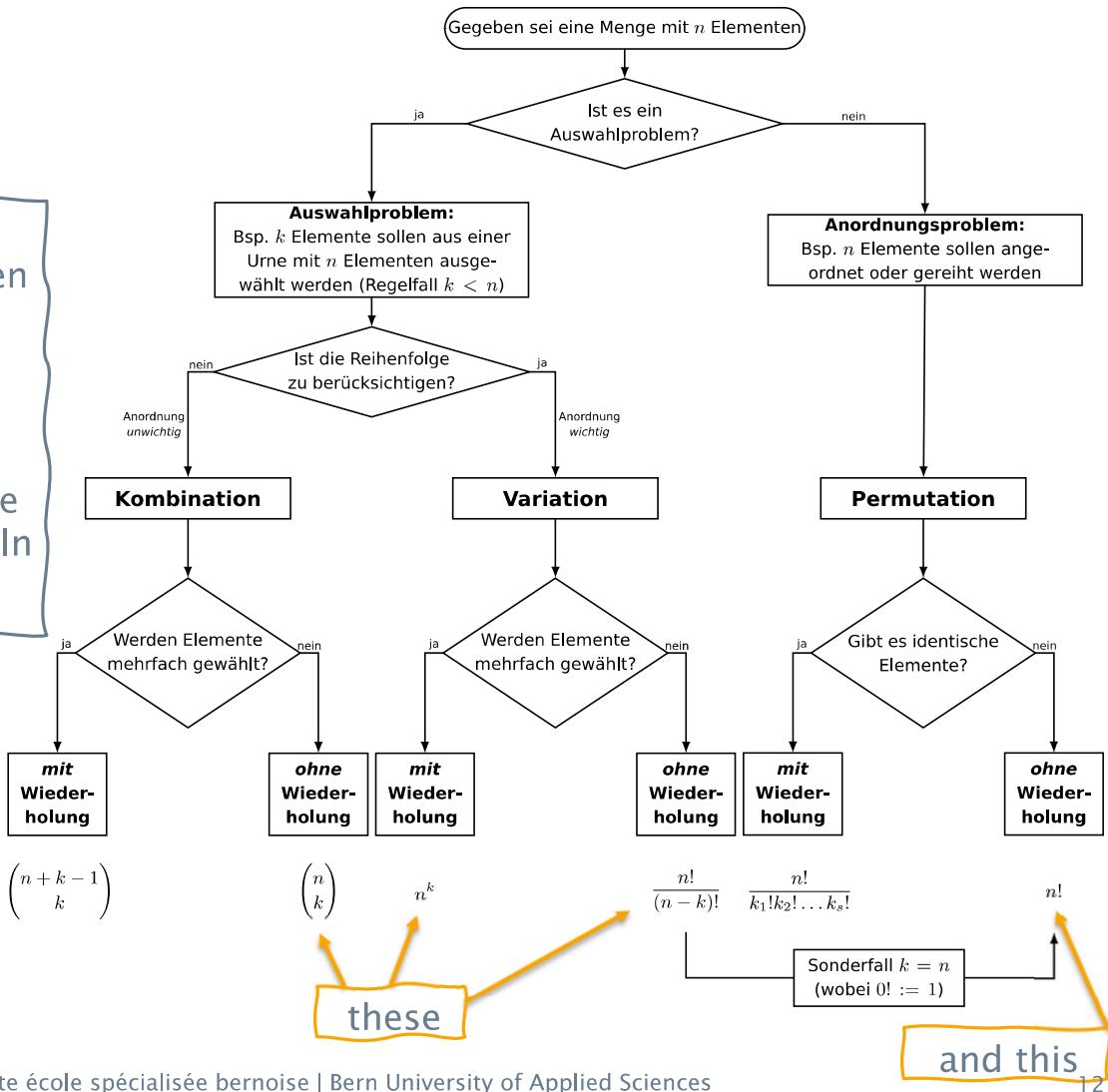
Wir finden also die Antwort auf unsere Frage, indem wir die 60 Möglichkeiten des Beispiels 12 durch 6 dividieren:

Es gibt also $60:6=10$ Möglichkeiten, aus den 5 Läufern eine Dreiergruppe auszuwählen. \diamond

Overview

In der rechten Darstellung werden alle Fälle gezeigt, d.h. sowohl ohne als auch mit Wiederholung.

Wir werden nur die markierten Formeln antreffen.



Some basic information

Libraries, programming

Libraries – best allies

List of libraries in R



Example of my workflow for visualisation
file:///Users/vidu/Downloads/Visualising_steps_gapminder.html

```
library(broom)
library(cluster)
library(clustree)
library(clValid)
library(data.table)
library(dplyr)
library(dtclust)
library(dtw)
library(factoextra)
library(forcats)
library(fpp2)
library(ggplot2)
library(ggrepel)
library(kableExtra)
library(lubridate)
library(magrittr)
library(mclust)
library(NbClust)
library(parallel)
library(RCurl)
library(scales)
library(splines)
library(som)
library(TTR)
library(tidyr)
library(tidyverse)
library(zoo)

library(grid)
library(openxlsx)
library(maps)
library(viridis)
#library(mvtsplot)
library(tabplot)
library(wesanderson)
```

Libraries – matplotlib

```
data = px.data.gapminder()

# And I need to transform my categorical column (continent) in a numerical value group1->1, group2->2...
data['continent']=pd.Categorical(data['continent'])

# For each year:
for i in data.year.unique():

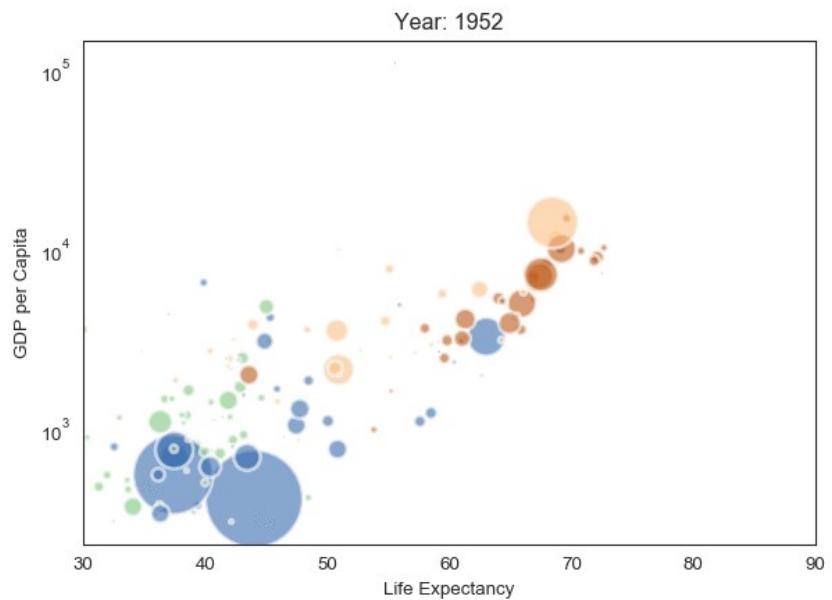
    # initialize a figure
    fig = plt.figure(figsize=(680/my_dpi, 480))

    # Change color with c and alpha. I map the continents to colors
    tmp=data[ data.year == i ]
    plt.scatter(tmp['lifeExp'], tmp['gdpPercap'],
                c=tmp['continent'].cat.codes,
                alpha=0.6, edgecolors="white")

    # Add titles (main and on axis)
    plt.yscale('log')
    plt.xlabel("Life Expectancy")
    plt.ylabel("GDP per Capita")
    plt.title("Year: "+str(i))
    #plt.ylim(0,100000)
    plt.xlim(30, 90)

    # Save it
    filename='Gapminder_step'+str(i)+'.png'
    plt.savefig(filename, dpi=96)
    plt.gca()

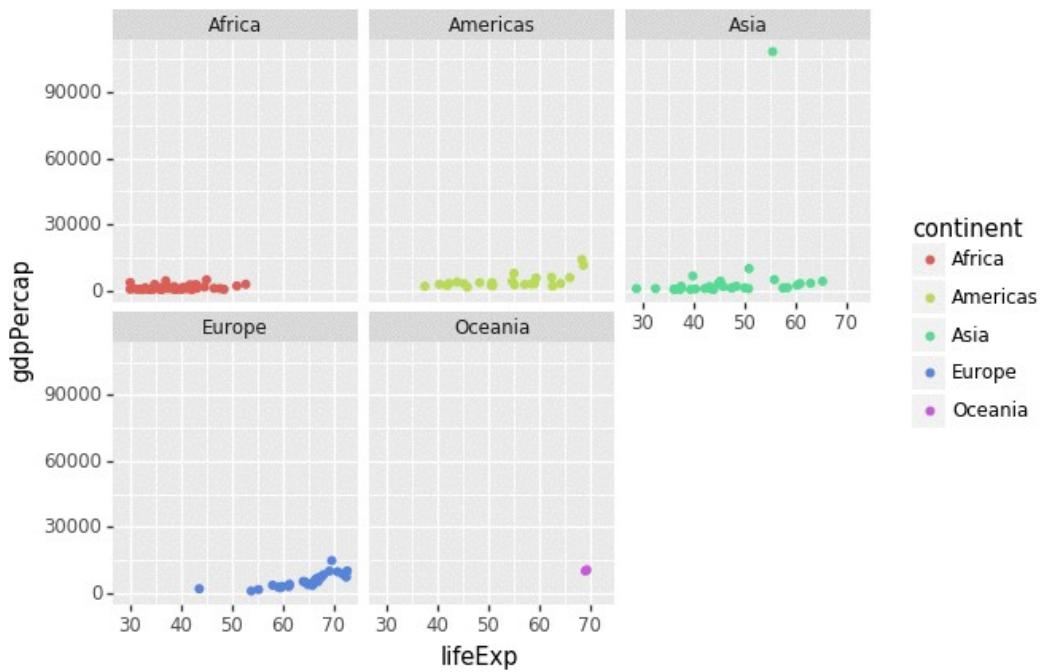
# Then use image magick (this is bash, not python)
#convert -delay 80 Gapminder*.png animated_g
```



Libraries – plotnine

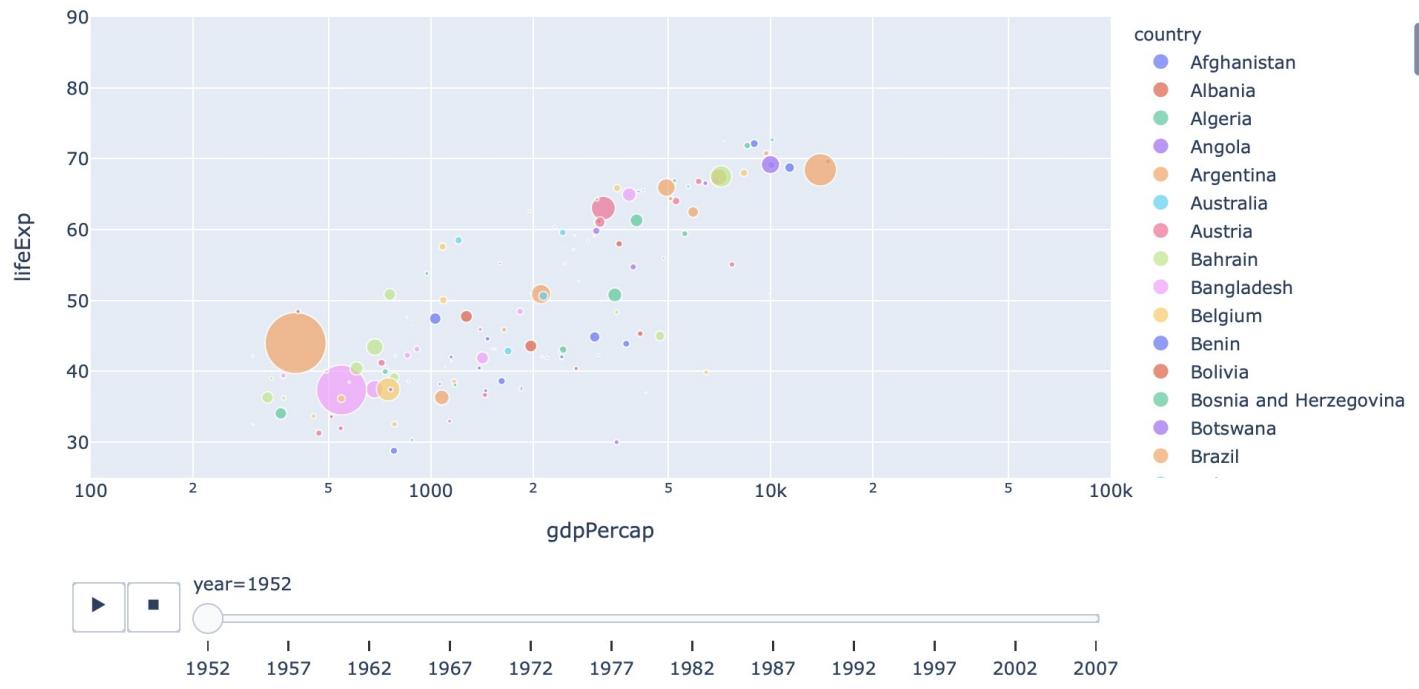
```
warnings.simplefilter("ignore")
from plotnine import *
# Using ggplot
# For each year:
for i in data.year.unique():
    p = (ggplot(data [ dat
        + geom_point()
        + facet_wrap('~-co
    )
    p

# Save it
filename='gg_Gapminder'
p.save(filename, dpi=9
# Then use image magick (t
#convert -delay 80 gg_Gapn
```



Libraries – plotly.express

```
: px.scatter(df, x="gdpPercap", y="lifeExp", animation_frame="year", animation_group="country",
            size="pop", color="country", hover_name="country",
            log_x = True,
            size_max=45, range_x=[100,100000], range_y=[25,90])
```



Libraries – pitfalls - order matters!!!

Beware of the order while loading libraries! Names are not exclusive. Better to label libraries or import only what you need.

Works fine

```
: from tidyverse import *
from dplyr import *
from pydataset import data

longley = data('longley')
longley
```

	GNP.deflator	GNP	Unemployed	Armed.Forces	Population	Year	Employed
1947	83.0	234.289	235.6	159.0	107.608	1947	60.323
1948	88.5	259.426	232.5	145.6	108.632	1948	61.122
1949	88.2	258.054	368.2	161.6	109.773	1949	60.171
1950	89.5	284.599	335.1	165.0	110.929	1950	61.187
1951	96.2	328.975	209.9	309.9	112.075	1951	63.221
1952	98.1	346.999	193.2	359.4	113.270	1952	63.639
1953	99.0	365.385	187.0	354.7	115.094	1953	64.989
1954	100.0	363.112	357.8	335.0	116.219	1954	63.761
1955	101.2	397.469	290.4	304.8	117.388	1955	66.019
1956	104.6	419.180	282.2	285.7	118.734	1956	67.857
1957	108.4	442.769	293.6	279.8	120.445	1957	68.169
1958	110.8	444.546	468.1	263.7	121.950	1958	66.513
1959	112.6	482.704	381.3	255.2	123.366	1959	68.655
1960	114.2	502.601	393.1	251.4	125.368	1960	69.564

Oh oh

```
: from tidyverse import *
from pydataset import data
from dplyr import *

longley = data('longley')
longley
```

```
-----  
TypeError                                         Traceback (most recent call last)
<ipython-input-6-c77ef828d8ae> in <module>
      3 from dplyr import *
      4
----> 5 longley = data('longley')
      6 longley
```

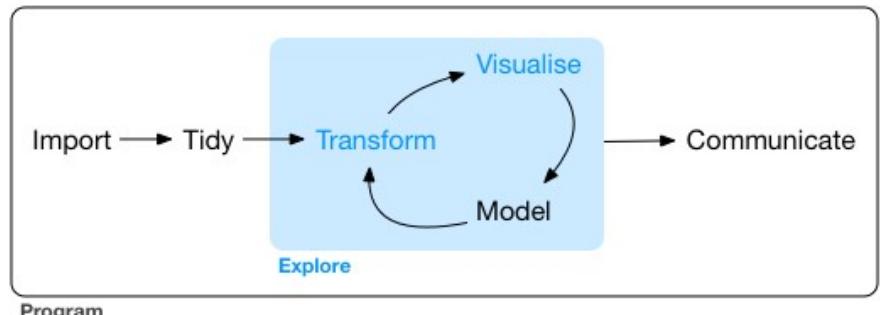
```
TypeError: 'module' object is not callable
```

Data wrangling – remember...

Basic idea is to leave nested programming and to create pipes.



[figure source](#)



[figure source](#)

We will use the packages **pandas**, **tidyverse** and **dplyr**.

Data wrangling – several options

A blog on modern programming in python

<https://nodata.science/dplyr-for-python.html>

Learn more about dplython functionalities from the blog

<https://www.r-bloggers.com/dpylthon-and-dplyr-for-python/>

Github page of Chris Riederer (dplython)

<https://github.com/dodger487/dplython/blob/master/dplython/dplython.py>

Pandas – in case you'd like a tutorial

10 Minutes to pandas

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Video

<https://www.youtube.com/watch?v=e60ItwIZTKM>

Reproducible research

Why using jupyter notebook is not a bad idea?

Clustering project

Cluster Analyse von Wassertemperaturen der Bundes-Messstationen
Dynamic Time Warping und k-Medoids
Dr. Vidushi Bigler-Mailart

IODA, Technik und Informatik, Berner Fachhochschule, Quellgasse 21, CH-2501 Biel/Bienne

Methodik

Die Verfahren *Dynamic Time Warping (DTW)* und *k-Medoids (PAM, engl. partitioning around medoids)* werden verwendet, um Tagesmittelwerte verschiedener Wassertemperatur-Messstationen in typologische Gruppen zu unterteilen. Diese Messreihen der Ähnlichkeit zwischen zwei zeitlichen Sequenzen, die in ihrer Geschwindigkeit variieren können, wird durch den DTW Algorithmus ermöglicht. Die Sequenzen werden in der Zeitdimension "verzögert" (engl. warped), damit ihre Ähnlichkeit unabhängig von nichtharmonischen Variationen in der Zeit genommen werden kann.

Die Tabelle 1 zeigt eine Liste von Kennzahlen (CVI, engl. *cluster validity indices*), die zur Bestimmung der optimalen Anzahl Gruppen eingesetzt werden können (siehe ebenfalls Figur 2).

Tabelle 1: Kennzahlen zur Bestimmung der Anzahl Cluster

Index	Information	Anzahl
COP	COP (Arbelaitz et al. (2013); zu minimieren)	9
Dunn	Dunn (Arbelaitz et al. (2013); zu maximieren)	7
DB	Davies-Bouldin (Arbelaitz et al. (2013); zu minimieren)	5
DBstat	modifiziertes Davies-Bouldin (Kan and Ramakrishna (2005); zu minimieren)	6
Sil	Silhouette (Rousseeuw (1987); zu maximieren)	9

Ergebnisse

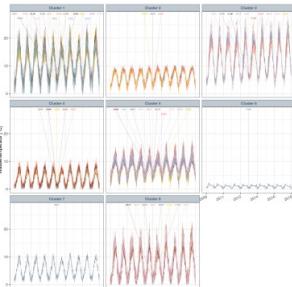
Figur 2 stellt die Ergebnisse der Cluster Analyse dar. Die Karte 3 zeigt die geographische Lage der Bundes-Messstationen und deren Gruppenzugehörigkeit.

Figur 4 ermöglicht eine visuelle Überprüfung der Ergebnisse. Die Werte der Messstationen werden diskretisiert und in drei Temperaturbereiche unterteilt - kalt (grau), moderate (rot) und warm (gelb). Die Y-Achse zeigt die zeitliche Verlauf. Die Messreihen der verschiedenen Gruppen werden gebündelt und durch graue Trennhüllen umrahmt. Die Abhängigkeiten und Differenzen in den Verhaltensstrukturen der Zeiteinheiten werden deutlich sichtbar. Klar erkennbare Muster kommen zur Geltung.

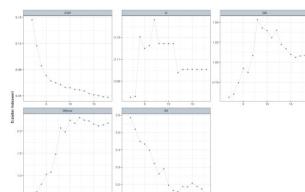
Quellen

Detaillierte Beschreibungen der verwendeten Verfahren und eine vollständige Liste der Referenzen sind im Hauptbericht zu finden.

1. Sakoe, H., Chiba (1978), "Dynamic programming algorithm optimization for spoken word recognition".
2. Kaufman, L. and Rousseeuw, P.J. (1987), "Clustering by means of Medoids, in Statistical Data Analysis Based on the L₁-Norm and Related Methods"



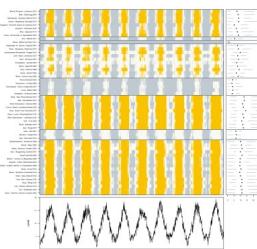
Figur 1: Ergebnis der Cluster Analyse mit dem DTW Algorithmus. Die Messreihen der Bundesstationen werden representativen Gruppen zugeschlagen.



Figur 2: Kennzahlen (CVI) zur Bestimmung der Anzahl Cluster



Figur 3: Geographische Lage der Bundes-Messstationen



Figur 4: Visuelle Überprüfung der Ergebnisse durch Mustererkennung; gruppierte Messreihen zeigen ähnliche Verhaltensstrukturen.



Berner Fachhochschule
Service Hochschule
K2SA
Konsortium für Optimierung und Daten Analyse

Cluster Analyse der Bundes- und kantonalen Messstationen

anhand von Langzeitwassertemperaturmessungen von Fließgewässern

Im Auftrag des Bundesamtes für Umwelt (BAFU)

Biel, November 2019

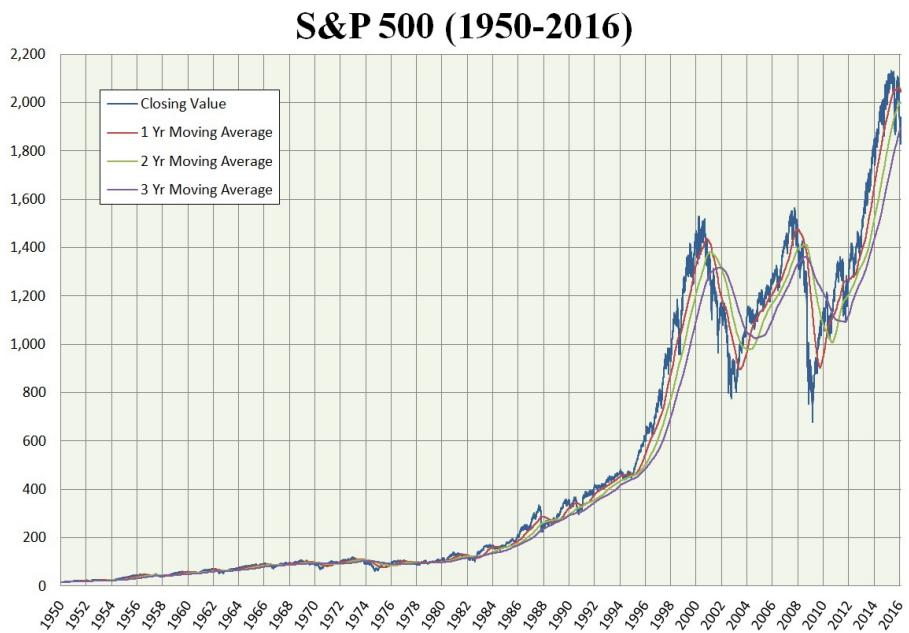
file:///Users/vidu/Documents/BFH/bafu/08_R_Code_Clustering/03_BAFU_Water_Temperature/Poter/Poster_DTW.html

For classes W2a & W2r

S&P 500

S&P 500 (more or less)

The S&P index measures the stock performance of 500 of the largest American companies.



[figure source](#)

S&P 472 with R-Code

```
setwd("~/Documents/BFH/bafu/08_R_Code_Clustering/01_Stockmarket_Prices")
library(quantmod)
library(openxlsx)
library(xts)
library(RCurl)

# Get the S&P 500 companys' information
download = getURL("https://raw.githubusercontent.com/datasets/s-and-p-500-companies/master/data/constituents.csv")
write.csv(download, file ="Infos")
SP500 = read.csv(text=download)
head(SP500)
# Drop some companies due to data errors
SP500 = SP500[-which(SP500$Symbol %in%
                      c("APC", "BHGE", "BBT", "BRK.B", "HRS", "HCP", "MON", "UA", "WELL", "AET",
                        "ANDV", "BF.B", "CA", "CSRA", "DPS", "DWDP", "EVHC", "ESRX", "GGP",
                        "LUK", "LLL", "KORS", "NFX", "PX", "RHT", "COL", "SCG", "SYMC",
                        "TMK", "TSS", "TWX", "WYN", "XL"))]
write.xlsx(SP500, file = "SP500.xlsx")
```



	Symbol	Name
1	MMM	3M Company
2	AOS	A.O. Smith Corp
3	ABT	Abbott Laboratories
4	ABBV	AbbVie Inc.
5	ACN	Accenture plc
6	ATVI	Activision Blizzard
		Information Technology

S&P 472 with R-code

```
# Acquire stock price information with quantmod package
stock_price = list()
stock_name = NULL
for(ticker in SP500$Symbol){
  print(ticker)
  stock = getSymbols(ticker, src='yahoo',from = '2015-07-01', auto.assign = F)
  stock_price = c(stock_price,list(stock))
  stock_name = c(stock_name,ticker)
}
names(stock_price) = stock_name

# I get a list of data.frames
> head(stock_price[["GOOGL"]])
  GOOGL.Open GOOGL.High GOOGL.Low GOOGL.Close GOOGL.Volume GOOGL.Adjusted
2015-07-01    543.66    545.81    539.76     543.30      1538000       543.30
2015-07-02    544.74    548.00    543.57     547.34      1406200       547.34
2015-07-06    542.25    548.58    542.10     545.62      1280700       545.62
2015-07-07    547.43    551.00    539.85     550.03      1678900       550.03
2015-07-08    545.60    548.28    541.20     541.70      1383100       541.70
2015-07-09    548.65    548.88    544.62     544.65      1488300       544.65
```

S&P 472 with R-code

```
# clean each data.frame
transform_df <- function(df){
  df <- fortify.zoo(df)
  int <- grep("Open", colnames(df))
  df$Symbol <- gsub(".Open", "", colnames(df)[int])
  colnames(df) <- c("Date", "Open", "High", "Low", "Close", "Volume", "Adjusted", "Symbol")
  df <- df[c("Symbol", "Date", "Open", "High", "Low", "Close", "Volume", "Adjusted")]
  return(df)
}

# row bind the data.frames to each other
temp <- transform_df(stock_price[[1]])
for (i in 2:length(names(stock_price))) {
  print(i)
  temp1 <- transform_df(stock_price[[i]])
  temp <- rbind(temp, temp1)
}
write.xlsx(temp, file = "stock_price.xlsx")
```



prepared
stock prices

S&P 500 (more or less)

Several ways are open to you and they do not all lead to Rome

- ▶ read Stine-Foster pages 148-153
- ▶ a very comprehensive tutorial
<https://pythonprogramming.net/getting-stock-prices-python-programming-for-finance/>
- ▶ figure out how to deal with such a large amount of data
- ▶ visualisations for time series data (see e.g. [link](#))
- ▶ define a clear question and see if you can answer it

Thank you for your time and attention.