Einführung Zeitreihen

Peter Büchel

HSLU I

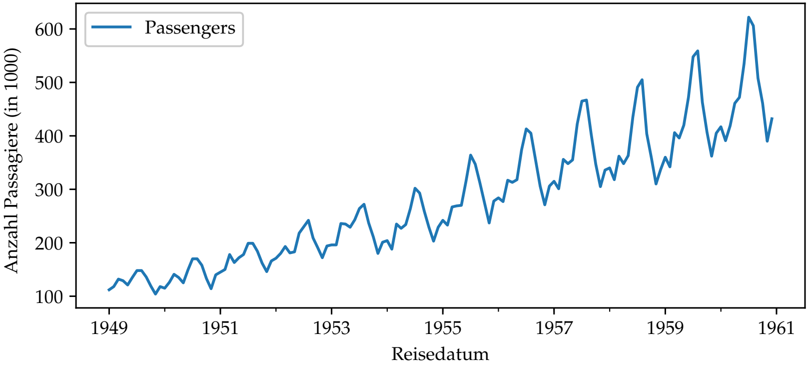
Stat: Block 11

Beispiel

- Tabelle zeigt die Passagierzahlen der Fluggesellschaft PAN AM (1927-1991) von 1949 bis 1960.

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

- Daten unübersichtlich $\rightarrow$ graphisch darstellen:



- Folgende Muster erkennbar:
  - Passagierdaten steigen jährlich $\rightarrow$ *Trend*
  - Innerhalb eines Jahres gibt es auch Unterschiede
    In Ferienzeit wurde mehr geflogen als sonst $\rightarrow$ *Saisonalität*
  - Beobachtungen *nicht* unabhängig voneinander
    Benachbarte Werte sind ähnlich $\rightarrow$ *serielle Korrelation*

- Dieser Datensatz wurde ursprünglich dazu verwendet um zukünfige Passagierzahlen vorherzusagen, damit der Kauf von Flugzeugen und die Nachfrage von Flugpersonal geplant werden kann

- Viele reale Messungen und Datenerfassungen resultieren in Datenmengen, die seriell korrelieren:
  - Überwachung von Maschinen: Temperatur, Druck, akkustische Emissionen, Vibrationen, ..., die in/ an oder um eine laufende Maschine (Motor, Generator, Kompressor,...) ändern sich nicht schlagartig.
    Nahe beieinander liegende Messwerte sind ähnlich.
  - Börse: Aktienpreise, Wechselkurse,... werden am Ende eines Handelstages aufgezeichnet. Diese ändern sich zwar zufällig von Tag zu Tag, aber nicht extrem (ausser in Ausnahmesituationen).
  - Umweltbeobachtungen: Temperaturen, Feuchtigkeit, Pollenkonzentration, Verschmutzung, Niederschläge, die bei einer bestimmten Wetterstation aufgezeichnet wurden.
    Die Temperatur von heute (z.B. $20\,^{\circ}\mathrm{C}$ in Luzern) wird sich nicht (oder kaum) auf $-10\,^{\circ}\mathrm{C}$ morgen senken. Sie wird wohl bei um die $20\,^{\circ}\mathrm{C}$ bleiben.
  - Bundesamt für Statistik: Bevölkerungsgrösse, Einkommen, Anzahl Unfälle, ändern sich nicht extrem von Jahr zu Jahr.

- Diese Art von Daten nennt man *Zeitreihen*

- Mehrere Ziele, die man mit Zeitreihen erreichen kann:
    - *Deskriptive Analyse*:
      Mit Hilfe von Übersichtsstatistiken und Visualisierungen grundsätzlichen Eigenschaften verstehen
    - *Modellierung und Interpretation*:
      Durch Modellierung des der Zeitreihe zu Grunde liegenden Prozesses erlaubt es, ein tieferes Verständnis zu erhalten
    - *Zerlegung*:
        - *Saisonalität*, insbesondere periodische Muster
        - *Trend*, eine allmähliche Änderung des Mittelwertes der Zeitreihe

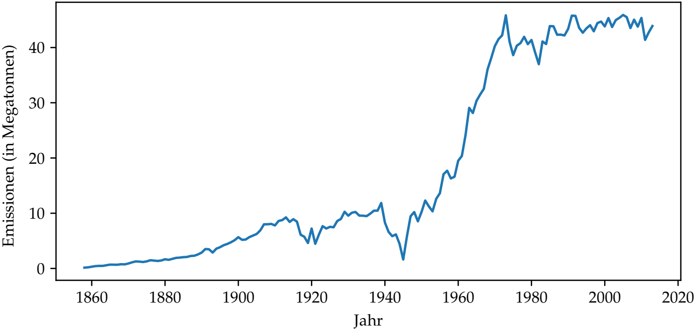
- *Vorhersage*:
  Mit Hilfe eines Modelles zukünftige Werte einer Zeitreihe vorhersagen
- *Regression*:
  Oft versucht man, eine Zeitreihe durch mehrere andere Zeitreihen zu erklären

- In diesem Modul nur die ersten 3 Punkte betrachten
- Vor mathematischer Modellierung, zuerst Beispiele

Beispiel: Kyoto Protokoll

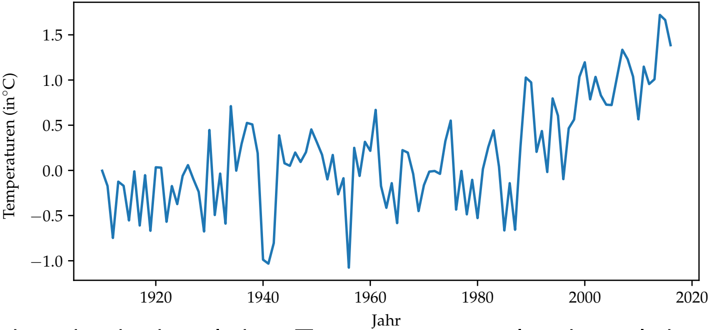
- Kyoto Protokoll: Zusatz zum Rahmenübereinkommen der Vereinten Nationen über Klimaänderungen
- Am 11. Dezember 1997 beschlossenen und am 16. Februar 2005 in Kraft getretenen
- Argumente für die Reduzierung der Treibhausgase hängen aus einer Kombination von Wissenschaft, Wirtschaft und Zeitreihenanalyse zusammen
- Entscheidungen, die in den nächsten Jahren gemacht werden, haben Einfluss auf die Zukunft unseres Planeten

- Abbildung:



- Abbildung: jährliche Emission vom Treibhausgas CO_2 in der Schweiz von 1858 bis 2013
- Keine saisonaler Effekt (jährliche Durchschnitte)
- Eigenartiger Trend erkennbar
- Emissionen nach 2. Weltkrieg zwischen 1950 und 1970 stark zugenommen

- Ähnlich bedrohlich: Globalen Temperaturerwärmung

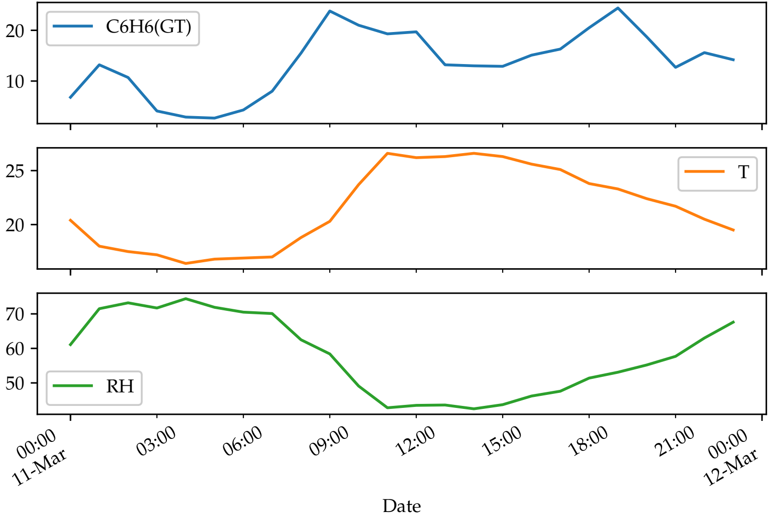


- Jährlichen durchschnittlichen Temperaturanomalien bezüglich zum Mittel zwischen 1910 und 2000 in Europa
- Aufwärtstrend, der um 1980 beginnt
- Zeitreihe sagt allerdings nichts über *Gründe* der Temperaturzunahme
- Abbildung weiter oben (CO_2): Korrelation zwischen dem CO-Ausstoss und der globalen Temperaturerwärmung unbestreitbar
- Das heisst nicht, dass es einen kausalen Zusammenhang geben muss

Beispiel: Luftqualität

- Datensatz von stündlichen Messungen aus einer Reihe von 5 chemischen Sensoren für Metalloxide, die in einem Multisensorgerät für chemische Luftqualität eingebettet sind
- Gerät wurde auf Strassenhöhe in einer deutlich verschmutzten Gegend in einer italienischen Stadt aufgestellt
- Daten wurden vom März 2004 bis Februar 2005 aufgezeichnet
- Insgesamt wurden 13 verschiedene Grössen gemessen

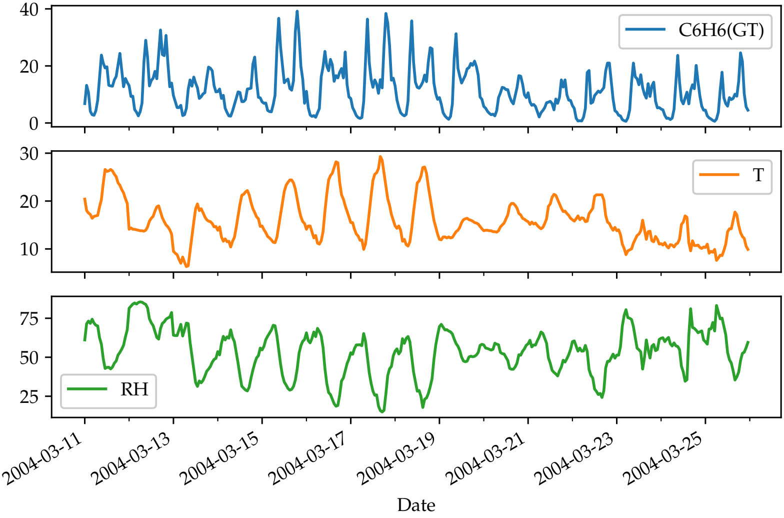
- Abbildung:



- Konzentration von Benzol (C_6H_6), die Lufttemperatur und die relative Luftfeuchtigkeit (in °C) an einem Tag aufgezeichnet

- Temperatur in der Nacht am tiefsten und am frühen Nachmittag am grössten
- Bei der Luftfeuchtigekeit ist es gerade umgekehrt
- Benzolkonzentration ist um 9 und 18 Uhr am grössten
  $\rightarrow$ Rushhour
- Gründe einer Saisonalität
- In folgender Abbildung: Werte über eine zweiwöchige Periode

- Abbildung:

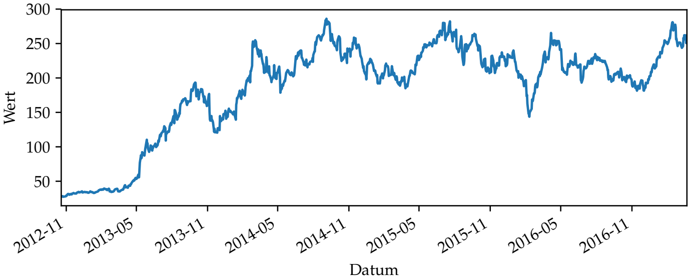


40
20
0
C6H6(GT)
30
20
10
T
75
50
25
RH
2004-03-11
2004-03-13
2004-03-15
2004-03-17
2004-03-19
2004-03-21
2004-03-23
2004-03-25
Date

Beispiel: Aktienkurs von Tesla

- Typisches Beispiel einer Zeitreihen: Aktienindizes, Wechselkurse, usw. in der Wirtschaft
- Aktienindizes werden oft analysiert und für Vorhersagen verwendet
- Trend eines Aktienkurses ist allerdings unmöglich vorherzusagen
- Hier: Aktienkurs von Tesla

- Abbildung:



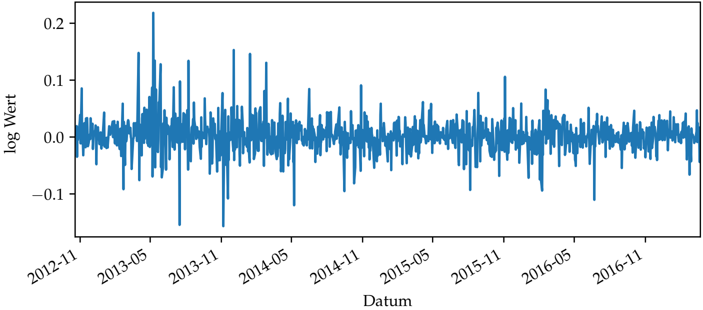
- Tagesabschlüsse von 1112 aufeinanderfolgender Handelstage, begonnen am 19 Dezember 2012

- Aktienindex von Tesla vom März bis Juni 2013 sehr stark zunehmend

- Februar 2016: scheinbarer Zusammenbruch des Kurse, der wieder von einem starken Anstieg gefolgt wurde

- Vergleichen Trends mit Ankündigungen von Tesla: Korrelationen, speziell mit Ankündigung der Models 3 im April 2016

- Abbildung:



- Anstatt Aktienkurs $\rightarrow$ sogenannten *log-return* angeben
- Veränderungen des Logarithmus des Index von Tag zu Tag
- log-returns sind eine Näherung der relativen Änderung (in Prozent) bezüglich des vorhergehenden Handelstages
- Kein Trend mehr vorhanden ist $\rightarrow$ Daten unkorreliert
- Konsequenz sind Vorhersagen des log-returns, die auf historischen Daten beruhen ein fruchtloses Unterfangen

Zeitreihen mit **pandas**

- Flugpassagiere auf **Python**-Befehle untersuchen:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

AirP = pd.read_csv("*AirPassengers.csv")
AirP.head()

AirP["TravelDate"] = pd.DatetimeIndex(AirP["TravelDate"])
AirP.set_index("TravelDate", inplace=True)
AirP.head()

AirP.plot()
plt.xlabel("Reisedatum")
plt.ylabel("Anzahl Passagiere (in 1000)")
```

- Stern beim Einlesen der Datei steht für den Pfad

- Lesen Datei ein

- Überprüfen mit `.head()` ob dies auch richtig gelang

```
AirP =
pd.read_csv("/home/bl/Dropbox/Statistics/Themen/Time_Series_Int
AirP.head()
```

```
##   TravelDate  Passengers
## 0   1/1/1949         112
## 1   2/1/1949         118
## 2   3/1/1949         132
## 3   4/1/1949         129
## 4   5/1/1949         121
```

- Spalte links ist der Index (Bezeichung der Zeilen)

- Wollen Daten als Index

- Spalte `TravelDate` muss zuerst in ein Datumformat umformt werden, das `pandas` versteht

- Die geschieht mit dem Befehl `DatetimeIndex`

```
AirP["TravelDate"] = pd.DatetimeIndex(AirP["TravelDate"])
```

- Daten noch als Index (Bezeichnung der Zeilen) übernehmen:

```
AirP.set_index("TravelDate", inplace=True)
```

- Der Anfang der Tabelle sieht dann so aus:

```
AirP.head()
```

```
##              Passengers
## TravelDate
## 1949-01-01          112
## 1949-02-01          118
## 1949-03-01          132
## 1949-04-01          129
## 1949-05-01          121
```

- Plotten:

```
AirP.plot()
plt.xlabel("Reisedatum")
plt.ylabel("Anzahl Passagiere (in 1000)")

plt.show()
```

Beispiel

- Untersuchen die Vierteljährliche Bierproduktion in Australien (in Megaliter) zwischen März 1956 und Juni 1994

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

AusBeer = pd.read_csv("*AustralianBeer.csv",sep=";",header=0)

AusBeer1 = AusBeer.copy()
AusBeer1.head()
```

```
##   Quarter  megalitres
## 0  1956Q1       284.4
## 1  1956Q2       212.8
## 2  1956Q3       226.9
## 3  1956Q4       308.4
## 4  1957Q1       262.0
```

- Weiter:

```
AusBeer1["Quarter"] = pd.DatetimeIndex(AusBeer["Quarter"])
AusBeer1.set_index("Quarter", inplace=True)

AusBeer1.head()
```

```
##             megalitres
## Quarter
## 1956-01-01       284.4
## 1956-04-01       212.8
## 1956-07-01       226.9
## 1956-10-01       308.4
## 1957-01-01       262.0
```

- Befehl `.describe()`:

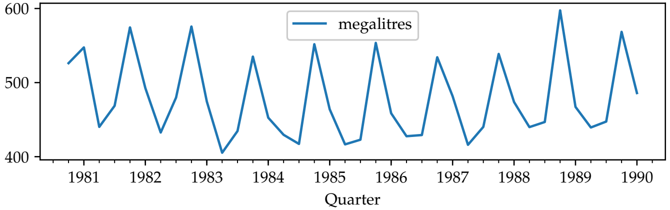
```
AusBeer1.describe()
```

```
##          megalitres
## count  154.000000
## mean   408.267532
## std     97.598588
## min    212.800000
## 25%    325.425000
## 50%    427.450000
## 75%    466.950000
## max    600.000000
```

- `.describe()` zeigt, die Anzahl Werte, der Mittelwert, die Standardabweichung, den minimalen Wert, das untere Quartil, den Median, das obere Quartil und den maximalen Wert

Teilmenge der Zeitreihe

- Können auch eine Teilmenge der Zeitreihe auswählen: Zeit zwischen September 1980 und März 1994:



- Saisonales Verhalten: Peaks gegen Ende des Jahres haben
- Dies entspricht dem australischen Sommer
- Diese Auswahl geschieht mit (amerikanisches Format).

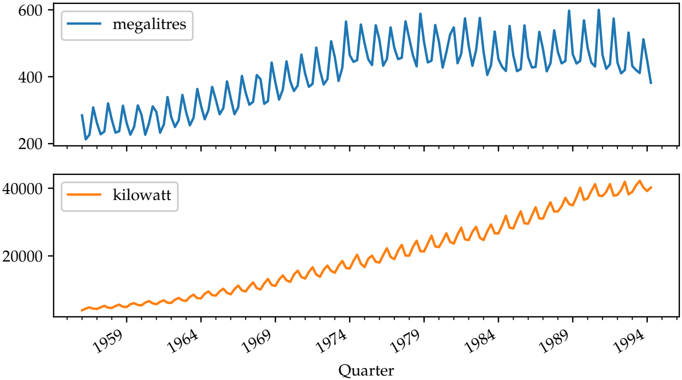
```
AusBeer1.loc["1980-9":"1990-3",:].plot()
```

Multivariate Zeitreihen

- Vierteljährlicher Stromverbrauch (in Millionen kWh in Australien mit vierteljährlicher Bierproduktion vergleichen

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

AusBeer = pd.read_csv("*AustralianBeer.csv",sep=";",header=0)
AusEl = pd.read_csv("*AustralianElectricity.csv",sep=";")

Aussie = AusBeer.copy()

# Hier wird der Datensatz um eine Spalte erweitert
Aussie["kilowatt"] = AusEl["kilowatt"]

Aussie["Quarter"]=pd.DatetimeIndex(Aussie["Quarter"])
Aussie.set_index("Quarter", inplace=True)

Aussie.plot(subplots=True)
```

- Abbildung:



- Plots zeigt einen wachsenden Trend für beide Grössen

- Teilsweise wegen steigender Bevölkerung in Australien von ungefähr 10 Mio auf ungefähr 18 Mio über die gleiche Periode

- Allerdings nahm die Produktion der Elektrizität um den Faktor 7 zu, während sich die Bevölkerung kaum verdoppelt hat.

Elementare Transformationen, Visualisierung und Zerlegung von Zeitreihen

- Analyse von Zeitreihen beginnt mit der Beschreibung, Transformation und Visualisierung der Daten
- Keine Modellierung der Daten
- Keine richtige Vorhersagen machen oder Vertrauensintervalle bilden
- Aber mit diesen Techniken wichtige Einsichten und ein tiefes Verständnis der Daten gewinnen
- Hier:
  - Die wichtigsten Datentransformationen bezüglich Zeitreihen beschreiben
  - Werkzeugkasten mit den wichtigsten Visualisierungen bereit stellen um Zeitreihen zu untersuchen
  - Zeitreihen in saisonale, irreguläre Komponenten bzw. Trend zerlegen.

Datentransformationen

- Oft wünschenswert oder sogar notwendig Zeitreihen zu transformieren bevor Anwendungen auf Modelle und Vorhersagen gemacht werden

- Insbesondere erwarten viele Methoden:
  - *Normalverteilung* oder zumindest *symmetrische Verteilung*
  - Ein *linearer Trend* zwischen den Daten und der Zeit
  - Eine zeitlich *konstante Varianz*

- Bsp: für sehr schiefe oder heteroskedastische (nicht konstante Varianz) Daten oft besser, nicht die originale Reihe brauchen

$$\{x_1, x_2, \ldots\}$$

- Transformierte Reihe

$$\{g(x_1), g(x_2), \ldots\}$$

Box-Cox-Transformationen

- Eine Familie von Transformationen, die besonders geeignet ist, um Schiefe und Varianz zu korrigieren: *Box-Cox-Transformationen*:

Box-Cox-Transformationen

Für eine Zeitreihe mit positiven Werten

$$\{x_1, x_2, \dots\}$$

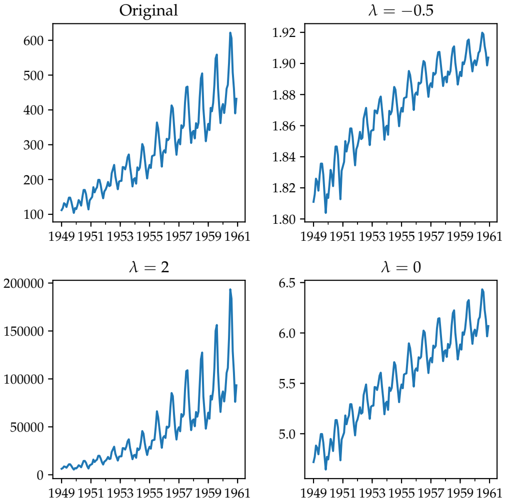
sind die Box-Cox-Transformationen definiert durch

$$g(x) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0. \end{cases}$$

- Ziel: Parameter λ, so wählen, dass die gewünschten Eigenschaften erfüllt sind

Beispiel: `AirPassengers`

- Abbildung:



- Box-Cox-Transformationen für verschiedene Werte von λ
- Originalen Daten der Trend und die Saisonalität offensichtlich ist
- Die Intensität des saisonalen Einflusses (d.h. die Varianz) ist mit den Jahren aber zunehmend
- Stabiles Bild für $\lambda = 0$: Linearer Trend und einen homogener saisonaler Effekt

- Code:

```
def boxcox(x,lambd):
    return np.log(x) if (lambd==0) else (x**lambd-1)/lambd

AirP["l_2"] = boxcox(AirP["Passengers"],2)
AirP["l_0"] = boxcox(AirP["Passengers"],0)
AirP["l_-05"] = boxcox(AirP["Passengers"],-.5)

plt.subplot(221)
AirP["Passengers"].plot()
plt.title("Original")
plt.xlabel("")

plt.subplot(222)
AirP["l_-05"].plot()
plt.title("lambda=-0.5")
plt.xlabel("")

plt.subplot(223)
AirP["l_2"].plot()
plt.title("lambda=2")
plt.xlabel("")

plt.subplot(224)
AirP["l_0"].plot()
plt.title("lambda=0")
plt.xlabel("")

plt.show()
```

Zeitachsentransformation

- Box-Cox-Familie von Transformationen kommt einer Modifikation der *Werte* einer Zeitreihe gleich
- Manchmal notwendig: *Zeitachse* zu transformieren
- Einfachste Beispiel: *Zeitverschiebung* oder *shifting*

Zeitverschiebungstransformation (time-shift)

Sei gegeben Zeitreihe

$$\{x_1, x_2, \dots\}$$

1. Die Zeitverschiebung durch einen *lag* von $k \in \mathbb{Z}$ ist definiert durch

$$g(x_i) = x_{i-k}$$

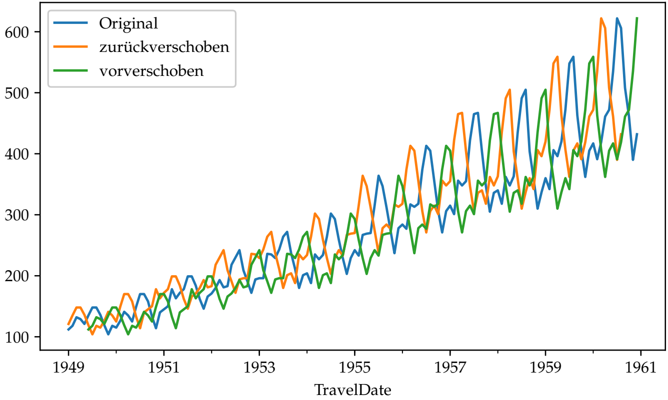
2. Für den Spezialfall $k = 1$ heisst die Zeitverschiebung *backshift*

$$B(x_i) = x_{i-1}$$

- k Schritte in Zeitreihe zurückgehen ($k > 0$) oder vorwärts ($k < 0$)

Beispiel: `AirPassengers`

- Abbildung:



- Zeitverschiebung für $k = 4$ und $k = -5$

- **shift**-Funktion von **pandas**

```
AirP["s_4"] = AirP["Passengers"].shift(4)
AirP["s_-5"] = AirP["Passengers"].shift(-5)

AirP["Passengers"].plot()
AirP["s_4"].plot()
AirP["s_-5"].plot()
plt.legend(["Original","zurueckverschoben","vorverschoben"])

plt.show()
```

- Rückwärtszeitverschiebung wird angewendet, falls *Differenzen* von Zeitreihen berechnet werden:

$$x_i - x_{i-1} = x_i - B(x_i)$$

- Oft Differenzen nehmen mit Box-Cox-Transformationen kombinieren

- Bsp: *log-returns* einer (finanziellen) Zeitreihe sind definiert durch

$$y_i = \log(x_i) - \log(x_{i-1}) = \log\left(\frac{x_i}{x_{i-1}}\right) = \log\left(\frac{x_i - x_{i-1}}{x_{i-1}} + 1\right) \approx \frac{x_i - x_{i-1}}{x_{i-1}}$$

- Letzte Gleichung: Taylor-Reihen Entwicklung der Logarithmus:

$$\log(s+1) = s - s^2/2 + \ldots$$

- D.h.: log-return Zeitreihe y_i nähert den relativen Anstieg der Zeitreihe x_i zu jedem Zeitpunkt an
- Diese Grösse wird oft für finanzielle Anwendungen studiert: Die ursprüngliche Reihe

$$\{x_1, x_2, \ldots\}$$

  scheint ein offensichtliches Muster zu haben
- aber die Reihe

$$\{y_1, y_2, \ldots\}$$

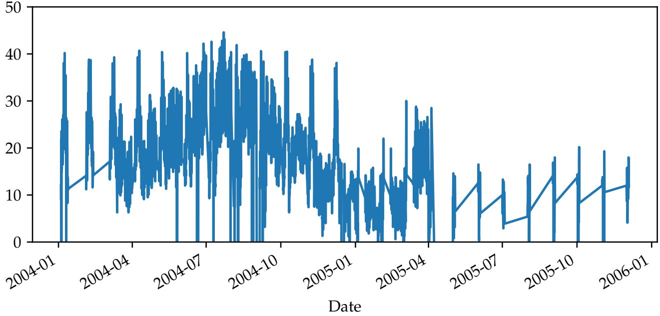
  ist oft sehr zufällig
- Siehe Graphiken zum Tesla Aktienkurs
- Tesla log-return schaut sehr zufällig aus trotz der gelegentlichen starken Fluktuationen, die sehr typisch für diese Art von Daten sind
- Analysten (und andere) versuchen die Wartezeit zwischen solchen Fluktuationen zu modellieren

Visualisierungen

- Wichtigster Teil der deskriptiven Statistik: Geeignete Visualisierung einer Datenmenge (und das gilt nicht nur Zeitreihen)
- *Zeitreihenplot* normalerweise erster Schritt in der Analyse von Zeitreihen
- Beispiele am Anfang dieses Blockes
- Diskrete Zeitpunkte werden durch Linien verbunden, obwohl sie in der Tat nicht stetig sind
- Dies macht `.plot()` von `pandas` automatisch.
- Bleim Plotten von Zeitreihe, dann hängt die Interpretierbarkeit stark von der Anzahl und der Glattheit der Daten ab
- Zeitreihen über grosse Zeiträume und vielen Datenpunkten sollten entsprechend aufgeteilt oder zusammengefasst werden

Beispiel: Luftqualität

- Luftqualität aus Beispiel früher weiter ausbauen
- Stündliche Messungen von verschiedenen Sensoren → nur Temperatur
- Abbildung zeigt vollständigen Plot

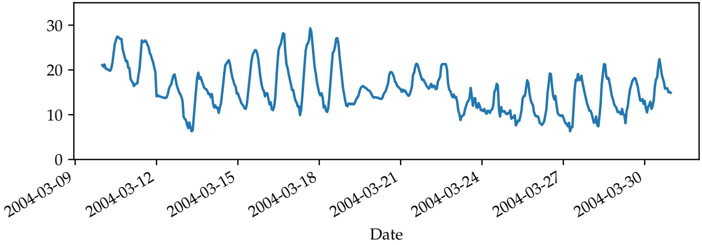


- Code:¶

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

AirQ = pd.read_csv("*AirQualityUCI.csv",sep=";",decimal=",")

AirQ1 = AirQ.copy()

# pandas kennt das Zeitformat in der Tabelle nicht:
#Punkt muss durch . ersetzt werden
AirQ1["Time"] = AirQ1["Time"].str.replace(".","-")

AirQ1["Date"] = pd.DatetimeIndex(AirQ1["Date"]+" "+AirQ1["Time"
AirQ1.set_index("Date", inplace=True)

# Einige Wert der Temperatur sind -200. Diese Zeilen werden weg
AirQ1 = AirQ1[AirQ1["T"] > -20]

AirQ1["T"].plot()

plt.show()
```

- Fokus auf 20 Tage, um Verhalten der Temperatur in grösserem Detail sehen

- Abbildung:



- Code:

```
AirQ4 = AirQ1.loc["2004-3-10":"2004-3-30","T"]
AirQ4.plot()
```

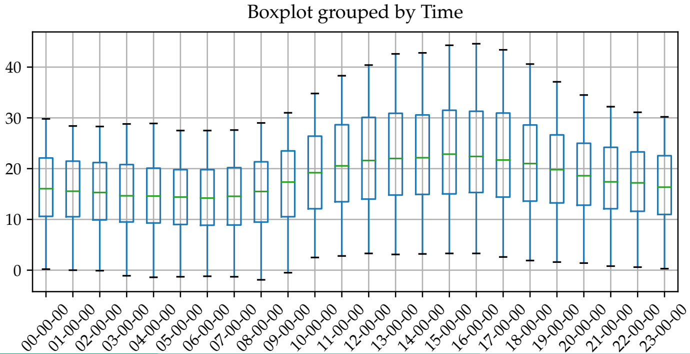
Figure: Temperatur über 20 Tage im März

Beispiel: Boxplot

- Zeitreihe **AirQ4** von vorherenthalten die stündlichen Lufttemperaturen über einen Zeitraum von 20 Tagen im März 2004 in einer italienischen Stadt

- Daten für jede Stunde über diese 20 Tage betrachten

- Abbildung: Boxplot dieser Daten nach Stunde gruppiert:



- Code:

```
AirQ1.boxplot("T",by="Time")

plt.show()
```

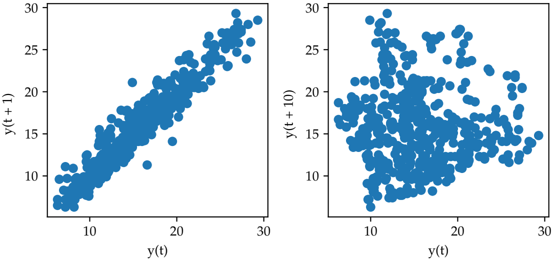
- Zusatz `by="Time"`: Für jede Stunde wird ein Boxplot erzeugt

lagged scatterplot

- Nützlicher Ansatz um graphisch die Korrelation von aufeinanderfolgenden Beobachtungen zu visualisieren
- Dabei wird die ursprüngliche Zeitreihe gegen eine zeitverschobene Zeitreihe aufzeichnet, also die Datenpunkte (x_i, x_{i-k})
- Die wird in `pandas` mit der `lag_plot`-Funktion gemacht

Beispiel: Luftqualität

- Lufttemperatur über die 20 Tage in Italien
- lag-plot für $k = 1$ und $k = 10$



- Streudiagramm mit lag 1 zeigt lineares Muster zeigt
- Korrelation zwischen benachbarten stündlichen Temperaturen
- lag von 10 Stunden zeigt ein ziemlich unspezifisches Streudiagramm
- Keine Korrelation von Temperaturen, die 10 Stunden auseinander liegen

- Code:

```
from pandas.plotting import lag_plot

plt.subplot(121)
lag_plot(AirQ4)

plt.subplot(122)
lag_plot(AirQ4, 10)

plt.show()
```

Zerlegung von Zeitreihen

- Beispiele früher: Viele Zeitreihen werden dominiert durch einen Trend und/oder saisonale Effekte

- Modelle in diesem Abschnitt basieren deswegen auf diesen Komponenten

- Einfaches additive Zerlegungsmodell ist gegeben durch

$$x_k = m_k + s_k + z_k$$

  wobei

  - k der Zeitindex
  - x_k die beobachteten Daten
  - m_k der Trend
  - s_k der saisonale Effekt
  - z_k ein Fehlerterm (i.A. Folge aus *korrelierten* zufälligen Variablen mit Mittelwert 0)

- Datensatz `AirPassengers`: saisonaler Effekt nimmt mit dem Trend
- In diesem Fall ist ein multiplikatives Modell geeigneter:

$$x_k = m_k \cdot s_k + y_k$$

- Wenn das Rauschen auch noch multiplikativ ist, dann ist der Logarithmus von x_k wieder linear

$$\log(x_k) = \log(m_k) + \log(s_k) + \log(y_k)$$

Bewegendes Mittel (moving average)

- Sehr einfache Methode den Trend m_k und den saisonalen Effekt s_k abzuschätzen ist mittels eines *moving average filter*:

Moving average filter

Sei $\{x_1, x_2, \ldots, x_n\}$ eine Zeitreihe und $p \in \mathbb{N}$.
Dann ist der *moving average filter* der Länge p definiert durch:

- Falls p ungerade, dann $p = 2l + 1$ und die gefilterte Folge ist definiert durch:

$$g(x_i) = \frac{1}{p}(x_{i-l} + \cdots + x_i + \cdots + x_{i+l})$$

- Falls p is gerade, dann $p = 2l$. Gefilterte Folge ist definiert durch:

$$g(x_i) = \frac{1}{p}\left(\frac{1}{2}x_{i-l} + x_{i-l+1} + \cdots + x_i + \cdots + x_{i+l-1} + \frac{1}{2}x_{i+l}\right)$$

Der Wert p wird bezeichnet als *Fensterbreite* oder *window width*.

- D.h.: moving average filter ersetzt i-ten Wert der Zeitreihe durch den Mittelwert der nächsten p Nachbarn

- Falls p ungerade ist, so ist das Fenster symmetrisch um x_i

- Falls p gerade, dann konstruiert man ein Fenster der Länge $p + 1$ (was dann ungerade ist), zählt dann aber jeweils nur die Hälfte an den Endpunkten

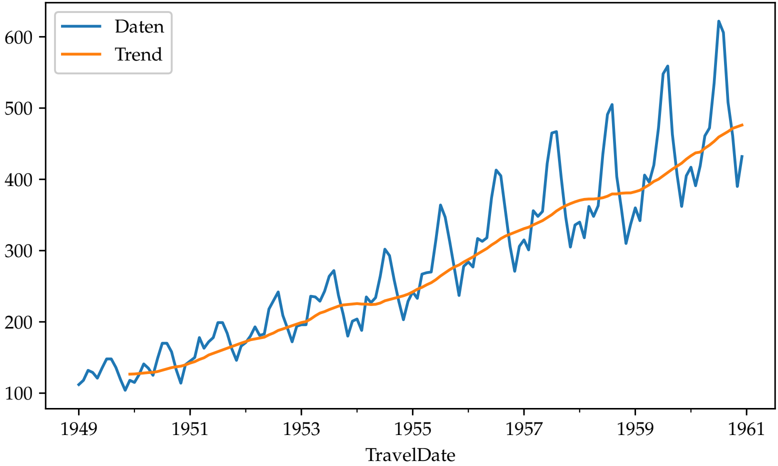
- Falls eine Zeitreihe ein Frequenz p hat (i.e. $p = 12$ für monatliche Daten, dann kann die Trendkomponente der Zeitreihe abgeschätzt werden durch einen moving average filter mit einer Fensterbreite p

- Da an jedem Zeitpunkt genau über eine ganze Periode gemittelt wird, so verschwinden die saisonalen Effekte und die Trendkomponente bleibt übrig

- Dies resultiert im Trendschätzer $\widehat{m}_k$

Beispiel: `AirPassenger`

- Schätzen Trend mit dem moving average filter



- `rolling(window=12).mean()`
- Code:

```
AirP["Trend"] = AirP["Passengers"].rolling(window=12).mean()

AirP["Passengers"].plot()
AirP["Trend"].plot()

plt.legend(["Daten","Trend"])

plt.show()
```

- Um den saisonalen additiven Effekt abzuschätzen, berechnet man

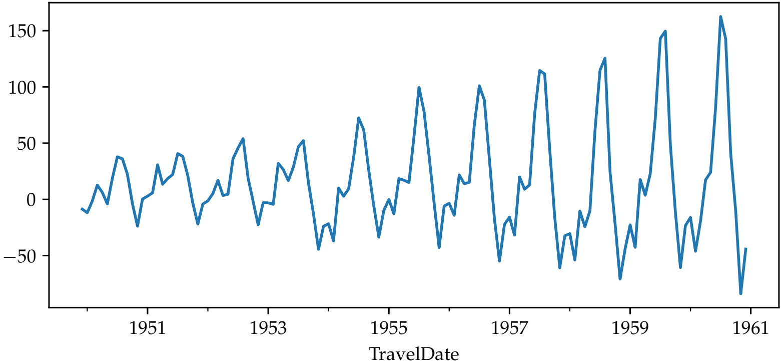
$$\widehat{s}_k = x_k - \widehat{m}_k$$

- Nun wird die Zeitreihe $\widehat{s}_k$ für jeden Punkt in einem Zyklus (Monat) gemittelt

- Erhalten eine einzige Schätzung für jeden Zykluspunkt (Monat).

- Datensatz `AirPassengers` und subtrahieren den geschätzten Trend

```
AirP["Season"] = AirP["Passengers"]-AirP["Trend"]

AirP["Season"].plot()

plt.show()
```

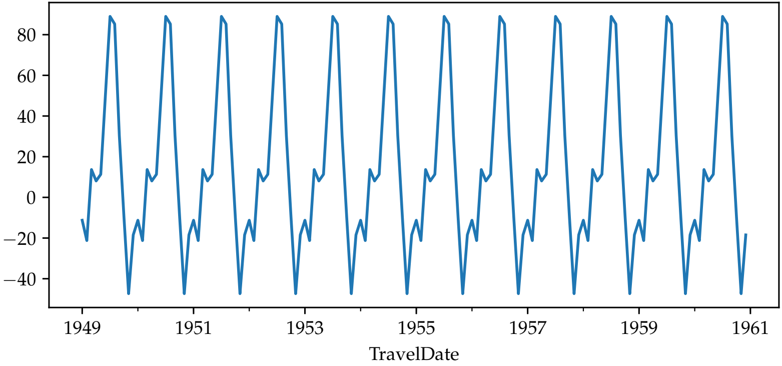


- Durchschnittliche Saisonalität

- Mittelwert der entsprechenden Monate genommen

```
# AirP["Season"] wird in eine Matrix umgewandelt
# mit den Monaten als Spalten (Jahre als Zeilen)

AirP2 = AirP["Season"].values.reshape((12,12))

# Entlang der Spalten (axis=0) wird  der Mittelwert genommen
# nanmean bedeutet, die NaN werden ignoriert

ave = np.nanmean(AirP2,axis=0)

# Der Vektor ave wird verzwölfacht,
# damit er wieder die gleiche Länge hat, wie AirP["Season"]

AirP["Season_ave"] = np.tile(A=ave, reps=12)

AirP["Season_ave"].plot()

plt.show()
```

- Plot:


80
60
40
20
0
−20
−40
1949
1951
1953
1955
1957
1959
1961
TravelDate

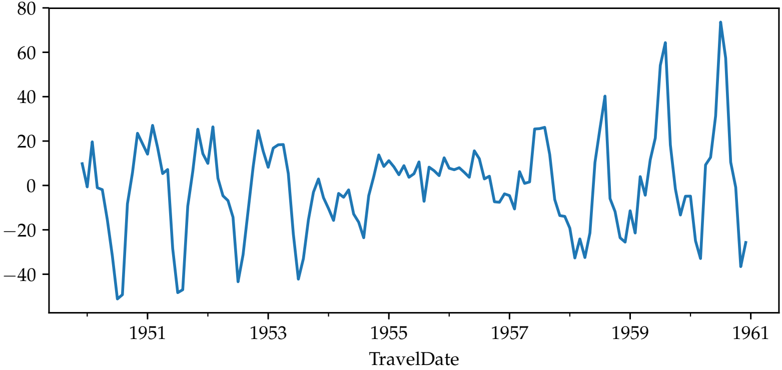
- Schlussendlich subtrahieren Schätzungen für Trend und Saisonalität
- Erhalten die Restterm (Residuen)

$$\widehat{r}_i = x_i - \widehat{m}_i - \widehat{s}_i$$

- Restterm sollte aus möglicherweise korrelierten Zufallsvariablen ohne Struktur/Periodizität bestehen

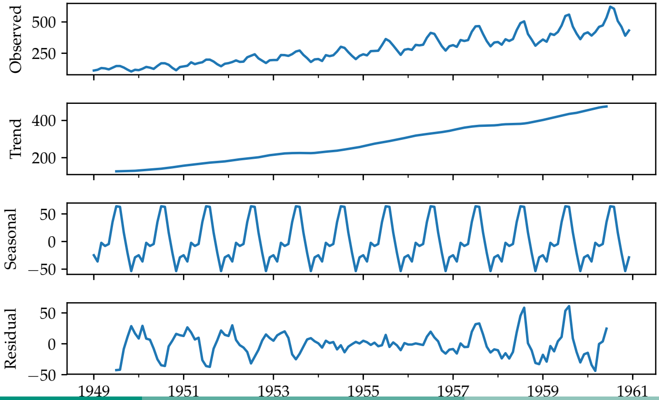
- Datensatz `AirPassengers` und subtrahieren den geschätzten Trend und Saisonalität von

```
AirP["Residual"] = AirP["Season"]- AirP["Season_ave"]

AirP["Residual"].plot()

plt.show()
```



- Abbildungen vorher lassen sich mit einem einzigen Befehl erzeugen

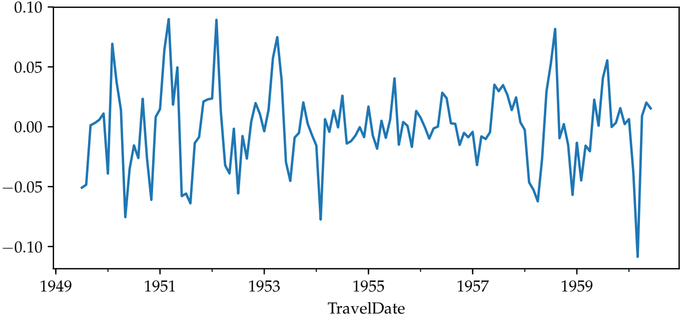
```
from statsmodels.tsa.seasonal import seasonal_decompose

seasonal_decompose(AirP["Passengers"], model="additive",
freq=12).plot()

plt.show()
```



- Restterm zeigt offensichtlich ein nichtzufälliges Verhalten des Restterms $\widehat{r}$
- Grund: lineares Zerlegungsmodell stimmt hier nicht
- Versuchen, die Schritte oben mit dem *Logarithmus* der `AirPassengers` durchzuführen
- Entspricht einem multiplikativen Modell

- Code:

```
seasonal_decompose(np.log(AirP["Passengers"]),
model="add").resid.plot()

plt.show()
```



- Im geschätzten Restterm der log-Daten ist der nichtzufällige Teil wesentlich vermindert

Bemerkungen

- Das Verfahren oben ist zwar sehr einfach, hat aber Nachteile:
  - Es fehlt Robustheit gegenüber Ausreissern in den Daten
  - Die Saisonalität wird konstant über die Zeit angenommen
- Es gibt bessere Lösungen, wie das STL-Verfahren (*seasonal decomposition of time series by loess*)
- Dies ist aber in Python nicht implementiert