Einführung in die Statistik mit Python

RS-eco

Biodiversity & Global Change Lab
Technische Universität München

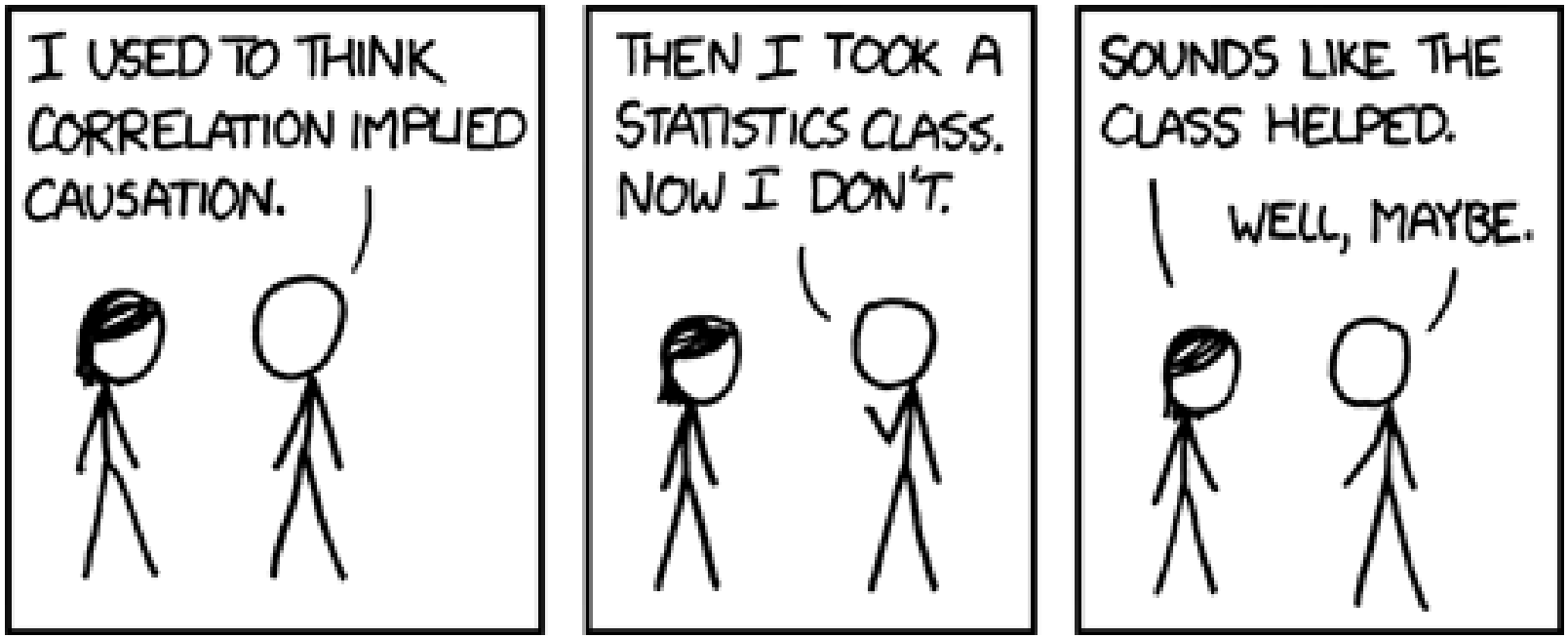
rs-eco@posteo.de

16. November 2019

Korrelation

Korrelation

- Korrelation misst die Assoziation zwischen zwei Variablen.
- Im Gegensatz dazu macht die lineare Regression die Vorhersage eines Wertes auf Grundlage des Wertes einer anderen Variablen
- Es gibt drei verschiedene Korrelations-Maße: **Pearson Produkt-Moment-Korrelation, Spearman Rank Korrelation & Kendall-tau Korrelation**



Pearson Produkt-Moment-Korrelation

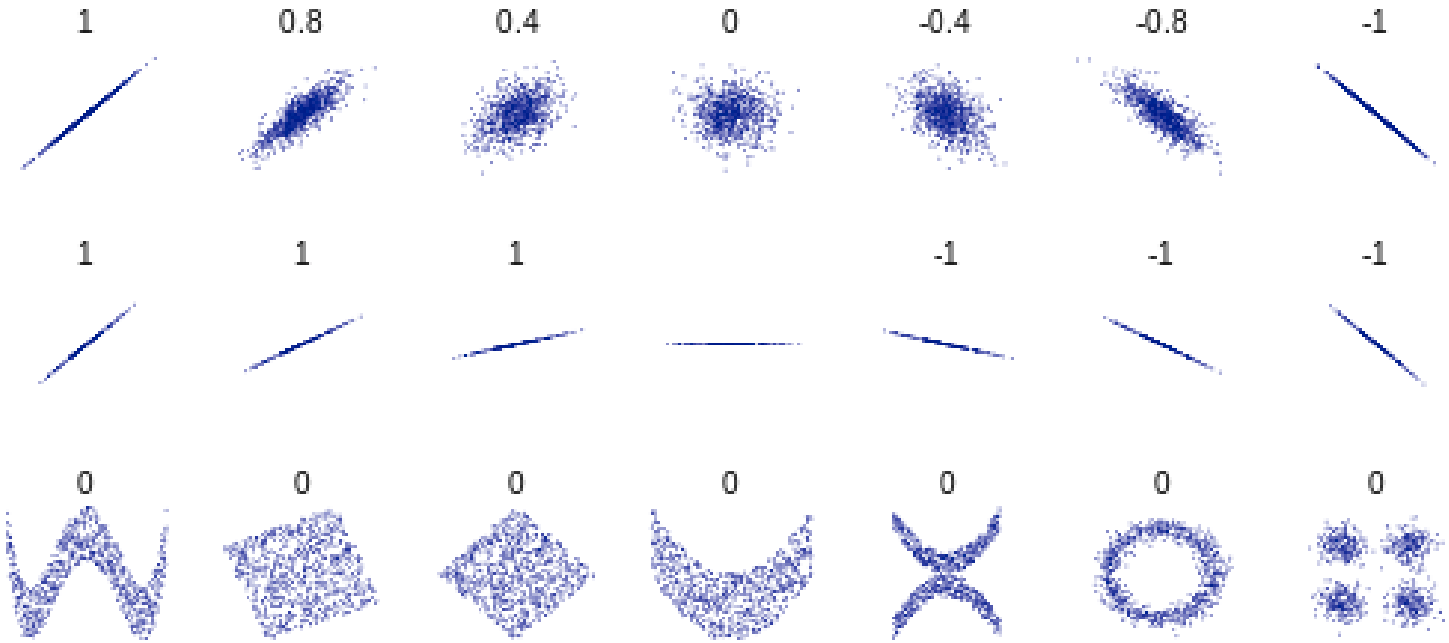
- Möchte man den linearen Zusammenhang zweier Merkmale bestimmen, nutzt man hierzu meist den Korrelationskoeffizienten.
- ein dimensionsloses Zusammenhangsmaß, das mit dem Symbol r abgekürzt wird und Werte zwischen -1 und 1 annehmen kann.
- Bei einem positiven linearen Zusammenhang, nähert sich der Koeffizient dem Wert „1“ an und beide Merkmale tendieren in eine Richtung.
- Bei einem negativen linearen Zusammenhang, also der Annäherung an den Wert „-1“, haben die Merkmale eine gegensätzliche Tendenz.
- Kein Zusammenhang besteht, wenn sich der Koeffizient dem Wert „0“ annähert.
- An dem Koeffizienten lassen sich also die Richtung des Zusammenhangs und die Stärke ablesen, aber nicht die Steigung!
- Berechnet wird der Korrelationskoeffizient, indem man die Kovarianz beider zu vergleichenden Merkmale durch das Produkt der Standardabweichungen beider Merkmale teilt.

Pearson Produkt-Moment-Korrelation

```
import math

def correlation(x, y):
  n = len(x)

  # Mittelwerte berechnen
  x_mn = sum(x) / n
  y_mn = sum(y) / n

  # Varianzen berechnen
  var_x = (1 / (n-1)) * sum(map(lambda xi: (xi - x_mn) ** 2 , x))
  var_y = (1 / (n-1)) * sum(map(lambda yi: (yi - y_mn) ** 2 , y))

  # Standardabweichungen berechenen
  std_x, std_y = math.sqrt(var_x), math.sqrt(var_y)

  # Kovarianz berechnen
  xy_var = map(lambda xi, yi: (xi - x_mn) * (yi - y_mn), x, y)
  cov = (1 / (n-1)) * sum(xy_var)

  # Korrelationskoeffizient nach Pearson
  r = cov / (std_x * std_y)
  return float(f"{r:.3f}")

  # Wohnungsgröße in Quadratmetern, Mietkosten
  sqrm = [20, 30, 40, 50, 60]
  cost = [300, 400, 600, 700, 1000]

  print(correlation(sqrm, cost)) #out: 0.981
```

Pearson Produkt-Moment-Korrelation

- Die Variablen müssen mindestens intervallskaliert (bzw. metrisch) sein.
- Beide Merkmal müssen einem linearen Zusammenhang folgen und Normal verteilt sein.
- Ist der Zusammenhang etwa exponentiell (Linearitätsbedingung nicht erfüllt), so ist das Ergebnis für den Zusammenhang relativ gering (obwohl beide Werte zum Beispiel in dieselbe Richtung weisen).



Spearman Rank Korrelation

- Der Pearson Korrelationskoeffizient besitzt den Nachteil, gegenüber Ausreißern empfindlich zu sein.
- Die Spearman Rangkorrelation ist ebenfalls ein Maß zur Berechnung eines bivariaten Zusammenhangs, aber hat diesen Nachteil nicht.
- Es lassen sich dabei jedoch bereits ordinalskallierte Daten nutzen, da hier die Ränge der Werte gebildet und eingesetzt werden (und nicht die Werte selbst).
- Bei der Rangvergabe werden die ursprünglichen Daten zunächst aufsteigend sortiert und anschließend die entsprechenden Ränge verteilt. Wird ein Rang mehrfach vergeben, dann wird der Mittelwert der Ränge gebildet.
- Zur Berechnung der Rang-Korrelation wird dieselbe Formel wie für die Pearson-Korrelation genutzt (Kovarianz geteilt durch das Produkt der Standardabweichungen), wobei man jedoch die korrespondierenden Rangwerte der Daten zur Berechnung einsetzt.

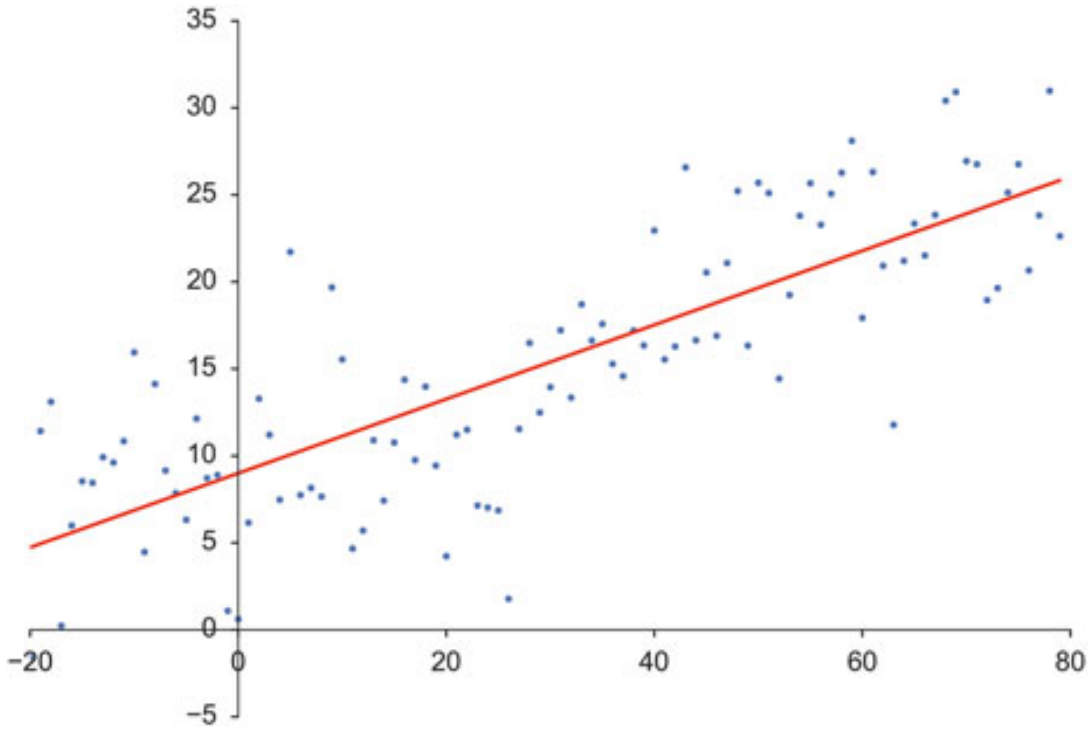
Spearman Rank Korrelation

```
from collections import Counter

def ranking(array):
  counts = Counter(array)
array_sorted = sorted(set(array))
rank = 1
rankings = {}
for num in array_sorted:
  count = counts.get(num)
if count == 1:
  rankings[num] = rank
rank += 1
else:
  rankings[num] = sum(range(rank, rank+count)) / count
rank += count
return [float(rankings.get(num)) for num in array]

# Beispiel für eine Rangkorrelation
eng = [12, 12, 3, 6, 10, 4, 15, 8]
deu = [14, 14, 5, 4, 11, 8, 10, 3]

eng_rank = ranking(eng) # [6.5, 6.5, 1.0, 3.0, 5.0, 2.0, 8.0, 4.0]
deu_rank = ranking(deu) # [7.5, 7.5, 3.0, 2.0, 6.0, 4.0, 5.0, 1.0]

correlation(eng_rank, deu_rank) #out: 0.639
```

Statistische Modellierung

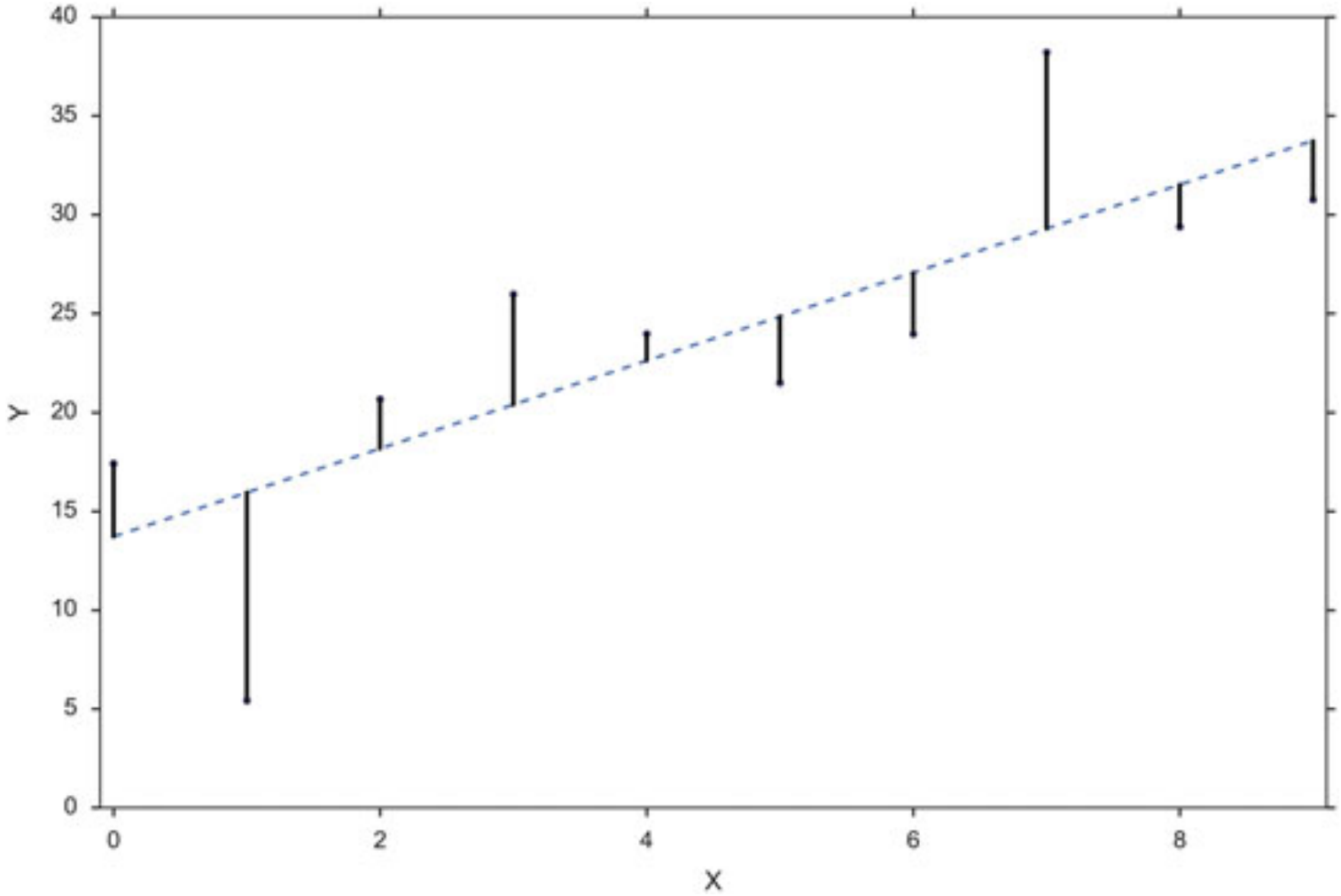
Lineare Regressions Modelle

- Wir können lineare Regressionen verwenden, um den Wert einer Variablen durch die Werte von einer oder mehreren anderen Variablen vorherzusagen.
- Wir suchen nach der am besten geeigneten Linie: **y = x*coeff + intercept + e**, wobei der Koeffizient die Steigung oder Neigung der Linie und e das Beobachtungsrauschen ist.



Lineare Regressions Modelle

- Die Residuen sind die Unterschiede zwischen Beobachtungswerten und Prognosewerten.



Lineare Regressions Modelle

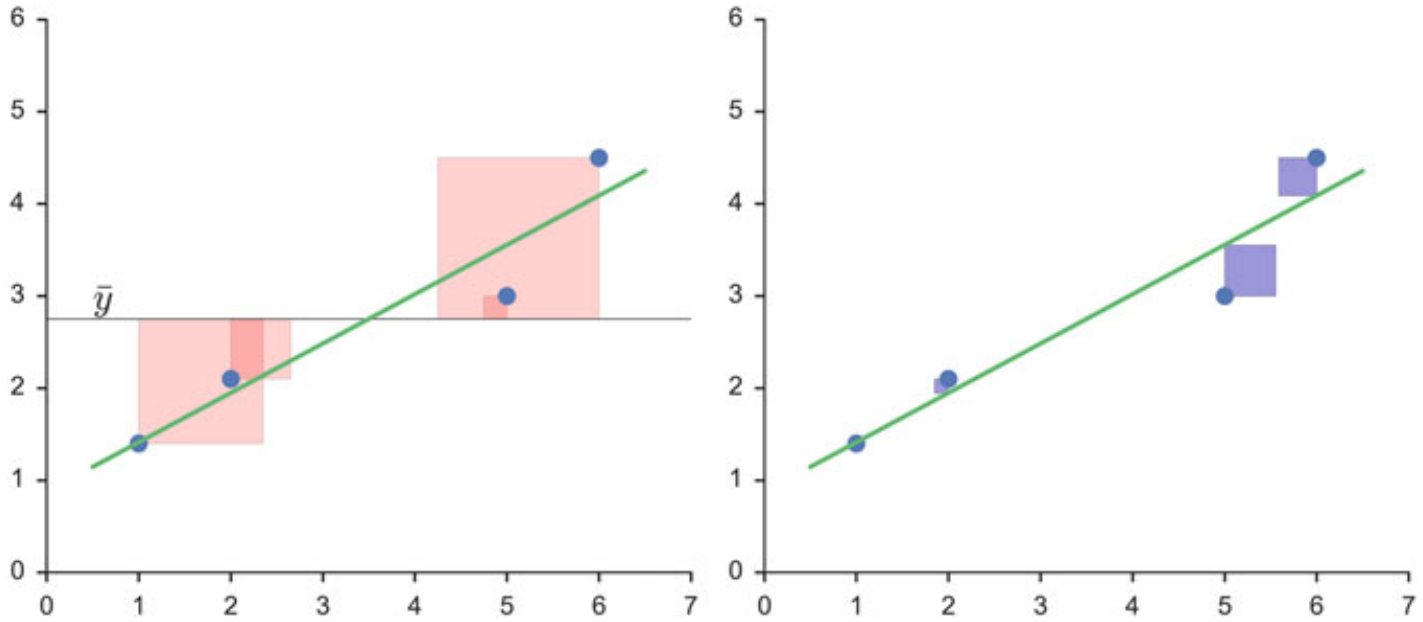
- Die Residuen sind die Unterschiede zwischen Beobachtungswerten und Prognosewerten.
- Die lineare Regressionsgleichung wird gelöst, um die Quadratsumme der Residuen zu minimieren.
- Lineare Regression wird manchmal auch als Ordinary Least-Squares (OLS) Regression bezeichnet.
- Beachtet, dass im Gegensatz zur Korrelation diese Beziehung zwischen x und y nicht mehr symmetrisch ist: Es wird davon ausgegangen, dass die x-Werte genau bekannt sind und dass die gesamte Variabilität in den Residuen liegt.

Modellkoeffizienten

- Die Koeffizienten oder Gewichte der linearen Regression sind in **result.params** enthalten und werden als Pandas-Objekt ausgegeben
- Die Standardfehler der Koeffizienten erhält man durch die Berechnung der Kovarianz-Varianz-Matrix für die geschätzten Koeffizienten der Prädiktorvariablen.
- Die Standardfehler sind die Quadratwurzeln der Elemente auf der Hauptdiagonale dieser Kovarianzmatrix.

Determinationskoeffizient

- Die "Variabilität" der Daten wird durch verschiedene Summen von Quadraten gemessen:
- SS_{mod} = Modellsumme der Quadrate/Erklärte Summe der Quadrate, oder die Summe der Quadrate für die Regression
- SS_{res} = Restsumme der Quadrate/die Summe der Fehler-Quadrate
- SS_{tot} = Gesamtsumme der Quadrate (= Stichprobenvarianz*(n - 1))



Determinationskoeffizient

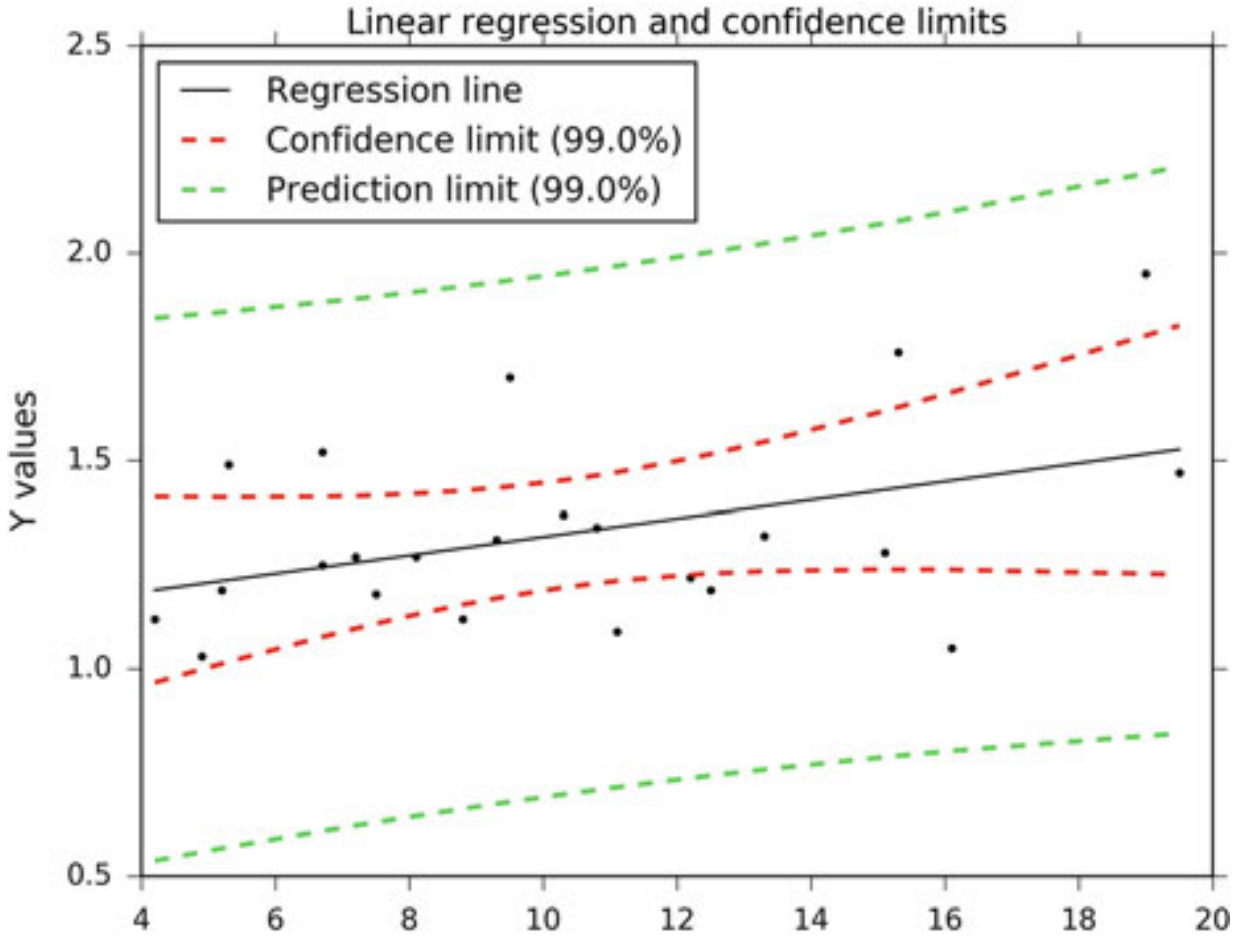
- Die "Variabilität" der Daten wird durch verschiedene Summen von Quadraten gemessen:
- SS_{mod} = Modellsumme der Quadrate/Erklärte Summe der Quadrate, oder die Summe der Quadrate für die Regression
- SS_{res} = Restsumme der Quadrate/die Summe der Fehler-Quadrate
- SS_{tot} = Gesamtsumme der Quadrate (= Stichprobenvarianz*(n - 1))
- Für lineare Regressionsmodelle: $SS_{mod} + SS_{res} = SS_{tot}$
- Mit diesen Ausdrücken läst sich ist das Bestimmtheitsmaß bestimmen: $R_2 = 1 - SS_{res}/SS_{tot} = SS_{mod}/SS_{tot}$.
- Für eine einfache lineare Regression ist der Bestimmtheitsmaßstab R^2 das Quadrat des Korrelationskoeffizienten r.
- R^2 ist einfacher zu interpretieren als der Korrelationskoeffizient r: Werte von R^2 nahe 1 entsprechen einer engen Korrelation, Werte nahe 0 einer schlechten Korrelation.

Konfidenzintervall

- Das Konfidenzintervall wird aus dem Standardfehler, dem p-Wert und dem kritischen Wert aus einem t-Test mit N - k Freiheitsgraden aufgebaut, wobei k die Anzahl der Beobachtungen und P die Anzahl der Modellparameter, d.h. die Anzahl der Prädiktorvariablen ist.
- Das Konfidenzintervall ist der Wertebereich, in dem wir erwarten würden, dass wir den Parameter von Interesse finden, basierend auf dem, was wir beobachtet haben.
- Es gibt zwei Konfidenzintervalle, ein Konfidenzintervall für den variablen Koeffizienten des Prädiktors und eins für den konstanten Term.
- Ein kleineres Konfidenzintervall deutet darauf hin, dass wir über den Wert des geschätzten Koeffizienten oder der konstanten Laufzeit zuversichtlich sind.
- Ein größeres Konfidenzintervall deutet darauf hin, dass es mehr Unsicherheit oder Varianz in der geschätzten Laufzeit gibt

Konfidenzintervall

- Die beiden Arten von Konfidenzintervallen (eines für die Daten und eines für die angepassten Parameter) gibt es auch bei Linienanpassungen.



Lineare Regression in Python

```
import pandas as pd
import numpy as np

## First, we generate simulated data according to the model:
x = np.linspace(-5, 5, 20)
np.random.seed(1)
y = -5 + 3*x + 4 * np.random.normal(size=x.shape)
data = pd.DataFrame({'x': x, 'y': y})

## Then we specify an OLS model and fit it:
from statsmodels.formula.api import ols
model = ols("y ~ x", data).fit()

## We can inspect the various statistics derived from the fit:
print(model.summary())
```

Lineare Regression in Python

OLS Regression Results

Dep. Variable:	y	R-squared:	0.804
Model:	OLS	Adj. R-squared:	0.794
Method:	Least Squares	F-statistic:	74.03
Date:	Wed, 13 Nov 2019	Prob (F-statistic):	8.56e-08
Time:	19:52:49	Log-Likelihood:	-57.988
No. Observations:	20	AIC:	120.0
Df Residuals:	18	BIC:	122.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-5.5335	1.036	-5.342	0.000	-7.710	-3.357
x	2.9369	0.341	8.604	0.000	2.220	3.654

Omnibus:	0.100	Durbin-Watson:	2.956
Prob(Omnibus):	0.951	Jarque-Bera (JB):	0.322
Skew:	-0.058	Prob(JB):	0.851
Kurtosis:	2.390	Cond. No.	3.03

Model-Ergebnisse

- n ist die Anzahl der Beobachtungen und k die Anzahl der Regressionsparameter, z.B. bei einer geraden Linie k = 2
- Die Freiheitsgrade (DF) des Modells sind die Anzahl der Prädiktoren oder erklärenden Variablen.
- Die DF der Residuen ist die Anzahl der Beobachtungen minus die Freiheitsgrade des Modells, minus 1 (für den Offset).
- Die meisten der in der Zusammenfassung aufgeführten Werte sind über das Ergebnisobjekt verfügbar.
- So kann beispielsweise der Wert R^2 durch **result.rsquared** ermittelt werden.

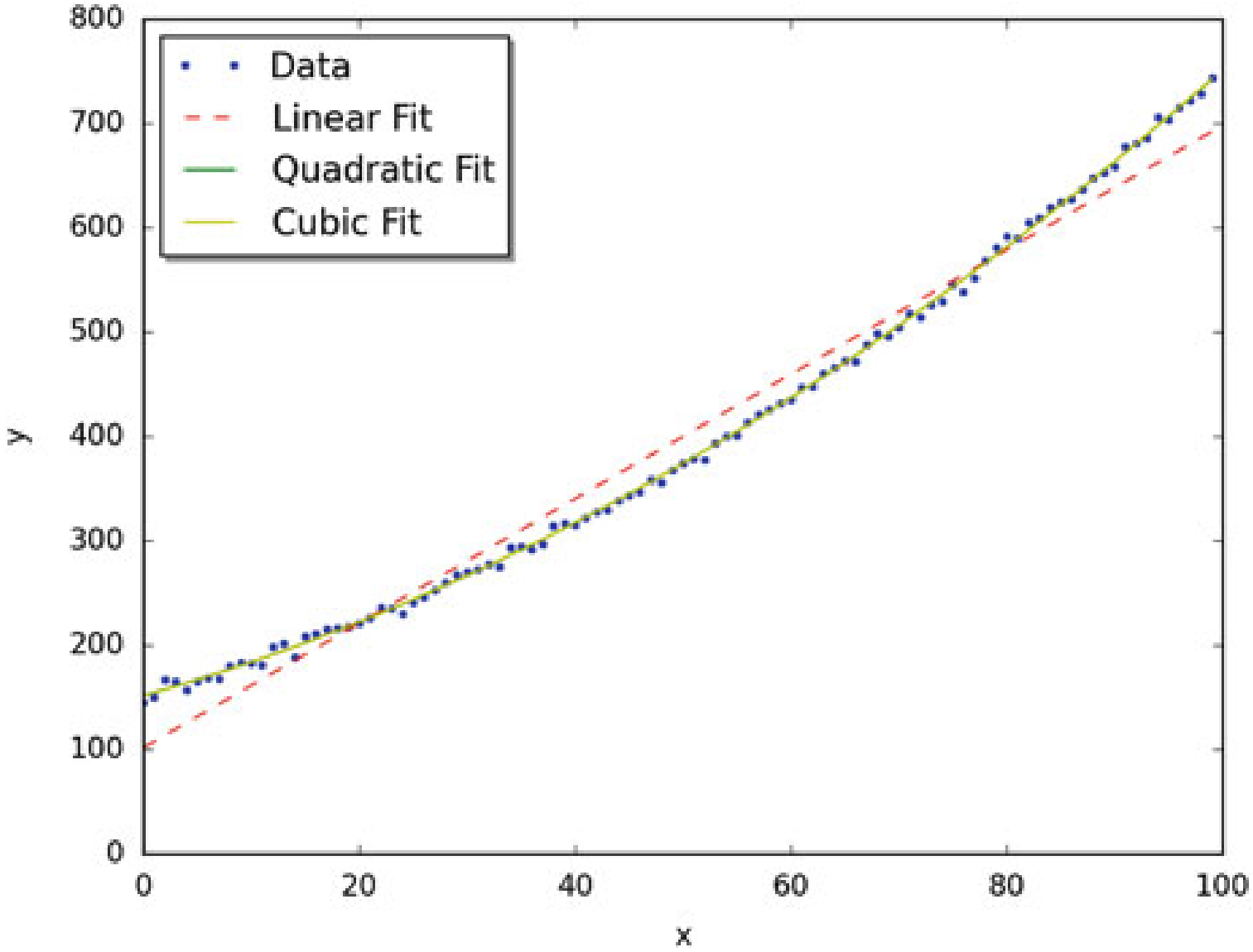
Angepasster R^2 Wert

- Für die Beurteilung der Qualität von Modellen bevorzugen viele Forscher den angepassten R 2-Wert, der häufig mit dem Balken über dem R angezeigt wird, der für eine große Anzahl von Parametern im Modell bestraft wird.

Quadratisches Polynom

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.formula.api as smf

''' Generate a noisy, slightly quadratic dataset '''
x = np.arange(100)
y = 150 + 3*x + 0.03*x**2 + 5*np.random.randn(len(x))
df = pd.DataFrame({'x':x, 'y':y})

# Fit the models, and show the results
Res1 = smf.ols('y~x', df).fit()
Res2 = smf.ols('y ~ x+I(x**2)', df).fit()
Res3 = smf.ols('y ~ x+I(x**2)+I(x**3)', df).fit()

print('The coefficients from the linear fit: {0}'.format(Res1.params))
print('The coefficients from the quadratic fit: {0}'.format(Res2.params))
print('The coefficients from the cubic fit: {0}'.format(Res3.params))
```

Quadratisches Polynom



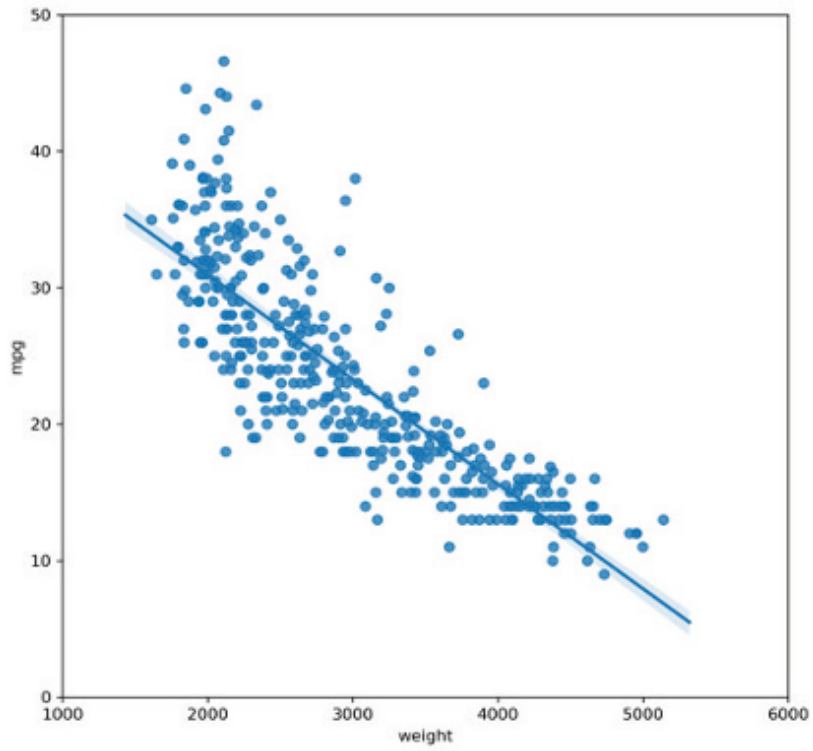
Finden der besten Lösung

- Wenn wir herausfinden wollen, welches das "bessere" Modell ist, können wir die Werkzeuge von statsmodels nutzen.
- Das Akaike Information Criterion (AIC) kann verwendet werden, um die Qualität des Modells zu beurteilen: Je niedriger der AIC-Wert, desto besser das Modell.

```
'''Solution with the tools from statsmodels'''
import statsmodels.api as sm

# Manually look at the results (AIC)
print(Res1.summary2())

# Or extract the AIC of all three Models
print('The AIC-value is {0:4.1f} for the linear fit,\n
      {1:4.1f} for the quadratic fit, and \n
      {2:4.1f} for the cubic fit'.format(Res1.aic, Res2.aic, Res3.aic))
```

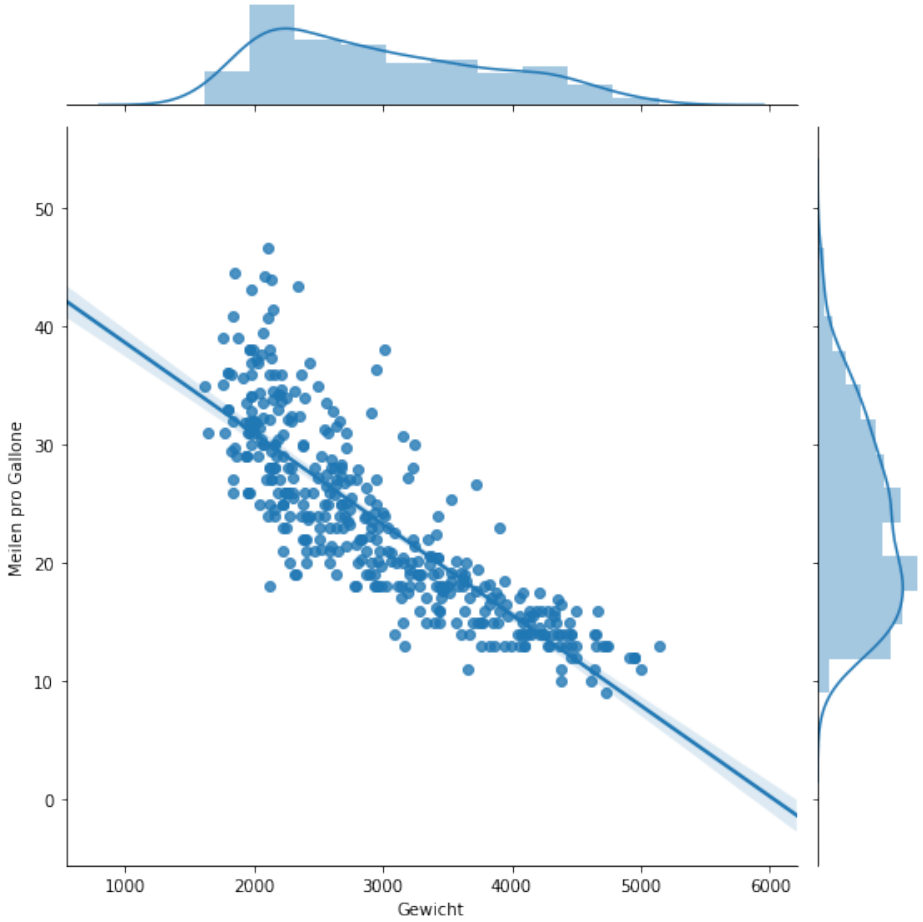
Zusammenhänge mit Regplot und Jointplot darstellen

```
import seaborn as sns

# Datensatz
cars = sns.load_dataset("mpg")

# Figure und Axes Objekt anlegen
fig, ax = plt.subplots(figsize=(8, 8))

# Grafik anlegen
conf = {"x": "weight", "y": "mpg",
  "data": cars, "ax": ax}
sns.regplot(**conf)

# Grafik anzeigen
plt.show()
```



Zusammenhänge mit Regplot und Jointplot darstellen

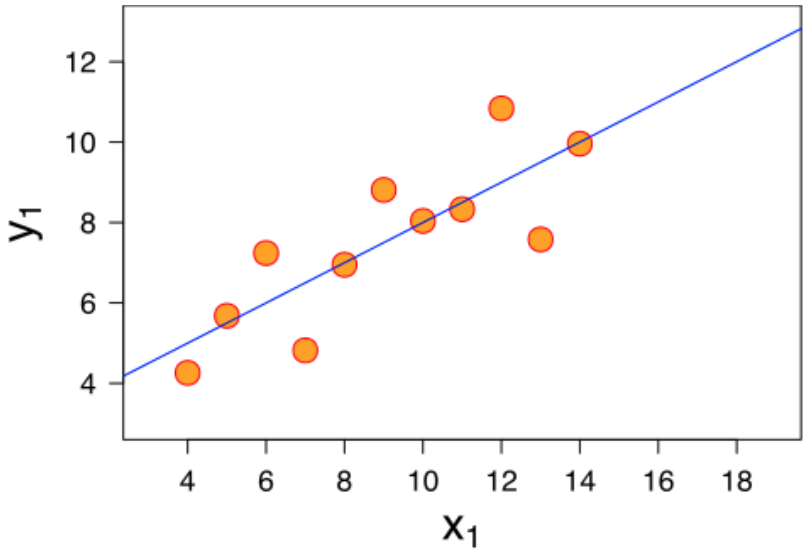
```
import seaborn as sns

# Datensatz
cars = sns.load_dataset("mpg")

conf = {"x": "weight", "y": "mpg",
  "kind": "reg", "data": cars,
  "height": 8, "ratio": 6}

f = sns.jointplot(**conf)
f.set_axis_labels("Gewicht",
                  "Meilen pro Gallone")

# Grafik anzeigen
plt.show()
```

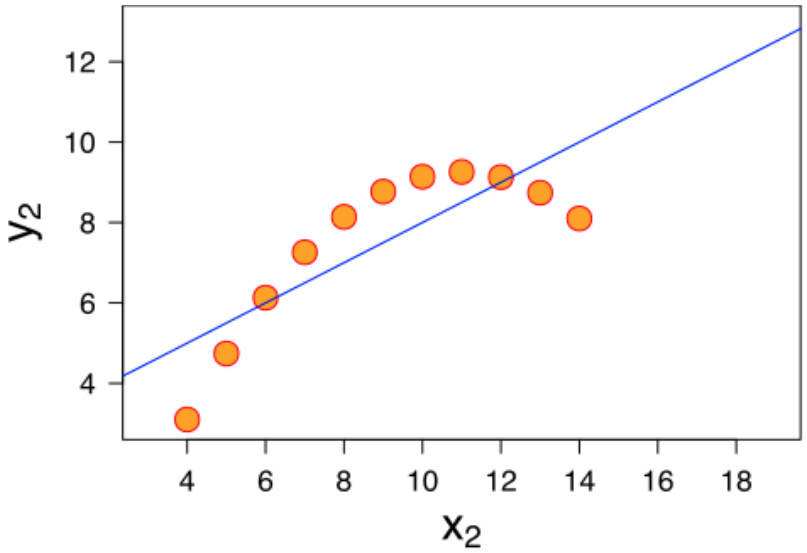


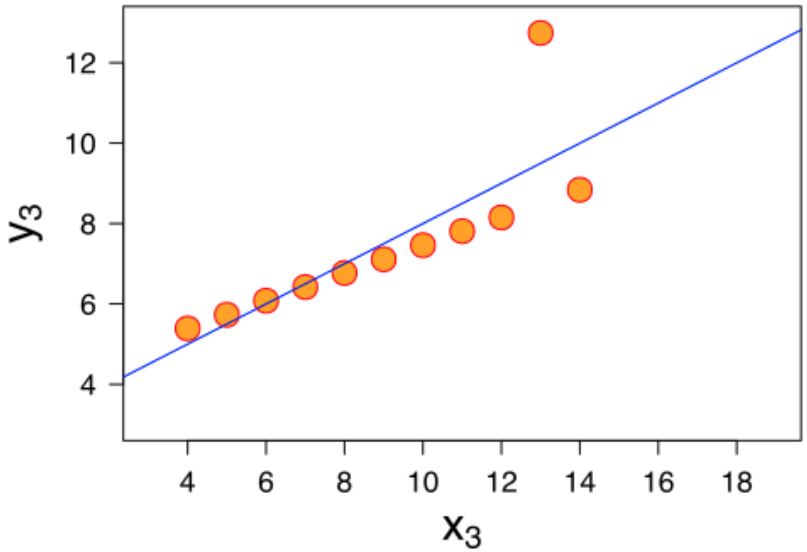
Grundannahmen linearer Regressionsmodelle

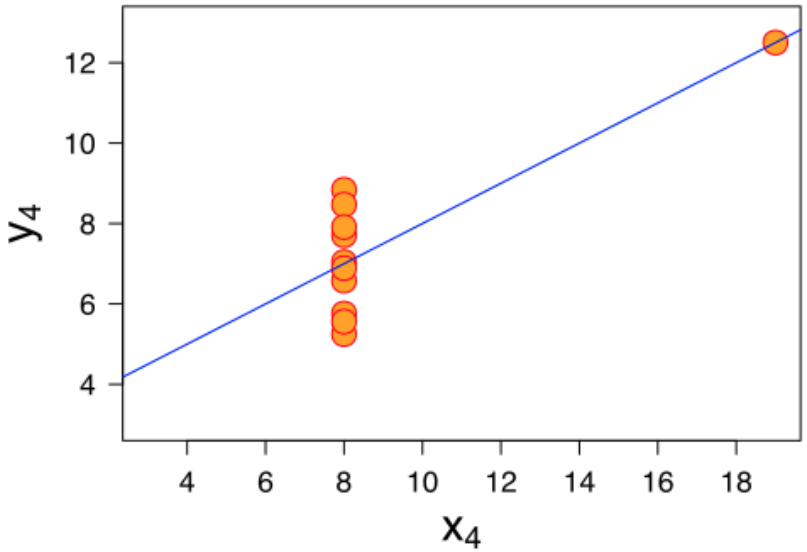
- Lineare Regressionsmodelle machen eine Reihe von Annahmen über die Prädiktorvariablen, die Antwortgrößen und deren Beziehung.
  1. Die unabhängigen Variablen (z.B. x) sind genau bekannt.
  2. Gültigkeit: Am wichtigsten ist, dass die Daten, die Sie analysieren, der Forschungsfrage zugeordnet werden sollten, die Sie zu beantworten versuchen. Das klingt offensichtlich, wird aber oft übersehen oder ignoriert, weil es unangenehm sein kann. So beschreibt beispielsweise eine lineare Regression eine quadratische Kurve nicht richtig.
  3. Additivität und Linearität: Die wichtigste mathematische Annahme des Regressionsmodells ist, dass seine deterministische Komponente eine lineare Funktion der einzelnen Prädiktoren ist.
  4. Gleiche Fehlervarianz
  5. Unabhängigkeit der Fehler von den Werten der unabhängigen Variablen.
  6. Unabhängigkeit der unabhängigen Variablen.

Anscombe-Quartett









Bootstrapping

- Eine weitere Art der Modellierung ist das Bootstrapping.
- Manchmal haben wir Daten, die eine Verteilung beschreiben, wissen aber nicht, um welche Art von Verteilung es sich handelt. Was können wir also tun, wenn wir z.B. Vertrauenswerte für den Mittelwert herausfinden wollen?
- Die Antwort ist Bootstrapping.
- Bootstrapping ist ein Schema des Resamplings, bei dem zusätzliche Proben wiederholt aus dem Ausgangsmuster entnommen werden, um Schätzungen der Variabilität zu erhalten.
- In einem Fall, in dem die Verteilung des Eingangsmusters unbekannt ist, ist das Bootstrapping besonders hilfreich, da es Informationen über die Verteilung liefert.
- Die Anwendung von Bootstrapping in Python wird durch das Paket scikits.bootstrap von Constantine Evans (http://github.org/cgevans/scikits-bootstrap) erheblich erleichtert.

Übung

Übung

- Ihr findet sämtliche Kursunterlagen unter: http://github.com/RS-eco/pyStats
- Öffnet das Jupyter Notebook für Tag 4 und:
  - Ladet den Datensatz **schoko.csv** in Python
  - Erstellt eine Korrelation zwischen Kakaogehalt und Nussanteil (visuell und statistisch)
  - Erklärt das Gewicht der Schokolade den Preis? Prüft dies visuell und statistisch
  - Plottet eine lineare Regression mit Konfidenzintervallen
  - Unterscheidet sich die Beziehung zwischen Gewicht und Schokolade für Bio und Nicht-Bio Schokolade?

Multivariate Datenanalyse

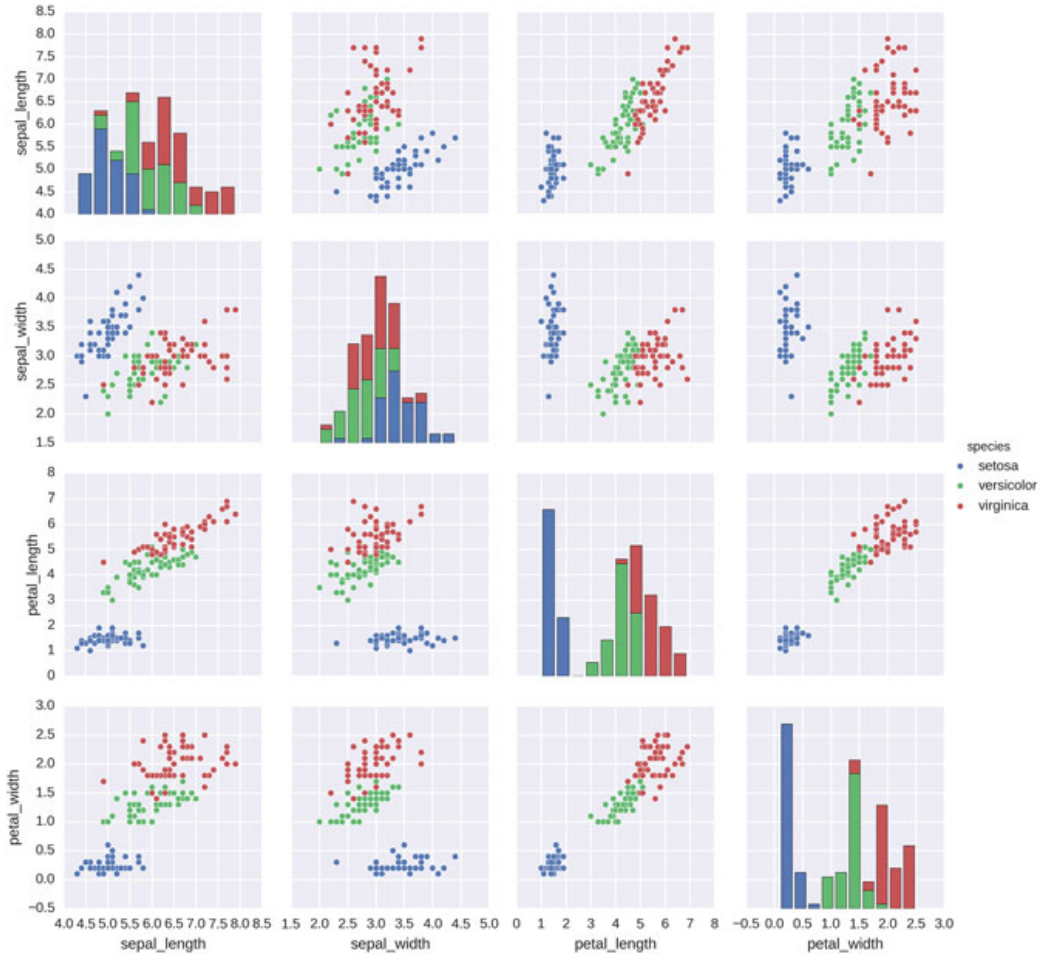
Visualisierung von multivariaten Korrelationen

- Für 3-6 Variablen, die miteinander in Beziehung stehen können, können wir mit einer Scatterplot-Matrix die Zusammenhänge zwischen den verschiedenen Variablen visualisieren.

```
# Load seaborn library
import seaborn as sns
sns.set()

# Load data
df = sns.load_dataset("iris")

# Create pairplot
sns.pairplot(df, hue="species", size=2.5)
```

Visualisierung von multivariaten Korrelationen

- Für 3-6 Variablen, die miteinander in Beziehung stehen können, können wir mit einer Scatterplot-Matrix die Zusammenhänge zwischen den verschiedenen Variablen visualisieren.



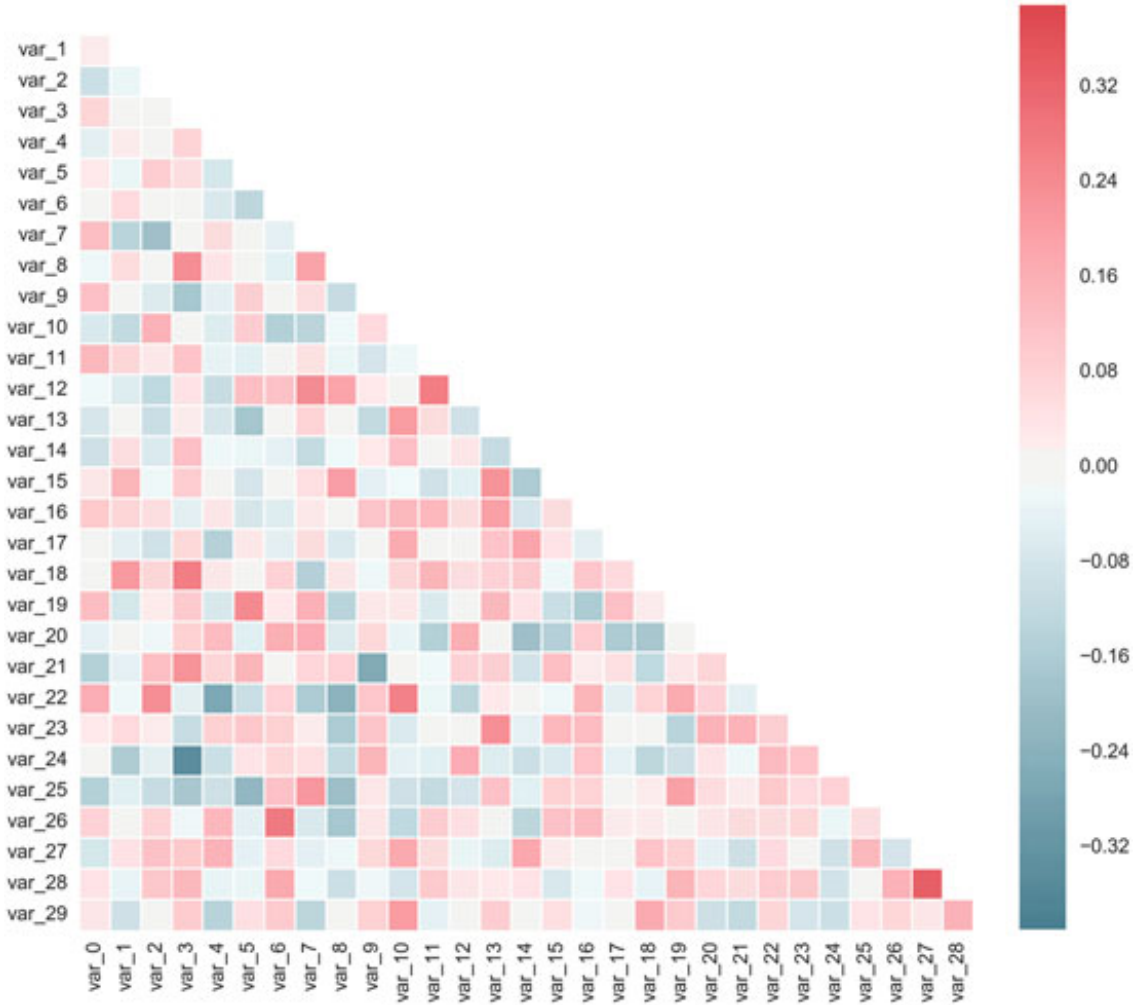
Korrelationsmatrix

- Eine elegante Möglichkeit, die Korrelation zwischen einer großen Anzahl von Variablen zu visualisieren, ist die Korrelationsmatrix.

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="darkgrid")

# Set seed for the random numbr generation
rs = np.random.RandomState(33)
# Create normally distribued dummy data,
# simulating 100 recordings from 30 different variables
d = rs.normal(size=(100, 30))

f, ax = plt.subplots(figsize=(9, 9))
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Calculate and visualize cross correlation
# between each possible combination of variables
sns.corrplot(d, annot=False, sig_stars=False,
             diag_names=False, cmap=cmap, ax=ax)
f.tight_layout()
```

Korrelationsmatrix

- Eine elegante Möglichkeit, die Korrelation zwischen einer großen Anzahl von Variablen zu visualisieren, ist die Korrelationsmatrix.



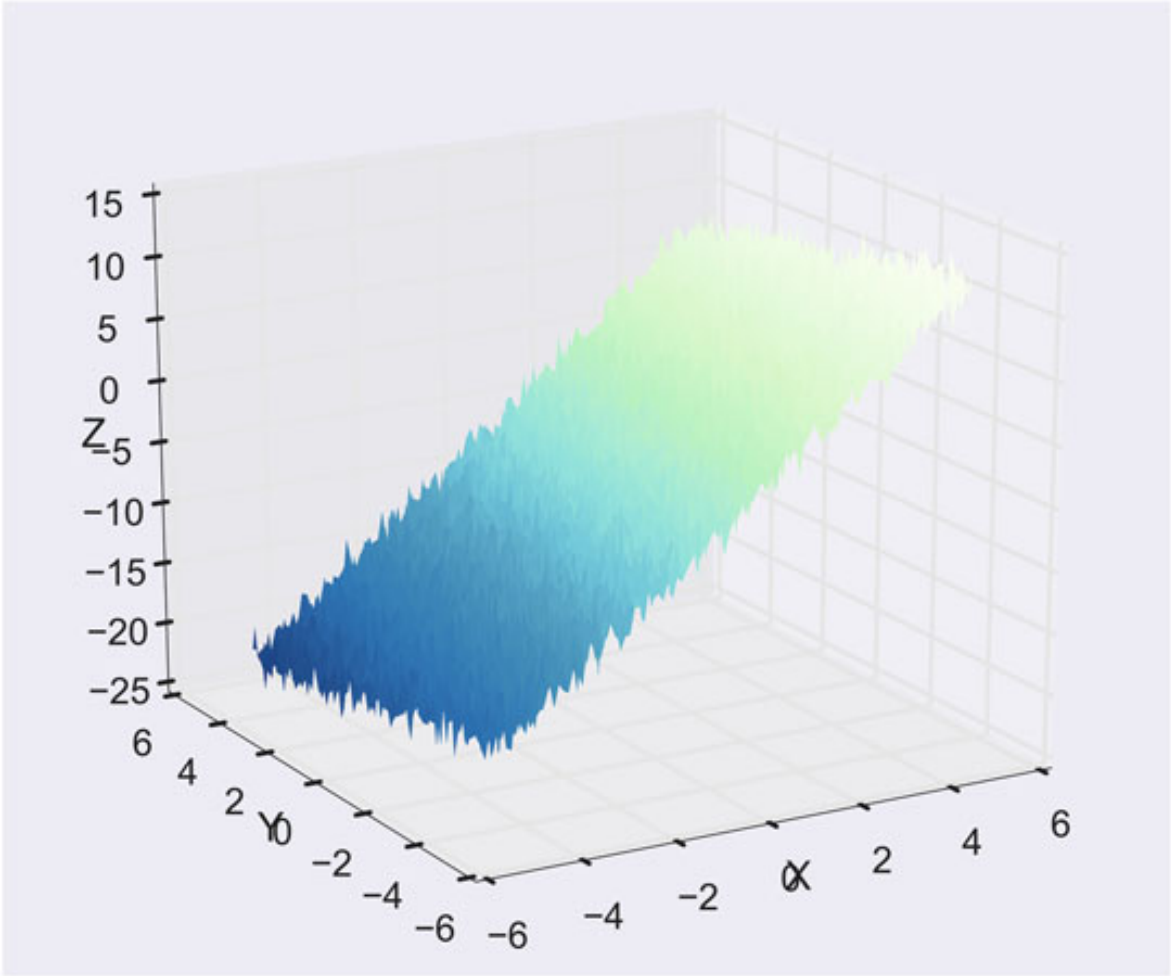
Multilineare Regression

- Für wirklich unabhängige Variablen, ist die multiple Regression eine einfache Erweiterung der einfachen linearen Regression.
- Multilineare Regression = ein lineares Modell, das eine Variable z (die abhängige Variable) mit 2 Variablen x und y erklärt:
  $z = x * c_1 + y * c_2 + i + e$

```
# Load iris dataset
data = sns.load_dataset("iris")

# Run model with two independent variables
model = ols('''sepal_width ~ species +
            petal_length''', data).fit()

# Print model summary
print(model.summary())
```

Multilineare Regression

- Ein solches Modell kann in 3D als Anpassung einer Ebene an eine Wolke von (x, y, z) Punkten gesehen werden.



Post-hoc Analyse: Varianz-Analyse (ANOVA)

- Im obigen Beispiel möchten wir testen, ob die Länge der Blütenblätter zwischen Iris versicolor und Iris virginica unterschiedlich ist, nachdem wir den Effekt der Blütenbreite entfernt haben.
- Dies kann als Test der Differenz zwischen dem Koeffizienten von Iris versicolor und Iris virginica in obigem linearen Modell formuliert werden (es ist eine Varianzanalyse, ANOVA).
- Dazu schreiben wir einen Kontrastvektor auf die geschätzten Parameter: Wir wollen "name[T.versicolor] - name[T.virginica]", mit einem "F-Test" testen.

```
print(model.f_test([0, 1, -1, 0]))
```

Daten transformieren: z-Transformation

- Wenn Werte auf unterschiedlichen Skalen gemessen wurden, können wir diese auch mithilfe der sogenannten z-Transformation bzw. z-Standardisierung vergleichbar machen.
- Diese Werte weisen die Besonderheit auf, dass sie einen Mittelwert von „0“ und eine Standardabweichung von „1“ besitzen. Ist der resultierende z-Wert für ein beobachtetes Datum also größer als „0“ (bzw. kleiner), lässt sich sagen, dass der Wert „überdurchschnittlich“ ist (bzw. unter dem Durchschnitt liegt).
- Für die Berechnung wird zum einen der Mittelwert benötigt sowie die Standardabweichung.
- Die jeweiligen z-Werte ergeben sich, indem man von jeder Beobachtung den Mittelwert der Beobachtungen abzieht und diesen Wert wiederum durch die Standardabweichung teilt.

Daten transformieren: z-Transformation

```
import math

def z_transform(array):
  n = len(array)
  mn = sum(array)/n
  var = (1/(n-1))*sum(map(lambda xi: (xi-mn)**2, array))
  std = math.sqrt(var)
  z = [(xi-mn)/std for xi in array]
  return z

pizza_de = [4.99, 7.99, 5.99, 4.99, 6.99]
pizza_us = [5.74, 9.19, 6.89, 5.74, 8.04]

z_de = z_transform(pizza_de)
z_us = z_transform(pizza_us)
z_de == z_us
```

Clusteranalyse

- Es ist die Aufgabe, eine Menge von Objekten so zu gruppieren, dass Objekte in derselben Gruppe oder demselben Cluster einander ähnlicher sind als diejenigen in anderen Gruppen oder Clustern.
- Es ist eine gängige Technik zur statistischen Datenanalyse.
- Die Clusteranalyse kann durch verschiedene Algorithmen erreicht werden, die sich stark unterscheiden können.
- Moderne Clusterbegriffe umfassen Gruppen mit geringen Abständen zwischen den Clustermitgliedern, dichte Bereiche des Datenraums, Intervalle oder bestimmte statistische Verteilungen.
- Daher ist die Clusteranalyse als solche keine triviale Aufgabe. Es handelt sich um eine interaktive mehrkriterielle Optimierung, die Trial-and-Error beinhaltet.
- Oft ist es auch notwendig, die Datenvorverarbeitung und die Parameter von Modellen oder Algorithmen zu modifizieren, bis das Ergebnis die gewünschten Eigenschaften erreicht.

Clusteranalyse

- Die Clusteranalyse, das Clustering von Probanden oder Variablen, wird zunächst durch ein Ähnlichskeits- und Distanzmaß zwischen zwei Probanden und später zwischen zwei Clustern durchgeführt.
- Diese Gruppen können mit hierarchischen oder nicht-hierarchischen Techniken durchgeführt werden.
- Die Cluster-Analyse ermöglicht es dem Individuen mit unterschiedlichen Merkmalen in einer großen Datenprobe schnell zu identifizieren.
- Wenn der Leser die Anzahl der Cluster kennt, ist ein nicht-hierarchisches Clustering ausreichend. Andernfalls muss eine hierarchische Analyse vorab durchgeführt werden.
- Hierarchisches Clustering: **Single linkage, Complete linkage, Average group linkage, Average linkage within groups, Centroid clustering, Ward method**
- Nicht-hierarchisches Clustering: **K-Means**

Distanz- und Ähnlichkeitsmaße

- Die Identifizierung von natürlichen Clustern von Probanden oder Variablen erfordert, dass die Ähnlichkeit zwischen diesen explizit gemessen werden muss.
- Es gibt mehrere Ähnlichkeitsmessungen (oder Nähe) oder Unähnlichkeit (oder Distanz), die je nach Variablentyp (Intervall, Frequenz oder Nominalwert) verwendet werden können.
- In der Clusteranalyse sind die häufigsten Kennzahlen: **Euklidische Distanz & Jaccardähnlichkeit**

Euklidischer Abstand

- Dies ist der Abstand zwischen zwei Punkten (p, q) in jeder beliebigen Dimension des Raumes
- Es ist das am Häufigsten verwendete Distanzmaß.
- Wenn die Daten dicht oder kontinuierlich sind, ist dies das beste Näherungsmaß.

```
### Euclidean distance matrix

# Import libraries
import scipy.spatial.distance as sp

# Filtering survey data
data = data[["sepal_length", "sepal_width", "petal_length", "petal_width"]]

# Calculate distance
dist = sp.pdist(data, 'euclidean')
df_dist = pd.DataFrame(sp.squareform(dist))
print(df_dist.head())
```

Jaccard-Koeffizient

- Dies ist ein Standardindex für binäre Variablen.
- Es ist definiert als der Quotient zwischen dem Schnittpunkt und der Vereinigung der paarweise verglichenen Variablen zwischen zwei Objekten.

```
### Jaccard similarity

# Import libraries
import scipy.spatial.distance as sp

# Calculate Jaccard similarity
dist = sp.pdist(data, 'jaccard')

# Turn into DataFrame
df_dist = pd.DataFrame(sp.squareform(dist))

# Print head
print(df_dist.head())
```

Ähnlichkeitsmaße für Variablen

- Wenn die Clusteranalyse darauf abzielt, Variablen zu gruppieren (und nicht Probanden oder Gegenstände), sind die geeigneten Ähnlichkeitsmaße die Korrelationskoeffizienten der Stichprobe.
- Bei kontinuierlichen Variablen ist der Pearson-Korrelationskoeffizient am besten geeignet.
- Für ordinale Variablen sollte der Spearman-Korrelationskoeffizient verwendet werden. Für nominale Variablen sollte der Leser den phi-Koeffizienten verwenden.

Hierarchische Clusteranalyse

- Hierarchische Techniken beziehen sich auf aufeinanderfolgende Schritte der Aggregation der betrachteten Subjekte, individuell.
- So, angesichts einer Reihe von N Elementen, die gebündelt werden sollen, und einer N $\times$ N Entfernungs-(oder Ähnlichkeits-)Matrix, ist der primäre Prozess des hierarchischen Clusterings:
  1. Beginnen Sie damit, jedes Element seinem Cluster zuzuordnen, so dass, wenn es N Elemente hat, es jetzt N Cluster hat, die jeweils nur ein Element enthalten. Die Abstände (Ähnlichkeiten) zwischen den Clustern sollen den Abständen (Ähnlichkeiten) zwischen den Elementen entsprechen, die sie enthalten.
  2. Finden Sie das nächstgelegene (am ähnlichsten) Clusterpaar und führen Sie es zu einem einzigen Cluster zusammen, so dass es nun einen Cluster weniger hat.
  3. Berechnen Sie Abstände (Ähnlichkeiten) zwischen dem neuen Cluster und jedem der alten Cluster.
  4. Wiederholen Sie die Schritte 2 und 3, bis alle Elemente zu einem einzigen Cluster der Größe N zusammengefasst sind.

Hierarchische Clusteranalyse

- Hierarchische Methoden von Clustern unterscheiden sich meist darin, wie diese Abstände berechnet werden.
- Die am häufigsten verwendeten Methoden sind: Single-Linkage, Complete Linkage, Average Group Linkage, Average Linkage within Groups, Centroid Clustering & Ward method
- Da es mehrere verfügbare Methoden gibt, ist die Existenz von Vor- und Nachteilen bei der Anwendung jeder einzelnen von ihnen sichtbar.
- Da die "beste" Methode zur Durchführung von hierarchischem Clustering nicht existiert, schlagen einige Autoren (Marôco, 2011) vor, verschiedene Methoden gleichzeitig zu verwenden.
- Wenn also alle Methoden ähnlich interpretierbare Lösungen liefern, kann man daraus schließen, dass die Datenmatrix natürliche Gruppierungen aufweist.

Hierarchische Clusteranalyse in Python

```
### Hierarchical clustering
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
Z_single = linkage(X, 'single') # Single linkage
Z_complete = linkage(X, 'complete') # Complete linkage

# Single-linkage clustering (method= ''single'')

# Calculate full dendrogram
plt.figure(figsize=(25, 10))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(Z_single,
           leaf_rotation=90., # rotates the x axis labels
           leaf_font_size=8., # font size for the x axis labels
)
plt.show()

# Complete-linkage clustering (method= ''complete'')
plt.figure(figsize=(25, 10))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(Z_complete,
           leaf_rotation=90., # rotates the x axis labels
           leaf_font_size=8., # font size for the x axis labels
)
plt.show()
```

Partitionierende Clusteranalyse

- Partitionierende Clusteringverfahren sind dazu gedacht, Elemente (und nicht Variablen) in einem Satz von Clustern zu gruppieren, deren Anzahl a priori definiert ist.
- Diese Methoden gelten schnell für Arrays mit großen Datenmengen, da es nicht notwendig ist, in jedem Schritt des Algorithmus eine neue Unähnlichkeitsmatrix zu berechnen und zu speichern.
- Es gibt verschiedene nicht-hierarchische Methoden, die sich vor allem dadurch unterscheiden, wie sie die erste Aggregation von Items in Clustern entfalten und wie die neuen Abstände zwischen den Zentroiden der Clustern und dem Item berechnet werden.
- Eine der Standardmethoden in den meisten statistischen Programmen ist das **K-means**.

k-Means Clustering in Python

```
### Clustering with k-means

from scipy.cluster.vq import whiten
from sklearn.cluster import KMeans

# Normalize variables values
std_survey_data = whiten(data, check_finite=True)

# K-means cluster analysis
kmeans = KMeans(n_clusters=3).fit(std_survey_data)
centroids = kmeans.cluster_centers_

# Plot data coloured by Clusters
plt.scatter(data['sepal_length'], data['sepal_width'],
            c=kmeans.labels_.astype(float), s=50, alpha=0.5)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
```

Übung

Übung

- Ihr findet sämtliche Kursunterlagen unter: http://github.com/RS-eco/pyStats
- Öffnet das Jupyter Notebook für Tag 4 und:
  - Ladet den Datensatz **schoko.csv** in Python
  - Erstelle einen pairplot() für Preis, Kakaogehalt und Kategorie und zusätzlich "Bio" als hue
  - Erstelle eine Korrelationsmatrix für alle Variablen
  - Erstelle eine multilineare Regression für Kakaogehalt, Anzahl der Inhaltsstoffe und Preis und plottet diesen Zusammenhang in 3D
  - Erstelle eine hierarchische Cluster-Analyse für jede Schokoladen-Tafel
  - Erstelle eine nicht-hierarchische Cluster-Analyse für den Schoko Datensatz

Vielen Dank für eure Aufmerksamkeit!