QQ-Plot
Vorzeichentest
Wilcoxontest
Mann-Whitney-Test

Peter Büchel

HSLU I

Stat: Block 08

Annahme: Normalverteilung

- Bis jetzt: Annahme der Normalverteilung bei Hypothesentests
- Wie kann man überprüfen, ob Daten normalverteilt sind?
- Es gibt mehrere Methoden: Hier graphische Methode

Beispieldatensatz Betondruckfestigkeit

k	$x_{(k)}$
1	24.4
2	27.6
3	27.8
4	27.9
5	28.5
6	30.1
7	30.3
8	31.7
9	32.2
10	32.8
11	33.3
12	33.5
13	34.1
14	34.6
15	35.8
16	35.9
17	36.8
18	37.1
19	39.2
20	39.7

- Messung: Betondruckfestigkeit von $n = 20$ verschiedenen Proben
- Wie gut können die Daten mit einer Normalverteilung beschrieben werden?

Idee

- Daten schon geordnet:

  24.4, 27.6, 27.8, 27.9, 28.5, 30.1, 30.3, 31.7, 32.2, 32.8, 33.3, 33.5, 34.1, 34.6, 35.8, 35.9, 36.8, 37.1, 39.2, 39.7

- Mittelwert und Standardabweichung:

```
import pandas as pd
x =
pd.Series([24.4,27.6,27.8,27.9,28.5,30.1,30.3,31.7,32.2,32.8,33.3,33.5,34.1,34.6,35.8,35.9,36.8,37.1,39.
x.mean()
x.std()
```

```
## 32.665000000000006
## 4.149733662981784
```

- Würden Daten einer Normalverteilung folgen, dann $\mathcal{N}(32.665, 4.15^2)$

- Nehmen Median und Quartile von x und der Normalverteilung:

```
import scipy.stats as st
x.quantile(q=[0.25, 0.5, 0.75])
pd.Series(st.norm.ppf(q=[0.25, 0.5, 0.75], loc=x.mean(), scale=x.std()))
```

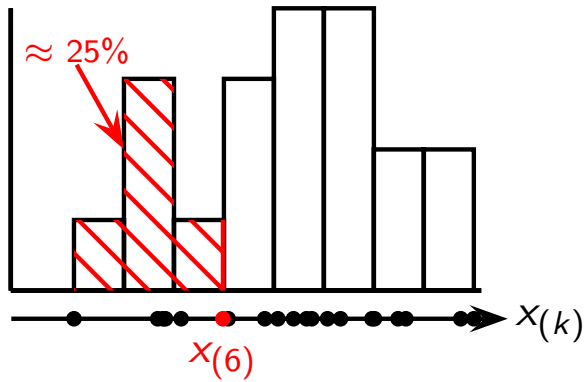
```
## 0.25    29.700
## 0.50    33.050
## 0.75    35.825
## dtype: float64
## 0    29.866047
## 1    32.665000
## 2    35.463953
## dtype: float64
```

- Stimmen diese Werte überein, so liegt Normalverteilung vor

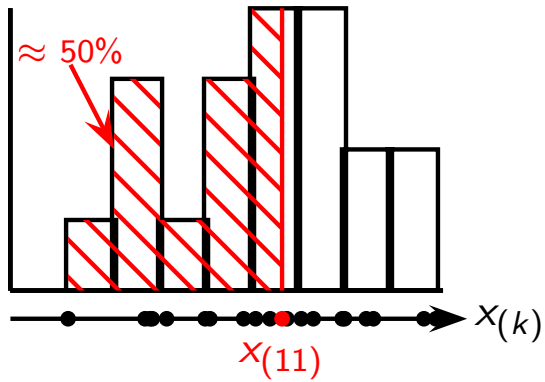
- Besser ersichtlich graphisch: Nächste Folie

QQ-Plot Betondruckfestigkeit: Graphisch

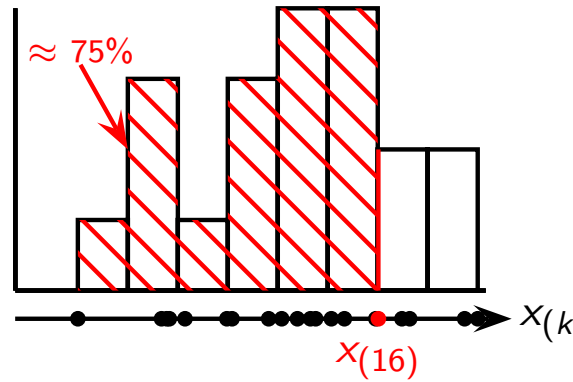
Empirisches 25% Quantil



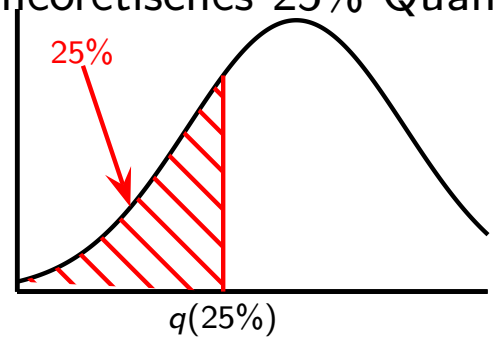
Empirisches 50% Quantil



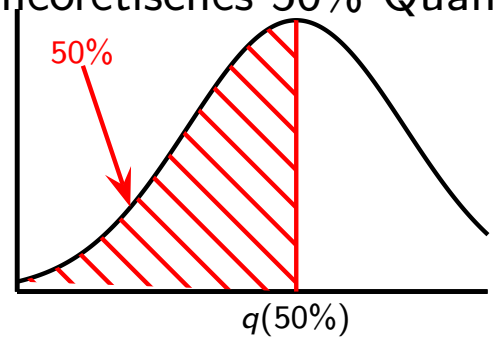
Empirisches 75% Quantil



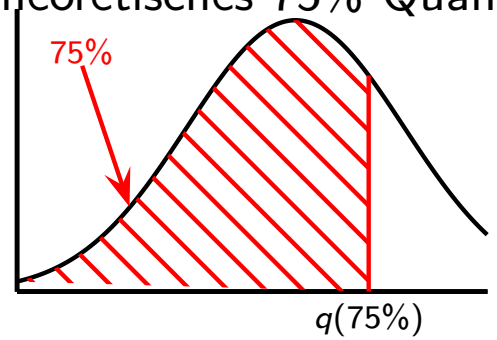
Theoretisches 25% Quantil



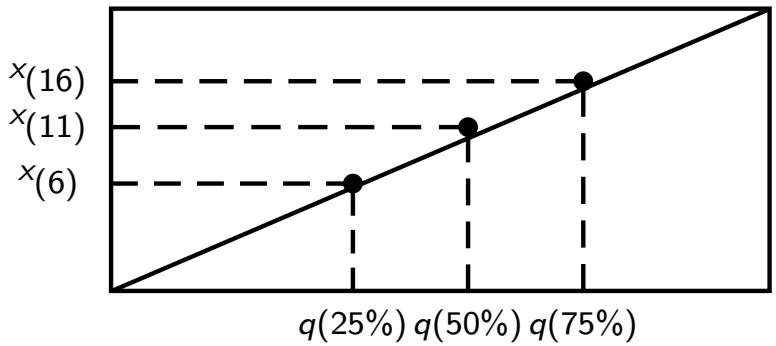
Theoretisches 50% Quantil



Theoretisches 75% Quantil



QQ-Plot

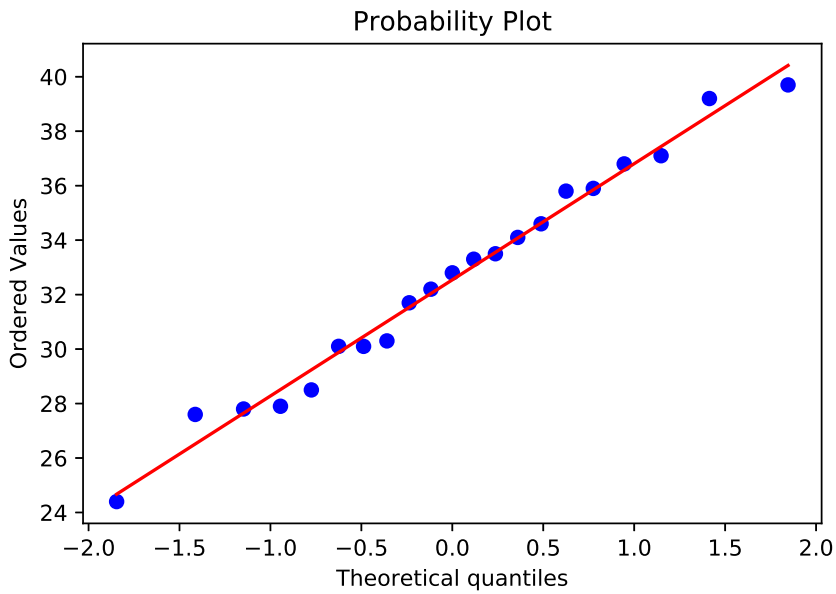


- Horizontale Achse: Theoretische Quantile (Normalverteilung)
- Vertikale Achse: Empirische Quantile (Daten)
- Liegt Normalverteilung vor, so liegen diese Punkte auf einer Geraden
- In Praxis: Es werden viele Quantile genommen, z.B. $q_{0.02}, \ldots, q_{0.98}$
- Siehe auch Jupyter Notebook: `qq_plot_py.ipynb`

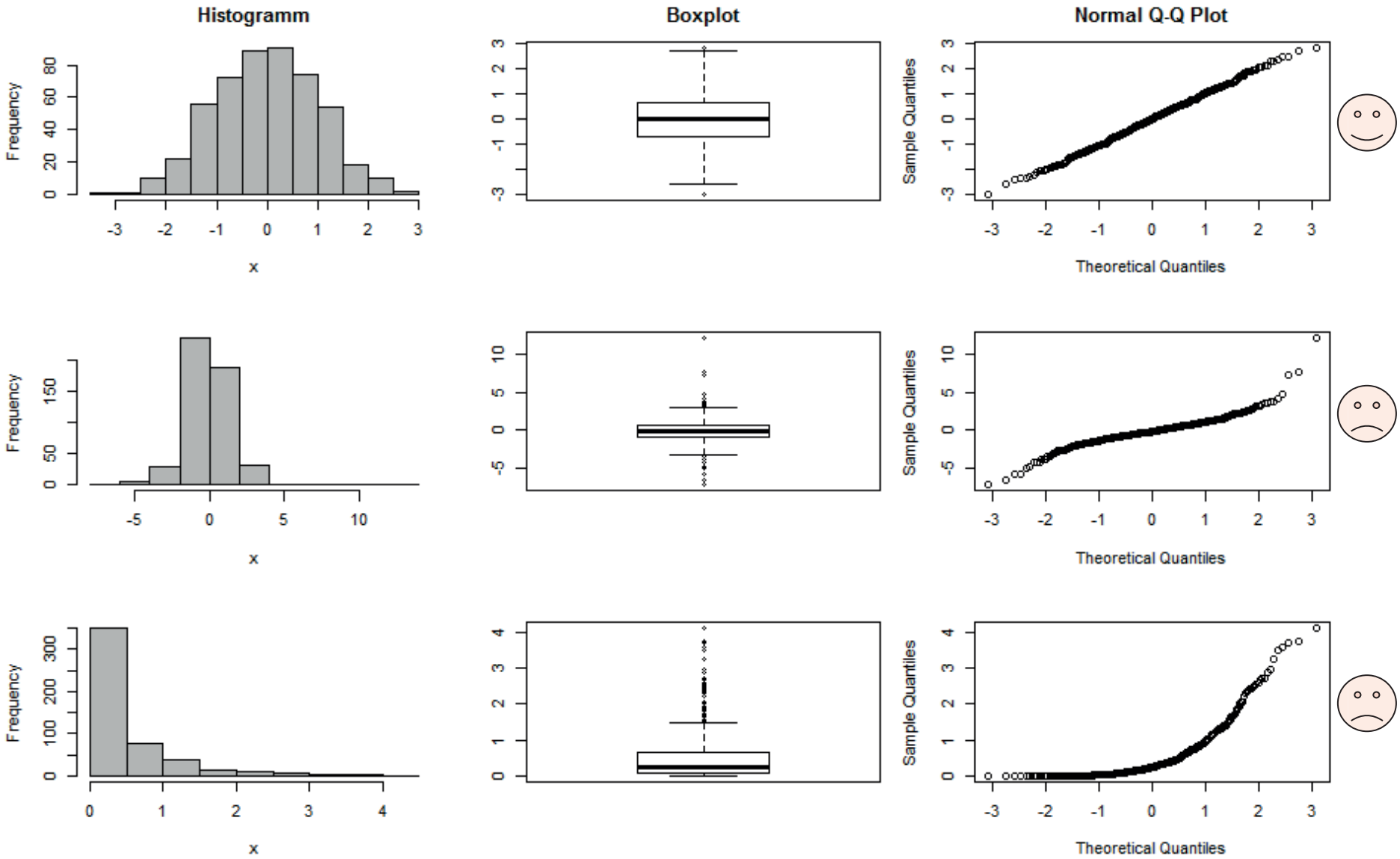
Normal-Plot mit Python

- Software: Theoretische Verteilung oft standardisiert ($\mathcal{N}(0,1)$
- Nur Umskalierung
- Aussage bleibt: Punkte auf Gerade, so liegt Normalverteilung vor
- Python-Befehl:

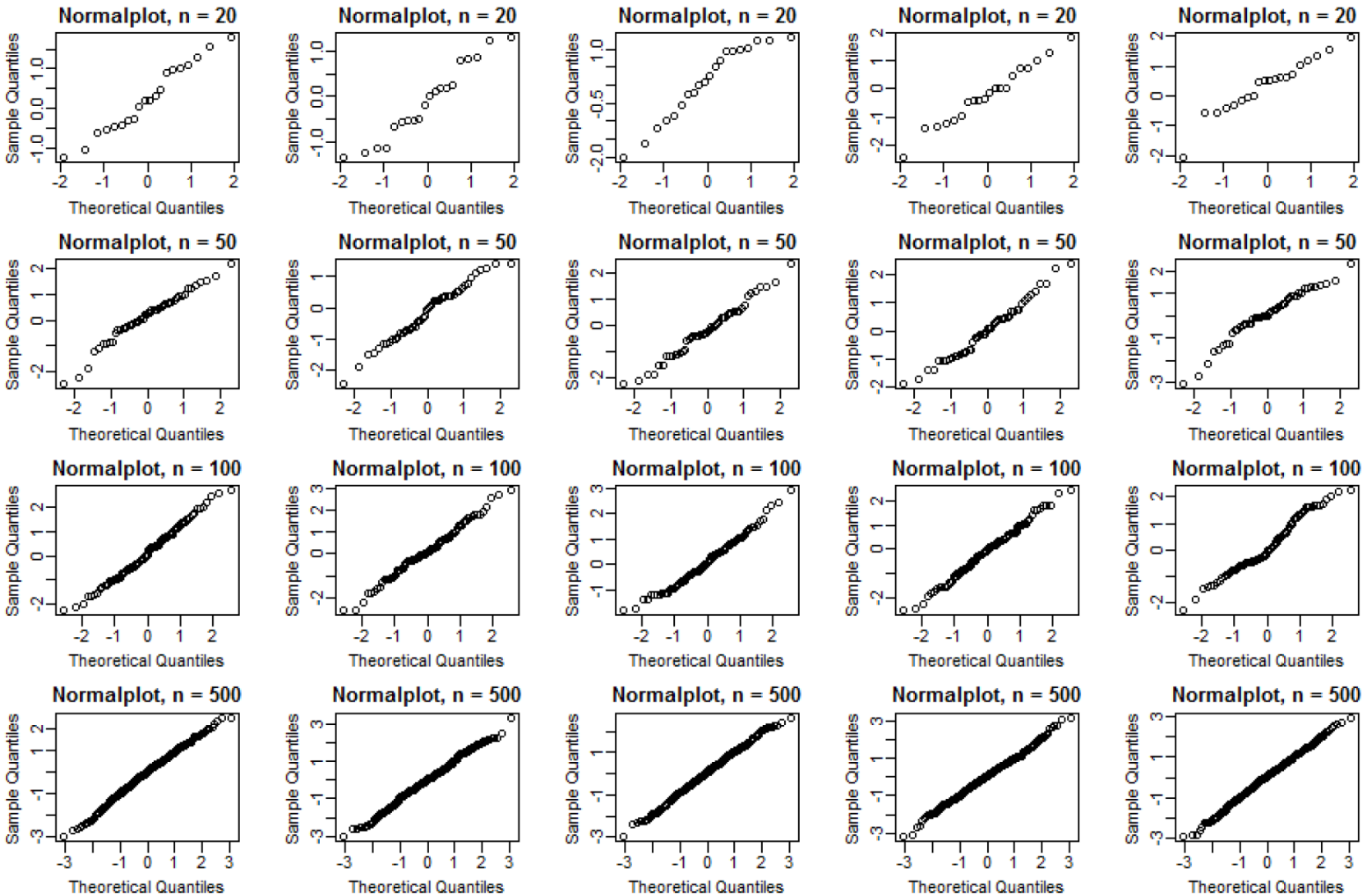
```
st.probplot(x, plot=plt)
```



Beispiele Normalplots für 3 Datensätze mit $n = 500$



Normalplots von simulierten Standardnormalverteilungen



Nicht-normalverteilte Daten

- Bis jetzt: Annahme der Normalverteilung
- Annahme der Normalverteilung sehr stark und oft nicht gegeben
- Was aber, wenn keine Normalverteilung vorliegt?
- Zwei Möglichkeiten:
  - Vorzeichentest: Sehr grob, kaum brauchbar
  - Wilcoxon-Test: Alternative zu t-Test, wenn Daten symmetrisch sind, aber nicht normalverteilt

Nicht-Normalverteilte Daten: Vorzeichentest

1. *Modell*:

$$X_1, \ldots, X_n \text{ iid}$$

wobei X_i eine beliebige Verteilung hat

2. *Nullhypothese*:

$$H_0: \ \mu = \mu_0 \quad (\mu \text{ Median})$$

Alternative:

$$H_A: \ \mu \neq \mu_0 \quad \text{(oder einseitige Variante)}$$

3. *Teststatistik*:

$$V\text{: Anzahl } X_i\text{'s mit } (X_i > \mu_0)$$

Verteilung der Teststatistik unter H_0:

$$V \sim \text{Bin}(n, \pi_0) \quad \text{mit } \pi_0 = 0.5$$

4. *Signifikanzniveau*:

$$\alpha$$

5. *Verwerfungsbereich für die Teststatistik*:

$$K = [0, c_u] \cup [c_o, n]$$

falls $H_A: \ \mu \neq \mu_0$.

Grenzen c_u und c_o müssen mit Binomialverteilung oder Normalapproximation berechnet werden

6. *Testentscheid*: Entscheide, ob der beobachtete Wert der Teststatistik im Verwerfungsbereich der Teststatistik liegt

Beispiel: Vorzeichentest

- Beobachtet:

$$x_1 = 13, \quad x_2 = 9, \quad x_3 = 17, \quad x_4 = 8, \quad x_5 = 14$$

- Angenommen:

$$H_0: \ \mu = \mu_0 = 10, \qquad H_A: \ \mu \neq 10$$

- Vorzeichen von $x_i - \mu_0$:

$$+, \ -, \ +, \ -, \ +$$

- Binomialtest mit

$$H_0: \ \pi = 0.5, \ H_A: \ \pi \neq 0.5, \ n = 5, \ x = 3 \ \text{(Anzahl „+")}$$

Beispiel: Vorzeichentest

- Antwort: Binomialtest mit **Python**

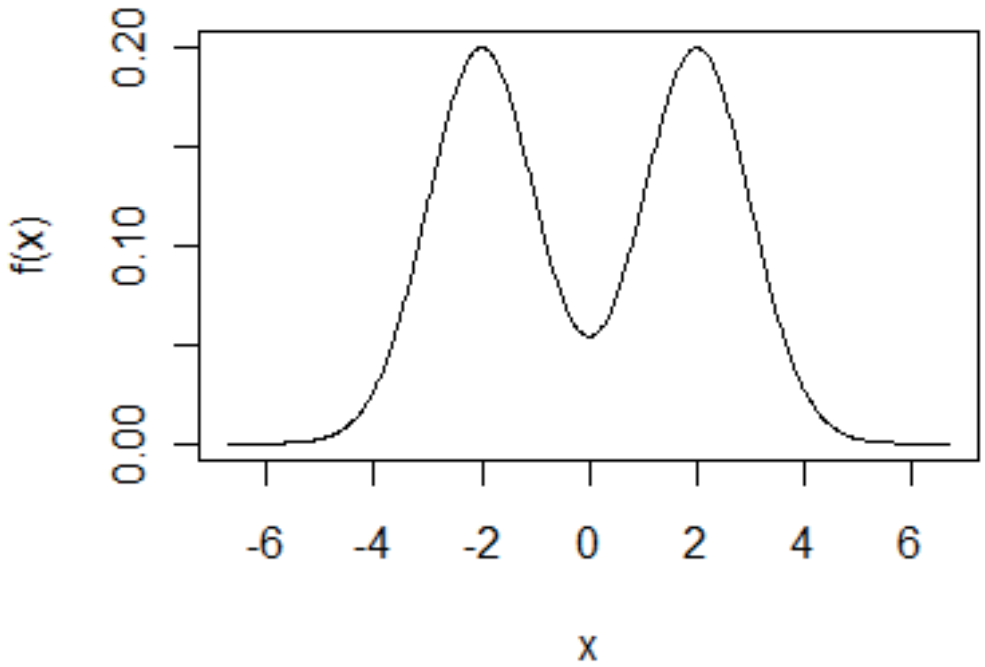
```
import scipy.stats as st

st.binom_test(x=3, n=5, p=0.5)
## 1.0
```

- Resultat ist p-Wert
- Nullhypothese beim Vorzeichentest hier wird nicht verworfen
- Vorteil vom Vorzeichentest: Keine Annahme an Verteilung
- Nachteil vom Vorzeichentest: Kleinere Macht

Nicht-normalverteilte Daten: Wilcoxon-Test

- Kompromiss zwischen Vorzeichen- und t-Test
- Annahme:

$$X_i \sim F \text{ iid}, \quad F \quad \text{ist symmetrisch}$$



- Teste Median μ : H_0 : $\mu = \mu_0$ (einseitig oder zweiseitig)
- Intuition der Teststatistik
  - Rangiere
$$|x_i - \mu_0| \to r_i$$
  - Gib Rängen ursprüngliches Vorzeichen von $(x_i - \mu_0)$ („signed ranks")
  - Teststatistik T: Summe aller Ränge, bei denen $(x_i - \mu_0)$ positiv ist
- Falls H_0 stimmt: Rangsumme nicht zu gross und nicht zu klein (in Mitte der gesamten Rangsumme)

Beispiel: Wilcoxon-Test

- Bsp: $H_0: \ \mu_0 = 0$
- Beobachte:

$$-1.9, \quad 0.2, \quad 2.9, \quad -4.1, \quad 3.9$$

- Absolutbeträge:

$$1.9, \quad 0.2, \quad 2.9, \quad 4.1, \quad 3.9$$

- Geordnet

$$0.2, \quad 1.9, \quad 2.9, \quad 3.9, \quad 4.1$$

- Ränge der Absolutbeträge:

$$1, \quad 2, \quad 3, \quad 4, \quad 5$$

- Rangsumme der posititven Gruppe: 1+3+4=8
  - Minimale Rangsumme: 0
  - Maximale Rangsumme: 1+2+3+4+5 = 15

Wicoxon-Test mit **Python**

- Code:

```
import scipy.stats as st
import numpy as np

x = np.array([-1.9, 0.2, 2.9, -4.1, 3.9])

st.wilcoxon(x, correction=True)
## /usr/local/lib/python3.7/dist-packages/scipy/stats/morestats.py:287
##   warnings.warn("Sample size too small for normal approximation.")
## WilcoxonResult(statistic=7.0, pvalue=1.0)
```

- Hier noch Warnung: Da Datengrösse mit 5 sehr klein

- Damit **Python** p-Wert richtig berechnen kann, muss $n > 20$

Übersicht der Tests

Test	Annahme				n_{min} bei $\alpha = 0.05$	Macht für ein Beispiel (1)
	σ_X bekannt	$X_i \sim N$	Symm. Verteilung	iid		
z	x	x	x	x	1	89 %
t		x	x	x	2	79 %
Wilcoxon			x	x	6	79 %
VZ				x	5	48 %

(1): $X_i \sim \mathcal{N}(\mu, \sigma^2)$, $n = 10$; $H_0 : \mu = 0$; $H_A : \mu \neq 0$; $\alpha = 0.05$
Macht berechnet für konkrete Alternative: $X_i \sim \mathcal{N}(1, 1)$

Wilcoxon-Test versus t-Test

- *Wilcoxon-Test* meistens *t-Test* oder *Vorzeichen-Test* vorzuziehen
- Er hat in vielen Situationen oftmals wesentlich *grössere Macht*, und selbst in den ungünstigsten Fällen ist er nie viel schlechter
- Wenn man trotzdem den t-Test verwendet, dann sollte man die Daten auch graphisch ansehen, damit wenigstens grobe Abweichungen von der Normalverteilung entdeckt werden
- Insbesondere sollte der *Normal-Plot* angeschaut werden

Vergleich von zwei Stichproben

- Mögliche Fragestellungen:
  - Vergleich von zwei Messverfahren (Messgerät A vs. Messgerät B): Gibt es einen signifikanten Unterschied?
  - Vergleich von zwei Herstellungsverfahren (A vs. B): Welches hat die besseren Eigenschaften (z.B. bzgl. einer Festigkeitsgrösse)?
  - Werden männliche Dozenten von weiblichen Studierenden besser als von männlichen Studierenden bewertet?
  - Sammeln jeweils Daten von zwei Gruppen

Gepaarte Stichproben

- Beispiel Messgeräte: Jeden Prüfkörper mit *beiden* Messgeräten messen
- Pro *Versuchseinheit* (hier: Prüfkörper) zwei Beobachtungen (einmal Gerät A und einmal Gerät B)
- Man spricht auch von *gepaarten Stichproben*
- Beide Beobachtungen sind *nicht* unabhängig, da an der *gleichen* Versuchseinheit zwei Mal gemessen wird!

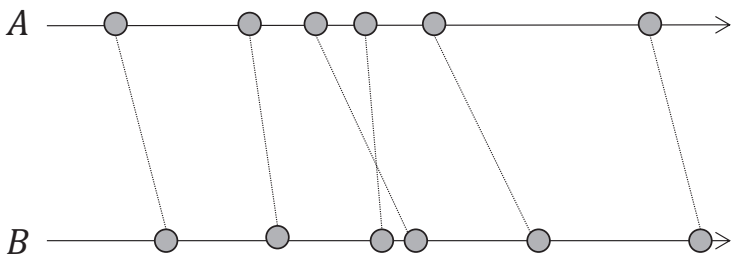
Ungepaarte (unabhängige) Stichproben

- Beispiel der beiden Herstellungsverfahren: Stichprobe von Verfahren A und eine andere Stichprobe von Verfahren B
- Beobachtungen sind hier *unabhängig*; „es gibt *nichts*, was sie verbindet"
- Man spricht von *ungepaarten (oder unabhängigen) Stichproben*

Unterscheidung gepaart versus ungepaarte Stichproben

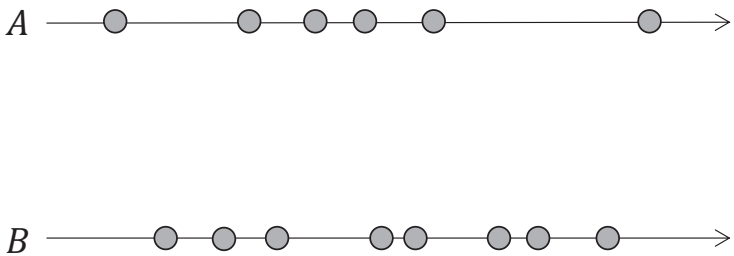
Gepaarte Stichproben

- Jede Beobachtung einer Gruppe kann eindeutig einer Beobachtung der anderen Gruppe zugeordnet werden
- Stichprobengrösse ist in beiden Gruppen zwangsläufig gleich



Ungepaarte Stichproben

- Keine Zuordnung von Beobachtungen möglich
- Stichprobengrössen können verschieden sein (müssen aber nicht!)
- Man kann die eine Gruppe vergrössern, ohne dass man die andere vergrössert

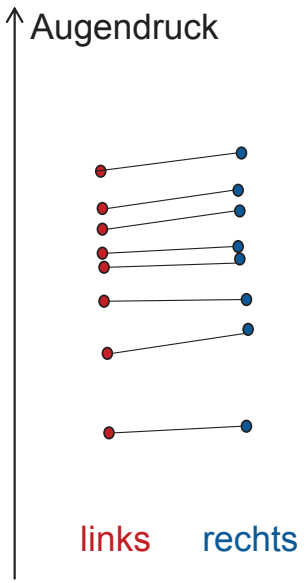


Gepaarte versus ungepaarte Stichproben

- Beispiel Augeninnendruck: Ein Auge wird behandelt, das andere nicht (gepaarter Test ist angebracht)

- Gemäss Vorraussetzungen dürfte auch ein ungepaarter Test angewendet werden

$H_0: \mu_X = \mu_Y$



Ungepaart:

Intuition Teststatistik: $T = \frac{\overline{X} - \overline{Y}}{\widehat{\sigma_{\overline{X}}}}$

Gepaart:
Differenz $D_i = X_i - Y_i$

Teststatistik $T = \frac{\overline{D}}{\widehat{\sigma_{\overline{D}}}}$

Statistischer Test für gepaarte Stichproben mit `Python`

- *Gepaarte Stichproben:*

$$X_i \sim \mathcal{N}(\mu_X, \sigma_X^2) \qquad \text{und} \qquad Y_i \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$$

- Betrachten Differenzen:

$$D_i = X_i - Y_i$$

- Führen einen t-Test durch mit der Teststatistik D_i

- Normalerweise Nullhypothese

$$\mathrm{E}(D) = \mu_D = 0$$

  also kein Unterschied

- Falls die Daten nicht normalverteilt: Wilcoxon- oder Vorzeichentest

Statistischer Test für gepaarte Stichproben mit **Python**

- Code

```
import scipy.stats as st
from scipy.stats import norm, t, binom
import numpy as np
from pandas import Series

vorher = Series([25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28])

nachher = Series([27, 29, 37, 56, 46, 82, 57, 80, 61, 59, 43])

st.ttest_rel(nachher, vorher)
## Ttest_relResult(statistic=4.271608818429545, pvalue=0.0016328499219
```

- **`ttest_rel`**: **`rel`** für *related*

- Output **`statistics:...`**: Wert mit dem der p-Wert berechnet wird (keine physikalische Bedeutung)

- Output `pvalue:...`: Wert für Testentscheid
- p-Wert ist mit 0.0016 unter Signifikanzniveau und somit wird Nullhypothese verworfen
- Es gibt einen statistisch signifikanten Unterschied zwischen `vorher` und `nachher`
- Vertrauensintervall:

```
vorher = Series([25, 25, 27, 44, 30, 67, 53, 53, 52, 60, 28])
nachher = Series([27, 29, 37, 56, 46, 82, 57,80, 61, 59, 43])
dif = nachher - vorher

t.interval(alpha=.95, df=dif.size-1, loc=dif.mean(),
scale=dif.std()/np.sqrt(dif.size))
## (4.91430993515407, 15.631144610300478)
```

- Unterschied auf 5 % Signifikanzniveau signifikant, weil P-Wert kleiner 5 %
- 95 %-Vertrauensintervall: Unterschieds in den Gruppenmittelwerten
- Mit 95 % W'keit ist Gruppenmittelwert von $\mathtt{x}$ um eine Zahl im Bereich

$$[4.91431, 15.63114]$$

grösser als der Gruppenmittelwert von $\mathtt{y}$

Statistischer Test für ungepaarte Stichproben mit **Python**

- *Ungepaarte Stichproben:* Daten X_i und Y_i normalverteilt, aber ungepaart
- Beispiel: Schmelzwärme von Eis: Zwei-Stichproben t-Test für ungepaarte Stichproben mit Nullhypothese $\mu_X = \mu_Y$:

```
x = Series([79.98, 80.04, 80.02, 80.04, 80.03, 80.03, 80.04,
79.97, 80.05,80.03, 80.02, 80.00, 80.02])

y = Series([80.02, 79.94, 79.98, 79.97, 80.03, 79.95, 79.97])

st.ttest_ind(x, y, equal_var=False)
## Ttest_indResult(statistic=2.839932638516127, pvalue=0.018660
```

- **ttest_ind**: **rel** für *independent*

- p-Wert ist 0.018 unter dem Signifikanzniveau und somit wird die Nullhypothese verworfen
- Es gibt einen statistisch signifikanten Unterschied zwischen den beiden Mittelwerten
- Unterschied auf 5 % Signifikanzniveau signifikant, weil p-Wert kleiner als 5 %
- Unterschied in den Gruppenmittelwerten

Bemerkungen

- `st.ttest_rel()` und `st.ttest_ind()` kennen nur zweiseitigen Test
- Bei einseitigem Test: Ausgegebener p-Wert halbieren
- `st.wilcoxon()` kennt ab SciPy Version 1.3.0 `alternative="..."` (`greater, less, two-sided`)
- Für frühere Versionen von SciPy kommt eine Fehlermeldung

Mann-Whitney U-Test (aka Wilcoxon Rank-sum Test)

- Falls Daten nicht normalverteilt
- $X_i \sim F,\ i = 1, \ldots, n;\ Y_j \sim G,\ j = 1, \ldots m$
  $H_0 :\ F = G$
  $H_A :\ F = G + \delta\ (\delta \neq 0)$ (oder einseitig)
  (d.h., Verteilungen sind verschoben, haben aber gleiche From)
- Code

```
x = Series([79.98, 80.04, 80.02, 80.04, 80.03, 80.03, 80.04,
79.97, 80.05,80.03, 80.02, 80.00, 80.02])

y = Series([80.02, 79.94, 79.98, 79.97, 80.03, 79.95, 79.97])

st.mannwhitneyu(x, y)
## MannwhitneyuResult(statistic=14.5, pvalue=0.007268822472369
```

- Nullhypothese wird verworfen

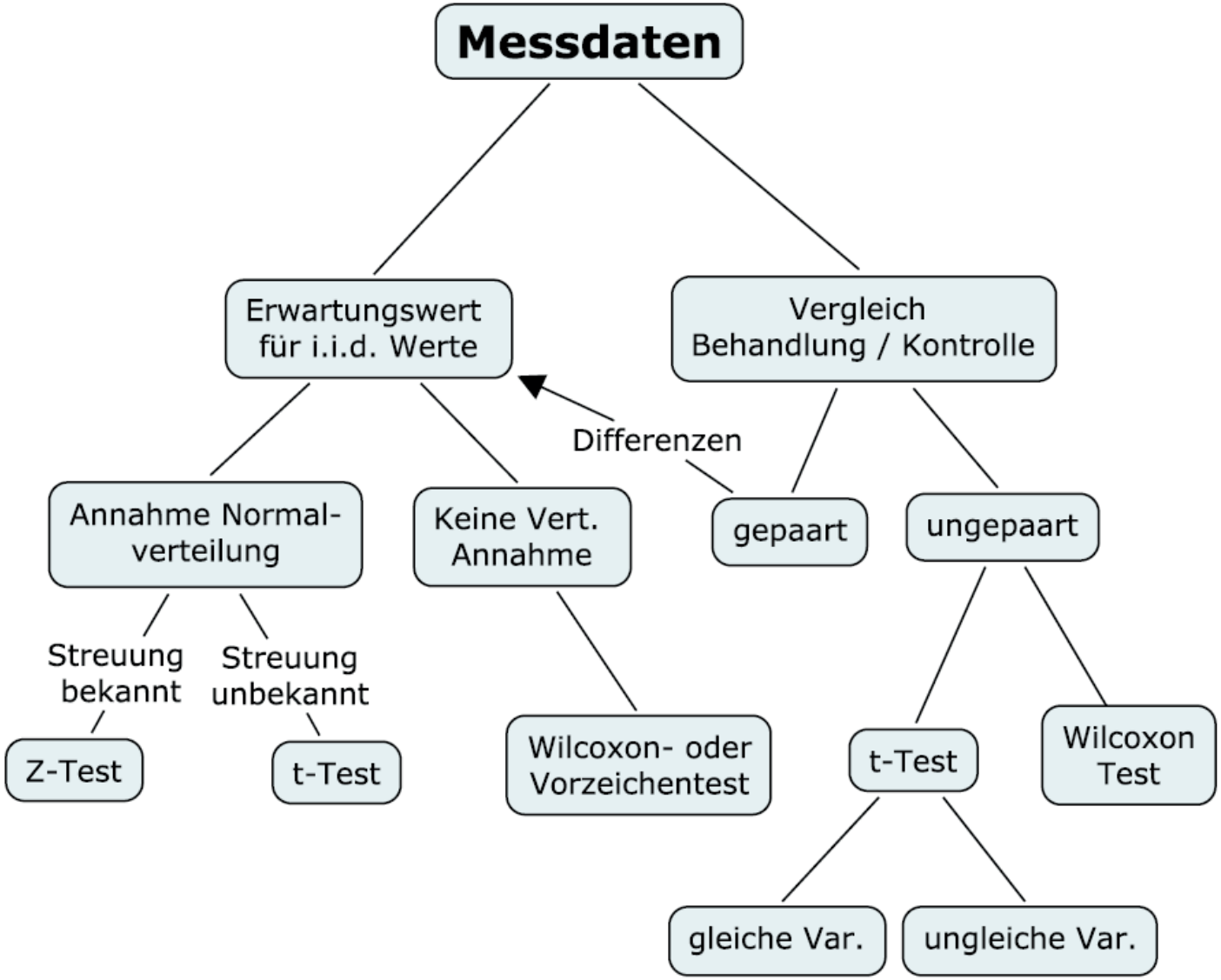
Übersicht: Tests für ungepaarte Stichproben

Test	**Annahme**				n_{min} **falls** ($n = m$) **bei** $\alpha = 0.05$	**Macht für ein Beispiel (1)**
	$\sigma_X = \sigma_Y$	$X_i \sim N$ $Y_i \sim N$	F, G haben gleiche Form	iid pro Gruppe		
t $(\sigma_X = \sigma_Y)$	x	x	x	x	2	57%
t $(\sigma_X \neq \sigma_Y)$		x		x	2	56%
MW U-Test	x		x	x	4	53%

(1): $X_i \sim \mathcal{N}(\mu_X, \sigma^2)$, $Y_i \sim \mathcal{N}(\mu_Y, \sigma^2)$ $n = m = 10$; $H_0 : \mu_X = \mu_Y$; $H_A : \mu_X \neq \mu_Y$; $\alpha = 0.05$

Macht berechnet für konkrete Alternative: $X_i \sim \mathcal{N}(0, 1)$, $Y_i \sim \mathcal{N}(1, 1)$

Übersicht Statistische Tests (stetige Verteilungen)



Python-Befehle

- Allgemein:

```
import scipy.stats as st
```

- Einstichprobenstest mit Datensatz `x`:
  - *t*-Test: `st.ttest_1samp(x, popmean=...)`
  - Wilcoxon-Test: `st.wilcoxon(x, alternative="...")`
- Zweistichprobenstest mit *gepaarten* Datensätzen `x,y`:
  - *t*-Test: `st.ttest_rel(x, y)`
  - Wilcoxon-Test: `st.wilcoxon(x, y, alternative="...")`
- Zweistichprobenstest mit *ungepaarten* Datensätzen `x,y`:
  - *t*-Test: `st.ttest_ind(x, y, equal_var=False)`
  - Wilcoxon-Test: `st.mannwhitneyu(x, y, alternative="...")`

Bemerkung

- `st.ttest_...` kennt keine Alternative und gibt immer den zweiseitigen p-Wert an
- Für einseitigen Test: Output des p-Wertes halbieren