Die Normalverteilung

Inhaltsverzeichnis

Mit zunehmender Stichprobengröße wird eine immer bessere Schätzung des Erwartungswerts erreicht. Mathematisch liegt dieser Beobachtung der zentrale Grenzwertsatz zugrunde. So werden beim Würfeln mit mehreren Würfeln weit vom Erwartungswert entfernte Wurfergebnis-

se immer unwahrscheinlicher. Dies lässt sich bereits mit wenigen Würfeln zeigen (siehe Beispiel).

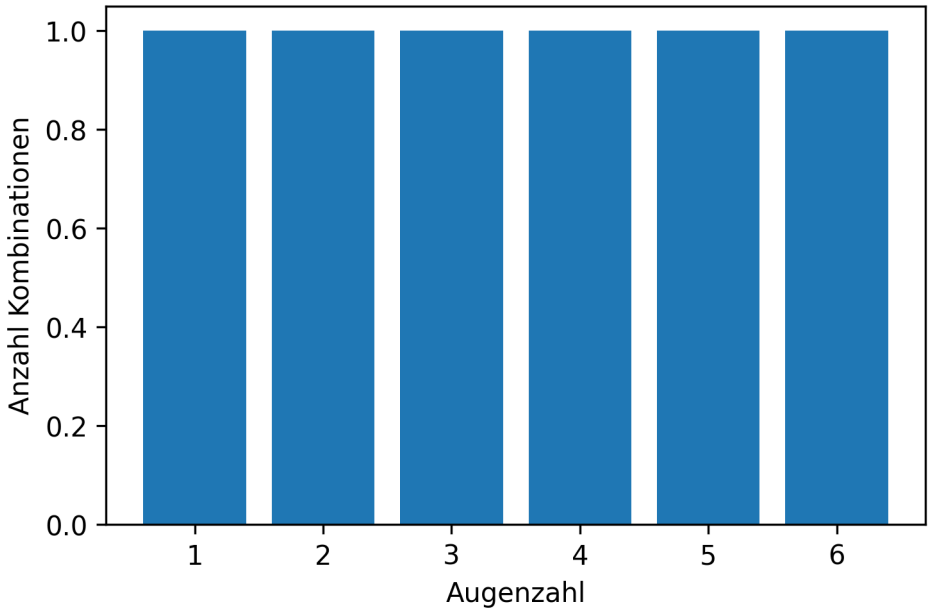
Hinweis 1: Häufigkeitsverteilung von Würfelergebnissen

Für einen Würfel gibt es 6 mögliche Ergebnisse, für 2 Würfel 6 * 6 mögliche Kombinationen, für 3 Würfel 6 * 6 * 6 Kombinationen und so weiter. Weil viele Kombinationen wertgleich sind, kommen Wurfergebnisse in der Nähe des Erwartungswerts häufiger vor als beispielsweise ein Einserpasch.

0.1 ein Würfel

```
ein_würfel = []

for i in range(1, 7):
  ein_würfel.append(i)

ein_würfel = pd.Series(ein_würfel)

print("Häufigkeitsverteilung der Augensumme:")
print(ein_würfel.value_counts(), "\n")
print(f"Durchschnitt: {ein_würfel.mean():.1f}")

plt.bar(ein_würfel.unique(), ein_würfel.value_counts())
plt.xlabel('Augenzahl')
plt.ylabel('Anzahl Kombinationen')
plt.show()
```

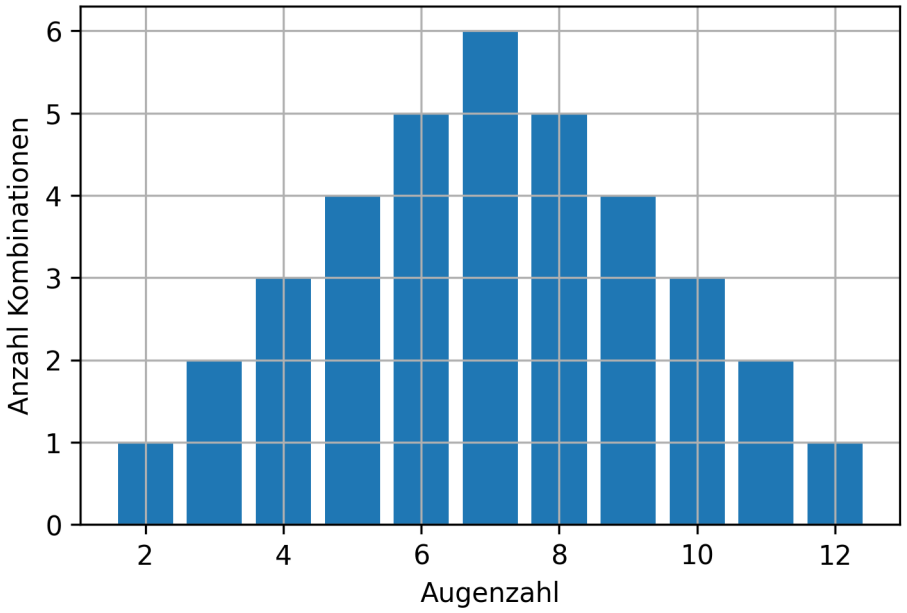
```
Häufigkeitsverteilung der Augensumme:
1    1
2    1
3    1
4    1
5    1
6    1
Name: count, dtype: int64

Durchschnitt: 3.5
```



0.2 zwei Würfel

```python
zwei_würfel = []

for i in range(1, 7):
  würfel_1 = i

  for j in range (1, 7):
    würfel_2 = j
    zwei_würfel.append(würfel_1 + würfel_2)

zwei_würfel = pd.Series(zwei_würfel)

print("Häufigkeitsverteilung der Augensumme:")
print(zwei_würfel.value_counts().sort_index(ascending = True), "\n")
print(f"Durchschnitt: {zwei_würfel.mean():.1f}")
print(f"Durchschnitt pro Würfel: {zwei_würfel.mean() / 2:.1f}")

plt.bar(zwei_würfel.unique(), zwei_würfel.value_counts().sort_index(ascending = True))
plt.xlabel('Augenzahl')
plt.ylabel('Anzahl Kombinationen')
plt.grid()
plt.show()
```

```
Häufigkeitsverteilung der Augensumme:
2     1
3     2
4     3
5     4
6     5
7     6
8     5
9     4
10    3
11    2
12    1
Name: count, dtype: int64

Durchschnitt: 7.0
Durchschnitt pro Würfel: 3.5
```

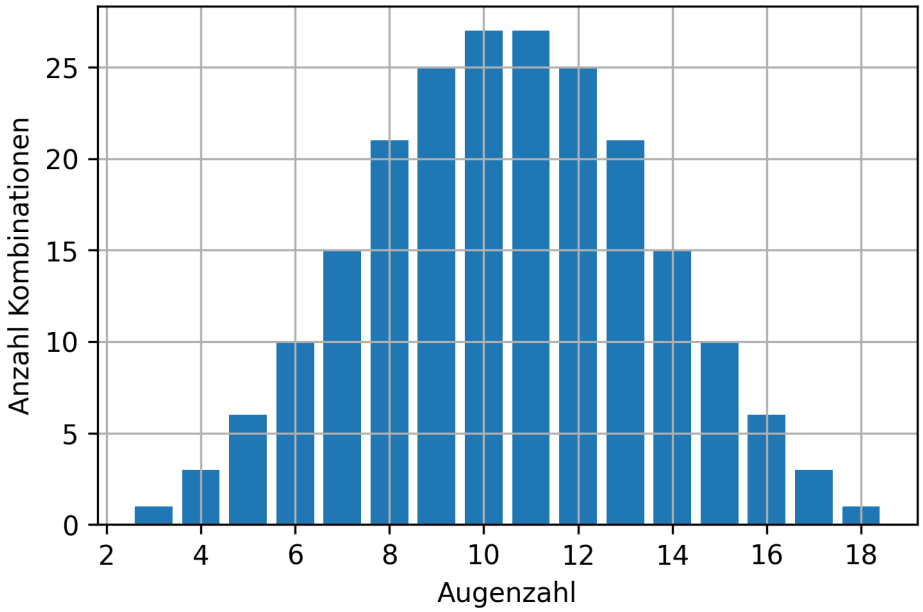


0.3 drei Würfel

```python
drei_würfel = []

for i in range(1, 7):
  würfel_1 = i

  for j in range (1, 7):
    würfel_2 = j

    for k in range (1, 7):
      würfel_3 = k
      drei_würfel.append(würfel_1 + würfel_2 + würfel_3)

drei_würfel = pd.Series(drei_würfel)

print("Häufigkeitsverteilung der Augensumme:")
print(drei_würfel.value_counts().sort_index(ascending = True), "\n")
print(f"Durchschnitt: {drei_würfel.mean():.1f}")
print(f"Durchschnitt pro Würfel: {drei_würfel.mean() / 3:.1f}")

plt.bar(drei_würfel.unique(), drei_würfel.value_counts().sort_index(ascending = True))
plt.xlabel('Augenzahl')
plt.ylabel ('Anzahl Kombinationen')
plt.grid()
plt.show()
```

```
Häufigkeitsverteilung der Augensumme:
3      1
4      3
5      6
6     10
7     15
8     21
9     25
10    27
11    27
12    25
13    21
14    15
15    10
16     6
17     3
```

```
18     1
Name: count, dtype: int64

Durchschnitt: 10.5
Durchschnitt pro Würfel: 3.5
```



Die mit steigender Stichprobengröße zu beobachtende Annäherung von Messwerten an einen in der Grundgesamtheit geltenden Erwartungswert gilt auch, wenn der Erwartungswert und die Varianz in der Grundgesamtheit unbekannt sind. Mit zunehmender Stichprobengröße nähern sich die Messwerte der Normalverteilung an, die nach ihrem Entdecker Carl Friedrich Gauß auch als Gaußsche Glockenkurve bekannt ist.

Die für größere Stichproben zu beobachtende Annäherung der Verteilung von Messwerten an die Normalverteilung kann anhand des Gewichts von Pinguinen aus dem Datensatz palmerpenguins gezeigt werden.

palmerpenguins

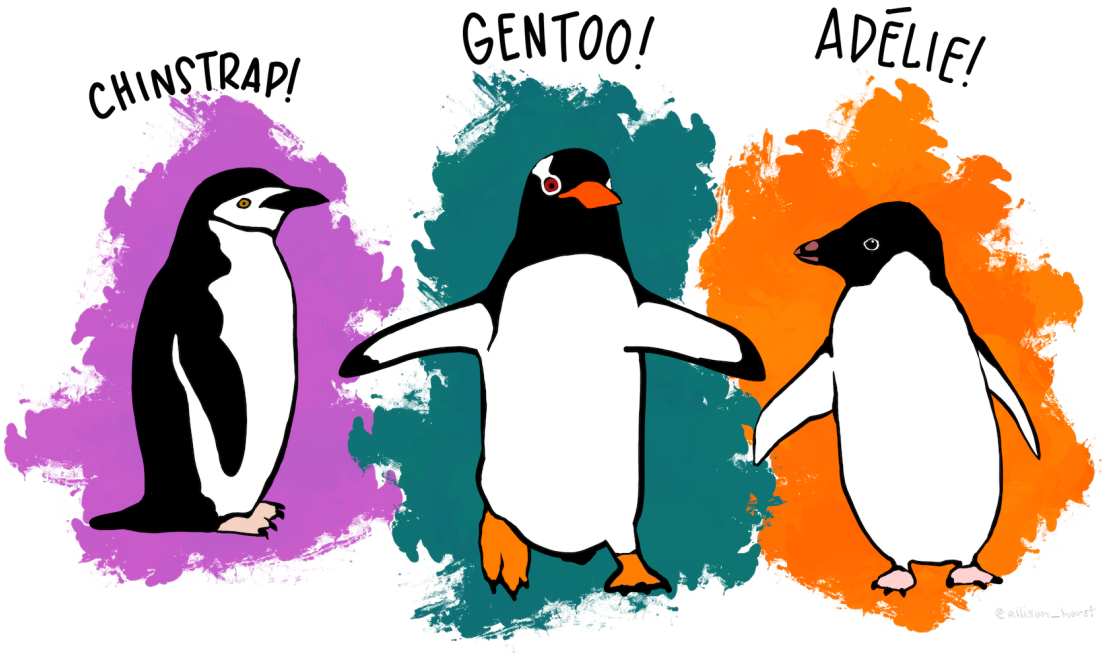


Abbildung 1: Pinguine des Palmer-Station-Datensatzes

Meet the Palmer penguins von @allison_horst steht unter der Lizenz CC0-1.0 und ist auf GitHub abrufbar. 2020

Der Datensatz steht unter der Lizenz CCO und ist in R sowie auf GitHub verfügbar. 2020

```
# R Befehle, um den Datensatz zu laden
install.packages("palmerpenguins")
library(palmerpenguins)
```

Horst AM, Hill AP und Gorman KB. 2020. palmerpenguins: Palmer Archipelago (Antarctica) penguin data. R package version 0.1.0. https://allisonhorst.github.io/palmerpenguins/. doi: 10.5281/zenodo.3960218.

```
penguins = pd.read_csv(filepath_or_buffer = "01-daten/penguins.csv")

# Tiere mit unvollständigen Einträgen entfernen
penguins.drop(np.where(penguins.apply(pd.isna).any(axis = 1))[0], inplace = True)

print(penguins.info(), "\n");
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 333 entries, 0 to 343
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            333 non-null    object
 1   island             333 non-null    object
 2   bill_length_mm     333 non-null    float64
 3   bill_depth_mm      333 non-null    float64
 4   flipper_length_mm  333 non-null    float64
 5   body_mass_g        333 non-null    float64
 6   sex                333 non-null    object
 7   year               333 non-null    int64
dtypes: float64(4), int64(1), object(3)
memory usage: 23.4+ KB
None
```

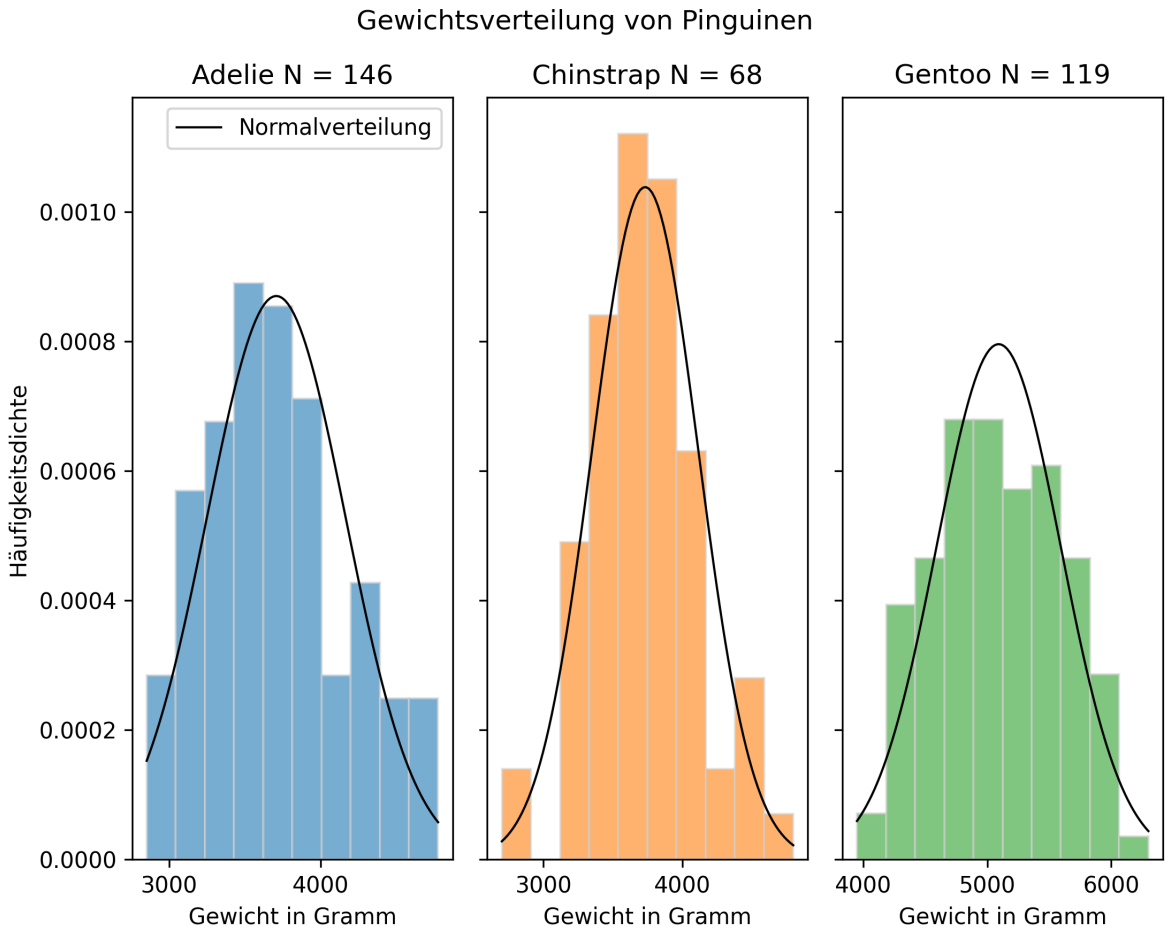
Der Datensatz enthält Daten für drei Pinguinarten.

```
print(penguins.groupby(by = penguins['species']).size())
```

```
species
Adelie       146
Chinstrap     68
Gentoo       119
dtype: int64
```

Unter anderen wurde das Körpergewicht in Gramm gemessen, das in der Spalte 'body_mass_g' eingetragen ist. Die Gewichtsverteilung der drei Spezies wird jeweils mit einem Histogramm dargestellt. Außerdem werden für jede Spezies der Stichprobenmittelwert und die Stichprobenstandardabweichung bestimmt. Mit diesen Werten kann eine Normalverteilungskurve berechnet und in das Histogramm eingezeichnet werden (wie das geht, wird in Beispiel 3 gezeigt). So kann optisch geprüft werden, ob die empirische Verteilung der Werte in der Stichprobe einer Normalverteilung mit den selben Werten für Mittelwert und Standardabspreichung entspricht.

0.4 Grafik



0.5 Code

```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (7.5, 6), sharey = True, layout = 'tight
plt.suptitle('Gewichtsverteilung von Pinguinen')

# Adelie
species = 'Adelie'
data = penguins['body_mass_g'][penguins['species'] == species]

## Histogramm
ax1.hist(data, alpha = 0.6, edgecolor = 'lightgrey', color = 'C0', density = True)
ax1.set_xlabel('Gewicht in Gramm')
```

```
ax1.set_ylabel('Häufigkeitsdichte')
ax1.set_title(label = str(species) + " N = " + str(data.size))

## Normalverteilungskurve
stichprobenmittelwert = data.mean()
stichprobenstandardabweichung = data.std(ddof = 1)
hist, bin_edges = np.histogram(data)
x_values = np.linspace(min(bin_edges), max(bin_edges), 100)
y_values =  1 / (stichprobenstandardabweichung * np.sqrt(2 * np.pi)) * np.exp(- (x_values - s

ax1.plot(x_values, y_values, color = 'black', linewidth = 1, label = 'Normalverteilung')
ax1.legend()

# Chinstrap
species = 'Chinstrap'
data = penguins['body_mass_g'][penguins['species'] == species]

## Histogramm
ax2.hist(data, alpha = 0.6, edgecolor = 'lightgrey', color = 'C1', density = True)
ax2.set_xlabel('Gewicht in Gramm')
ax2.set_title(label = str(species) + " N = " + str(data.size))

## Normalverteilungskurve
stichprobenmittelwert = data.mean()
stichprobenstandardabweichung = data.std(ddof = 1)
hist, bin_edges = np.histogram(data)
x_values = np.linspace(min(bin_edges), max(bin_edges), 100)
y_values =  1 / (stichprobenstandardabweichung * np.sqrt(2 * np.pi)) * np.exp(- (x_values - s

ax2.plot(x_values, y_values, color = 'black', linewidth = 1)

# Gentoo
species = 'Gentoo'
data = penguins['body_mass_g'][penguins['species'] == species]

## Histogramm
ax3.hist(data, alpha = 0.6, edgecolor = 'lightgrey', color = 'C2', density = True)
ax3.set_xlabel('Gewicht in Gramm')
ax3.set_title(label = str(species) + " N = " + str(data.size))

## Normalverteilungskurve
stichprobenmittelwert = data.mean()
```

```
stichprobenstandardabweichung = data.std(ddof = 1)
hist, bin_edges = np.histogram(data)
x_values = np.linspace(min(bin_edges), max(bin_edges), 100)
y_values =  1 / (stichprobenstandardabweichung * np.sqrt(2 * np.pi)) * np.exp(- (x_values - s

ax3.plot(x_values, y_values, color = 'black', linewidth = 1)

plt.show()
```

Die Normalverteilung ist eine Dichtekurve, an die sich der Verlauf eines Histogramms mit einer gegen unendlich gehenden Anzahl von Messwerten und einer gegen Null gehenden Klassenbreite annähert.

> **!** Wichtig 1: Histogramm
>
> Das Histogramm ist eine grafische Darstellung der Häufigkeitsverteilung kardinal skalierter Merkmale. Die Daten werden in Klassen, die eine konstante oder variable Breite haben können, eingeteilt. Es werden direkt nebeneinanderliegende Rechtecke von der Breite der jeweiligen Klasse gezeichnet, deren Flächeninhalte die (relativen oder absoluten) Klassenhäufigkeiten darstellen. Die Höhe jedes Rechtecks stellt dann die (relative oder absolute) Häufigkeitsdichte dar, also die (relative oder absolute) Häufigkeit dividiert durch die Breite der entsprechenden Klasse.

Die Dichtefunktion der Normalverteilung beschreibt, welcher Anteil der Werte innerhalb eines bestimmten Wertebereichs liegt. Bei der Berechnung der relativen Häufigkeiten in Beispiel 2 haben wir gesehen, dass die Summe der relativen Häufigkeiten 1 ist. Dies entspricht der Fläche unterhalb der Dichtekurve.

Die Dichtefunktion der Normalverteilung ist definiert als:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Die Form der Normalverteilung ergibt sich aus dem Faktor $e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ der Funktionsgleichung. Das Maximum der Funktion liegt am Punkt $x = \mu$. Von dort fällt sie symmetrisch ab und nähert sich der x-Achse an. Der Abfall der Funktion erfolgt umso schneller, je kleiner σ ist. Die Wendepunkte der Kurve liegen jeweils eine Standardabweichung vom Mittelwert entfernt.

Eine Normalverteilung mit dem Mittelwert $\mu = 0$ und einer Standardabweichung $\sigma = 1$ heißt Standardnormalverteilung.

0.12 Normalverteilung anpassen

Um die Verteilung in einem Datensatz durch eine Normalverteilung anzunähern, werden dessen Mittelwert und Standardabweichung in die Funktionsgleichung der Normalverteilung eingesetzt. Mit Python können die Berechnungen direkt vorgenommen werden. In der Handhabung einfacher sind die vom Paket SciPy bereitgestellten Funktionen, die im nächsten Abschnitt ausführlicher vorgestellt werden. Das folgende Beispiel zeigt die Berechnung und Visualisierung mit Python und mit SciPy.

Hinweis 3: Dichtekurven berechnen und darstellen

Betrachten wir die Verteilungskennwerte der Gruppe der Meerschweinchen, die eine Dosis von 2 Milligramm Vitamin C erhielten.

```
print(verteilungskennwerte(dose2), "\n");

dose2_mean = verteilungskennwerte(dose2, output = False)[1]
dose2_std = verteilungskennwerte(dose2, output = False)[4]

print("Exakter Mittelwert:", dose2_mean)
print("Exakte Standardabweichung:", dose2_std)
```

```
N: 20
arithmetisches Mittel: 26.10
Stichprobenfehler: 0.84
```

```
Stichprobenvarianz: 14.24
Standardabweichung: 3.77
None

Exakter Mittelwert: 26.1
Exakte Standardabweichung: 3.7741503052098744
```

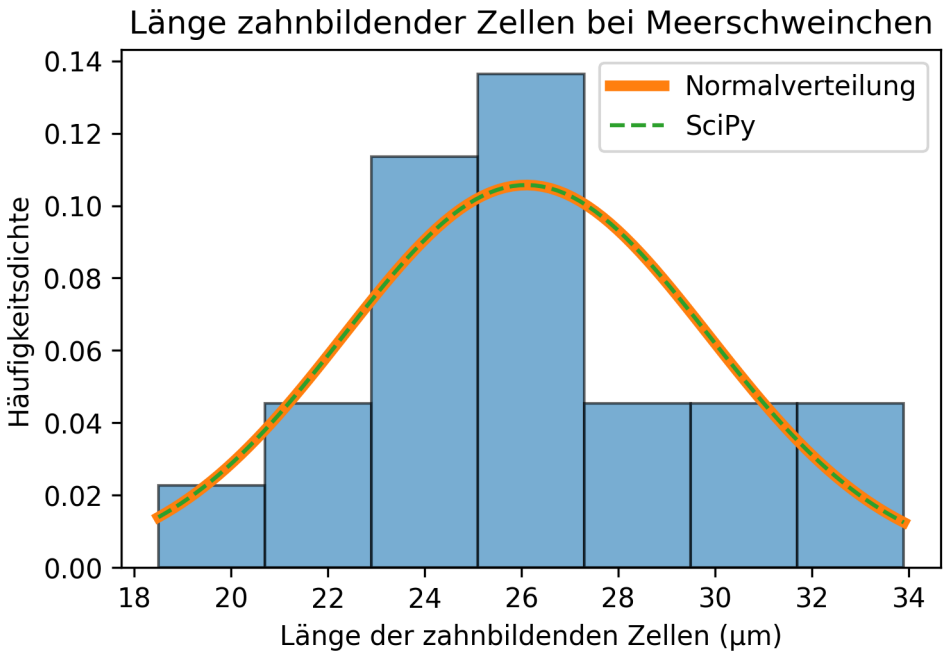
Wenn wir die Standardabweichung und das arithmetische Mittel in die Normalverteilungsfunktion einsetzen, erhalten wir:

$$f(x) = \frac{1}{3.7742\sqrt{2\pi}}\ e^{-\frac{1}{2}\left(\frac{x-26.10}{3.7742}\right)^2}$$

$$f(x) = 0.1057 \times e^{-\frac{1}{2}\left(\frac{x-26.10}{3.7742}\right)^2}$$

In Python können die Berechnungen umgesetzt und grafisch dargestellt werden:

```
# Histogram der Häufigkeitsdichte zeichnen
plt.hist(dose2, bins = 7, density = True, edgecolor = 'black', alpha = 0.6);
plt.title('Länge zahnbildender Zellen bei Meerschweinchen')

# Achsenbeschriftung
plt.xlabel('Länge der zahnbildenden Zellen ( m)')
plt.ylabel('Häufigkeitsdichte')

# Normalverteilung berechnen.
hist, bin_edges = np.histogram(dose2, bins = 7)

x_values = np.linspace(min(bin_edges), max(bin_edges), 100)

## Normalverteilungsfunktion mit Python berechnen
y_values =  1 / (dose2_std * np.sqrt(2 * np.pi)) * np.exp(- (x_values - dose2_mean) ** 2 /
plt.plot(x_values, y_values, label = 'Normalverteilung', lw = 4)

## scipy
y_values_scipy = scipy.stats.norm.pdf(x_values, loc = dose2_mean, scale = dose2_std)
plt.plot(x_values, y_values_scipy, label = 'SciPy', linestyle = 'dashed')

plt.legend()
plt.show()
```



Die Verteilung der Länge zahnbildender Zellen bei Meerschweinchen, die eine Dosis von 2 Milligramm Vitamin C erhielten, könnte einer Normalverteilung entsprechen. Aufgrund der geringen Stichprobengröße ist dies aber schwer zu beurteilen.

Quelle: Skript MB S. 51-54

0.13 Das Paket SciPy

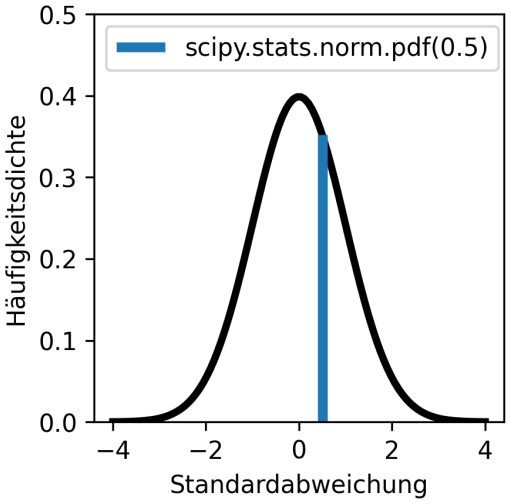
Funktionen zur Berechnung von Dichtekurven können über Paket SciPy importiert werden. Das Modul stats (statistical functions) umfasst zahlreiche Funktionen zum Testen von Hypothesen. Funktionen für die Normalverteilung werden wie folgt aufgerufen:

```
import scipy
print("Häufigkeitsdichte der Normalverteilung bei x = 0:", scipy.stats.norm.pdf(0), "\n")
```

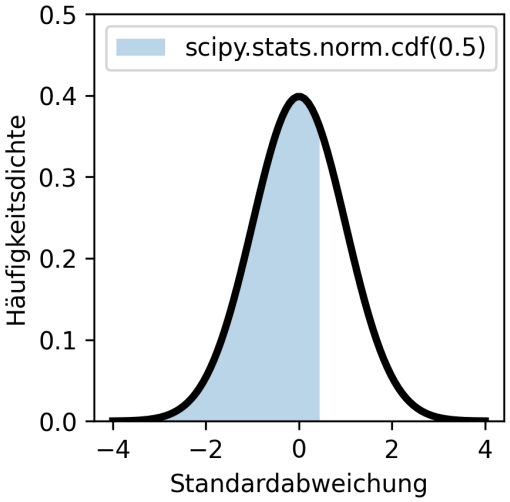
```
Häufigkeitsdichte der Normalverteilung bei x = 0: 0.3989422804014327
```

Für die Normalverteilung sind vier Funktionen relevant:

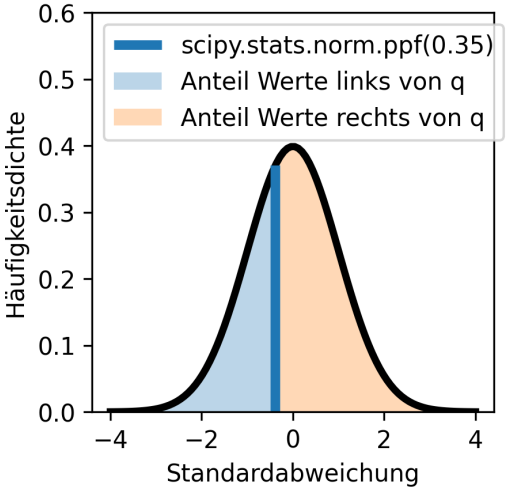
Mit den Parametern `loc = mittelwert` und `scale = standardabweichung` kann die Form der Normalverteilung angepasst werden. Standardmäßig wird die Standardnormalverteilung



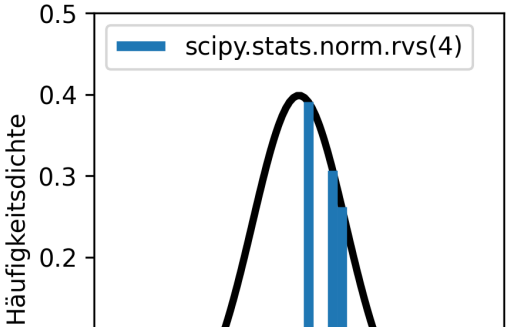
Beschreibung
Die Funktion `scipy.stats.norm.pdf(x)` berechnet die Dichte der Normalverteilung am Punkt `x` (PDF = probability density function). `x` kann auch ein array sein - so wurde die linksstehende Kurve mit dem Befehl `scipy.stats.norm.pdf(np.linspace(-4, 4, 100))` berechnet.



Beschreibung
Die Funktion `scipy.stats.norm.cdf(x)` berechnet den Anteil der Werte links von `x` (CDF = cumulative density function).



Beschreibung
Die Funktion `scipy.stats.norm.ppf(q)` ist die Quantilfunktion der Normalverteilung und die Umkehrfunktion der kumulativen Häufigkeitsdichtefunktion (CDF). Die Funktion berechnet für $0 \leq q \leq 1$ den Wert `x`, links von dem der Anteil `q` aller Werte liegt und rechts von dem der Anteil `1-q` liegt (PPF = percentile point function).



Beschreibung
Die Funktion `scipy.stats.norm.rvs(size)` zieht `size` Zufallszahlen aus der Normalverteilung.

mit `loc = 0` und `scale = 1` berechnet. Die Parameter der Funktionen können Einzelwerte (Skalare) oder auch Arrays bzw. Listen sein.

0.14 Aufgaben Normalverteilung

Möglicherweise haben Sie schon einmal von Mensa International gehört, einer Vereinigung für Hochbegabte. Wer Mitglied in dieser Vereinigung werden möchte, soll einen höheren Intelligenzquotienten (IQ) haben als 98 % der Bevölkerung seines:ihres Herkunftslandes (Wikipedia).

1. Wenn der durchschnittliche IQ 100 und die Standardabweichung 15 beträgt, welchen IQ müssten Sie haben, um bei Mensa International aufgenommen zu werden?
2. Mensa International ist nicht die einzige Organisation ihrer Art, andere Organisationen haben sogar noch strengere Kriterien. Welcher IQ wird benötigt, um hier Mitglied zu werden?

- Intertel (Kriterium: IQ aus dem höchsten 1 %)
- Triple Nine Society (Kriterium: IQ aus dem höchsten 0,1 %)
- Prometheus Society (Kriterium: IQ aus dem höchsten 0,003 %)

3. Der IQ ist nicht mit angeborener Intelligenz gleichzusetzen und auch abhängig davon, wie viel Gelegenheit man zum Gehirntraining hatte, etwa durch den Schulbesuch. Der niedrigste durchschnittliche IQ wurde mit 71 im Land Niger gemessen. Angenommen Sie hätten einen IQ von 100. Würden Sie in Niger das Kriterium der Mensa International erfüllen?

Musterlösung Normalverteilung

Aufgabe 1: Einen IQ von mehr als …

```
print(scipy.stats.norm.ppf(loc = 100, scale = 15, q = 0.98))
```

```
130.80623365947733
```

Aufgabe 2: Sie benötigen einen IQ von mindestens…

```
print(scipy.stats.norm.ppf(loc = 100, scale = 15, q = 0.99))
print(scipy.stats.norm.ppf(loc = 100, scale = 15, q = 0.999))
print(scipy.stats.norm.ppf(loc = 100, scale = 15, q = 1 - (0.003 / 100)))
```

```
134.8952181106126
146.3534845925172
160.19216216677682
```

Aufgabe 3: Nicht ganz.

```
print(scipy.stats.norm.cdf(loc = 71, scale = 15, x = 100))
```

```
0.9734024259789904
```

Übrigens: Wie der Spiegel berichtet, schneiden Studierende mit mittelmäßigem Intelligenzquotienten ebenso erfolgreich ab wie Hochbegabte, vorausgesetzt sie sind neugierig genug und arbeiten gewissenhaft.

0.15 Konfidenzintervalle

Die Grundidee in der Statistik ist, dass von Stichprobenwerten auf den tatsächlichen Wert in der Grundgesamtheit geschlossen werden kann. Die Überlegung ist wie folgt:

1. Wenn eine Stichprobe aus einer Grundgesamtheit gezogen wird, dann streuen die Stichprobenwerte normalverteilt um den Mittelwert der Grundgesamtheit. Bei einer Normalverteilung liegen

   - 68,27 % aller Werte im Intervall ± 1 s,
   - 95,45 % aller Werte im Intervall ± 2 s und
   - 99,73 % aller Werte im Intervall ± 3 s.

2. Mit der gleichen Wahrscheinlichkeitsverteilung liegt der unbekannte Mittelwert der Grundgesamtheit um einen zufälligen Wert aus der Stichprobe.

3. Der Erwartungswert kann mit einer gewissen Wahrscheinlichkeit aus dem Standardfehler des Mittelwerts einer Stichprobe geschätzt werden. Man wählt dazu ein Konfidenzniveau, also eine Vertrauenswahrscheinlichkeit, dass der Erwartungswert tatsächlich im Bereich der Schätzung liegt. Der umgekehrte Fall, dass der Erwartungswert nicht im Bereich der Schätzung liegt, wird Signifikanz- oder Alphaniveau genannt und mit dem griechischen Buchstaben α (alpha) gekennzeichnet. α liegt im Bereich 0 - 1, das Konfidenzniveau ist $1 - \alpha$ (siehe: Fehler 1. und 2. Art).

   - der Erwartungswert liegt in 68,27 % aller Fälle im Intervall ± 1 $\frac{s}{\sqrt{n}}$,
   - der Erwartungswert liegt in 95,45 % aller Fälle im Intervall ± 2 $\frac{s}{\sqrt{n}}$ und
   - der Erwartungswert liegt in 99,73 % aller Fälle im Intervall ± 3 $\frac{s}{\sqrt{n}}$.

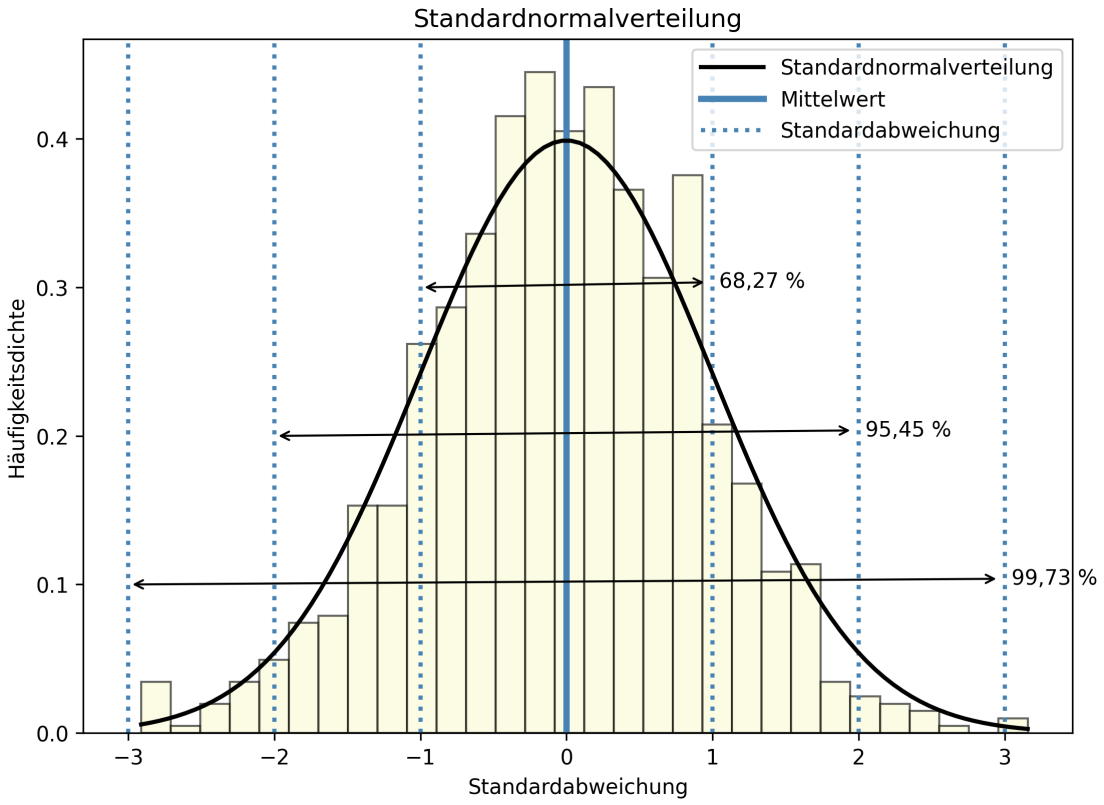
Häufig wird das Alphaniveau $\alpha = 0.05$ bzw. das Konfidenzintervall 95 % gewählt, was ± 1.96 $\frac{s}{\sqrt{n}}$ entspricht. Dies gilt aber nur für große Stichproben. Für kleine Stichprobengrößen folgen die Stichprobenmittelwerte der t-Verteilung, die im nächsten Abschnitt vorgestellt wird.

hier könnte / müsste man noch einseitige und zweiseitige Hypothesentests und den Begriff "Alpha-Halbe" einführen. Das ließe sich auch gut grafisch mit nur nach rechts gehenden und beidseitigen Pfeilen darstellen.

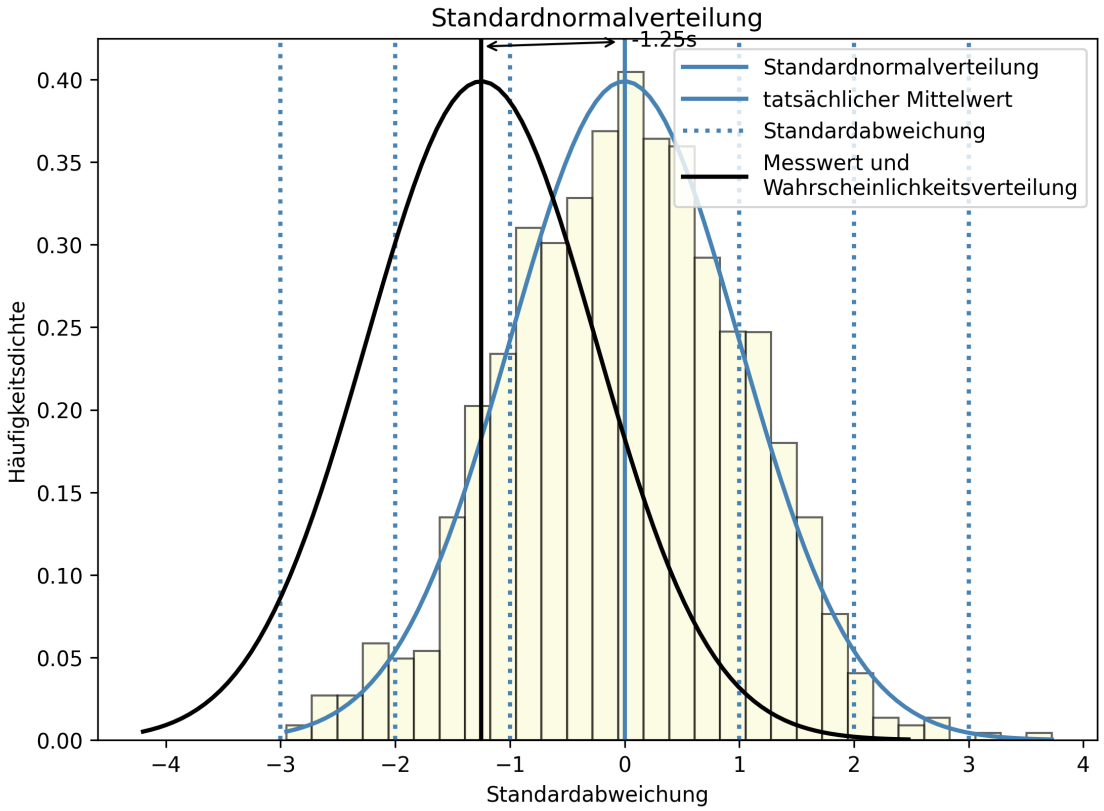
Im folgenden Beispiel wird die Idee, dass mit einer gewissen Wahrscheinlichkeit vom Stichprobenmittelwert auf den Mittelwert der Grundgesamtheit (Erwartungswert) geschlossen werden kann, noch einmal grafisch dargestellt.

Hinweis 4: Prinzip der schließenden Statistik

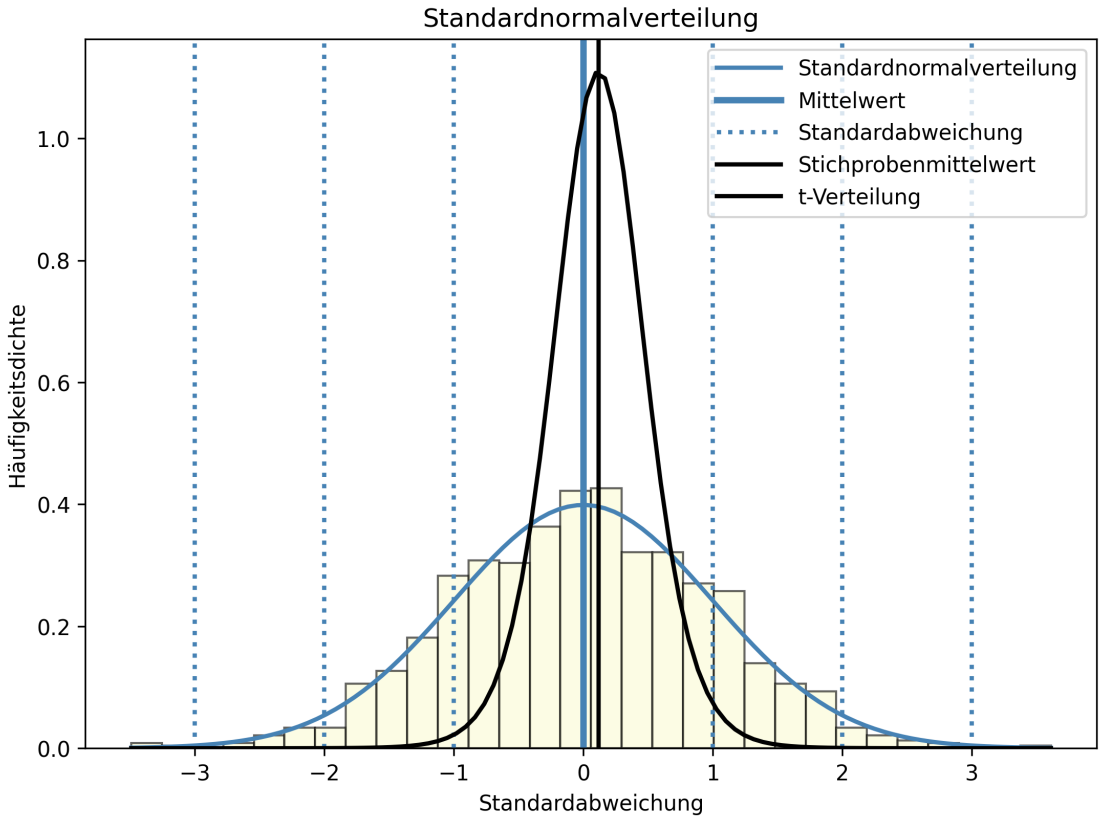
0.16 Standardnormalverteilung



0.17 Einzelner Messwert



0.18 Stichprobe N = 12



Ein Stichprobenmittelwert streut inform einer Normalverteilungskurve mit sigma = stichprobenfehler. Diese Dichtekurve ist erheblich schmaler und höher als die Normalverteilungskurve eines einzelnen Messwerts. Dies liegt an der geringen Standardabweichung in der Stichprobe von ~ 0.2, was die Kurve staucht.

0.19 Code

Code für das Panel Stichprobe N = 12

```
import numpy as np
import matplotlib.pyplot as plt
import scipy

# Parameter der Standardnormalverteilung
mu, sigma = 0, 1  # Mittelwert und Standardabweichung

# Daten generieren
seed = 4
np.random.seed(seed = seed)
data = np.random.default_rng().normal(mu, sigma, 1000)

# Grafik
plt.figure(figsize = (8.5, 6))

# Histogramm plotten
array, bins, patches = plt.hist(data, bins = 30, density = True, alpha = 0.6, color = 'ligh

# Mittelwert einzeichnen
mean_line = plt.axvline(mu, color = 'steelblue', linestyle = 'solid', linewidth = 3)

# positive und negative Standardabweichungen einzeichnen
pos_std_lines = [plt.axvline(mu + i * sigma, color = 'steelblue', linestyle = 'dotted', lin
neg_std_lines = [plt.axvline(mu - i * sigma, color = 'steelblue', linestyle = 'dotted', lin

# Normalverteilungskurve
x_values = np.linspace(min(bins), max(bins), 100)
y_values = 1 / (sigma * np.sqrt(2 * np.pi)) * np.exp(- (x_values - mu) ** 2 / (2 * sigma **
normal_dist_curve = plt.plot(x_values, y_values, color = 'steelblue', linestyle = 'solid',

# Stichprobe
N = 12
np.random.seed(seed = 4)
stichprobe = np.random.default_rng().normal(mu, sigma, N)

stichprobenstandardabweichung = stichprobe.std(ddof = 1)
stichprobenmittelwert = stichprobe.mean()
standardfehler =  stichprobenstandardabweichung / np.sqrt(len(stichprobe))

# Histogramm berechnen
# hist, bins = np.histogram(stichprobe, bins = 30, density = True)

# Standardfehlerkurve Stichprobe
# x_values = np.linspace(min(bins), max(bins), 100)
x = np.linspace(stichprobenmittelwert - 4 * stichprobenstandardabweichung, stichprobenmitte
y_values = scipy.stats.t.pdf(x = x_values, df = N - 1, loc = stichprobenmittelwert, scale =

# Stichprobenmittelwert einzeichnen
mean_stichprobe = plt.axvline(stichprobenmittelwert, color = 'black', linestyle = 'solid',

# Verteilungskurve einzeichnen
stichprobe_dist_curve = plt.plot(x_values, y_values, color = 'black', linestyle = 'solid',
```

0.20 Die t-Verteilung

Wikipedia

Anzahl der Freiheitsgrade

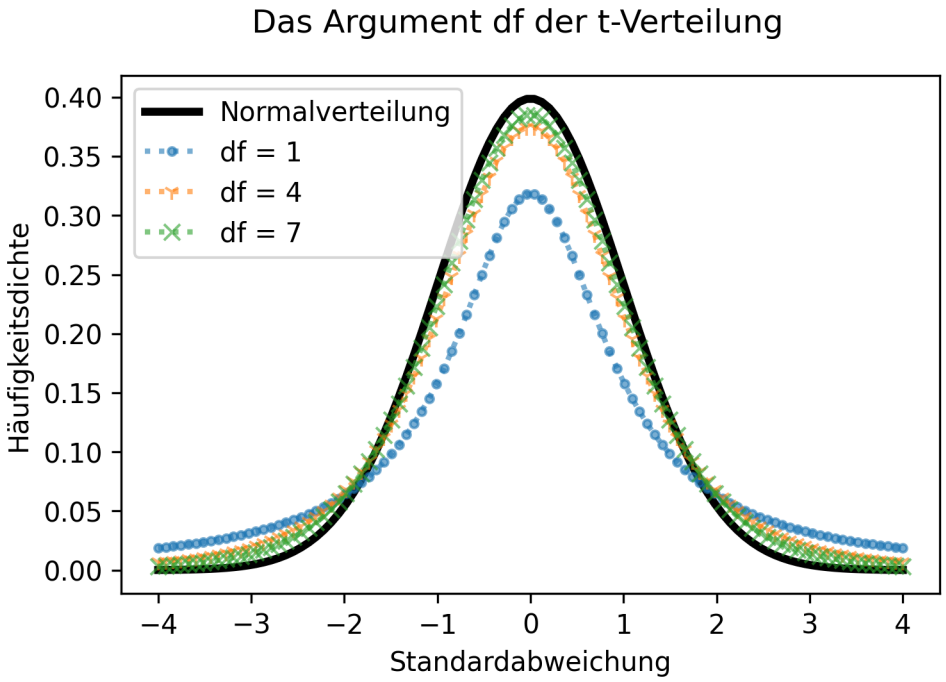
> **!** Wichtig 2: Anzahl Freiheitsgrade
>
> "Die Anzahl unabhängiger Information, die in die Schätzung eines Parameters einfließen, wird als Anzahl der Freiheitsgrade bezeichnet. Im Allgemeinen sind die Freiheitsgrade einer Schätzung eines Parameters gleich der Anzahl unabhängiger Einzelinformationen, die in die Schätzung einfließen, abzüglich der Anzahl der zu schätzenden Parameter, die als Zwischenschritte bei der Schätzung des Parameters selbst verwendet werden. Beispielsweise fließen n Werte in die Berechnung der Stichprobenvarianz ein. Dennoch lautet die Anzahl der Freiheitsgrade $n-1$, da als Zwischenschritt der Mittelwert geschätzt wird und somit ein Freiheitsgrad verloren geht."
> Anzahl der Freiheitsgrade (Statistik). von verschiedenen Autor:innen steht unter der Lizenz CC BY-SA 4.0 ist abrufbar auf Wikipedia. 2025

-
- Gammafunktion

-

-
-
-
-

0.21 Grafik


Das Argument df der t-Verteilung
Normalverteilung
df = 1
df = 4
df = 7
Häufigkeitsdichte
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4
Standardabweichung

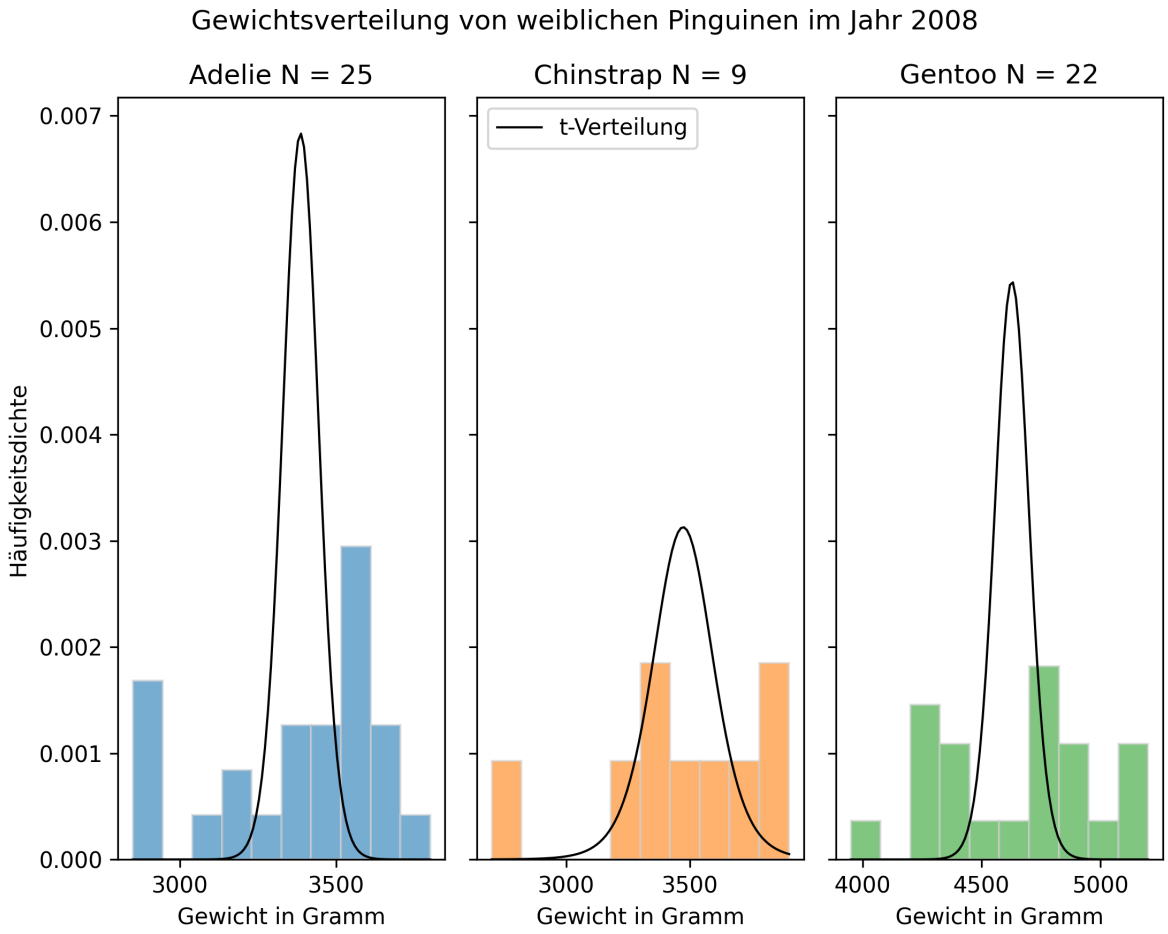
0.22 Code

```
x_values = np.linspace(-4, 4, 100)

# Normalverteilung
y_values = scipy.stats.norm.pdf(x_values)
plt.plot(x_values, y_values, color = 'black', lw = 3, label = 'Normalverteilung')
# plt.ylim(bottom = 0, top = 0.5)

# t-Verteilungen
marker = [".", "1", "x"]

[plt.plot(x_values, scipy.stats.t.pdf(x_values, df = (i + (i - 1) * 2)), linestyle = 'dotted

plt.suptitle('Das Argument df der t-Verteilung')
plt.xlabel('Standardabweichung')
plt.ylabel('Häufigkeitsdichte')
plt.legend(loc = 'upper left')
plt.show()
```

-
-
-
  - –
  - –

```
print(penguins.groupby(by = [penguins['species'], penguins['sex'], penguins['year']]).size()
```

0.23 Grafik



0.24 Code

```
year = 2008
sex = 'female'
species = 'Adelie'

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (7.5, 6), sharey = True, layout = 'tight
plt.suptitle('Gewichtsverteilung von weiblichen Pinguinen im Jahr 2008')

# Adelie
data = penguins['body_mass_g'][(penguins['species'] == species) & (penguins['sex'] == sex) &
stichprobengröße = data.size
```

```
## Histogramm
ax1.hist(data, alpha = 0.6, edgecolor = 'lightgrey', color = 'C0', density = True)
ax1.set_xlabel('Gewicht in Gramm')
ax1.set_ylabel('Häufigkeitsdichte')
ax1.set_title(label = str(species) + " N = " + str(stichprobengröße))

## t-Verteilung des Stichprobenmittelwerts
stichprobenmittelwert = data.mean()
stichprobenstandardabweichung = data.std(ddof = 1)
standardfehler = stichprobenstandardabweichung / np.sqrt(stichprobengröße)
hist, bin_edges = np.histogram(data)
x_values = np.linspace(min(bin_edges), max(bin_edges), 100)
y_values = scipy.stats.t.pdf(x_values, loc = stichprobenmittelwert, scale = standardfehler, c

ax1.plot(x_values, y_values, color = 'black', linewidth = 1, label = 't-Verteilung')
ax1.legend(loc = 'upper left')

# Chinstrap
species = 'Chinstrap'

data = penguins['body_mass_g'][(penguins['species'] == species) & (penguins['sex'] == sex) &
stichprobengröße = data.size

## Histogramm
ax2.hist(data, alpha = 0.6, edgecolor = 'lightgrey', color = 'C1', density = True)
ax2.set_xlabel('Gewicht in Gramm')
ax2.set_title(label = str(species) + " N = " + str(stichprobengröße))

## t-Verteilung des Stichprobenmittelwerts
stichprobenmittelwert = data.mean()
stichprobenstandardabweichung = data.std(ddof = 1)
standardfehler = stichprobenstandardabweichung / np.sqrt(stichprobengröße)
hist, bin_edges = np.histogram(data)
x_values = np.linspace(min(bin_edges), max(bin_edges), 100)
y_values = scipy.stats.t.pdf(x_values, loc = stichprobenmittelwert, scale = standardfehler, c

ax2.plot(x_values, y_values, color = 'black', linewidth = 1)

# Gentoo
species = 'Gentoo'

data = penguins['body_mass_g'][(penguins['species'] == species) & (penguins['sex'] == sex) &
```

```
stichprobengröße = data.size

## Histogramm
ax3.hist(data, alpha = 0.6, edgecolor = 'lightgrey', color = 'C2', density = True)
ax3.set_xlabel('Gewicht in Gramm')
ax3.set_title(label = str(species) + " N = " + str(stichprobengröße))

## t-Verteilung des Stichprobenmittelwerts
stichprobenmittelwert = data.mean()
stichprobenstandardabweichung = data.std(ddof = 1)
standardfehler = stichprobenstandardabweichung / np.sqrt(stichprobengröße)
hist, bin_edges = np.histogram(data)
x_values = np.linspace(min(bin_edges), max(bin_edges), 100)
y_values = scipy.stats.t.pdf(x_values, loc = stichprobenmittelwert, scale = standardfehler, c

ax3.plot(x_values, y_values, color = 'black', linewidth = 1)

plt.show()
```

0.25 Aufgabe Konfidenzintervalle

1.

2.

Tipp 1: Tipp und Musterlösung

Folgende Schritte helfen Ihnen bei der Lösung:

1. Bestimmen Sie den Stichprobenmittelwert $\bar{x}$.
2. Bestimmen Sie die Stichprobenstandardabweichung s, die Stichprobengröße N und den Standardfehler $\frac{s}{\sqrt{N}}$.
3. Bestimmen Sie die z- oder t-Werte der Normal- bzw. t-Verteilung für das gewählte Konfidenzniveau - für einen zweiseitigen Hypothesentest $\frac{\alpha}{2}$ und $1 - \frac{\alpha}{2}$
4. Berechnen Sie das Konfidenzintervall $\bar{x} \pm t_{\alpha/2} \frac{s}{\sqrt{n}}$.

Musterlösung

Alphaniveau definieren und Pinguine auswählen

```
alpha = 1 - 0.9
data = penguins[(penguins['sex'] == sex) & (penguins['year'] == year)]
```

1. Stichprobenmittelwerte bestimmen.

```
penguin_means = data['body_mass_g'].groupby(by = data['species']).mean()
print(penguin_means)
```

```
species
Adelie       3386.000000
Chinstrap    3472.222222
Gentoo       4627.272727
Name: body_mass_g, dtype: float64
```

2. Stichprobenstandardabweichung, Stichprobengröße und Standardfehler bestimmen.

```
penguin_stds = data['body_mass_g'].groupby(by = data['species']).std(ddof = 1)
penguin_sizes = data['body_mass_g'].groupby(by = data['species']).size()
penguin_stderrors = penguin_stds / np.sqrt(penguin_sizes)

print("Stichprobenstandardabweichungen:\n", penguin_stds)
print("\nStichprobengrößen:\n", penguin_sizes)
print("\nStandardfehler:\n", penguin_stderrors)
```

```
Stichprobenstandardabweichungen:
 species
Adelie       288.862712
Chinstrap    370.903551
Gentoo       339.722321
Name: body_mass_g, dtype: float64
```



```
Stichprobengrößen:
 species
Adelie       25
Chinstrap     9
Gentoo       22
Name: body_mass_g, dtype: int64
```