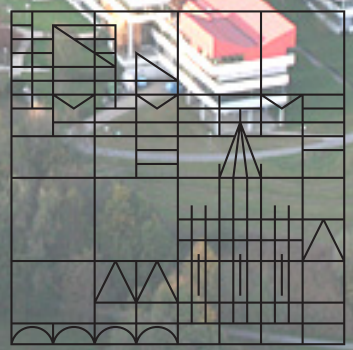
Computerphysik II

Teil 1 - Statistik und Datenanalyse

S. Gerlach

WiSe 2021/22

Universität
Konstanz



Statistik und Datenanalyse

INHALT

1. Kombinatorik
2. Diskrete Wahrscheinlichkeitsverteilungen
3. Numerische Integration
4. Kontinuierliche Wahrscheinlichkeitsverteilungen
5. Schätzmethoden und Anpassung von Funktionen

Kombinatorik - Permutation

Permutation: Anordnung von n Elementen ($n = 3$: 123, 132, 321, 213, 231, 312, 321)
Anzahl:

$$P_n = n!$$

Anzahl mit Wdh.:

$$P_n = \frac{n!}{n_1! n_2! \cdots n_k!}$$

Berechnung:
$n! = 1 \cdot 2 \cdots n$ (iterativ)
$n! = n(n-1)!$ (rekursiv)
In Python: `math.factorial(int)`

Kombinatorik - Permutation

Für große n (Striling-Formel):

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n$$

bzw. $\lg(n!) \approx n(\ln n - 1)$.

Anwendungen Fakultät:

- Binomialkoeffizient (siehe Kombination)
- Binomischer Lehrsatz
- Pascalsches Dreieck
- Exponentialfunktion ($\sum_{n=0}^{\infty} \frac{x^n}{n!}$)
- Taylorreihe
- Leibniz-Produktregel
- ...

Kombinatorik - Variation

Variation: Auswahl von k aus n Elementen (2 aus 3: 12, 13, 21, 23, 31, 32)

Reihenfolge wichtig

Anzahl:

$$V_{n,k} = \frac{n!}{(n-k)!}$$

Anzahl mit Wdh.:

$$V_{n,k}^{w} = n^k$$

Kombinatorik - Kombination

Kombination: Auswahl von k aus n Elementen (2 aus 3: 12, 13, 23)

Reihenfolge egal

Anzahl:

$$C_{n,k} = \frac{V_{n,k}}{P_k} = \frac{n!}{k!(n-k)!} = \binom{n}{k}.$$

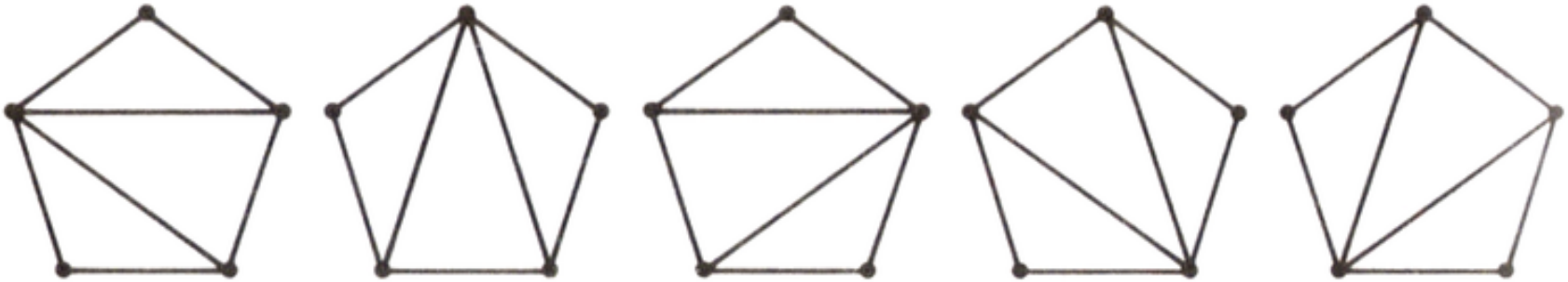
```
1 from scipy.misc import comb
2 print ("%d over %d =" % (n, k), comb(n, k, exact=True))
```

In Python: `itertools` (s. Übung)

Tipp: `list(...)` erzeugt eine iterierbare Liste, `set` eine Menge (ohne Wiederholungen)

Kombinatorik - Beispiel

Catalan-Zahlen



https://de.wikipedia.org/wiki/Catalan-Zahl

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

$$C_{n+1} = \frac{2(2n+1)}{n+2} C_n$$

Diskrete Wahrscheinlichkeitsverteilungen

Zuerst: Zufallsexperimente mit diskreter Anzahl an Ergebnissen.

1. Beispiel: **Binomialverteilung**
Mit welcher Wahrscheinlichkeit gibt es k positive Ergebnisse bei n Versuchen eines Bernoulli-Experiments ("Münzwurf")?

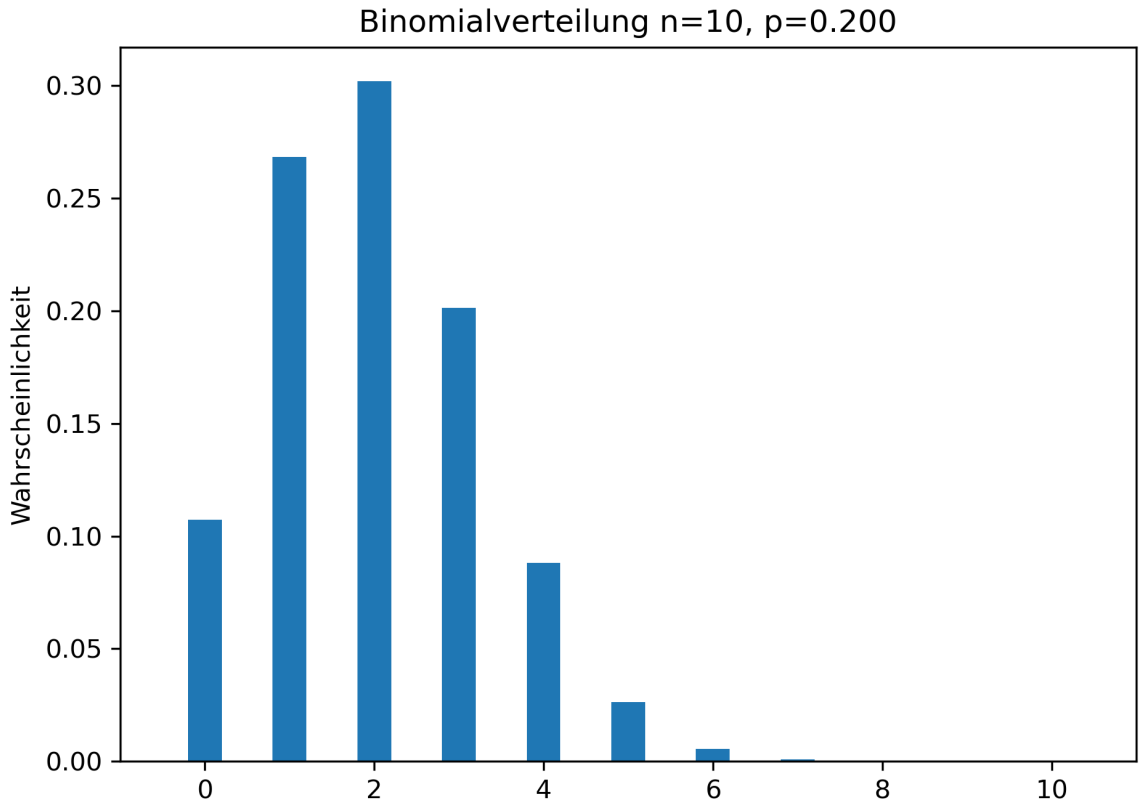
$$\boxed{P(k) = B_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k}}$$

Mit $k = 0, \cdots, n$. $\binom{n}{k}$ ist die Anzahl der Kombinationen, da egal ist, wann die positiven Ergebnisse auftreten.

Ideale Münze ($p = 1/2$): $P(k) = \binom{n}{k} \frac{1}{2}^k \frac{1}{2}^{n-k} = \frac{1}{2^n} \binom{n}{k} = \frac{C_{n,k}}{V_{n,2}}$

Binomialverteilung

```
1 from scipy.stats import binom
2 import pylab as pl
3
4 k=pl.arange(0, n + 1, 1)
5 pl.bar(k,binom.pmf(k,n,p),width=0.4,align="center")
```

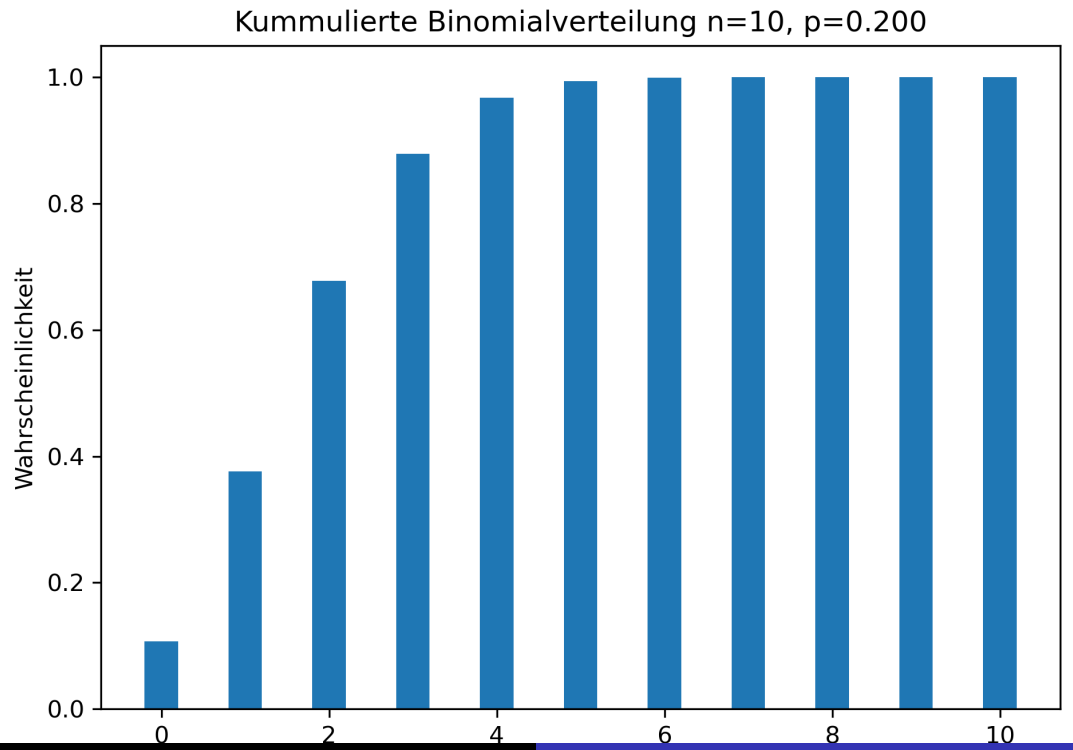


Kumulative Binomialverteilung

Wieviele Ereignisse sind kleiner als ein Wert k_0: Summierte Wahrscheinlichkeit

$$F_K(k_0) = P(K < k_0) = \sum_{k=0}^{\lfloor k_0 \rfloor} \binom{n}{k} p^k (1-p)^{n-k}$$

```
1 pl.bar(k0,binom.cdf(k0,n,p),width=0.4,align="center")
```



Binomialverteilung

Eigenschaften:

- **Normierung**:
  $\sum_{k=0}^{n} P(k) = \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \overset{\text{binom.Lehrsatz}}{=}$
  $(p + (1-p))^n = 1$
- **Erwartungswert**: $\mu = \langle k \rangle = \sum_{k=0}^{n} kP(k) = \cdots = np$
  (= Summe der einzelnen Erwartungswerte)
- **Varianz**: $\sigma^2 = \sum_{k=0}^{n} (k-\mu)^2 P(k) = \langle (x - \mu^2) \rangle = \cdots =$
  $\langle x^2 \rangle - \langle x \rangle^2 = \cdots = np(1-p)$.

Damit ist die **relative Breite**:

$$\frac{\sigma}{\mu} = \sqrt{\frac{1-p}{np}} \sim \frac{1}{\sqrt{n}}$$

D.h. Mittelwert lässt sich für $n \to \infty$ beliebig genau bestimmen.

Binomialverteilung - Beispiele

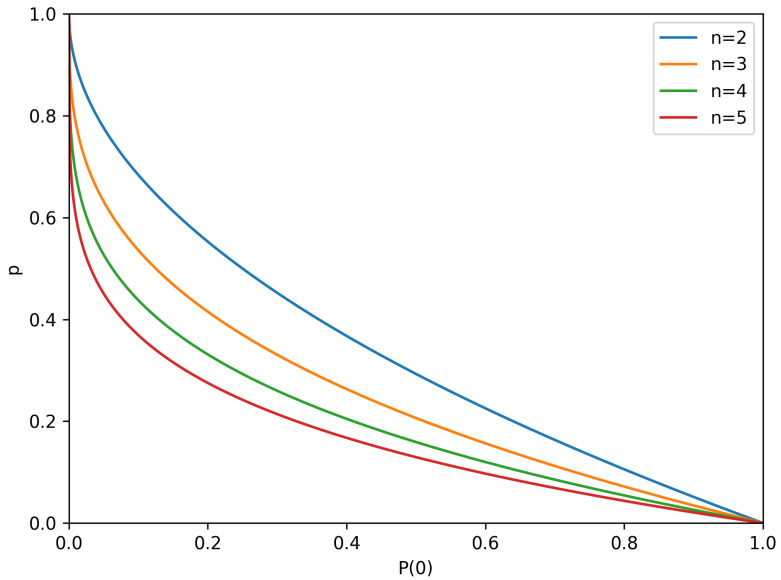
1. **Qualitätssicherung**:
Situation: Produkte am Fliessband sollen kontrolliert werden. Gesucht ist die Fehlerquote p.
Idee: Messe Wahrscheinlichkeit für Null Fehler $P(0)$ bei Auswahl von jeweils n Teilen:

$$P(0) = B_{n,p}(k=0) = \binom{n}{0} p^0 (1-p)^{n-0} = (1-p)^n$$

Damit

$$\boxed{p = 1 - \sqrt[n]{P(0)}}.$$

Beispiel: $n = 3$, gemessen $P(0) = 0,25 \rightarrow p = 0,37$.



Binomialverteilung - Beispiele

2. **Ising-Spinkette**: s. Buch S. 248

Poissonverteilung

Grenzfall $n \to \infty$ bei konstantem Erwartungswert $\lambda = np$ (d. h. $p \to 0$):

$$\boxed{P_\lambda(k) = \frac{\lambda^k}{k!} \mathrm{e}^{-\lambda}} \qquad \textbf{Poissonverteilung}$$

Anwendung: viele, aber seltene Ereignisse

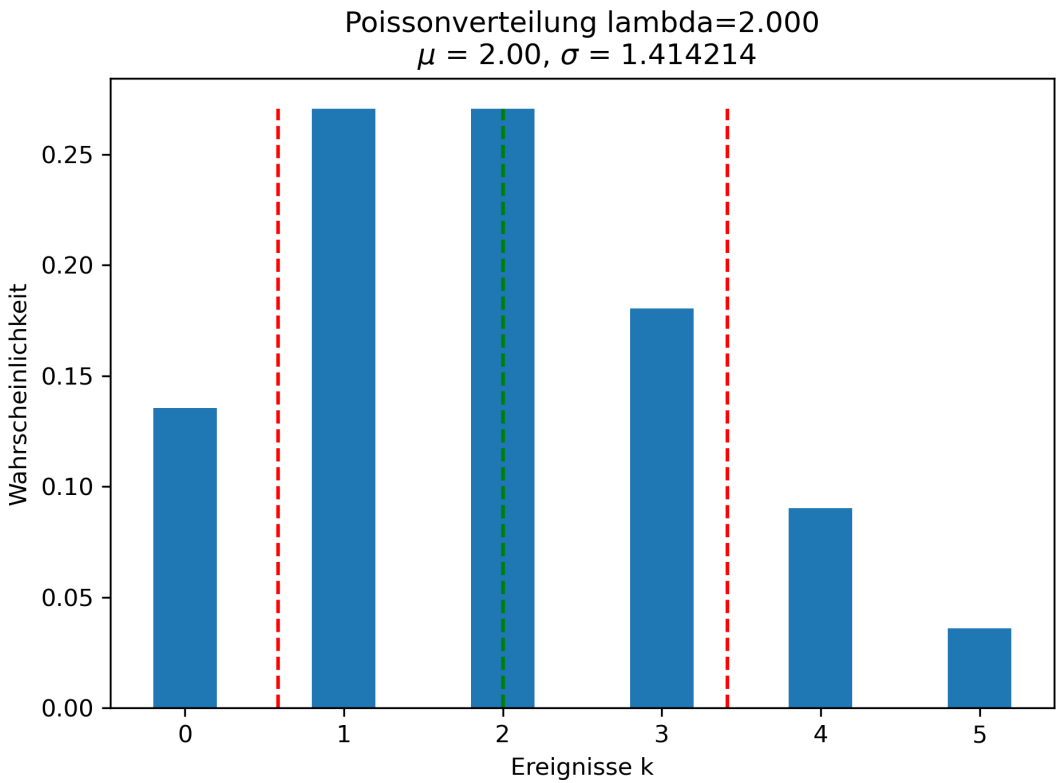
```
1 from scipy.stats import poisson
2 pl.bar(k, poisson.pmf(k, l), width=0.4, align="center")
```

Poissonverteilung

Eigenschaften:

- Normierung:
  $\sum_{k=0}^{n} P(k) = \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} e^{-\lambda} = e^{-\lambda} \sum_{k=0}^{\infty} \frac{\lambda^k}{k!} = e^{-\lambda} e^{\lambda} = 1.$
- Erwartungswert: $\mu = \sigma^2 = \cdots = \lambda$

Relative Breite: $\frac{\sigma}{\mu} = \sim \frac{1}{\sqrt{\lambda}}$.

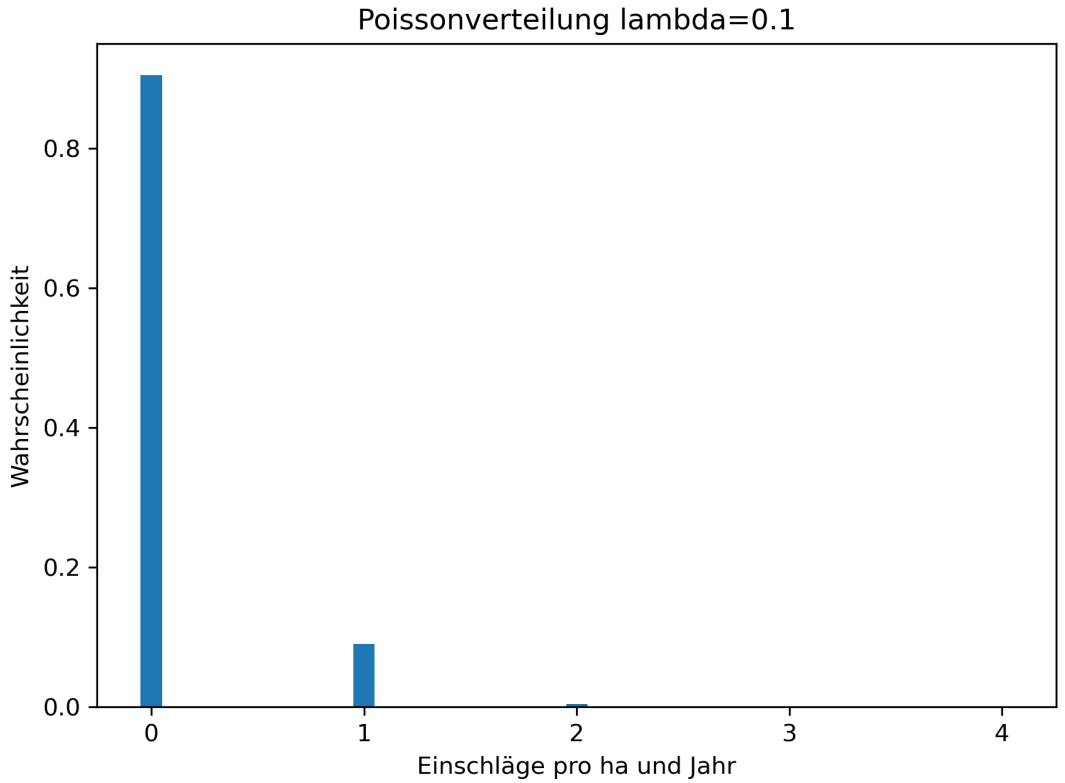


Poissonverteilung - Beispiel 1 - Blitzeinschläge

Im Mittel wurden 0,1 Blitzeinschläge pro Hektar in Deutschland registriert. Mit welcher Wahrscheinlichkeit schlägt kein, ein, zwei, ... Blitze auf einen Hektar ein?

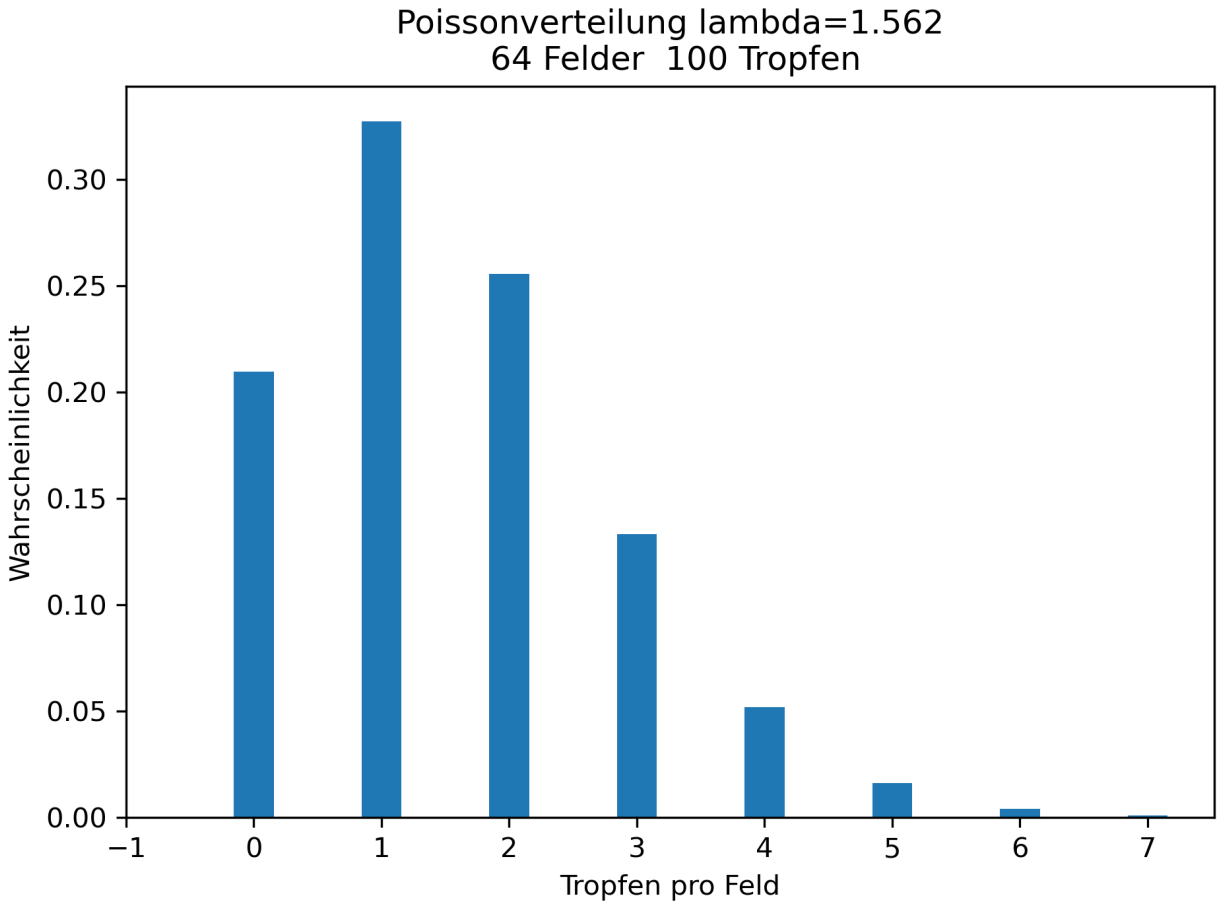
$\mu = 0,1 = \lambda$

$\rightarrow$ 90% kein Einschlag, 10% ein Einschlag.



Poissonverteilung - Beispiel 2 - Schachbrett

Es regnet auf ein Schachbrett. Wieviele Tropfen pro Feld erwartet man?
$N = 64$ Felder, $n = 100$ Tropfen: $\lambda = \mu = n/N \approx .1,56$:
Immerhin mit 5% Wahrscheinlichkeit 4 Regentropfen im Feld.

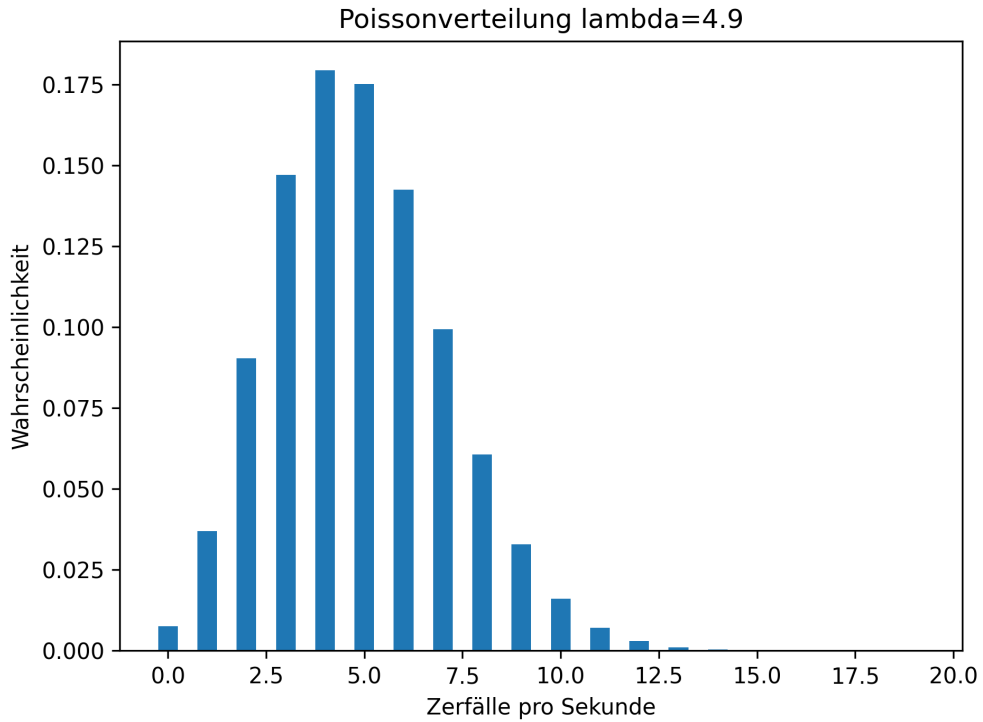


Poissonverteilung - Beispiel 3 - Radioaktiver Zerfall

Die Zerfälle pro Zeiteinheit Δt folgen für $\Delta t \ll t_{\mathrm{H}}$ der Poissonverteilung mit $\lambda = pN$. $p \approx \alpha \Delta t$ ist die Wahrscheinlichkeit für einen Zerfall, N die Anzahl der Teilchen.
Beispiel: $N = 10^{18}$ Atome, $t_{\mathrm{H}} = 4,5$ Mrd. Jahre. Damit $\alpha = \ln 2 / t_{\mathrm{H}}$ und für $\Delta t = 1$ s:

$$\lambda = \alpha N \Delta t \approx 4,9.$$

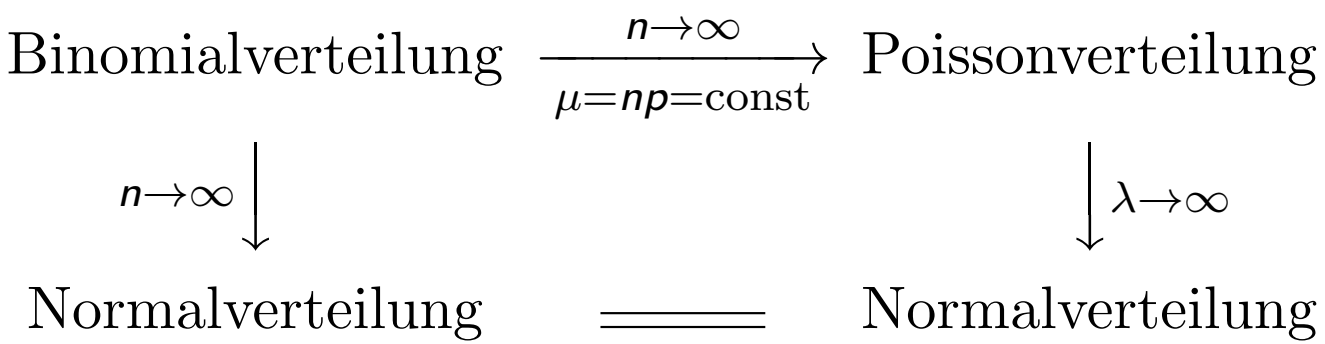
Es gibt also im Mittel 5 Zerfälle pro Sekunde mit der Verteilung:



Übergang

Mehr **Beispiele**: Übungsblatt

Diskrete Verteilungen oft unhandlich. Betrachte **Grenzfälle**:



→ **Kontinuierliche Wahrscheinlichkeitsverteilungen**

Numerische Integration

Um mit kontinuierlichen Wahrscheinlichkeitsverteilungen zu arbeiten, müssen wir nun nicht mehr Summieren, sondern integrieren. Numerisch brauchen wir dafür die **Numerische Integration**.

Dafür sind wichtig:

- Newton-Cotes-Formeln und zusammengesetze Formeln
- Uneigentliche Integrale
- Gauß-Quadratur

$\rightarrow$ siehe Buch Kapitel 9.5

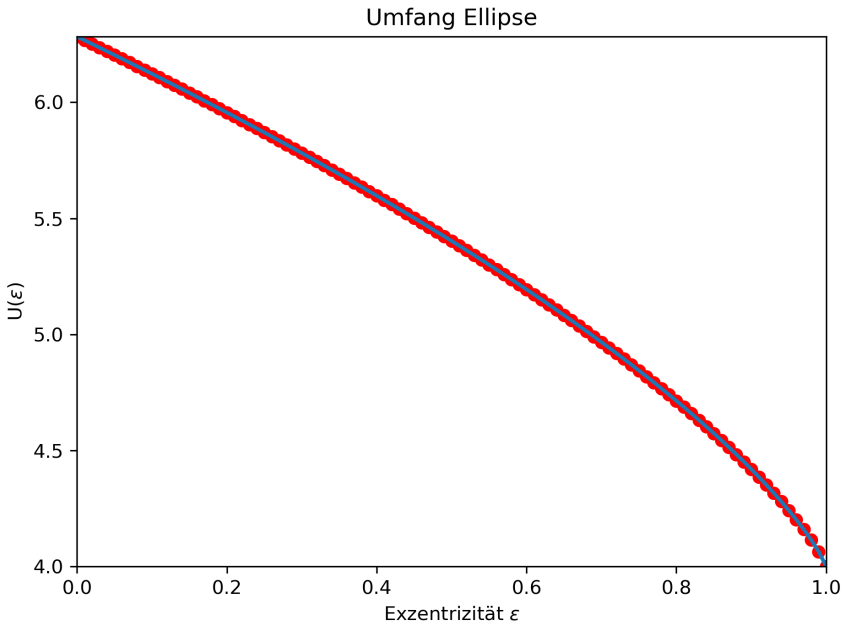
Num. Integration - Beispiel: Kurvenlänge einer Ellipse

Mit $\gamma(t) = \begin{pmatrix} a\cos t \\ b\sin t \end{pmatrix}$ $(t \in [0, 2\pi])$:

$$|\gamma(t)| = \sqrt{a^2 \sin^2 t + b^2 \cos^2 t} = a\sqrt{1 - \varepsilon^2 \cos^2 t}.$$

Damit ist die Kurvenlänge:

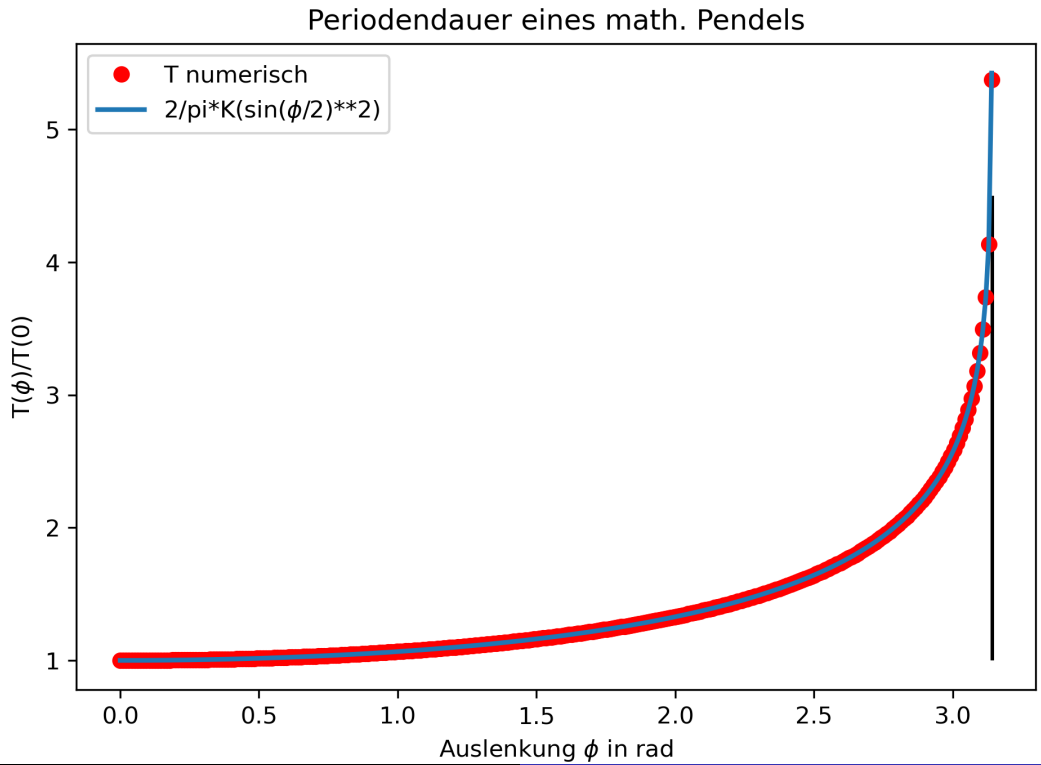
$$L = \int_0^{2\pi} |\gamma(t)| \, \mathrm{d}t = 4a \underbrace{\int_0^{\pi/2} \sqrt{1 - \varepsilon^2 \cos^2 t} \, \mathrm{d}t}_{E(\varepsilon^2), \text{Elliptisches Integral}}.$$



Num. Integration - Beispiel: Periodendauer des math. Pendels

$$T = \frac{4}{\omega} \int_0^{\pi/2} \frac{1}{\sqrt{1 - \sin^2 \frac{\varphi_0}{2} \sin^2 \varphi}} \, \mathrm{d}\varphi = \frac{4}{\omega} \, \mathrm{K}(\sin^2 \frac{\varphi_0}{2}).$$

`scipy.special.ellipk(k**2)`



Kontinuierliche Verteilungen

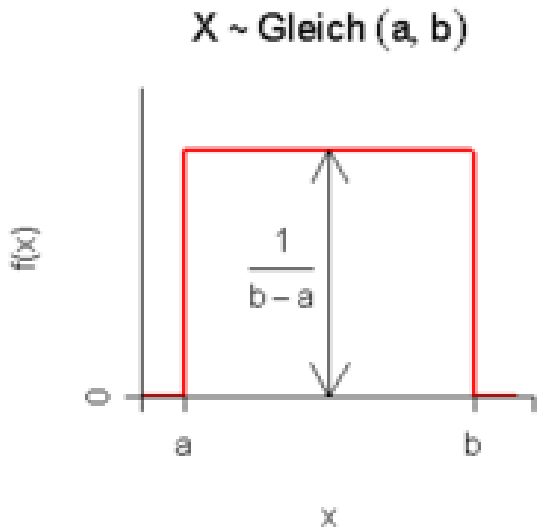
$\rightarrow$ siehe Buch Kapitel 18.1.3

- Gleichverteilung
- Exponentialverteilung (Buch)
- Lorentz-Verteilung
- Normalverteilung (Buch)

Beispiele: Maxwell-Boltzmann (Buch)
Mehr Beispiele: Übung

Gleichverteilung

$$\boxed{P(x) = \frac{1}{b-a} \quad (a \leq x \leq b)}$$

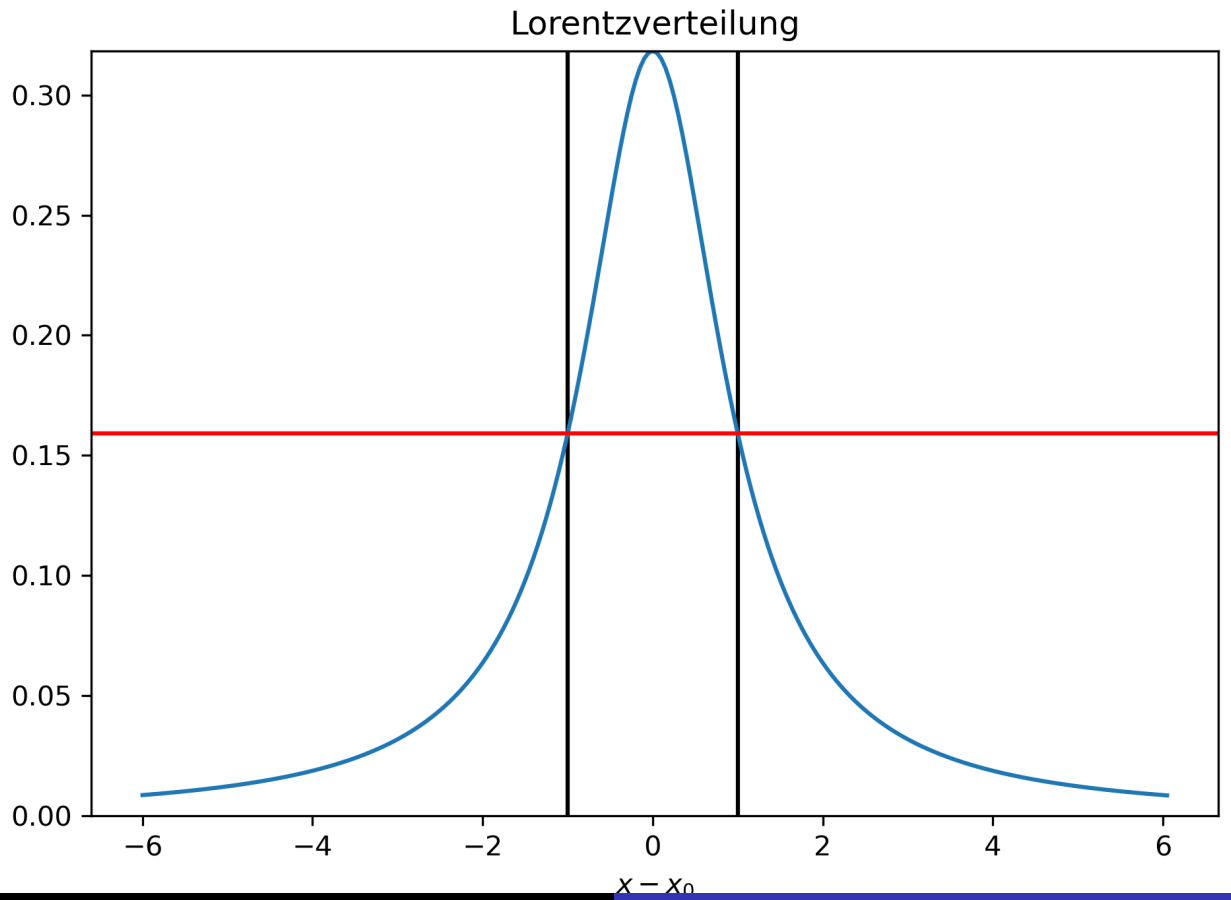


- Normierung: $\int_{-\infty}^{\infty} P(x)\,\mathrm{d}x = \int_a^b \mathrm{d}x = .. = 1$
- Erwartungswert: $\mu = \int_a^b xP(x)\,\mathrm{d}x = .. = \frac{a+b}{2}$
- Varianz: $\sigma^2 = \int_a^b (x-\mu)^2 P(x)\,\mathrm{d}x = .. = \frac{(b-a)^2}{12}$
- $\rightarrow$ Standardabweichung für $[0,1]$: $\sigma_{[0,1]} = 1/\sqrt{12}$

Lorentz-Verteilung

(In der Mathematik: Cauchy-Lorentz-Verteilung)

$$f(x) = \frac{1}{\pi} \frac{\gamma}{\gamma^2 + (x - x_0)^2}$$



Lorentz-Verteilung

Ursprung: Fourier-Trafo vom exp. Zerfall
Daher bekannt aus: Resonanzkurve, Spektrallinie,
"Breit-Wigner"-Resonanz

Eigenschaften:

- Erwartungswert: $\mu = \int_{-\infty}^{\infty} x f(x)\,\mathrm{d}x = \infty$!
- Varianz: $\sigma^2 = \int_{-\infty}^{\infty} x^2 f(x)\,\mathrm{d}x = \infty$!

Verwende daher Median x_0 und FWHM (*full width half maximum*)
$\Gamma = 2\gamma$ als Linienbreite.

Vertrauensintervall: $\int_{-\Gamma/2}^{\Gamma/2} = 50\%$.
CDF: $F(x) = \int_{-\infty}^{x} f(x')\,\mathrm{d}x' = \frac{1}{\pi}\,\mathrm{atan}(\frac{x-x_0}{\gamma}) + \frac{1}{2}$.

Schätzmethoden

Suche anhand einer Stichprobe die **besten Parameter** einer Verteilung bzw. Funktion
$\rightarrow$ **Anpassung** der Verteilung/Funtion an die Stichprobe/Messwerte durch Optimierung der Parameter
2 wichtige **Verfahren**:

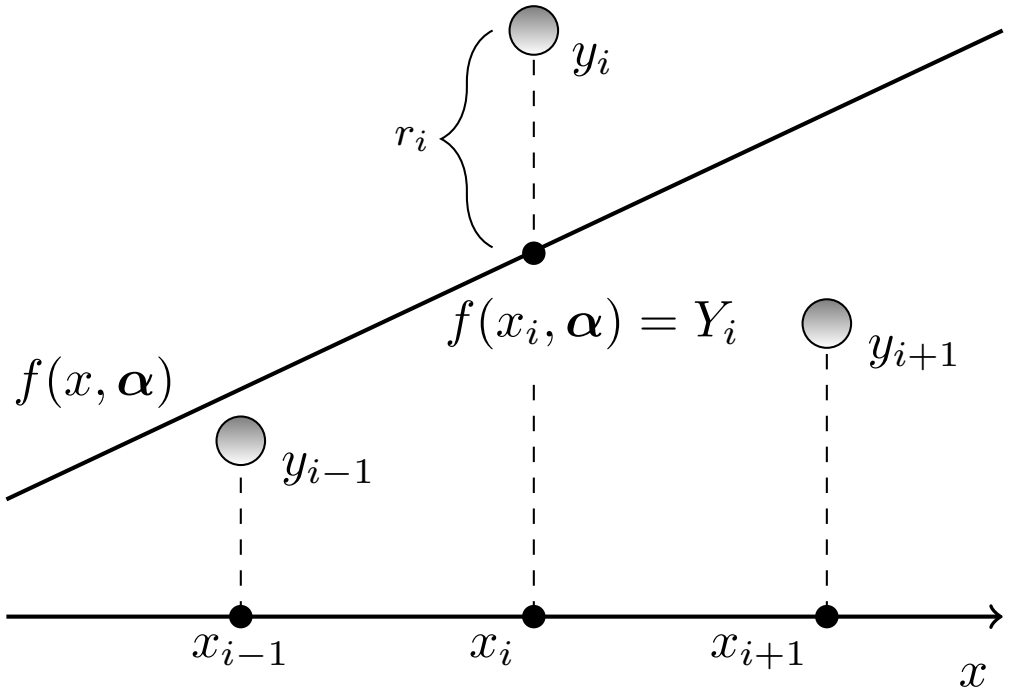
- **Kleinste-Quadrate-Methode** (*least square*, LS) s. Kapitel 9.6
- **Maximum Likelyhood-Methode** [ML] $\rightarrow$ s. Kapitel 18.2.1

Anpassung an Daten

Anpassung (*fit*) einer Modellfunktion f an (Mess-)Daten durch **Optimierung der Modellparameter**.

Messwerte: $y_i(x_i)$ $(i = 1, \cdots, n)$
Modellfunktion: $f(x; \underbrace{\alpha_1, \cdots, \alpha_m}_{=\boldsymbol{\alpha}, \text{Modellparameter}})$

$\rightarrow$ Suche Parameter $\boldsymbol{\alpha}$, sodass die Abweichungen (Residuen) des Modell von den Daten möglichst klein sind.



Kleinste-Quadrate-Methode

$\rightarrow$ Minimiere die quadratische Abweichungen der Modellfunktion (f) von den Daten (y):

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{N} (y_i - f(x_i, \boldsymbol{\alpha}))^2 = \min_{\boldsymbol{\alpha}} \sum_{i=1}^{N} r_i^2(\boldsymbol{\alpha}) = \min_{\boldsymbol{\alpha}} \chi^2(\boldsymbol{\alpha}). \qquad (1)$$

χ^2: sum of squared residuals (SSR) / residual sum of squares (RSS)

Beispiele:

- **Lineare Anpassung**/Regression: Modellfunktion ist linear ($f(x; a, b) = a + bx$) $\rightarrow$. Optimale Parameter lassen sich direkt berechen. ($\rightarrow$ Kapitel 9.6.1)
- **Polynomregression**: Modellfunktion ist ein Polynom ($\rightarrow$ Kapitel 9.6.2)
- **Polynomapproximation**: Anpassung einer bel. Funktion mit einem Polynom ($\rightarrow$ Kapitel 18.2.2)
- Allgemeiner Fall: **Nichtlineare Anpassung**

Lineare Regression

Modellfunktion: $f(x; a, b) = a + bx$, $\langle x \rangle = \frac{1}{n} \sum_i x_i$

$$\min_{\boldsymbol{\alpha}} \chi^2(\boldsymbol{\alpha}) = \min_{a,b} \sum_{i=1}^{n} (y_i - a - bx_i)^2 :$$

$$0 = \frac{\mathrm{d}\chi^2}{\mathrm{d}b} = -2 \sum_i (y_i - a - bx_i) x_i$$

$$\sum_i x_i y_i = b \sum_i x_i^2 + a \sum_i x_i$$

$$\langle xy \rangle = b \langle x^2 \rangle + a \langle x \rangle \tag{2}$$

$$0 = \frac{\mathrm{d}\chi^2}{\mathrm{d}a} = -2 \sum_i (y_i - a - bx_i)$$

$$\sum_i y_i = b \sum_i x_i + a \sum_i$$

$$\langle y \rangle = b \langle x \rangle + a \tag{3}$$

$$\begin{pmatrix} \langle y \rangle \\ \langle xy \rangle \end{pmatrix} = \begin{pmatrix} 1 & \langle x \rangle \\ \langle x \rangle & \langle x^2 \rangle \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \tag{4}$$

Lineare Regression

$$a = \langle y \rangle - b\langle x \rangle \tag{5}$$

$$b = \frac{\langle xy \rangle - \langle x \rangle \langle y \rangle}{\langle x^2 \rangle - \langle x \rangle^2} = \frac{\mathrm{cov}(x,y)}{\sigma_x^2} \tag{6}$$

Ergebnis:

$$\begin{pmatrix} b \\ a \end{pmatrix} = \frac{1}{\sigma_x^2} \begin{pmatrix} \mathrm{cov}(x,y) \\ \langle x^2 \rangle \langle y \rangle - \langle x \rangle \langle xy \rangle \end{pmatrix} \tag{7}$$

Kovarianz:

$$\mathrm{cov}(x,y) = \langle (x - \langle x \rangle)(y - \langle y \rangle) \rangle = \langle xy + \langle x \rangle \langle y \rangle - x\langle x \rangle - y\langle x \rangle \rangle = \langle xy \rangle - \langle x \rangle \langle y \rangle$$

$$\mathrm{cov}(x,y) = 0 \rightarrow \langle xy \rangle = \langle x \rangle \langle y \rangle (\text{unkorreliert})$$

$$\mathrm{cov}(x,x) = \langle (x - \langle x \rangle)^2 \rangle = \langle x^2 \rangle - \langle x \rangle^2 = \sigma_x^2$$

Kovarianzmatrix:

$$C(x,y) = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) \end{pmatrix} = \begin{pmatrix} \sigma_x^2 & \mathrm{cov}(x,y) \\ \mathrm{cov}(x,y) & \sigma_y^2 \end{pmatrix} \tag{8}$$

Lineare Regression

Korrelationskoeff. (norm. Kovarianz):

$$\rho_{x,y} = \frac{\mathrm{cov}(x,y)}{\sqrt{\sigma_x^2 \sigma_y^2}}$$

$\rightarrow$ Correlation.png

Beispiel:

x_i	1	2	3
y_i	1	3	2

$\langle x \rangle = 2$, $\langle x \rangle = 2$, $\langle xy \rangle = 13/3$, $\langle x^2 \rangle = \langle y^2 \rangle = 14/3$
$\rightarrow \sigma_x^2 = \langle x^2 \rangle - \langle x \rangle^2 = 2/3, \sigma_y^2 = \langle y^2 \rangle - \langle x \rangle^2 = 2/3$
$\mathrm{cov}(x,y) = \langle xy \rangle - \langle x \rangle \langle y \rangle = 1/3$

$$\begin{aligned} b &= 1/3/2/3 = 1/2 \\ a &= 2 - 1/2 * 2 = 1 \end{aligned}$$

$$\boxed{f(x) = x/2 + 1}$$

$$\rho_{x,y} = \frac{1/3}{\sqrt{2/3 \cdot 2/3}} = 1/2 > 0$$

Lineare Regression

Über die genaue Bezeichnung und ihre Abkürzungen gibt es international keine Einigkeit. Die natürliche deutsche Abkürzung für die Residuenquadratsumme bzw. die Summe der (Abweichungs-)Quadrate der Restabweichungen (oder: "Residuen"), ist SAQRest, oder SQR. Die englische Abkürzung SSR ist vieldeutig und führt zu anhaltenden Verwechslungen: Sowohl Sum of Squared Residuals (Residuenquadratsumme) als auch Sum of Squares due to Regression (Regressionsquadratsumme) werden als SSR abgekürzt. Allerdings wird die Regressionsquadratsumme oft auch als erklärte Quadratsumme (Sum of Squares Explained) bezeichnet, deren natürliche englische Abkürzung SSE ist. Die Abkürzungsproblematik wird dadurch verschärft, dass die Residuenquadratsumme oft auch als Fehlerquadratsumme (Sum of Squares Error) bezeichnet wird, deren natürliche englische Abkürzung ebenfalls SSE ist (diese Bezeichnung ist besonders irreführend, da die Fehler und die Residuen unterschiedliche Größen sind). Des Weiteren findet sich für Residuenquadratsumme ebenfalls die englische Abkürzung RSS, statt der Abkürzung SSR, da statt der Bezeichnung Sum of Squared Residuals, oft auch die Bezeichnung Residual Sum of Squares verwendet wird. Auch diese englische Abkürzung kann mit der Regressionsquadratsumme verwechselt werden, die im Englischen auch als Regression Sum of Squares bezeichnet, deren natürliche englische Abkürzung auch hier RSS ist.
(https://de.wikipedia.org/wiki/Residuenquadratsumme)

Lineare Regression

- $\sum_i (y_i - \langle y \rangle)^2 = \sigma_y^2$: sum of squared total (**SST**) $\rightarrow$ "gesamte" Variation
- $\sum_i (f(x_i) - \langle y \rangle)^2 = \sigma_{\hat{y}}^2$: sum of squares explained (**SSE**), *sum of squares due to regression (SSR), regression sum of squares (RSS)* $\rightarrow$ "vorhergesagte" Variation
- $\sum_i (f(x_i) - y_i)^2$: sum of squared residuals (**SSR**), *residual sum of squares (RSS), sum of squared errors (SSE)* $\rightarrow$ "restliche" Variation

$$\boxed{SST = SSE + SSR}$$

Güte der Anpassung: $R^2 = \frac{SSE}{SST} \rightarrow 1$ (Wieviel der Variationen wird durch das Model erklärt)
Güte der Anpassung: $\chi^2 \rightarrow 0$ (Wieviel der Variationen bleibt übrig)
Unsicherheit:
$MSE = SSR/(n-p) = SST(1-R^2)/(n-p)$, p - Anzahl Parameter
$\Delta b = \sqrt{MSE}/\sigma_x$
$\rightarrow$ `linreg.py`

Lineare Regression

NumPy:
`x.mean()`, `x.var()`, `numpy.cov(x,y)`
`bias` - Normierung mit N, statt N-1
→ `linreg-numpy.py`
SciPy:
`scipy.stats.linregress()`

- intercept: a
- slope: b
- rvalue: $\sqrt{R^2}$
- pvalue: Hypothesentest auf b=0
- stderr: σ_b
- intercept_stderr: σ_a (ab SciPy 1.6)

→ `linreg-scipy.py`

Regression mit Unsicherheiten

Berücksichtigung von Unsicherheiten:

$$y_i \pm \sigma_i$$

$$\chi^2 = \sum_i w_i (y_i - f(x_i, \boldsymbol{\alpha}))^2$$

Gewichtung: $w_i = 1/\sigma_i^2$
Ergebnis (lineare Regression):

$$\chi(a,b)^2 = \sum_i \left(\frac{y_i - a - b x_i}{\sigma_i} \right)^2$$

$$\begin{pmatrix} \langle y \rangle \\ \langle xy \rangle \end{pmatrix} = \begin{pmatrix} 1 & \langle x \rangle \\ \langle x \rangle & \langle x^2 \rangle \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

mit $\langle x \rangle = \sum_i w_i x_i$, etc. (vgl. Mittelwertbildung von diskreten Verteilungen)

Polynomregression, Polynomapproximation

- Erweiterung: **Anpassung eines Polynoms** an Daten:
  Buch Kap. 9.6.2
  numpy.polyfit
  polyreg-numpy.py
  polyreg-numpy-interpolation.py
- Anwendung Kleinste-Quadrate: (globale) **Näherung einer Funktion** durch ein Polynom:
  Buch Kap. 18.2.2
  polyfit.py
  polyfit-legendre.py

Nichtlineare Anpassung

Lösung des Optimierungsproblems für eine allg. Modellfunktion mithilfe von Minimierungsmethoden der Linearen Algebra
($\rightarrow$ Kapitel 18.2.3)

Idee:
Minimierung durch Anwendung des (mehrdimensionalen) Newton-Verfahrens auf den Gradienten der Modellfunktion. Bei Verwendung von $\chi^2(\boldsymbol{\alpha})$ ergibt sich das **Gauß-Newton-Verfahren**.
Um die Konvergenz zu garantieren, führt man einen Parameter λ ein und erhält endlich den **Levenberg-Marquardt-Algorithmus**.
($\rightarrow$ Kapitel 11.3)

Nichtlineare Anpassung

(Iteratives) Finden von Nullstellen mit dem Newton-Verfahren:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Mehrdimensionales Newton-Verfahren:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - J_f^{-1}(\mathbf{x}_n)\mathbf{f}(\mathbf{x}_n)$$

$J_f = (\partial_{x_j} f_i)$ - Jacobi-Matrix
bzw. löse LGS

$$J(\mathbf{x}_n)(\underbrace{\mathbf{x}_{n+1} - \mathbf{x}_n}_{\Delta\mathbf{x}_n}) = -\mathbf{f}(\mathbf{x}_n)$$

und berechne $\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_n$.

Nichtlineare Anpassung

Optimierungsproblem: Finde Nullstellen der Ableitung von $f(\mathbf{x})$

$$\mathbf{x}_{n+1} = \mathbf{x}_n - J^{-1}_{\nabla f}(\mathbf{x}_n)\boldsymbol{\nabla} f(\mathbf{x}_n)$$

mit $J_{\nabla f} = H_f = (\frac{\partial^2 f}{\partial_{x_i}\partial_{x_j}})$ - Hesse-Matrix

bzw.

$$H_f \Delta\mathbf{x}_n = -\nabla f(\mathbf{x}_n)$$

Wende **Newton-Verfahren** auf **Methode der kleinsten Quadrate** an:
Suche Minimum von $\chi^2(\boldsymbol{\alpha})$:

$$\mathbf{0} = \nabla_{\boldsymbol{\alpha}}\chi^2(\boldsymbol{\alpha}) = 2J^{\mathrm{T}}_{\mathbf{r}}(\boldsymbol{\alpha}) \cdot \mathbf{r}(\boldsymbol{\alpha})$$

In Newton-Verfahren:

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n - J^{-1}_{\nabla\chi^2}(\boldsymbol{\alpha}_n) \cdot \nabla\chi^2(\alpha_n)$$

$$\boxed{\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n - (J^{\mathrm{T}}_{\mathbf{r}} \cdot J_{\mathbf{r}})^{-1} \cdot J^{\mathrm{T}}_{\mathbf{r}}(\boldsymbol{\alpha}_n) \cdot \mathbf{r}(\boldsymbol{\alpha}_n)}$$

bzw.

$$(J^{\mathrm{T}}_{\mathbf{r}} \cdot J_{\mathbf{r}})\Delta\boldsymbol{\alpha}_n = -J^{\mathrm{T}}_{\mathbf{r}}(\boldsymbol{\alpha}_n) \cdot \mathbf{r}(\boldsymbol{\alpha}_n)$$

genannt **Gauss-Newton**

Nichtlineare Anpassung

Gauss-Newton Problem: Keine garantierte Konvergenz (vgl. Newton-Verfahren)

Lösung: erzwinge Abstieg durch Marquardt-Parameter λ_n:

$$(J_{\mathbf{r}}^{\mathrm{T}} \cdot J_{\mathbf{r}} + \lambda_n D)\Delta\boldsymbol{\alpha}_n = -J_{\mathbf{r}}^{\mathrm{T}}(\boldsymbol{\alpha}_n) \cdot \mathbf{r}(\boldsymbol{\alpha}_n)$$

$\lambda \to 0$: Gauss-Newton
$\lambda \to \infty$: $\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n - \gamma_n \nabla\chi^2(\boldsymbol{\alpha}_n)$ Gradientensuchverfahren

$D = \mathrm{diag}(J^{\mathrm{T}} \cdot J)$ (Levenberg):

$$\boxed{(J^{\mathrm{T}} \cdot J + \lambda_n \,\mathrm{diag}(J^{\mathrm{T}} \cdot J))\Delta\boldsymbol{\alpha}_n = -J^{\mathrm{T}}(\boldsymbol{\alpha}_n) \cdot \mathbf{r}(\boldsymbol{\alpha}_n)}$$

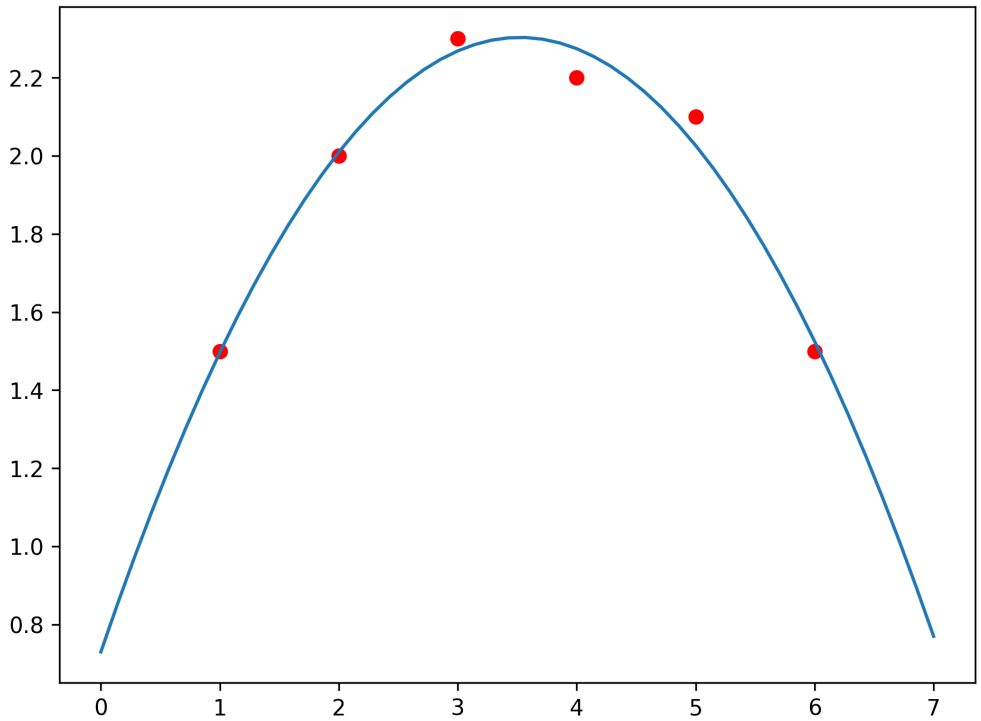
genannt **Levenberg-Marquardt-Algorithmus**

Nichtlineare Anpassung - Python

```
popt, pcov = scipy.optimize.curve_fit(f,x,y)
```

popt - opt. Parameter
pcov - Kovarianzmatrix der Parameter

→ `nonlin-scipy-curve_fit.py`



SciPy

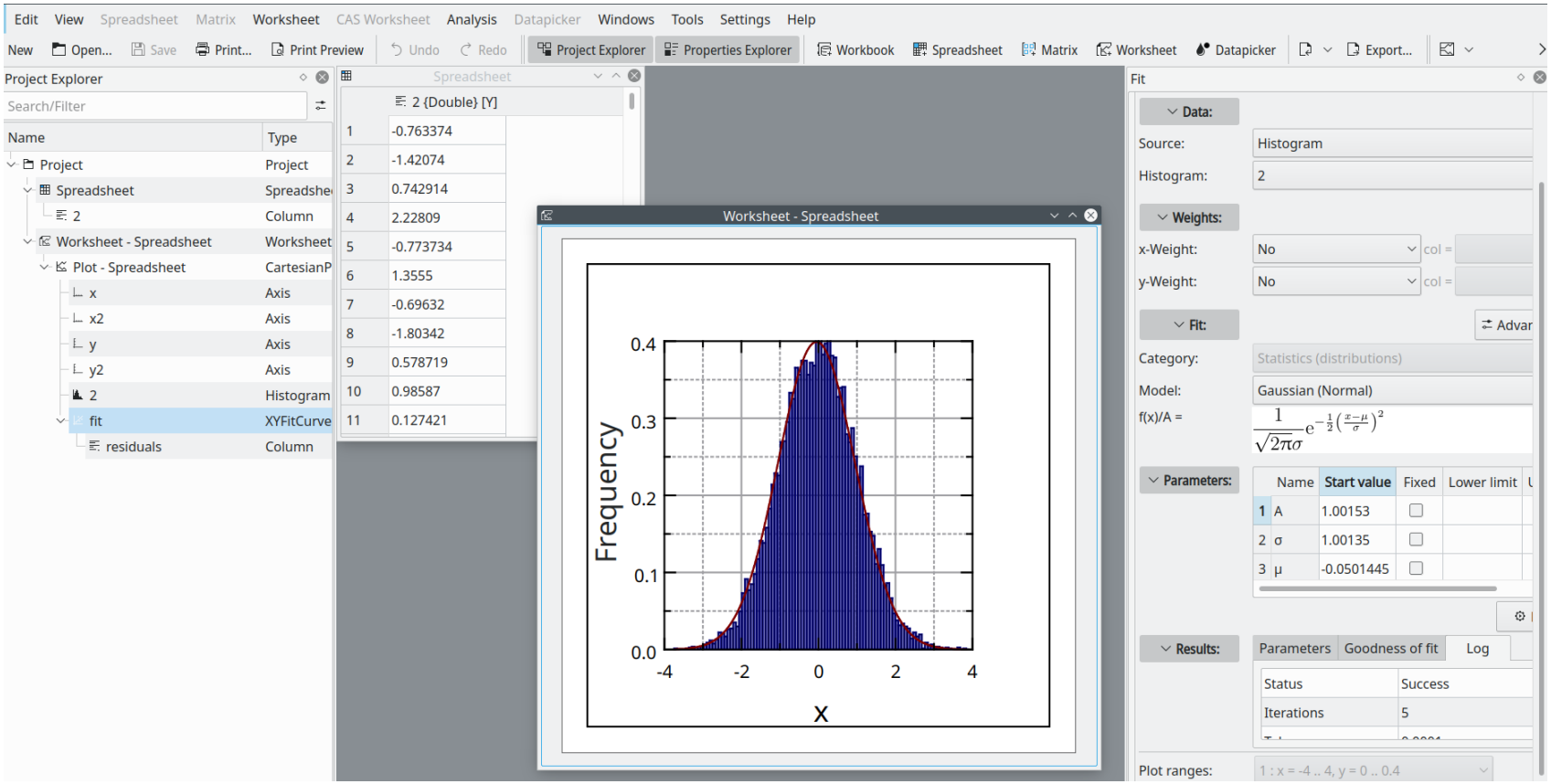
Nichtlineare Anpassung - Programme

Nichtlineare Anpassung von x-y-Daten mithilfe von Gnuplot:

```
1  gnuplot> f(x) = a + b*x
2  gnuplot> fit f(x) 'data.dat' using 1:2 via a, b
3  ...
4  iter chisq delta/lim lambda   a             b
5  0 3.43e+07  0.00e+00 4.08e+01 1.000000e+00  1.000000e+00
6  1 5.35e+03 -6.42e+08 4.08e+00 4.285327e-01 -6.638261e-03
7  2 5.01e+03 -6.87e+03 4.08e-01 6.004075e-02 -1.161502e-03
8  3 5.01e+03 -3.01e-01 4.08e-02 5.758351e-02 -1.124643e-03
9  ...
10 Final set of parameters      Asymptotic Standard Error
11 ==========================   ==========================
12 a               = 0.0575835    +/- 0.01416      (24.59%)
13 b               = -0.00112464  +/- 0.0002452    (21.8%)
14 ...
```

Nichtlineare Anpassung - Programme

GUI-Programme mit Fitfunktionen: LabPlot, Origin, Igorpro, ...



Maximum-Likelihood-Methode

Suche **Parameter** α einer vermuteten/bekannten **Verteilung** $f(y;\alpha)$ anhand einer Stichprobe y_i (n unabhängige Messungen).
Maximum-Likelihood: Die besten Parameter ergeben die **höchste Wahrscheinlichkeit** die Stichprobe zu erhalten.

Likelihood-Funktion: Datenpunkte unabhängig, d.h. Wahrscheinlichkeiten multiplizieren sich:

$$P = f(y_1, y_2, ..., y_n; \boldsymbol{\alpha}) = f(y_1; \boldsymbol{\alpha}) \cdots f(y_n; \boldsymbol{\alpha}) = \prod_{i=1}^{n} f(y_i; \boldsymbol{\alpha}) =: L(\boldsymbol{\alpha})$$

Lösung:

$$\left.\frac{\partial L(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}\right|_{\hat{\boldsymbol{\alpha}}} = \mathbf{0} \quad \text{bzw.} \quad \left.\frac{\partial \ln \mathbf{L}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}\right|_{\hat{\boldsymbol{\alpha}}} = \mathbf{0}$$

(**Log-Likelihood-Funktion** meist einfacher zu berechnen, da Logarithmus von Produkt = Summe)

Maximum-Likelihood-Methode - Beispiel (Fische im Bodensee)

Stichprobe: Einen Tag angeln. Ergebnis: 14 Felchen, 6 Saiblinge.
Likelyhood-Funktion (Binomialverteilung):

$$L(p) = B_{n,k}(p) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$\ln L(p) = ln \binom{n}{k} + k \ln p + (n-k) \ln(1-p)$$

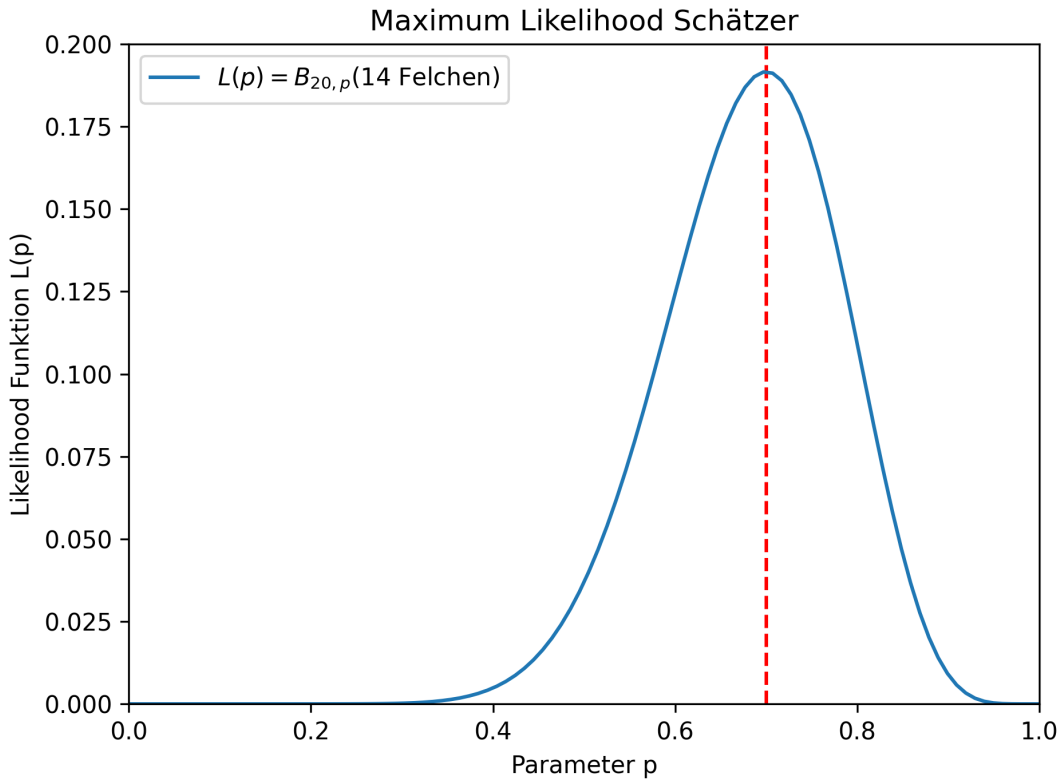
$$\frac{\mathrm{d}}{\mathrm{dp}} \ln L(p)|_{\hat{p}} = \frac{x}{\hat{p}} - \frac{n-k}{1-\hat{p}} \stackrel{!}{=} 0$$

$$\rightarrow \boxed{\hat{p} = \frac{k}{n}}$$

Maximum-Likelihood-Methode - Beispiel (Fische im Bodensee)

Stichprobe: Einen Tag angeln. Ergebnis: 14 Felchen, 6 Saiblinge.

$$\hat{p} = \frac{k}{n} = 14/21 = 70\,\%$$



Maximum-Likelihood-Methode - Poissonverteilung

$$f(k;\lambda) = \frac{\lambda^k}{k!}\mathrm{e}^{-\lambda}$$

Stichprobe $k_1, .., k_n$. Likelihood-Funktion:

$$L(\lambda) = \frac{\lambda^{\sum_i k_i}}{\prod_i k_i!}\mathrm{e}^{-n\lambda}$$

$$\ln L(\lambda) = \sum_i (k_i \ln \lambda - \ln(k_i!) - \lambda)$$

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} \ln L(\lambda)|_{\hat{\lambda}} \stackrel{!}{=} 0$$

$$\boxed{\hat{\lambda} = \frac{1}{n}\sum_i k_i = \langle k \rangle}$$

Maximum-Likelihood-Methode - Exponentialverteilung

$$f(x;\lambda) = \lambda \mathrm{e}^{-\lambda x} \quad (\lambda > 0)$$

Stichprobe $x_1, ..., x_n$. Likelihood-Funktion:

$$L(\lambda) = \lambda^n \mathrm{e}^{-\lambda \sum_i x_i}$$

$$\ln L(\lambda) = n \ln \lambda - \lambda \sum_i x_i$$

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} \ln L(\lambda)|_{\hat{\lambda}} \stackrel{!}{=} 0$$

$$\boxed{\hat{\lambda} = \frac{1}{\frac{1}{n}\sum_i x_i} = \frac{1}{\langle x \rangle}}$$

Maximum-Likelihood-Methode - Normalverteilung

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

Stichprobe $x_1, ..., x_n$. Likelihood-Funktion:

$$L(\mu,\sigma^2) = \frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}\sum_i (x_i-\mu)^2}$$

$$\ln L(\mu,\sigma^2) = -\frac{n}{2}(\ln(2\pi) - \ln\sigma^2) - \frac{1}{2\sigma^2}\sum_i (x_i-\mu)^2$$

$$\frac{\partial}{\partial\mu}\ln L(\mu,\sigma^2)|_{\hat{\mu}} = -\frac{1}{\sigma^2}\sum_i (x_i-\hat{\mu}) \stackrel{!}{=} 0$$

$$\boxed{\hat{\mu} = \frac{1}{n}\sum_i x_i = \langle x\rangle}.$$

analog ($\frac{\partial}{\partial\sigma^2}$):

$$\hat{\sigma}^2 = \frac{1}{n}\sum_i (x_i-\mu)^2.$$