4 Schätzung

Markus Lippitz
10. Mai 2022

Ziele: Sie können die Parameter eines Modells *schätzen*, das einen Datensatz beschreibt.

- Punktschätzung: chi-quadrat, maximum likelihood
- Intervallschätzung
- Regression

Weitere Aufgaben:

- Sie detektieren in einem Intervall von 1 Sekunde 10 Photonen. Wie hell ist die Lichtquelle mit 95% Wahrscheinlichkeit?
- Erzeugen sie eine Summe von 2 P-Verteilungen und schätzen / bestimmen die Parameter durch verschiedene Methoden. Vergleichen sie die ‚wahren' Parameter mit den geschätzten Parametern und deren geschätzten Fehlern.

Literatur: Stahel Kap. 7, 9, 13, Bevington Kap. 4, 6–8, Meyer Kap. 29,30,32,33

Überblick

'Schätzen' nennt man es in der Statistik, wenn man aufgrund von gemessenen Daten in einer Stichprobe eine Aussage über die Wirklichkeit macht. In der Physik würde man das 'die Parameter eines Modells bestimmen' nennen. Schon in der Gauss'schen Fehlerfortpflanzung hatten Sie gesehen, dass oft ein Intervall von Werten die Daten ähnlich gut beschreibt. Die Bestimmung dieses Intervalls nennt man 'Intervall-Schätzung'. Schließlich will man manchmal die Frage beantworten, ob eine gewissen Menge an Messwerten vereinbar ist mit einem gewissen Parameter. Ist der Würfel gezinkt, wenn 5 mal hintereinander die 6 gewürfelt wird? Diese Frage beantworten statistische Tests.

Punktschätzung

Wir haben also eine Stichprobe, können mit diesen Daten alles mögliche anstellen, und möchten auf Eigenschaften der Ausgangs-Verteilung schließen. Wir könnten beispielsweise das arithmetische Mittel aller Werte der Stichprobe berechnen und dies als Schätzwert für den Parameter μ der Normalverteilung nehmen. Man sagt, das arithmetische Mittel $\bar{X}$ ist ein 'Schätzer' für μ. Schätzer von Variablen bezeichnet man mit einem Hut über der Variablen, also hier $\hat{\mu}$. Diese sind natürlich selbst wiederum Zufallsvariablen und gehorchen einer Verteilung.

Bias

Wann ist ein Schätzer gut? Wie kann man verschiedene Schätzer für die gleiche Variable vergleichen? Wir möchten erreichen, dass de Schätzung T möglichst nahe am wirklichen Wert θ liegt, und die Abweichungen dazwischen rein zufällig sind. Ein Maß für die Abweichung ist der **mittlere quadratische Fehler** (mean square error, mse)

$$\mathcal{E}\left\langle (T-\theta)^2 \right\rangle$$

Das kann man umschreiben als

$$\mathcal{E}\left\langle ((T-\mathcal{E}\left\langle T\right\rangle) + (\mathcal{E}\left\langle T\right\rangle - \theta))^2 \right\rangle = \mathcal{E}\left\langle (u+v)^2 \right\rangle$$

Der Term mit u^2 ist die Varianz von T, der Term mit $2uv$ verschwindet, weil $\mathcal{E}\left\langle T-\mathcal{E}\left\langle T\right\rangle\right\rangle = 0$, und der Term mit v^2 beleibt, so dass

$$\mathcal{E}\left\langle (T-\theta)^2 \right\rangle = \mathbf{var}\left\langle T\right\rangle + (\mathcal{E}\left\langle T\right\rangle - \theta)^2$$

Hier ist der zweite Term der *systematische Fehler* oder bias des Schätzers. Selbst durch ausgiebiges Messen wird T niemals θ erreichen, wenn dieser Term nicht Null ist. Ein Schätzer heist 'erwartungstreu', 'biasfrei' oder 'unbiased', wenn dieser Term verschwindet.

Bsp. Halbwertszeit λ einer Exponentialverteilung

Als Beispiel betrachten wir die Exponentialverteilung

$$f(x) = \lambda e^{-\lambda x}$$

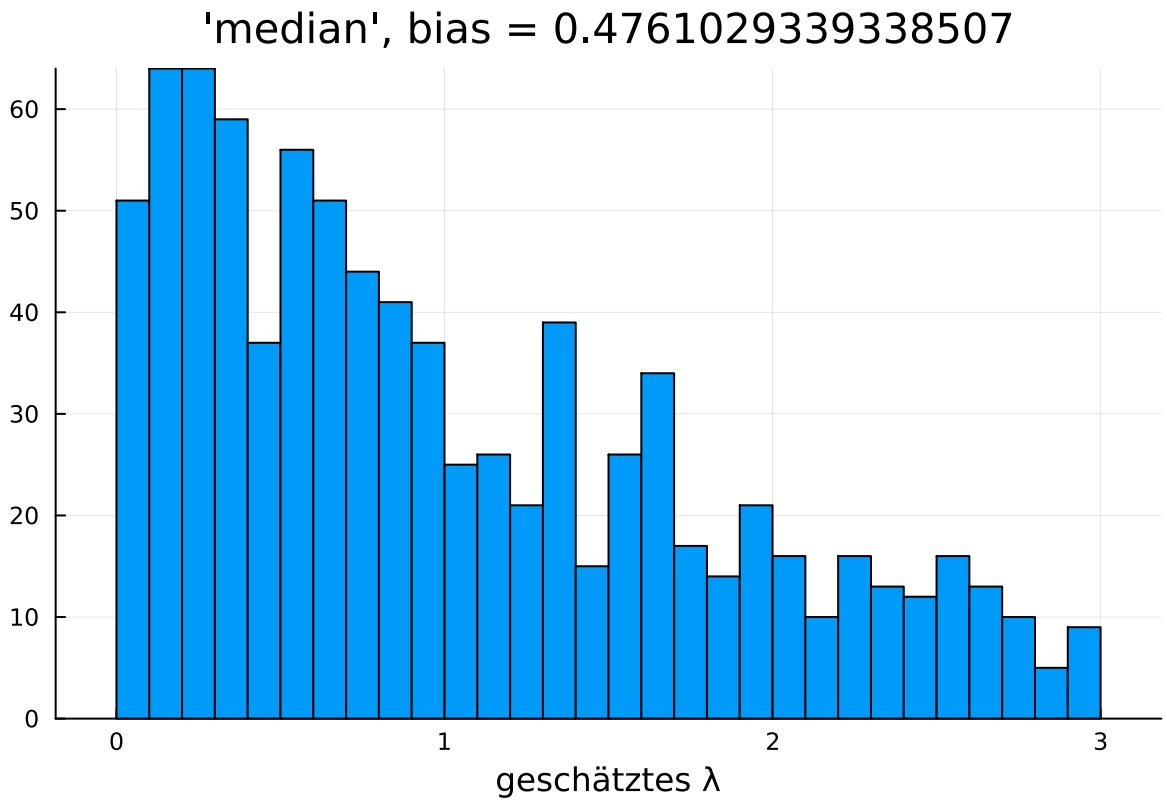
Wir ziehen n Zufallszahlen aus dieser Verteilung und wollen daraus den Parameter λ bestimmen. Die Halbwertszeit τ der Verteilung, also $f(\tau) = \lambda/2$ ist erreicht bei $x = \ln 2/\lambda$, daher der Name der Parameters.

Wir vergleichen als Schätzer die Mittelwertbildung und die Bildung des Medians. Letzterer muss noch durch $\ln 2$ geteilt werden. Für die 'wahre' Verteilung gilt $\lambda = 1$.

1

```
@bind n_mittel Slider(1:10; show_value=true)
```

- (●) median
- ( ) mean




```
let
    N_Z = 1000    # Anzahl Punkte in Z-Verteilung
    x = range(0,3; step=0.1)

    zufall = rand(Exponential(1),  N_Z, n_mittel) # exponentialverteilt, λ=1
    if (schaetzer == "median")
        lambda_hat = median(zufall; dims=2) / log(2) # median along n_mittel
    else
        lambda_hat = mean(zufall; dims=2) # mean along n_mittel
    end
    bias = mean(lambda_hat) - 1 ;

    h1 = StatsBase.fit(Histogram, vec(lambda_hat), x)  # histogram
    plot( h1, legend=false,
        title="'$(schaetzer)', bias = $(bias)",
        xlabel="geschätztes λ")

end
```

Zunächst einmal sieht man, dass bei kleinem n beide Schätzer deutlich streuen. Beide Schätzer sind für große n erwartungstreu, bias free. Für kleines n besitzt aber der Median einen deutlichen bias.

Schätzung der Varianz

Jetzt kommen wir endlich zu dem komischen Faktor $\frac{1}{n-1}$ bei der Schätzung der Varianz.

Wir hatten schon immer definiert, dass

$$\hat{\mathbf{var}} = \hat{\sigma^2} = \frac{1}{n-1}\left(\sum_i X_i^2 - n\bar{X}^2\right)$$

Der Erwartungswert davon ist

$$\mathcal{E}\left\langle\hat{\sigma^2}\right\rangle = \frac{1}{n-1}\left(\sum_i \mathcal{E}\left\langle X_i^2\right\rangle - n\mathcal{E}\left\langle\bar{X}^2\right\rangle\right)$$

Wir nehmen an, dass $\mathcal{E}\left\langle X_i\right\rangle = 0$. Falls das nicht der Fall sein sollte, könnten wir immer die X_i verschieben, ohne die Varianz zu ändern. Damit ist $\mathcal{E}\left\langle X_i^2\right\rangle = \sigma^2$ und $\mathcal{E}\left\langle\bar{X}^2\right\rangle = \sigma^2/n$. Insgesamt bekommen wir

$$\mathcal{E}\left\langle\hat{\sigma^2}\right\rangle = \frac{1}{n-1}\left(\sum_i \mathcal{E}\left\langle X_i^2\right\rangle - n\mathcal{E}\left\langle\bar{X}^2\right\rangle\right) = \frac{1}{n-1}\left(n\sigma^2 - n\frac{1}{n}\sigma^2\right) = \sigma^2$$

Die mit dem Faktor $\frac{1}{n-1}$ definierte (empirische) Varianz $\hat{\sigma}$ ist also ein erwartungstreuer Schätzer der Varianz σ einer Verteilung. Der Vor-Faktor ist notwendig, um bias free zu sein. Dies ist unabhängig von der Verteilung, aus der die X_i gezogen sind. Über die haben wir hier keine Annahme gemacht.

Im nächsten Kapitel werden wir das aus dem Gesichtspunkt der Freiheitsgrade noch einmal betrachten. Bei n Messwerten haben wir n Freiheitsgarde. Einen davon verbrauchen wir zur Bestimmung des Mittelwerts μ. Damit sind nur noch $n-1$ übrig, die in die Varianz eingehen.

Maximum Likelihood Methode

Schätzer liefern als einen Schätzwert für den Parameter einer Wahrscheinlichkeitsverteilung auf Basis von gegebenen Messwerten. Aber wie kommt man an einen Schätzer, wenn es nicht gerade der Mittelwert ist? Dieses Problem löst die Maximum Likelihood Methode (ungefähr 'Methode der größten Plausibilität'). Wie wir weiter unten sehen werden ist sie die Basis für die Bestimmung von Modell-Parameter über 'kleinste Quadrate', aber vielfältiger bzw. mit weniger Voraussetzungen einsetzbar.

Wir folgen hier dem **Beispiel** von Stahel, Kapitel 7.4. Wir führen ein Experiment durch, werfen eine Münze o.ä. Dies gelingt mit einer Wahrscheinlichkeit p. Wir führen das so oft durch, bis es einmal misslingt. Die Wahrscheinlichkeit, dass es nach dem Versuch x misslingt ist geometrisch verteilt, also

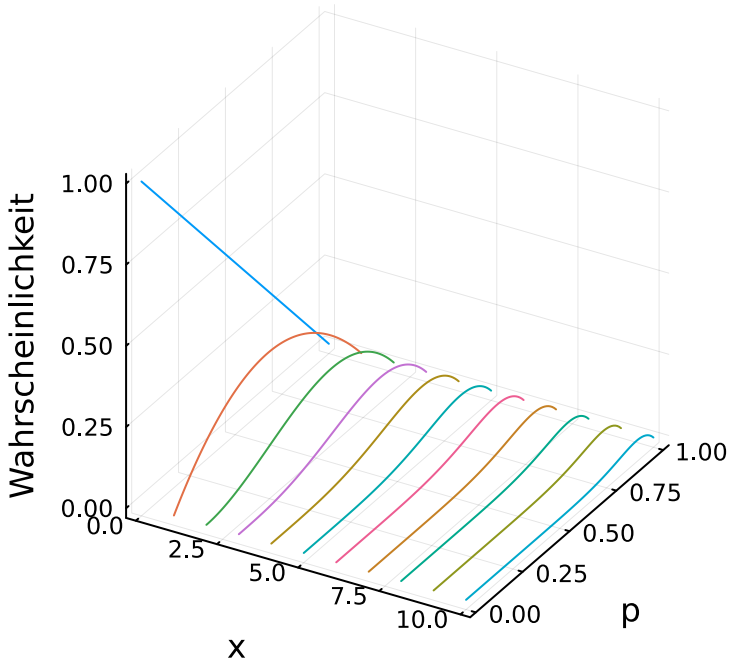
$$P\left\langle X = x \right\rangle = (1-p)\,p^x$$

Wenn wir feststellen, dass der Versuch nach $x = 3$ geglückten Durchgängen zum erstem mal misslungen ist, was können wir dann über p sagen?

Die Idee ist, die Rolle von Parameter p und Variable x zu vertauschen. Wir betrachten also x als Parameter, schließlich kennen wir $x =$, und p als freie Variable. In der folgenden Grafik folgen wir also einer Zeile konstanten x. Der plausibelste Wert von p ist der, der die Wahrscheinlichkeit P maximiert, bei gegebenen x. Das kann man in diesem Fall analytisch tun,

$$\hat{p} = \frac{x}{x+1}$$

ist aber auch sonst immer numerisch möglich.




```
let
    p = range(0, 1; length=100)
    xs = range(0, 10)

    plot3d()
    for x in xs
        wk = @. ((1 - p) * p^x)
        plot3d!(x .+ zeros(size(p)), p, wk)
    end
    plot3d!(xlabel="x", ylabel="p", zlabel="Wahrscheinlichkeit",legend=false)
end
```

Das geht natürlich nicht nur mit einem Messwert x, sondern auch wenn mehrere x_i gemessen wurden. Dann müssen wir die Gesamt-Plausibilität als Funktion des Parameters θ maximieren, also

$$P_\theta \langle X_1 = x_1, X_2 = x_2, \dots X_n = x_n \rangle = \prod p_\theta \langle x_i \rangle$$

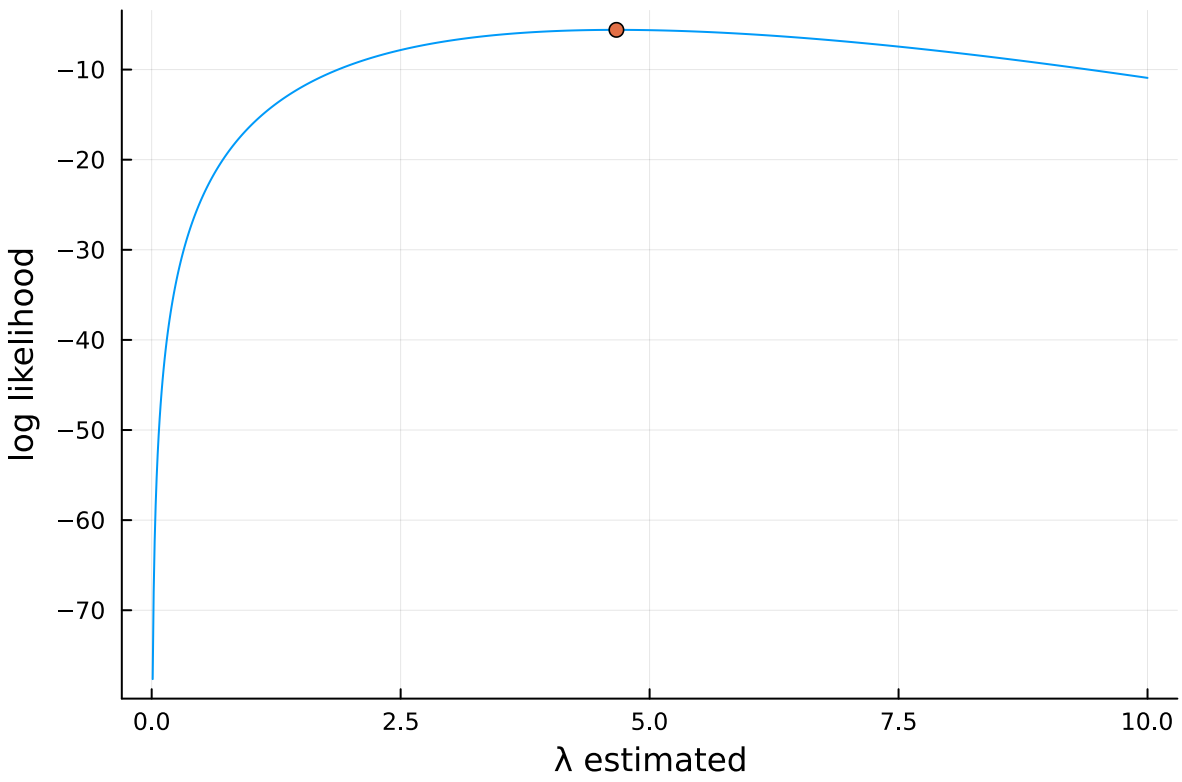
Das wird einfacher, wenn man auf beiden Seiten den Logarithmus zieht. So erhält man die Log-Likelihood-Funktion L

$$L \langle x_1, x_2 \dots x_n; \theta \rangle = \sum \ln p_\theta \langle x_i \rangle$$

Wir maximieren also $L \langle x_1, x_2 \dots x_n; \theta \rangle$ durch Variation des Parameter(-Satzes) θ. Das gefundene θ ist dann der plausibelste Parameter, der die gemessenen x_i beschreibt.

Bsp: Photonen zählen

Betrachten wir als Beispiel einen sehr schwachen Laserstrahl. Wir messen in n aufeinander folgenden Zeit-Intervallen die Zahl der Photonen x_i und wollen daraus den Mittelwert λ der zugrundeliegenden Poisson-Verteilung bestimmen.




```
begin
    n = 3
    λ_true = 5
    data = rand(Poisson(λ_true), n)

    # calcluate pdf  at given λ for each element of 'data' and sum log values
    L(λ_est) = sum(log.(pdf.(Poisson(λ_est), data)))

    λ_est_range = range(0,2 * λ_true; length=1000)
    plot(λ_est_range, @. L(λ_est_range); legend=false,
        xlabel="λ estimated", ylabel="log likelihood")

    res = Optim.optimize(x -> -1 .* L(x), 0, 1e3) # minimize -L
    scatter!([res.minimizer], [-1 .* res.minimum]) # mark maximum
end
```

Natürlich ist in diesem Fall der Mittelwert auch ein guter Schätzer.

▸ (4.66667, 4.66667)

```
mean(data), res.minimizer
```

Lineare Regresssion bei Poisson-Verteilung

Eine Stärke der Maximum-Likelihood Methode ist, dass auch andere Verteilungen als die Normalverteilung verwendet werden können, beispielsweise die Poisson-Verteilung. Unser Modell sei weiterhin $y_0 = a + b\,x$, aber die Wahrscheinlichkeit, einen Wert y_i zu finden folge einer Poisson-Verteilung mit $\lambda = y_0 = a + b\,x$, also

$$P_i = \frac{\lambda^{y_i}}{y_i!}\exp(-\lambda) = \frac{(a+bx_i)^{y_i}}{y_i!}\exp(-(a+bx_i)) \quad .$$

Damit wird

$$\mathcal{L}(a,b) = \sum (y_i \log(a+bx_i)) - \sum (a+bx_i) + \mathrm{const.}$$

Diese Funktion wird maximal, wenn

$$\begin{aligned} N = &\quad \sum \frac{y_i}{a+bx_i} \\ \sum x_i = &\quad \frac{x_i\,y_i}{a+bx_i} \end{aligned}$$

gleichzeitig erfüllt sind. Eine Lösung für a, b findet sich nur numerisch.

Methode der kleinsten Quadrate

Wir hatten schon im Kapitel zur beschreibenden Statistik die Methode der kleinsten Quadrate erwähnt. Man kann ein Modell an Daten anpassen, indem man die Parameter des Modells so variiert, dass die Summe der quadratischen Abweichung zwischen Modell und Daten minimal wird. Bei gegebenen x_i und gemessenen y_i sucht man also Parameter θ eines Modells $y = f(x_i;\theta)$ so dass

$$\mathrm{Summe\ der\ Residuen} = \sum_i \left(y_i - f(x_i;\theta)\right)^2$$

minimal wird.

Warum funktioniert das? Die Begründung liefer die Maximum Likelihood Methode. Aus dieser Sichtweise beschreibt das Modell $f(x_i;\theta)$ den Erwartungswert einer Normalverteilung mit der Standardabweichung σ, die für alle Messpunkte identisch ist. Die Wahrscheinlichkeit, einen Wert y_i zu messen, obwohl das Modell doch $f(x_i;\theta)$ vorhersagt, ist also

$$p_\theta(x_i) = c\,e^{-\frac{(y_i - f(x_i;\theta))^2}{\sigma^2}}$$

Der Vorfaktor c normiert die Verteilung, ist aber für alle i identisch, so dass wir ihn bei der Optimierung vernachlässigen können. Die Log-Likelihood Funktion ist dann

$$L\left\langle x_1, x_2 \ldots x_n; \theta \right\rangle = \sum \ln p_\theta \left\langle x_i \right\rangle = -\frac{c}{\sigma^2} \sum_i (y_i - f(x_i;\theta))^2$$

Wir maximieren L, minimieren also die Summe der Residuen. Die Methode der kleinsten Quadrate ist also die Maximum Likelihood Methode unter der Annahme einer Normalverteilung der (konstanten) Messunsicherheit.

Die erste Konsequenz ist, dass wir eine variable Messunsicherheit σ_i berücksichtigen können, in dem wir

$$\textbf{Summe der gewichteten Residuen} = \chi^2 = \sum_i \frac{(y_i - f(x_i;\theta))^2}{\sigma_i^2}$$

minimieren.

Die zweite Konsequenz ist, dass wir die Verteilung der Messunsicherheit im Auge behalten müssen. Sollte es begründete Zweifel an deren Normalverteilung geben, dann sollte man von der Methode der kleinsten Quadrate Abstand nehmen und direkt die Maximum Likelihood Methode anwenden. Dies ist beispielsweise der Fall, wenn ein Modell an Photonenzahlen angepasst werden soll, und die typischen Photonenzahlen unterhalb von ungefähr 10 liegen. Deren Fehler ist dann gerade Poisson-verteilt.

Bsp. Modell an Photonen-Zahlen anpassen

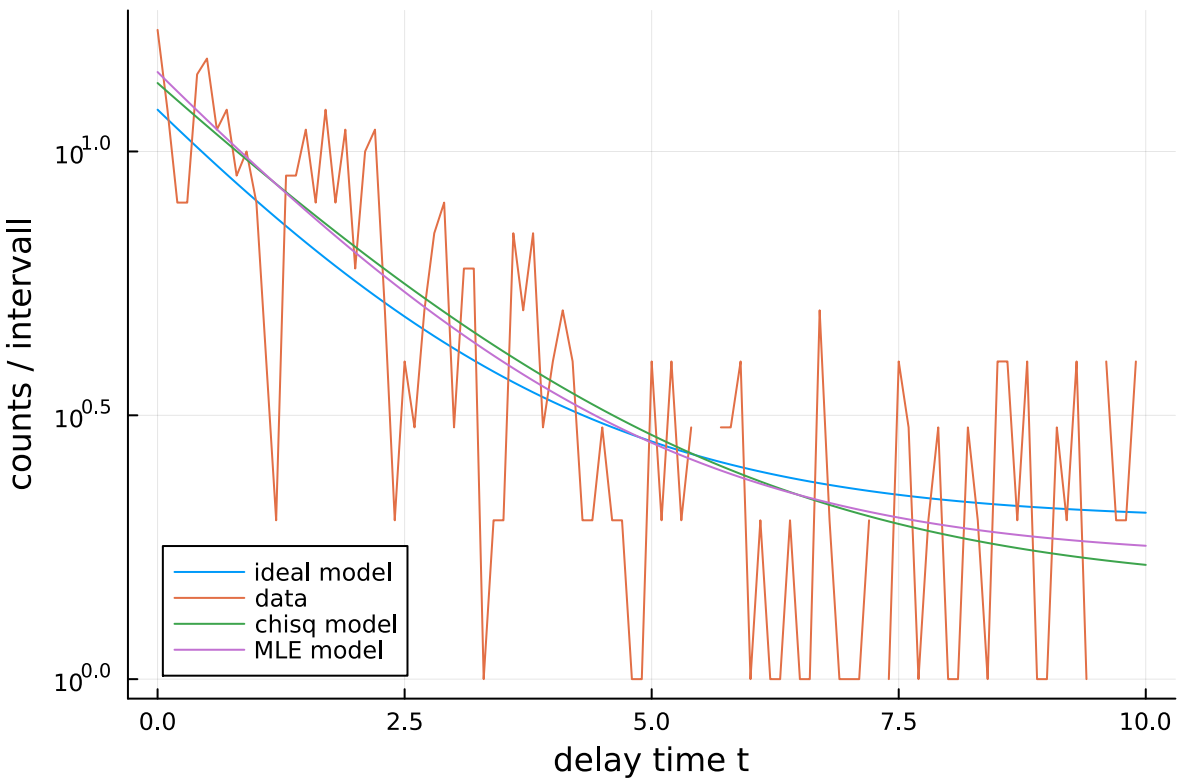
Beim zeitkorrelierten Einzelphotonenzählen bestimmt man die Anzahl an detektierten Fluoreszenz-Photonen eines Moleküls als Funktion des zeitlichen Abstands zum anregenden Laserpuls. Typische Abstände liegen im Bereich von einer Nanosekunde und sind durch die Lebenszeit des angeregten Zustands im Molekül bestimmt. Die Idee ist, diesen dadurch zu bestimmen. Man findet einen exponentiellen Abfall der Zählereignisse mit dem zeitlichen Abstand zur Anregung. Die Zerfallskonstante ist gerade die Lebenszeit des Zustands.

Das Modell ist also

$$f(t; \tau, c_{fl}, c_{bg}) = c_{fl}\, e^{-t/\tau} + c_{bg}$$

wobei t die Zeit zwischen Laserpuls und Detektion des Photons ist und τ die Lebenszeit des angeregten Zustands beschreibt. Die beiden c_i bestimmen den Helligkeit des Moleküls und die des Untergrunds, beispielsweise aufgrund von Raumlicht oder der Dunkelzählrate des Detektors.

Die gemessene Photonenzahl in einem Intervall $[t, t + dt]$ ist also Poisson-Verteilt um einen Mittelwert, der durch f gegeben ist.




```julia
begin
    t = range(0, 10; step= 0.1)

    f(t, τ, c_fl, c_bg) = c_fl * exp(- t / τ) + c_bg
    f(p::Vector) = f.(t, p[1], p[2], p[3])

    p_model = [2, 10, 2]
    data_p = rand.(Poisson.(f(p_model)))

    lower = zeros(size(p_model))
    upper = 1 ./ lower # = inf
    initial = ones(size(p_model))

    L(p::Vector) = -1 .* sum(log.(pdf.(Poisson.(f(p)),data_p)))
    res_MLE = optimize(L,  lower, upper, initial, Fminbox(NelderMead()))

    chisq(p::Vector) = sum((data_p .- f(p)).^2 )
    res_quadrate = optimize(chisq,  lower, upper, initial, Fminbox(NelderMead()))

    plot(t, f(p_model), yaxis=:log10, label="ideal model")
    plot!(t, map(x -> iszero(x) ? NaN : x, data_p), # clip zero values for log plot
        yaxis=:log10, xlabel="delay time t", ylabel="counts / intervall",
        label="data")
    plot!(t, f(res_quadrate.minimizer), yaxis=:log10, label="chisq model")
    plot!(t, f(res_MLE.minimizer), yaxis=:log10, label="MLE model")
end
```

```
[2.34646, 12.0024, 1.47734]
```

```
1 res_quadrate.minimizer
```

```
[2.0727, 12.4519, 1.68966]
```

```
1 res_MLE.minimizer
```

Da hatte ich den Unterschied dramatischer in Erinnerung

Intervallschätzung

Mit der Punktschätzung haben wir den plausibelsten Wert des Parameters gefunden, der die Verteilung beschreibt, aus der unsere Messdaten gezogen wurden. Nun geht es um die Frage, welche anderen Werte dieser Parameter auch noch annehmen könnte. Wir sind also auf der Suche nach einem Intervall, in dem der wirkliche Parameter dann mit einer gewissen Wahrscheinlichkeit liegen wird. Dieses Intervall nennt man **Konfidenzintervall**.

Bsp. geometrische Verteilung

Betrachten wir noch einmal als Beispiel die geometrische Verteilung von oben. Mit der Wahrscheinlichkeit p gelingt ein Versuch, nach x Versuchen tritt zum ersten mal ein Misserfolg auf. Die Wahrscheinlichkeit für einen Misserfolg nach x Versuchen ist also

$$P\left\langle X = x \right\rangle = (1-p)\,p^x$$

Wir messen einen Misserfolg nach $x = 3$ versuchen. Wie oben gesehen ist der plausibelste Parameter

$$\hat{p} = \frac{x}{x+1} = \frac{3}{4}$$

Wir suchen eine untere (p_u) und obere (p_o) Intervallgrenze, die jeweils eine Funktion des gemessenen Wertes x ist, so dass der wahre Parameter p mit 95% Wahrscheinlichkeit in diesem Intervall liegt, also

$$\mathcal{P}\left\langle p_u(x) < p < p_o(x) \right\rangle = 0.95$$

unabhängig vom wahren p und ggf. auch anderen Parametern der Verteilung. Das Gleichheitszeichen wird manchmal als 'gleich oder etwas größer' interpretiert. Insbesondere bei diskreten Verteilungen ist ein Gleichheit nicht zu erreichen.

Man kann die Intervallgrenze auf verschiedenem Weg bekommen, siehe bspw. wikipedia. Hier folgen wir wiederum Stahel und benutzen ein Ergebnis aus dem Test von Hypothesen. Ohne das hier näher zu begründen drehen wir wieder Variable und Parameter um. Wir wählen die obere Grenze des Intervalls so, dass unser Messwert x auf der *unteren* 2.5%-Perzentil liegt, also

$$P_{p_o}\left\langle X \leq x \right\rangle = 2.5\%$$

Die kumulative Dichtefunktion beträgt also 0.025 bzw $1 - 0.025$ an der unteren bzw oberen Intervallgrenze. In Julia finden wir diese Grenz-Parameter über eine Nullstellensuche:

```
function grenzen(x)
    # in Julia, p is defined as (1-p) compared to our eq. above ...
    # seach for p, so that cdf crosses 2.5%
    r = optimize(p -> (cdf(Geometric(1-p), x) -  0.025).^2, 0, 1)
    oben = r.minimizer
    if (x >= 1)
        r = optimize(p -> (cdf(Geometric(1-p), x-1) -  (1- 0.025)).^2, 0, 1)
        unten = r.minimizer
    else
        unten = 0
    end
    return unten, oben
end;
```

```
(0.292402, 0.993691)
```

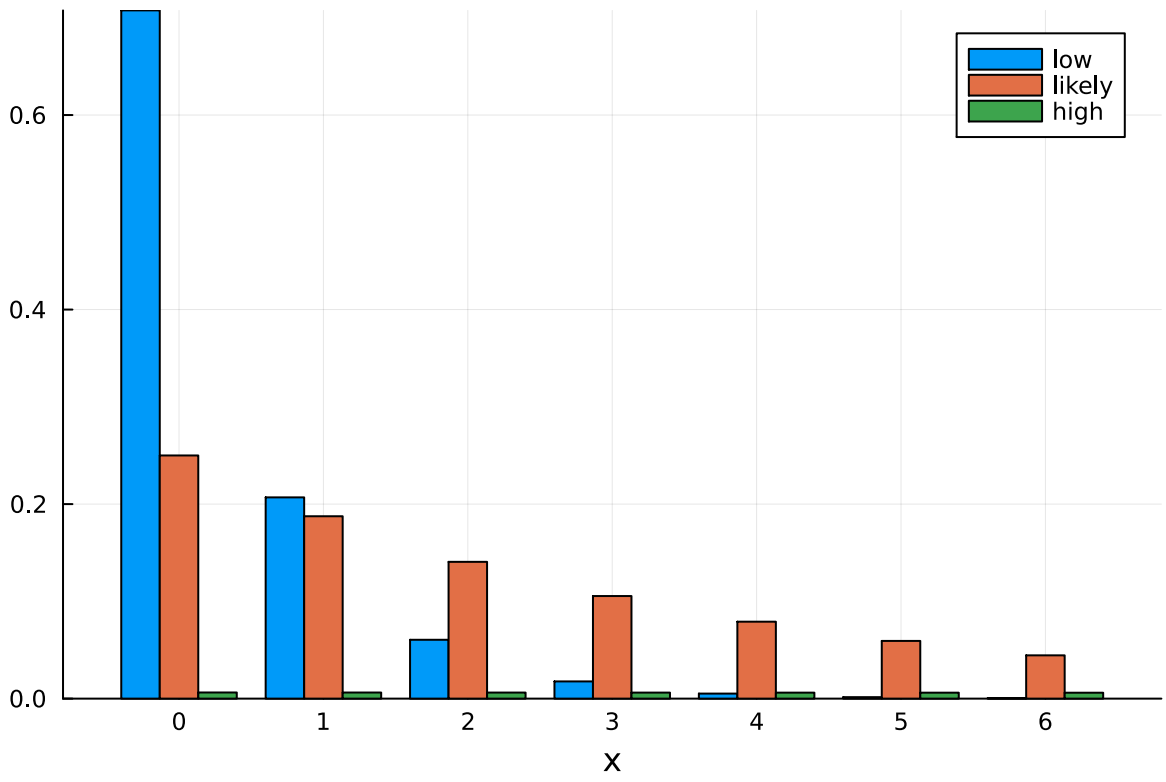
```
grenzen(3)
```

Testweise summieren wir die Wahrscheinlichkeiten für die Fälle $\mathbf{0-3}$ bzw. $\mathbf{3-\infty}$ erfolgreiche Versuche bei dem gegebenen p_u und p_o, um sicherzugehen, dass jeweils 2.5% außerhalb liegen.

```
(0.025, 0.025)
```

```
let
    xm = 3
    (plow, phigh) = grenzen(xm)
    wk_below = sum([pdf.(Geometric(1-plow) ,x) for x = (xm:20)])
    wk_high =  sum([pdf.(Geometric(1-phigh),x)  for x = (0:xm)])
    wk_below, wk_high
end
```

Dies sind die drei charakteristischen Verteilungen, wenn wir $x = 3$ gemessen haben. Die Plausibelste ist die mit $p = 3/4$, die anderen beiden sind die Grenzfälle.

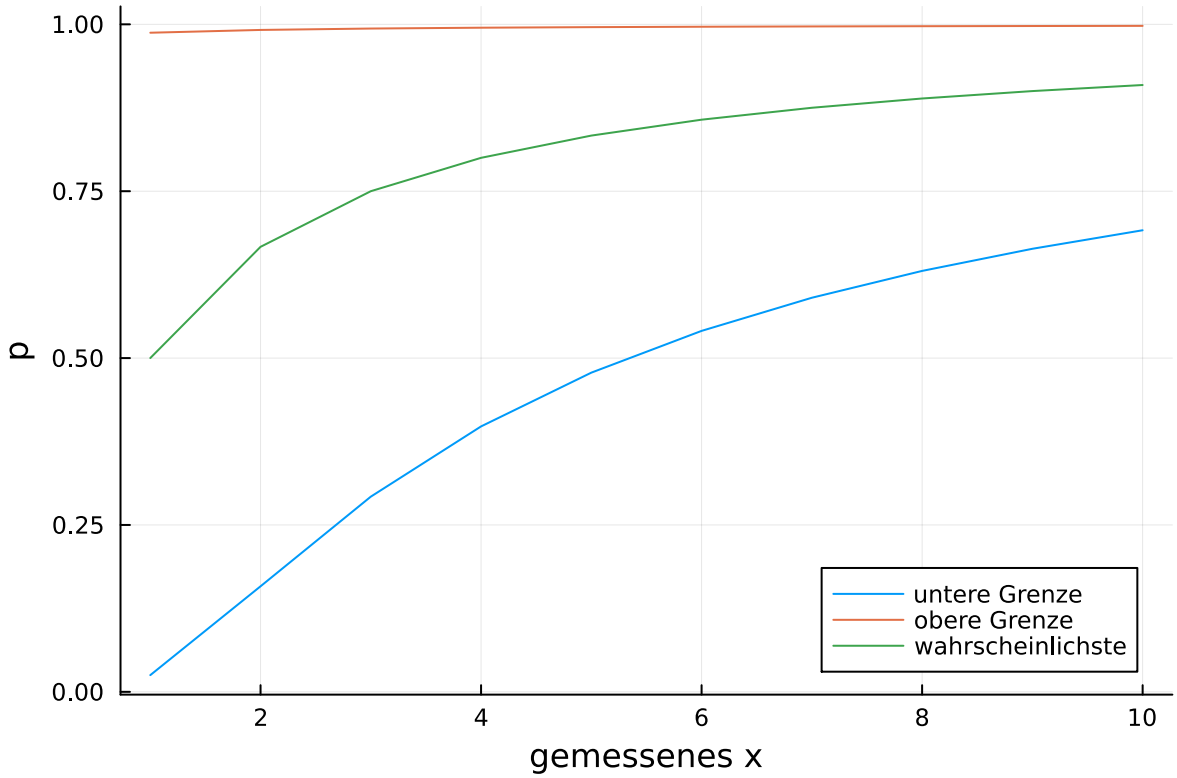



```
let
    xm = 3
    (plow, phigh) = grenzen(xm)
    plikely = xm ./ (xm +1)

    xr = 6;
    xs = range(0, xr)

    groupedbar( [pdf.(Geometric(1-p),x) for x in xs, p in [plow, plikely, phigh]]
    , xticks=(1:xr+1, string.(0:xr)), xlabel="x", label = ["low" "likely"
    "high"])
end
```

Als Funktion des gemessenen x sieht das Intervall so aus




```
let
    xs = range(1,10)
    plot([grenzen(x)[1] for x in xs], label="untere Grenze")
    plot!([grenzen(x)[2] for x in xs], label="obere Grenze")
    plot!([x / (x+1) for x in xs], label="wahrscheinlichste", xlabel="gemessenes
    x", ylabel="p",legend=:bottomright)
end
```

Test der Intervallgrenzen

Lassen sie uns das noch einmal andersrum betrachten. Wir geben uns eine geometrische Verteilung mit bekanntem p vor, ziehen daraus eine Zufallszahl x und schätzen aufgrund dieser Zahl die Grenzen des Konfidenzintervalls. In 95% der Fälle müsste unser vorgegebenes p in diesem Intervall liegen, in 5% der Fälle nicht.

```
0.02180000000000004
```

```
let
    p_true = 0.75
    n = 10000
    data = rand(Geometric(1-p_true), n)
    grenz = [grenzen(x) for x in data]
    n_innerhalb= count(g -> (g[1] < p_true < g[2]), grenz)
    p_ausserhalb = 1 - n_innerhalb/n
end
```

Unsere Intervallgrenzen sind etwas zu groß. Zu selten liegt der wahre Wert außerhalb der Grenzen. Dies ist ein Problem bei diskreten Verteilungen. Hier sind die Messwerte diskret,und damit notgedrungen auch die berechneten Grenzen des Konfidenzintervalls. Dadurch kann man die Wahrscheinlichkeit nicht exakt auf den gewünschten Wert einstellen.

Normalverteilung

Wenn man die Verteilungsfunktion kennt, kann man die Grenzen des Konfidenzintervalls wie oben gezeigt berechnen. Oft ist die Verteilung eine Normalverteilung. Dies nimmt man notfalls auch an, wenn man es nicht besser weiss.

Im ersten Kapitel hatten wir uns schon die Integrale über $[\mu - n\sigma, \mu + n\sigma]$ angeschaut

```
▸ [0.682689, 0.9545, 0.9973]
```

```
[cdf(Normal(0, 1), n) - cdf(Normal(0, 1), -n) for n in (1,2,3)]
```

Ein $\pm\2\sigma$-Intervall beinhaltet also 95.45% der Fälle. Eigentlich suchen wir ja ein Intervall mit genau 95%. Das liegt aber in der Nähe, nämlich bei $\pm 1.96\sigma$, siehe den 'Minimizer' hier

```
Results of Optimization Algorithm
 * Algorithm: Brent's Method
 * Search Interval: [0.000000, 3.000000]
 * Minimizer: 1.959964e+00
 * Minimum: 2.553723e-19
 * Iterations: 12
 * Convergence: max(|x - x_upper|, |x - x_lower|) <= 2*(1.5e-08*|x|+2.2e-16): true
 * Objective Function Calls: 13
```

```
optimize( n -> ( cdf(Normal(0, 1), n) - cdf(Normal(0, 1), -n) - 0.95).^2, 0,3)
```

Ein Problem ist hier, dass die Standardabweichung σ hier die **wahre** Standardabweichung ist. Wir kennen aber typischerweise nur eine **geschätzte** Standardabweichung, die wir auf Grundlage unserer Messwerte schätzen müssen. Diesen zusätzlichen Faktor diskutieren wir im nächsten Kapitel.

Bsp. Gauß'sche Fehlerrechnung

Die Gauß'sche Fehlerrechnung kann als Methode der Intervallschätzung gesehen werden. Wir haben mehrere Messwerte y_i an Stellen x_i, die wir durch ein Modell $f(x; p_j)$ mit den Parametern p_j beschreiben wollen. Wir finden die p_j, indem wir die quadratische Abweichung minimieren, also nach den p_j ableiten und Null setzen, also

$$\frac{\partial}{\partial p_j} \sum_i (y_i - f(x_i; p_k))^2 = 0$$

So erhalten wir ein Gleichungssystem zur Bestimmung der p_j, das wir lösen.

Das Konfidenzintervall um diese optimalen p_j erhalten wir durch Gauß'sche Fehlerfortpflanzung durch dieses Gleichungssystem: Die Unsicherheit σ_j im Paramter p_j ist die partielle Ableitung des Parameters nach allen Messwerten y_i multipliziert mit deren Unsicherheit σ_i, und alles quadratisch addiert, also

$$\sigma_j^2 = \sum_i \sigma_i^2 \left(\frac{\partial p_j}{\partial y_i} \right)^2$$

Dabei haben wir die Bestimmungsgleichung für die p_j in der Nähe der optimalen p_j linearisiert, weil nur die erste Ableitung eingeht.

Als Beispiel betrachetn wir jetzt eine Gerade

$$y = a + bx$$

Die Koeffizienten werden bestimmt über

$$\hat{b} = cor_{xy} \frac{\sigma_y}{\sigma_x} \quad \text{und} \quad \hat{a} = \bar{y} - \hat{b}\bar{x}$$

wobei σ_x bzw. σ_y die Standardabweichung über alle Werte x_i bzw. y_i ist und cor_{xy} der Korrealtions-Koeffizient (siehe auch Kapitel 2).

Unter der Annahme von identischen Unsicherheiten in allen Messwerten ist ein guter Schätzer der Unsicherheit des einzelnen Messpunkts die mittlere quadratische Abweichung zum Modell, also

$$\hat{\sigma} = \frac{1}{N-2} \sum_i \Big(y_i - (\hat{a} + \hat{b} x_i) \Big)^2$$

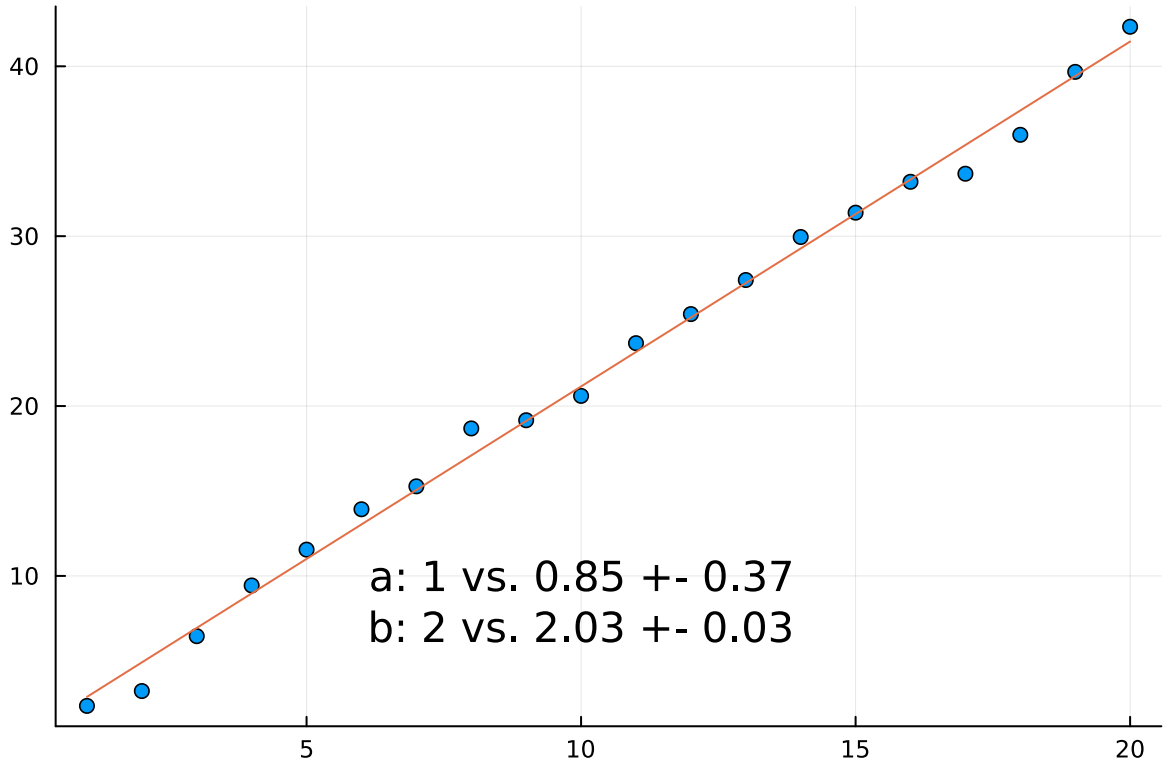
Die 2 in $N-2$ stammt von den zwei durch $\hat{a}$ und $\hat{b}$ verbrauchten Freiheitsgarden der Messung, analog zur Schätzung der Standardabweichung oben.

Damit bekommt man durch Fehlerfortpflanzung (siehe Bevington Kap 6.4 und Stahel Kap. 13.2)

$$\hat{\sigma_a} = \hat{\sigma} \sqrt{\frac{\sum x_i^2}{\Delta}} \quad \text{und} \quad \hat{\sigma_b} = \hat{\sigma} \sqrt{\frac{N}{\Delta}} = \frac{\hat{\sigma}}{\sigma_x} \frac{1}{\sqrt{N-1}}$$

mit $\Delta = \sigma_x^2 \, N(N-1)$.

In Julia ist das




```
let
    a_true = 1
    b_true = 2
    σ_true = 1
    x = range(1, 20)
    N = length(x)
    y_true = a_true .+ b_true .* x
    y = rand(Normal(0, σ_true), N) .+ y_true

    # estimate (a,b) from data
    b = cor(x,y) * std(y)  / std(x)
    a = mean(y) - b * mean(x)
    y_fit = a .+ b .* x

    # estimate σ, Bevington eq. 6.15
    σ = sqrt(sum( (y .- (a .+ b.* x)).^2)  / (N + 2))

    # estimate σ_a, σ_b, Bevington eq. 6.23
    d = std(x)^2 * N * (N-1)
    σ_a = σ * sqrt(sum(x.^2) / d)
    σ_b = σ * sqrt( N / d )

    #plot everything
    scatter(x,y)
    plot!(x, y_fit, legend=false)
    annotate!(10,10, "a: $(a_true) vs. $(round(a, digits=2)) +- $(round(σ_a,
    digits=2))")

```

```
28      annotate!(10,7, "b: $(b_true) vs. $(round(b, digits=2)) +- $(round(σ_b,
        digits=2))")
    end
```

Bedingung an χ^2

Wir hatten oben gesehen, dass die Summe der gewichteten Residuen χ^2

$$\chi^2 = \sum_i \frac{(y_i - f(x_i; p))^2}{\sigma_i^2}$$

minimal wird für die wahrscheinlichsten Parameter p. Der Erwartungswert für dieses Minimum ist die Anzahl der Freiheitsgrade ν der Messung (Bevington, Kap. 4.4)

$$\mathcal{E}(\chi^2) = \nu = n - n_c$$

wobei die Messung n Datenpunkte umfasst und n_c Parameter im Modell verwendet ('verbraucht') wurden. Manchmal definiert man

$$\chi^2_\nu = \chi^2/\nu \quad \text{mit} \quad \mathcal{E}(\chi^2_\nu) = 1$$

Wie ändert sich χ^2 mit den Parametern p_j? Die erste Ableitung ist nach Definition Null. Die zweite ist verknüpft mit der Unsicherheit σ_j im Parameter p_j (aus der Verbindung zur Log-Likelihood-Funktion $\mathcal{L}$, Bevington, eq. 8.10)

$$\frac{\partial^2 \chi^2}{\partial p_j^2} = \frac{2}{\sigma_j^2}$$

bzw

$$\sigma_j^2 = \left(\frac{\partial^2 \mathcal{L}(p_j)}{\partial p_j^2} \right)^{-1}$$

oder andersherum

$$\mathcal{L}(p_j \pm \sigma_j) = \mathcal{L}(p_j) - \frac{1}{2}$$

Bei mehr als einem Parameter stellt die Kontur bei $\mathcal{L}_{\max} - 1/2$ also auch die Kovarianz der Unsicherheit in den Parametern dar.

Bootstrapping

Wenn eine Normalverteilung der Unsicherheit nicht angenommen werden kann, beispielsweise bei Poisson-Verteilungen mit $\lambda < 10$, dann ist 'bootstrapping' die Lösung. Wikipedia schreibt 'selten auch Münchhausen-Methode genannt', aber genau das ist die Idee. Man zieht sich an den Haaren aus dem Sumpf!

Wir haben einen Datensatz aus N Messwerten, die wir in n Intervalle eines Histogramms einsortieren und mit unserem Modell des Histogramms vergleichen wollen. Histogramm-Balken-Höhen sind gezählte Ereignisse, also Poisson-verteilt. Wir passen an unser Histogramm ein Modell mittels der Maximum-Likelihood Methode an und bestimmen so die Parameter. Um die Unsicherheit in den Parametern zu bestimmen, erzeugen wir einen neuen Datensatz aus dem alten, originalen ('an dem Haaren aus dem Sumpf!'). Dazu ziehen wir zufällig N Werte aus dem originalen Datensatz *mit Zurücklegen*. Einzelne originale Messwerte können also mehrfach im neuen Datensatz vorkommen, aber der Gesamtumfang bleibt N. An diesen neuen Datensatz passen wir unser Modell wieder an und bestimmen wieder die Parameter. Dies wiederholen wir so oft, dass wir glauben, den zentralen Grenzwertsatz zu erfüllen, z.B. 100-mal. Die Unsicherheit der Parameter ergibt sich aus der Standardabweichung der Verteilung der so ermittelten 100 Varianten der Parameter. Details finden sich z.B. in Bevington & Robinson: Data reduction and error analysis, oder in Press et al. Numerical Recipes.

```
1 using Distributions, StatsBase, LinearAlgebra, Plots
```

```
1 using PlutoUI
```

```
1 using StatsPlots
```

```
1 using Optim
```

Inhalt


```
TableOfContents(title="Inhalt")
```