Semester Report - Introduction

Thomas Baldauf

Winter semester 2017/18

1 Introduction

In the first chapter, we got to know two basic principles of probability. The first is Kolmogorov's definition of probability, and the second is Bayes' theorem.

1. Kolmogorov's definition of probability: Define a set S

   - For every subset A in S, $P(A) \geq 0$
   - For disjoint subsets $A \cap B = \emptyset, P(A \cup B) = P(A) + P(B)$, where A and B are disjoint
   - $P(S) = 1$

   The basic idea is to introduce a set S of all possible outcomes of a probabilistic process (sample space). A, B are elements of a set of considered outcomes F. They are then subsets of this set, and $\{A_i\}$ are again subsets of the subset $A \subseteq S$. We define a probability space as the triple (S, F, P). Here, P is called probability measure function and maps outcomes to probabilities.

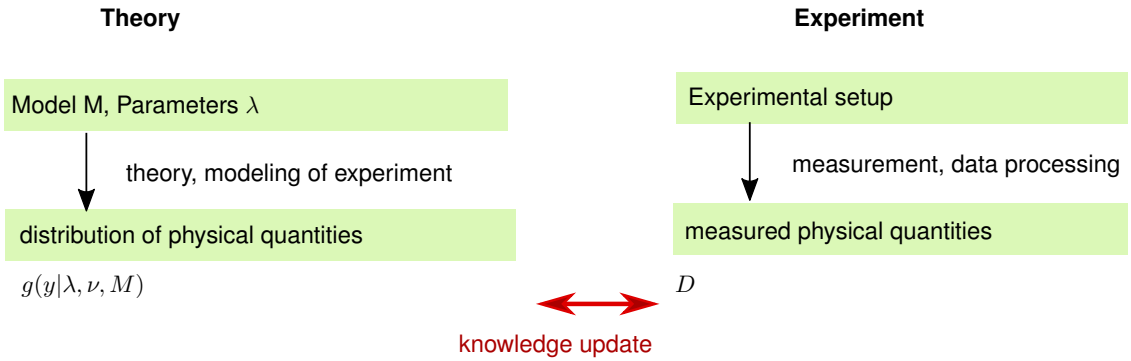
2. Bayes' theorem: $P(A|B) = \dfrac{P(A \cap B)}{P(A)} = \dfrac{P(B|A)P(A)}{\sum_{i=1}^{N} P(B|A_i)P(A_i)}$

   Bayes' theorem follows directly from the definition of conditional probability. The last equality arises from the law of total probability. We called Bayes' theorem the "theorem of scientific knowledge updates", as it gives the probability of an outcome B given a prior knowledge $P(A)$. Bayes' theorem can also be used to determine parameters of a model, and gives the probability of this parameters given some data measured.

We also had a look on the physical relevance of Bayes' theorem, or more general: augmenting our scientific knowledge using probabilistic methods. We defined scientific knowledge as a justified true belief. Beliefs are very vague in science, as we can have beliefs about anything. A *justified* belief as to be a logically consistent, explaining known measurements. It also has to be able to predict possible future experiments. The principle of Occam's razor argues that simple models, which can easily be re-tested and verified by measurements, are in favor of many physicists. Roughly speaking, we can update

- the model itself (i.e. the belief)
- the parameters within the model

In the context of the latter considerations, we introduced the following knowledge updating scheme for scientific knowledge:



where g is a simulated experimental outcome as a function of the model M, the model parameters λ and of the additional parameters ν, also called nuisance parameters.
We can distinguish two Schools of Analysis, the Frequentist and the Bayesian school. In the classical, Frequentist view, we incorporate no prior information, only the predicted frequency distribution of observable model parameters and the observed data itself are used. In the Bayesian school, the model parameters are infered from Bayes' theorem. We can thus also obtain information on the probability of the model parameters.
As an example, we discuss a particle detector in the Frequentist and in the Bayesian way of thinking.

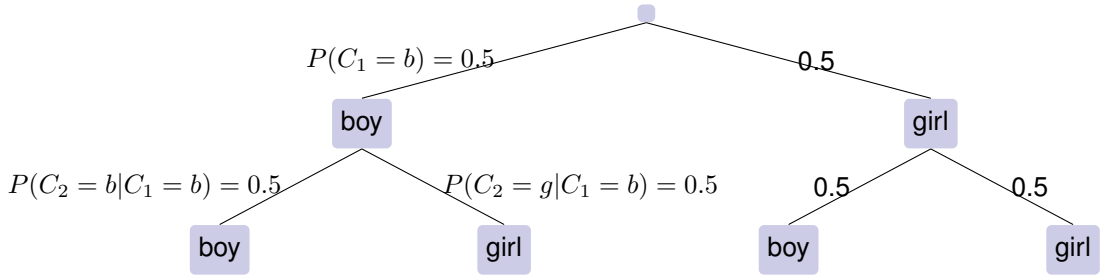
2 Tasks

2.1 Jane and her children

You meet Jane on the street. She tells you she has two children, and has pictures of them in her pocket. She pulls out one picture, and shows it to you. It is a girl. What is the probability that the second child is also a girl ? Variation: Jane takes out both pictures, looks at them, and is required to show you a picture of a girl if she has one. What is now the probability that the second child is also a girl ?
Jane has two children, let us call them C_1, C_2.
Where "g" stands for girl and "b" for boy. For the first part, we consider the right part of the branch at depth one, i.e. on the "girl" node. We can consider either the probability $P(C_2 = g|C_1 = g)$ or $P(C_2 = g \wedge C_1 = g)$.

$$P(C_2 = g|C_1 = g) = 0.5$$
$$P(C_2 = g \wedge C_1 = g) = P(C_1 = g) \cdot P(C_2 = g) = 0.25$$

We can also use Bayes' theorem



$$P(C_2 = g|C_1 = g) = \frac{P(C_1 = g|C_2 = g)P(C_2 = g)}{P(C_1 = g|C_2 = g)P(C_2 = g) + P(C_1 = g|C_2 = b)P(C_2 = b)}$$
$$= \frac{P(C_1 = g|C_2 = g) \cdot 0.5}{0.5 \cdot 0.5 + 0.5 \cdot 0.5} = P(C_1 = g|C_2 = g)$$

So for the variation task, it does not make difference if Jane takes out the picture of the first or the second child, probabilities will stay the same. This statement is based on the assumption that $P(g) = P(b) = 0.5$.

2.2 Probability?

Go back to section 1.2.3 and come up with more possible definitions for the probability of the data.

Within the context of the model, one can assign various definitions of probability, but only one (or few of them) might seem to be appopriate. For example, in section 1.2.3. of the script, the famous heads and tails problem was discussed, where a coin is flipped N times, and the results are written up. The ”counterintuitive” result is that the probability of ”only heads” is the same as any other configuration, allthough we would like to consider ”only heads” intuitively as less probable. That is why we used the binomial coeffitient to express this kind of probability. We have in principle two definitions of the probability:

- the probability that a certain (elementary) event happens one time (neglecting prior knowledge), like 0.5 for ”heads” and 0.5 for ”tails”, number of outcomes of a certain event per total number of outcomes. This is the classical interpretation of probability.
- the frequency of a certain physical process, e.g. the probability of a nuclear decay (having many nuclei decaying at a time and counting the decay frequency), called frequentist approach.
- the degree of confidence in a stochastic process, e.g. the probability an ensurance calculates for the car accident risk of young drivers (Bayesian probability)
- the ”uniqueness” of an event (like ”only heads” or ”only tails”) in comparison to all possible events (”HHTH” or ”HTTH” ... ”two times tails”), e.g. binomial distribution.
- credence or degree of believe, e.g. in Casinos, where people believe to be winners, or the proposal that neutrinos are faster than the speed of light (which cannot quite be corroborated

by prior knowledge about neutrinos). This concept is called subjectivism. Heads and tails could express if some measured values are in accordance with a model or not.

- propensity approach, trying to make use of the law of large numbers, and trying to find the reason behind a stochastic process. Instead of flipping a coin 100 times, we can understand the physical pre-conditions for the coin-flip model, and derive a basic physical principle (with stochastic nature).
- logical appoach. Saying "Probably Trump was elected because of his extraordinary personality" is not meant to be a degree of believe. It rather expresses a logical consequence. E.g., heads and tails could be a logical consequence of an experiment.

A proper mathematical definition of probability was given by Kolmogorov.

2.3 Particle detector

Your particle detector measures energies with a resolution of 10%. You measure an energy, call it E. What probabilities would you assign to possible true values of the energy ? What can your conclusion depend on?

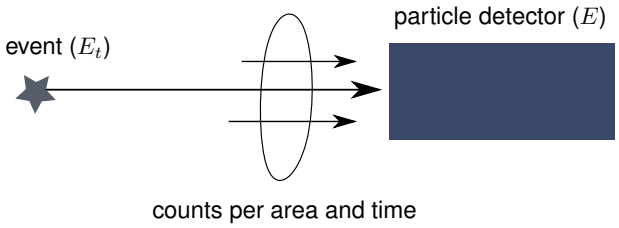


Figure 1: Scheme for the detection of energies using a particle detector.

Let us consider the situation shown in figure 1. At a certain spatial position, particles with exactly same energy E_t are created. The particle detector meausres always different energies E for these same energies. The energies E fluctuate around the true value E_t.
We do not have any information about the response of our detector. Thus, we could assume that any true energy value E_t in the interval $[0.9E, 1.1E]$ is equally probable. We end up with a box distribution

$$p_{\text{box}}(E_t) = \begin{cases} \dfrac{1}{0.2E} = \text{const.} & \text{for } E_t \in \{0.9E, 1.1E\} \\ 0 & \text{else} \end{cases}$$

The factor 0.2 arises from the normalization condition

$$\int p(E_t) dE_t \stackrel{!}{=} 1.$$

As we have a particle detector, we could assume a Poisson distribution as the detector's response function. The Poisson distribution can be approximmated by the Gauss distribution. The 10% could then refer to the standard deviation σ of the Gauss distribution (or even multiples of σ). We would get the following probability:

$$p_{\text{Gauss}}(E_t) = \text{Gauss}(E_t, \sigma = 0.1E).$$

The Gauss distribution,

$$p_{\text{Gaus}}(E_t, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{E_t - E}{\sigma}\right)^2\right\},$$

is already normalized. Alternatively, we can use the FWHM which is defined as $\text{FWHM} = 2\sqrt{2\ln 2}\sigma$ instead of the standard deviation to express the resolution.
The conculsion on the true value definitely depends on the definition of the detector response function!

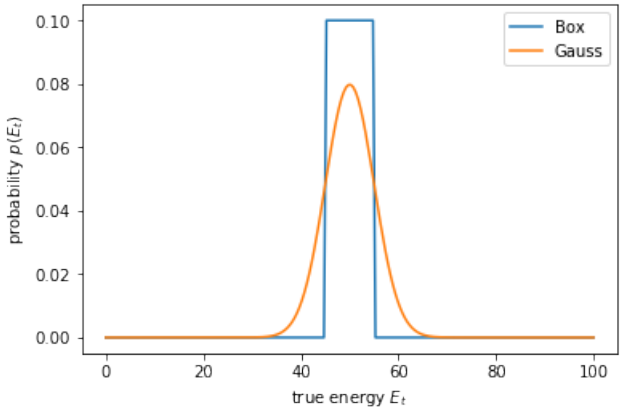


Figure 2: Comparison of the two models for the detector response. In this example, E is chosen to be 50 (arbitrary) energy units.

We can observe a count rate of the detector by tuning through the energies and counting the events per time at that certain energy, for example in a Gamma spectrum. Then, the observed count rate is the convolution of the detector response with the true signal. If the detector response were a delta function, we could measure the true count rate with inifinite accuracy. We can measure the detector response function by shooting particles of well-known energy, and measuring the detected peak FWHM.

2.4 Mongolian Swamp Fever

Mongolian swamp fever is such a rare disease that a doctor only expects to meet it once every 10000 patients. It always produces spots and acute lethargy in a patient; usually (I.e., 60% of cases) they suffer from a raging thirst, and occasionally (20% of cases) from violent sneezes. These symptoms can arise from other causes: specifically, of patients that do not have the disease: 3% have spots, 10% are lethargic, 2% are thirsty and 5% complain of sneezing. These four probabilities are independent.

What is your probability of having Mogolian swamp fever if you go to the doctor with all or with any three out of four of these symptoms ? (From R.Barlow)

Going to the doctor and telling him or her what symptoms you have gives the doctor some data $\mathcal{D}$ about your health status. We are given the probabilities $p(X_i|S)$, which are the probabilites of the symptoms X_i, given a person is infected with the fever (S) or not ($\bar{S}$). These probabilities are the likelihoods of the data, and are visualized in figure 3 together with the prior probabilities.

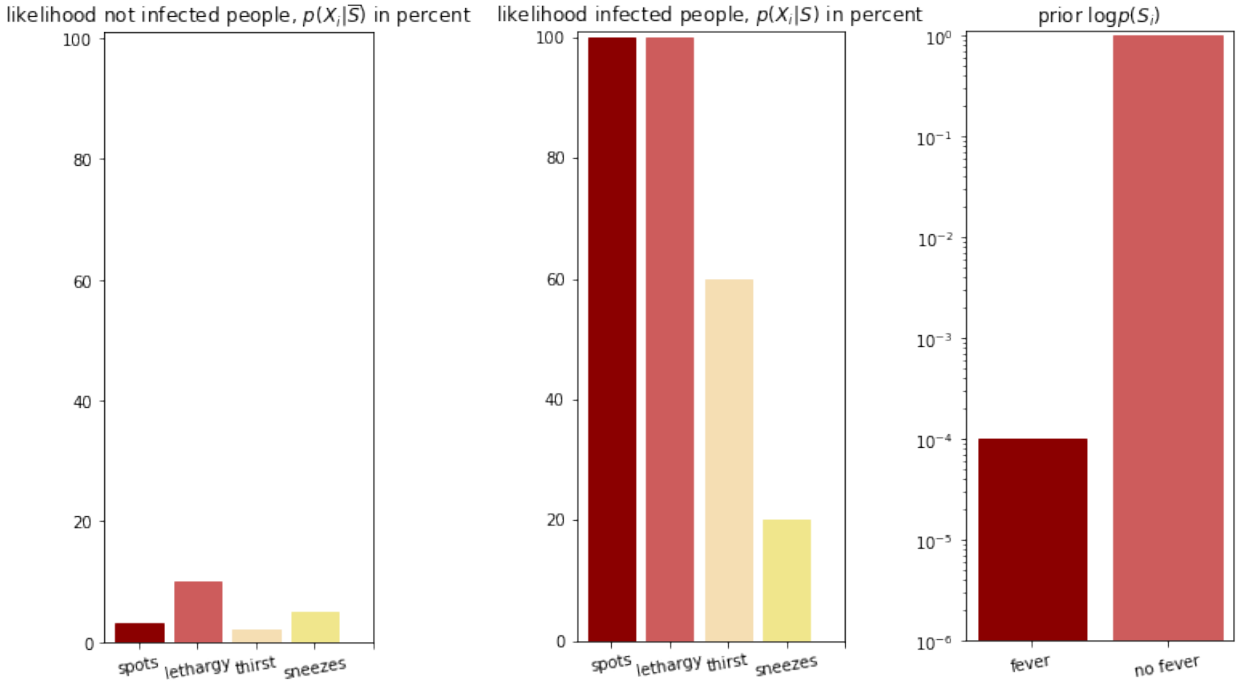


Figure 3: Likelihoods for infected people (left) and not infected people (middle) in percent. The right figure shows the prior probabilities $p(S)$ and $P(\bar{S})$.

$$
\begin{aligned}
P(\text{"spots"}|S) &= 1.0, & P(\text{"spots"}|\bar{S}) &= 0.03 \\
P(\text{"lethargy"}|S) &= 1.0, & P(\text{"lethargy"}|\bar{S}) &= 0.1 \\
P(\text{"thist"}|S) &= 0.6, & P(\text{"thirst"}|\bar{S}) &= 0.02 \\
P(\text{"sneezing"}|S) &= 0.2, & P(\text{"sneezing"}|\bar{S}) &= 0.05
\end{aligned}
$$

Bayes theorem states:

$$
\begin{aligned}
p(S|X_i) &= \frac{p(X_i|S)P(S)}{p(X_i)} \\
p(\bar{S}|X_i) &= \frac{p(X_i|\bar{S})P(\bar{S})}{p(X_i)}
\end{aligned}
$$

Where we call $p(S|X_i)$ the posterior probability, $p(X_i|S)$ the likelihood and $P(S)$ the prior probability. $p(X_i)$ is the evidence. We know the prior probability by the estimation that only one in 10,000 patients is infected with the fever, so $p(S) = (10000)^{-1}$. Therefore, he prior distribution is just a constant distribution.
We can directly calculate the probability if we know a patient shows one symptom. If the patient shows more than one symptom, we have to deal with a joint probability. So, reading the task carefully again, we want the following probabilities:

$$
\begin{aligned}
&p(S \mid \text{"lethargy"}, \text{"thirst"}, \text{"spots"}) \\
&p(S \mid \text{"spots"}, \text{"lethargy"}, \text{"sneezes"}) \\
&p(S \mid \text{"sneezes"}, \text{"thirst"}, \text{"spots"}) \\
&p(S \mid \text{"thirst"}, \text{"lethargy"}, \text{"sneezes"})
\end{aligned}
$$

To get the wanted probabilities, given all observed symptoms, we use Bayes' theorem:

$$
\begin{aligned}
P(S|X_i) &= \frac{P(X_i|S)P(S)}{P(X_i)} \\
P(S|X_i, X_j) &= \frac{P(X_i, X_j|S)P(S)}{P(X_i, X_j)} \\
P(S|X_i, X_j) &\overset{X_i \perp X_j}{=} \frac{P(X_i|S)P(X_j|S)}{P(X_i, X_j|S) + P(X_i, X_j|\bar{S})} P(S) \\
P(S|\{X_i\}_{i=1}^N) &= \frac{\left[\prod_{i=1}^N P(X_i|S)\right] P(S)}{\left[\prod_{i=1}^N P(X_i|S)\right] P(S) + \left[\prod_{i=1}^N P(X_i|\bar{S})\right] P(\bar{S})}
\end{aligned}
$$

This results in

$$
\begin{aligned}
p(S \mid \text{"leth."}, \text{"th."}, \text{"sp."}) &= \frac{1.0 \cdot 0.6 \cdot 1.0}{0.0001 \cdot (1.0 \cdot 0.6 \cdot 1.0) + 0.9999\,(0.1 \cdot 0.02 \cdot 0.03)} \approx 0.5000 \\
p(S \mid \text{"sp."}, \text{"leth."}, \text{"sn."}) &= \frac{1.0 \cdot 1.0 \cdot 0.2}{0.0001 \cdot (1.0 \cdot 1.0 \cdot 0.2) + 0.9999\,(0.03 \cdot 0.1 \cdot 0.05)} \approx 0.1177 \\
p(S \mid \text{"sn."}, \text{"th."}, \text{"sp."}) &= \frac{0.2 \cdot 0.6 \cdot 1.0}{0.0001 \cdot (0.2 \cdot 0.6 \cdot 1.0) + 0.9999\,(0.05 \cdot 0.02 \cdot 0.03)} \approx 0.2857 \\
p(S \mid \text{"th."}, \text{"leth."}, \text{"sn."}) &= \frac{0.6 \cdot 1.0 \cdot 0.2}{0.0001 \cdot (0.6 \cdot 1.0 \cdot 0.2) + 0.9999\,(0.02 \cdot 0.1 \cdot 0.05)} \approx 0.1072 \\
p(S \mid \text{"th."}, \text{"leth."}, \text{"sn."}, \text{"sp."}) &= \frac{0.6 \cdot 1.0 \cdot 0.2 \cdot 1.0}{0.0001 \cdot (0.6 \cdot 1.0 \cdot 0.2 \cdot 1.0) + 0.9999\,(0.02 \cdot 0.1 \cdot 0.05 \cdot 0.03)} \approx 0.8000
\end{aligned}
$$

Binomial and Multinomial Distribution

Thomas Baldauf

Winter semester 2017/18

1 Introduction

1.1 Binomial Distribution

The Binomial distribution models situations with two possible outcomes, that have fixed probabilities $p, 1-p$, and a fixed number of trials. The efficiency of a "counting apparatus" can be modeled by the Binomial distribution, as you can count the number of trials from a calibration source, and record the counts registered in the apparatus. The two events can be "particle/no particle" or even "particle of type A/particle of type B". The Binomial distribution is given by

$$P(r|N,p) = \frac{N!}{r!(N-r)!} p^r q^{N-r}. \tag{1}$$

It has expectation $\mathrm{E}[r] = Np$ and variance $\mathrm{Var}[r] = Np(1-p)$. Its mode is

$$r^* = 0 \text{ for } p = 0, \text{ and } r^* = N \text{ for } p = 1 \tag{2}$$

or more generally

$$r^* = \begin{cases} \lfloor (N+1)p \rfloor \text{ for } p \neq 0, 1 \text{ and } (N+1)p \notin \{1, \ldots, N\} \\ \lfloor (N+1)p \rfloor \text{ and } \lfloor (N+1) \rfloor - 1 \text{ for } (N+1)p \in \{1, \ldots, N\}. \end{cases} \tag{3}$$

In the limit $N \to \infty, p \neq 0$, the Binomial distribution approaches a Gaussian distribution:

$$P(r|N,p) \approx \frac{1}{\sqrt{2\pi Np(1-p)}} \exp\left(-\frac{1}{2} \frac{(r-Np)^2}{Np(1-p)} \right) \tag{4}$$

1.2 Confidence Intervals

There exist many definitions of confidence intervals. They give each probability a rank. The probability is summed up (accumulated) in a special way (defined by the interval of choice) until a certain value is reached (which is called confidence level α)

1.2.1 Central Interval

The central interval is defined as the interval $[r_1, r_2]$ where

$$P(r < r_1) \leq \alpha/2 \tag{5}$$
$$P(r > r_2) \leq \alpha/2 \tag{6}$$

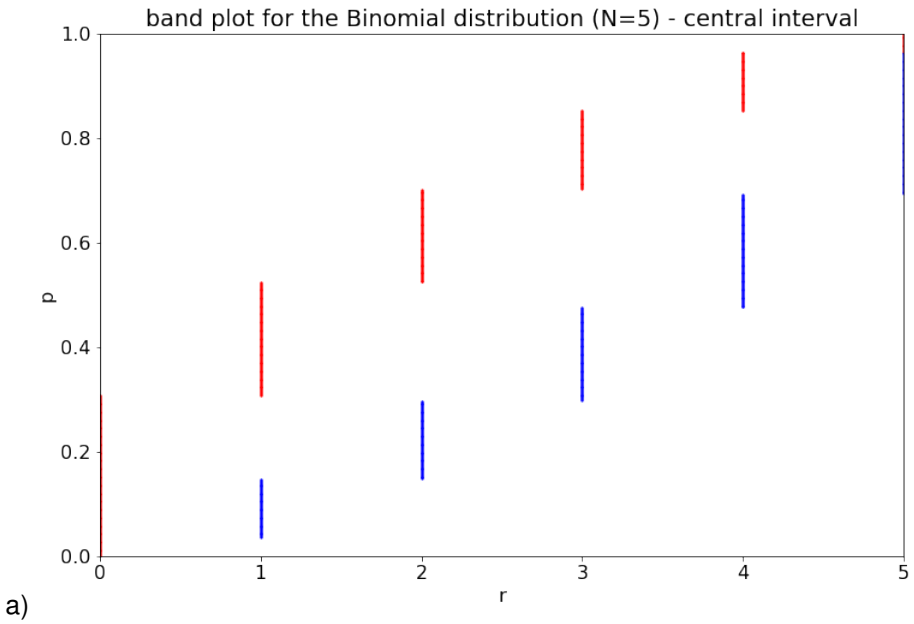
where P is the probability distribution under consideration, and r denotes a possible outcome of an experiment. Technically, we sum up the probability content from the right and left tail until we reach more or equal than $\alpha/2$.

1.2.2 Smallest Interval

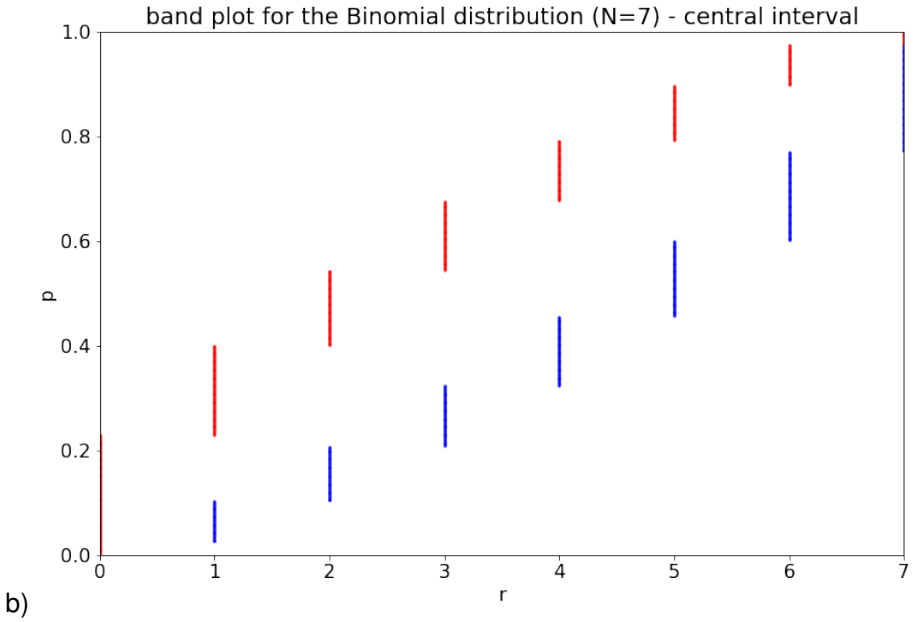
The smallest interval is the smallest interval around the mode containing at least a certain probability $1 - \alpha$ is reached. We start with the mode and add more and more values, in the order of their probbility, until we have collected enough probability. If there are two equallly probbale outcomes, both are added.

1.2.3 Neyman Confidence Level construction

If the success probability p is unknown, we have to estimate a confidence interval for p. This is known as the Neyman Confidence Level. We start by constructing a band plot for a certain value of $1 - \alpha$,N, using the smallest or central interval. Then, we can measure the (real) number of successes r_D. We can find a range of p such that r_D lies within the 68% central/smallest interval. The p range obtained by this procedure is the $1 - \alpha$ Neyman confidence interval for p. An example for $N = 5$ and $N = 7$ is shown in figure 1.



a)



b)

Figure 1: Bandplot for the smallest 68% confidence level interval for $N = 5$ (a) and $N = 7$ (b). These plots were generated by code B.

1.3 Bayesian Analysis of Binomial data

We can perform an analytical Bayesian analysis on Binomial data. That is

$$P(p|N,r) = \frac{P(r|N,p)P_0(p)}{P(r|N)} = \frac{P(r|N,p)P_0(p)}{\int P(r|N,p)P_0(p)\,\mathrm{d}\,p}. \tag{7}$$

As many other well-known probability distributions, the Binomial distribution has a well-known conjugate prior, which is the Beta distribution. This means that using a Binomial prior results in a Beta posterior. Also, as we found in the lecture, using a flat prior and Binomial likelihood results in the following posterior distribution

$$P(p|N,r) = \frac{(N+1)!}{r!(N-r)!}p^r(1-p)^{N-r} \tag{8}$$

In the context of the posterior distribution, we can define the median, which is the value p_{med} with

$$F(p_{\text{med}}) = \int_0^{p_{\text{med}}} P(p|N,r)\,\mathrm{d}\,p = \int_{p_{\text{mex}}}^1 P(p|N,r)\,\mathrm{d}\,p = \alpha/2 \tag{9}$$

Where α defines the $1-\alpha$ central interval.
We can also compute the mode p^*. It is r/N for a flat prior, and the smallest interval is defined via

$$1-\alpha = \int_{p_1}^{p_2} P(p|r,N)\,\mathrm{d}\,p, \quad P(p_1|r,N) = P(p_2|r,N), \quad p_1 < p^* < p_2 \tag{10}$$

If we use a non-flat prior $P(p,N_1,r_1)$ and a likelihood $P(r_2|N_2,p)$, we get

$$P(p|N,r) = \frac{(N_1+N_2+1)!}{(r_1+r_2)!(N_1+N_2-r_1-r_2)!}p^{r_1+r_2}(1-p)^{N_1+N_2-r_1-r_2} \tag{11}$$

where $r = r_1 + r_2, N = N_1 + N_2$, we can see that it yields the same result as in the flat prior case, but with the data points added together.

The Binomial distribution can also be generalized to the Multinomial distribution.

2 Tasks

2.1 Mean and standard deviation

For the following function

$$P(x) = xe^{-x},\ 0 \le x < \infty \tag{12}$$

(a) Find the mean and standard deviation. What is the probability content in the interval (mean-standard deviation, mean+standard deviation)?

(b) Find the median and 68% central interval

(c) Find the mode and 68% smallest interval

The mean of x is the expectation value of x:

$$\mathbb{E}[x] = \int_0^\infty x P(x)\,\mathrm{d}\,x = \int_0^\infty x^2 e^{-x}\,\mathrm{d}\,x = 2. \tag{13}$$

The expectation value of x^2 is

$$\mathbb{E}[x^2] = \int_0^\infty x^2 P(x)\,\mathrm{d}\,x = 6. \tag{14}$$

The standard deviation is

$$\sigma = \sqrt{\mathbb{E}[x^2] - \mathbb{E}[x]^2} = \sqrt{6-4} = \sqrt{2}. \tag{15}$$

The probability content within the standard deviation is

$$q = \int_{\mu-\sigma}^{\mu+\sigma} P(x)\,\mathrm{d}\,x = \int_{2-\sqrt{2}}^{2+\sqrt{2}} x e^{-x}\,\mathrm{d}\,x = 0.7375 \tag{16}$$

The median is defined as

$$F(x_{\text{med}}) = \int_0^{x_{\text{med}}} P(x)\,\mathrm{d}\,x = 0.5 \tag{17}$$

We explicitly calculate the integral, and get the equation

$$-\exp(-x_{\text{med}})(x_{\text{med}}+1) + 1 = 0.5 \tag{18}$$

We can solve this equation numerically and get

$$x_{\text{med}} \approx 1.6784 \tag{19}$$

(we do not take care about the negative solution). We can also see the solution graphically (see figure 2).
The central probability interval is then given by

$$F(x_1) = \int_0^{x_1} P(x)\,\mathrm{d}\,x = \int_{x_2}^\infty P(x)\,\mathrm{d}\,x = \frac{\alpha}{2} \tag{20}$$

where α denotes the probability that the parameter p lies outside of this interval. We get for the integral

$$\int_a^b x e^{-x}\,\mathrm{d}\,x = -e^{-b}(b+1) + e^{-a}(a+1) \tag{21}$$

this yields two equations for x_1 and x_2, with $\alpha = 0.68$.

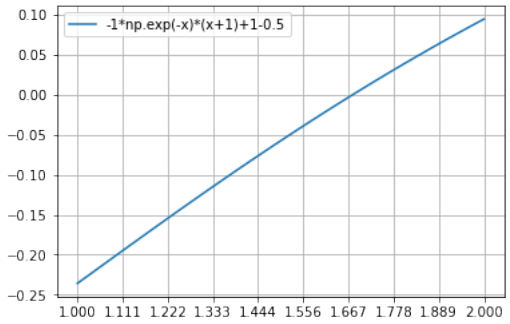


Figure 2: The graphical solution of the equation $-\exp(-x_{\text{med}})(x_{\text{med}}+1)+1=0.5$

$$-e^{-x_1}(x_1+1)+1 = \frac{1-0.68}{2} \Rightarrow x_1 = 0.71 \tag{22}$$

$$e^{-x_2}(x_2+1) = \frac{1-0.68}{2} \Rightarrow x_2 = 3.29 \tag{23}$$

Then the central probability Interval is

$$\Rightarrow [x_1, x_2] \approx [0.71, 3.29] \tag{24}$$

Last but not least, we calculate the mode and the smallest interval. The mode is given by the x value for which $P(x)$ is maximized. Becaus it only depends on one variable, we can simply calculate the derivative with respect to x:

$$\frac{dP(x)}{dx} = e^{-x} - xe^{-x} = 0 \Rightarrow x^* = 1 \tag{25}$$

The shortest interval is defined as the interval where

$$1-\alpha = \int_{x_1}^{x_2} P(x)dx,\ p(x_1) = p(x_2),\ x_1 < p^* < x_2 \tag{26}$$

where the x_1,x_2 are not to be confused with the central probability interval. They are now the limits of the shortest interval.
We get two equations for two variables:

$$0.68 = -e^{-x_2}(x_2+1) + e^{-x_1}(x_1+1) \tag{27}$$

$$x_1 e^{-x_1} = x_2 e^{-x_2} \tag{28}$$

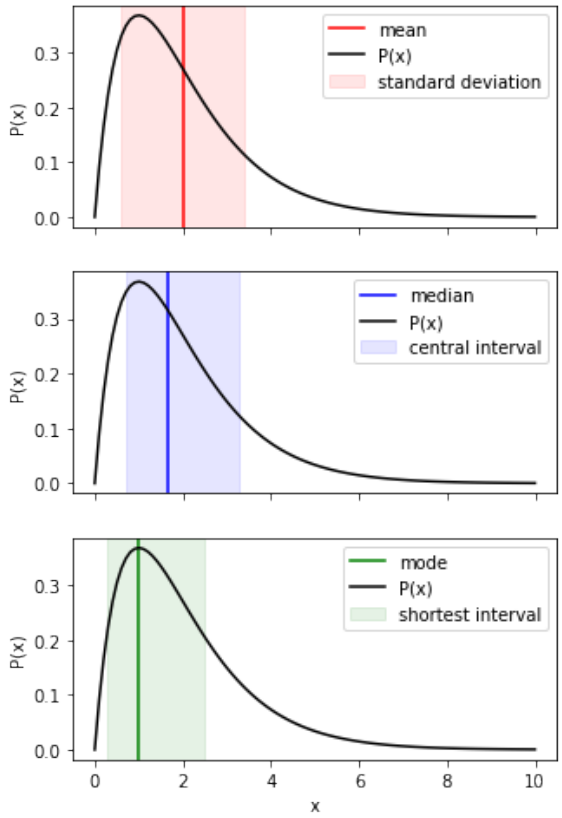
This is a non-linear equation system which can be solved numerically or graphically. The result is:

$$x_1 = 0.2706, x_2 = 2.4899 \tag{29}$$

The shortest interval is

$$[x_1, x_2] \approx [0.27, 2.49] \tag{30}$$

In conclusion, we got the following results:



estimator	x	interval	$[x_1, x_2]$	probability content
median	1.68	central interval	$[0.71, 3.29]$	0.68
mode	1.00	shortest interval	$[0.27, 2.49]$	0.68
mean	2.00	standard deviation	$[0.59, 3.41]$	0.74

Figure 3: Confidence intervals for $P(x)$

2.2 Trials and Successes

Energy	Trials	Successes
0.5	100	0
1.0	100	4
1.5	100	20
2.0	100	58
2.5	100	92
3.0	1000	987
3.5	1000	995
4.0	1000	998

Table 1: Calibration data for an apparatus.

Consider the data in table 1. Starting with a flat prior for each energy, find an estimate for the efficiency (success parameter p) as well as an uncertainty. For the estimate of the parameter, take the mode of the posterior probability for p and use the smallest interval to find the 68% probability range. Make a plot of the result.
The binomial distribution is

$$P(r|N,p) = \binom{N}{r} p^r (1-p)^r \tag{31}$$

where p is the success parameter, N is the nummber of trials, and r is the number of successes. We cannot calculate this direct probability because we have no information about the success parameter p. Thus, we have to perform a Bayesian analysis. From Bayes' theorem, we get a probability distribution for the success parameter. We get exactly the same result like in 8:

$$P(p|N,r) = \frac{(N+1)!}{r!(N-r)!} p^r (1-p)^{N-r} \tag{32}$$

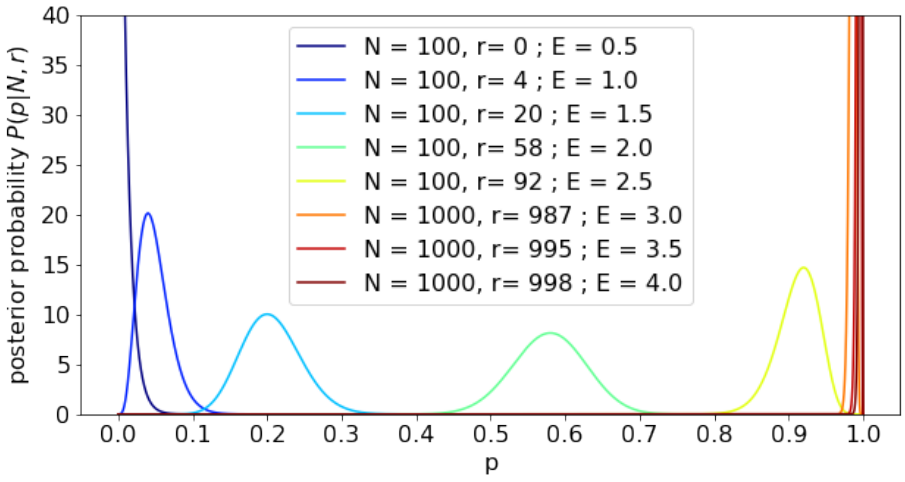


Figure 4: Plot of the posterior probabilites for each data row in table 1.

The mode of a binomial distribution is known, $p^* = r/N$, so we can easily calculate the modes for all rows of the data table. The 68% probability range is calcaulated using the smallest interval. Therefore, we have to find $p_1 < p^* < p_2$ such that

$$1 - \alpha = \int_{p_1}^{p_2} P(p|N,r)dp \tag{33}$$

where $1 - \alpha = 0.68$. Code A computes the approximate boundaries for the samllest interval. The results are:

E	p_1	p_2	probability content	mode of p
0.5	0.0	0.00380	0.6808	0
1.0	0.03240	0.04870	0.6809	0.04
1.5	0.18401	0.21672	0.6807	0.20
2.0	0.55975	0.60006	0.6812	0.58
2.5	0.90839	0.93069	0.6873	0.92
3.0	0.98549	0.98839	0.6873	0.987
3.5	0.99399	0.99579	0.6914	0.995
4.0	0.99729	0.99849	0.6840	0.998

Mind that the probability content (i.e. $1 - \alpha$) is slightly more than 68 percent this is typical for the construction of such intervals, as the probability content is accumulated until it is over a threshold value. It would be coincidence if it were exactly 0.68.
The resulting uncertainties can now be plotted as error bars. The result is shown in figure 5. We can see that for the energies far from the saturation, the detector efficiency is more uncertian. This means that we do know when the detector gives no signal output, and we know when it does, but in between, its efficiency is less exactly known.

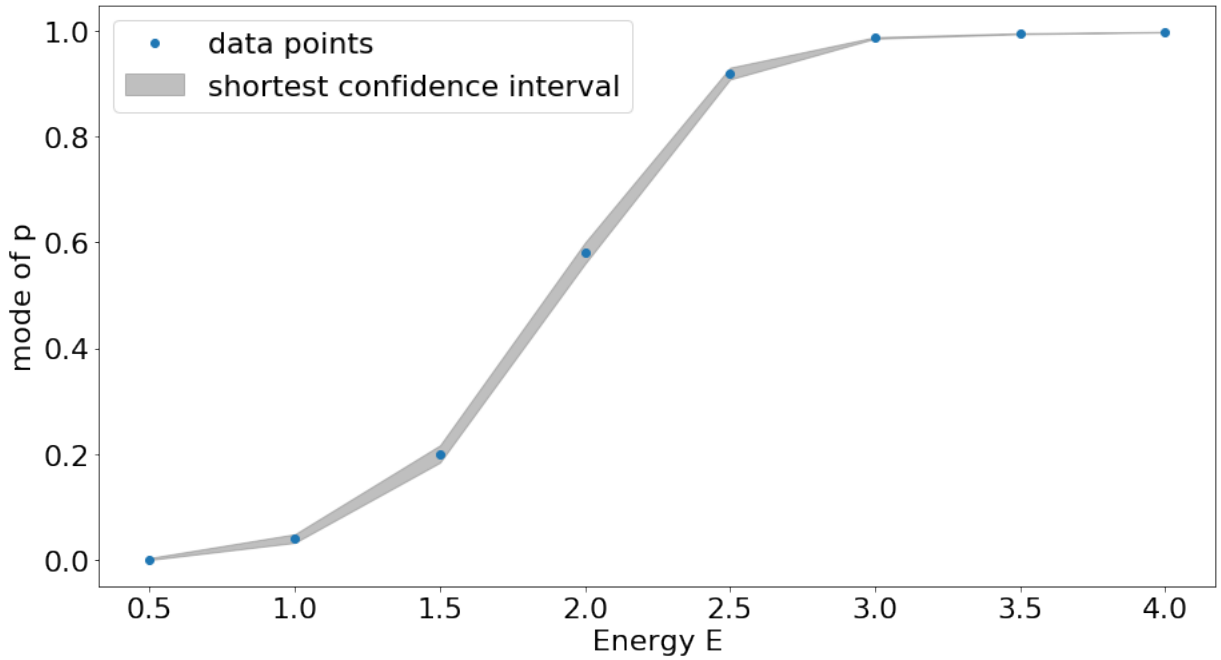


Figure 5: Plot of the uncertainties in efficiency for all data points. The gray bars visualize the determined smallest 68% confidence interval.

2.3 Frequentist perspective

Analyze the data in table 1 from a frequentist perspective by finding the 90% confidence level interval for p as a function of energy. Use the Central Interval to find the 90% CL interval for p.

We can use code D to generate the central interval. In the frequentist persepctive (code B), we can generate a band plot for the corresponding parameters ($N = 100$ and $N = 1000$, $1 - \alpha = 90\%$) and read from the plot if the observed success rate lies within the desired confidence interval. Figure 6 shows the band plots for the corresponding numbers of trials. Instead of "looking it up", we can output the interval directly in python as well. A summary of the obtained values can be seen in table 2.

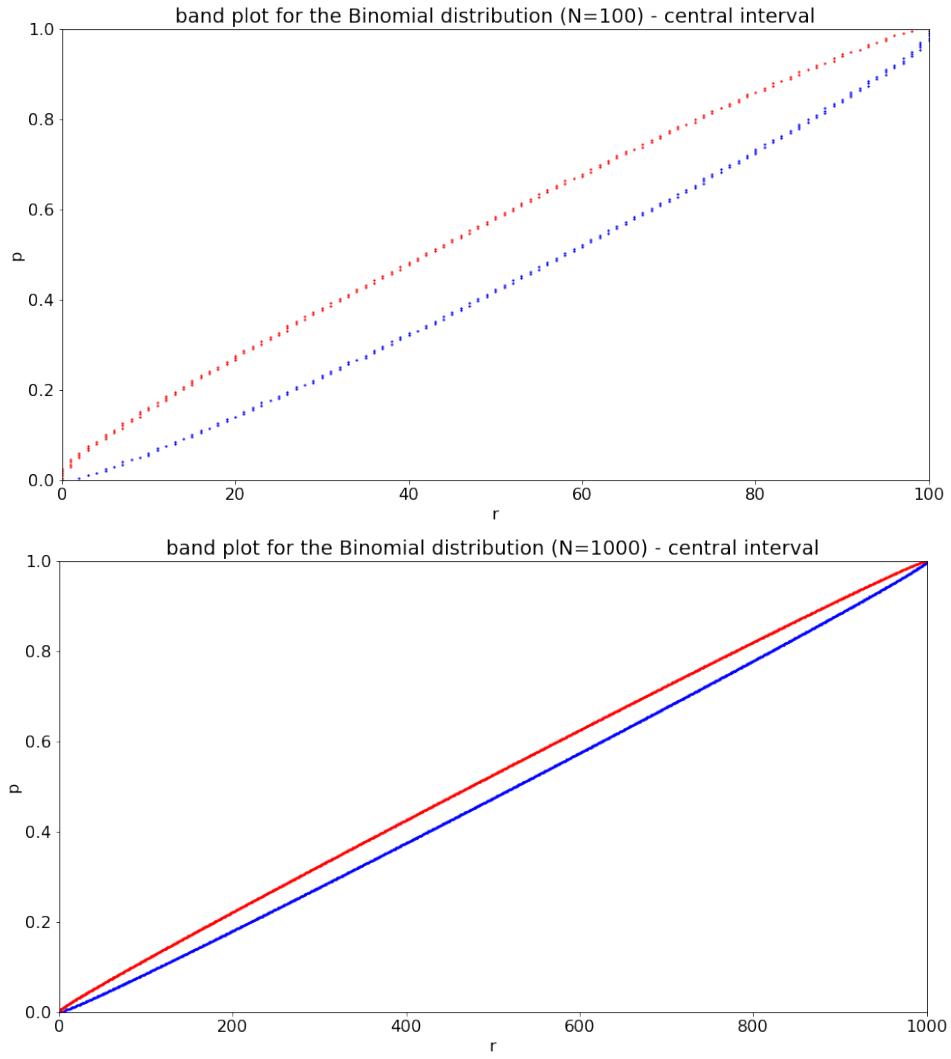


Figure 6: Band plot for a Binomial process with $N = 100$ (top) and $N = 1000$ (bottom).

Energy E	success rate r	p_{min}	p_{max}
0.5	0	0.0	0.025
1.0	4	0.015	0.085
1.5	20	0.141	0.276
2.0	58	0.497	0.663
2.5	92	0.864	0.954
3.0	987	0.980	0.992
3.5	995	0.990	0.998
4.0	998	0.994	0.999

Table 2: List of values for the observed energies and success rates, corresponding to the band plots shown in figure 6

2.4 Multiple usage of data

Let us se what happens if we reuse the same data multiple times. We have N trials and measure r successes. Show that if you reuse the data n times, starting at first with a flat prior and then using the posterior from one use of the data as the prior for the next use, you get

$$P_n(p|r,N) = \frac{(nN+1)!}{(nr)!(nN-nr)!} p^{nr}(1-p)^{n(N-r)} \tag{34}$$

What are the expectation value and variance for p in the limit $n \to \infty$?
Bayes' theorem states

$$P(\vec{p}|\vec{r},N) = \frac{P(\vec{r})P_0(\vec{p})}{\int P(\vec{r}|N,\vec{p})P_0(\vec{p})\,\mathrm{d}\,\vec{p}} \tag{35}$$

Like in the lecture, we can start with a binomial distribution of one data set and a flat prior. The resulting posterior distribution is:

$$P(p|N_1,r_1) = \frac{(N_1+1)!}{r_1!(N_1-r_1)!} p^{r_1}(1-p)^{N_1-r_1}. \tag{36}$$

Re-using this data as the new prior leads us to

$$P(p|N_2,r_2) = \frac{P(r_2|N_2,p)P(p|N_1,r_1)}{\int P(r_2|N_2,p)P(p|N_1,r_1)\,\mathrm{d}\,p} \tag{37}$$

In the lecture we derived the solution. Rewriting $2r := r_1 + r_2$ and $2N = N_1 + N_2$, we obtain again the prior probability, but with the added paramters.

$$P(p|N_1,N_2,r_1,r_2) = \frac{(N_1+N_2+1)!}{(r_1+r_2)!(N_1+N_2-r_1-r_2)!} p^{r_1+r_2}(1-p)^{N_1+N_2-r_1-r_2} \tag{38}$$

If we now perform n such iterations, we get

$$P(p|N,r) = \frac{(nN+1)!}{(nr)!(nN-nr)!} p^{nr}(1-p)^{n(N-r)} \tag{39}$$

where all $N_i = N$ and all $r_i = r$ are equal (we measure exactly N trials with r successes n times). At this point, we notice that the result obtained is nothing else but the Beta distribution, which is the conjugate prior of the Binomial distribution. The beta distribution is defined as

$$\mathrm{Beta}(p, r+1, N-r+1) = \frac{1}{B(r+1, N-r+1)} p^r (1-p)^{N-r} \tag{40}$$

where $B(r+1, N-r+1)$ is defined via the Gamma funcion:

$$\frac{1}{B(r+1, N-r+1)} := \frac{\Gamma(N+2)}{\Gamma(r+1)\Gamma(N-r+1)} = \frac{(N+1)!}{r!(N-r)!} \tag{41}$$

where $\Gamma(n) = (n-1)!$ is the Gamma function for integers. The final result for the posterior probability can thus be written as

$$P(p, N_1, N_2, r_1, r_2) = \mathrm{Beta}(p, \tilde{r}+1, \tilde{N}-\tilde{r}+1) \tag{42}$$

where we defined $\tilde{N} = nN, \tilde{r} = nr$. Obtaining the expected value and the variance for this expression is now easy, as the values are well known for the Beta distribution. We end up with

$$\mathbb{E}[p] = \frac{\tilde{r}+1}{\tilde{r}+1+\tilde{N}-\tilde{r}+1} = \frac{\tilde{r}+1}{\tilde{N}+2} = \frac{nr+1}{nN+2} = \frac{r+\frac{1}{n}}{N+\frac{2}{n}} \tag{43}$$

In the limit for $n \to \infty$, we obtain

$$\lim_{n\to\infty} \mathbb{E}[p] = \lim_{n\to\infty} \frac{r+\frac{1}{n}}{N+\frac{2}{n}} = \frac{r}{N} \tag{44}$$

The variance is a bit more complicated:

$$\mathrm{Var}[p] = \frac{(\tilde{r}+1)(\tilde{N}-\tilde{r}+1)}{(\tilde{r}+1+\tilde{N}-r+1)^2(\tilde{r}+1+\tilde{N}-\tilde{r}+1+1)} \tag{45}$$

in the limit, we get

$$\lim_{n\to\infty} \mathrm{Var}[p] = \lim_{n\to\infty} \frac{1}{n} \frac{(r+\frac{1}{n})(N-r+\frac{1}{n})}{(N+\frac{2}{n})^2(N+\frac{3}{N})} = 0 \cdot \frac{r(N-r)}{N^3} = 0 \tag{46}$$

We put all results together:

distribution $P_n(p\|N,r)$	expected value ($n \to \infty$)	variance ($n \to \infty$)
$\mathrm{Beta}(p, nr+1, nN-nr+1)$	r/N	0

Table 3: Properties of the posterior distribution after n repetitions, and its first and second moment in the limit $n \to \infty$.

A Code for estimation of apparatus efficiency- Bayesian perspective

```
import numpy as np
from math import factorial as fac
import matplotlib.pyplot as plt
import matplotlib

matplotlib.rcParams.update({'font.size': 16})

def posterior(p,N,r):
    return (fac(N+1)/(fac(r)*fac(N-r)))*(p**r)*((1-p)**(N-r))

p = np.linspace(0,1,10000)
N = np.array([100,100,100,100,100,1e3,1e3,1e3],dtype=int)
r = np.array([0,4,20,58,92,987,995,998],dtype=int)
E = np.array([0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0],dtype=float)

plt.figure(figsize=(10,5))
for i in range(len(N)):
    plt.plot(p,posterior(p,N[i],r[i]),label="N = %s, r= %s ; E = %s"%(str(N[i]),str(r[i]),str(E[i])),color=plt.cm.jet(i*40))

plt.legend(loc=9)
plt.xlabel("p")
plt.ylabel("posterior probability $P(p|N,r)$")
plt.ylim([0,40])
plt.xticks(np.linspace(0,1.0,11,endpoint=True))
plt.show()


#compute smallest interval

alpha = .68
confis = []

for i in range(len(E)):
    post = posterior(p,N[i],r[i])
    p_vals = p

    post = post/np.sum(post)

    print(np.sum(post))

    values = []

    for k in range(len(p)):
        values.append([post[k],p_vals[k]])

    values = sorted(values,key = lambda x: x[0]) #sort by probability content

    #print(values)

    confi_inter = [] #set of points inside confidence interval

    cum = 0#cumulative probability
    while(cum <= 1-alpha):
        cum_old = cum
        nxt = values.pop()
        cum += nxt[0]
        confi_inter.append(nxt[1])

    print("E=",E[i],min(confi_inter),max(confi_inter),np.round(1-cum_old,4))
    confis.append([min(confi_inter),max(confi_inter)])
```

B Code for estimation of apparatus efficiency - Frequentist perspective

```
import numpy as np
from scipy.special import binom

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

import time
import progressbar
matplotlib.rcParams.update({'font.size': 16})



def binom_dist(r,N,p):
    """
    returns the binomial distribution for
    sucess rate r, trial number N, probability p"""
    q = 1-p
    return binom(N,r)*(p**r)*(q**(N-r))


def binom_mode(N,p):
    if(p == 0):
        return 0,None
    elif(p== 1):
        return N,None
    elif(p != 0 and p!= 1 and ((N+1)*p)%1 != 0 ):
        return int((N+1)*p),None #floor value
    else: #(N+1)p is an integer => double mode
        return int((N+1)*p), int((N+1)*p)-1




def bar_plot(N,alpha,sample_points=500):
    """
    Generates a bar plot (Neymann construction)
    of the central interval for a given trial number N and confidence alpha
    sample points determine the resolution on p axis"""

    results = []

    plt.figure(figsize=(15,8))

    bar = progressbar.ProgressBar() #progress bar

    for i,p in enumerate(np.linspace(0,1,sample_points)):

        bar.update(100*i/sample_points)

        result = central_interval(binom_dist,N,p,alpha)

        if(result[1] != result[0]):
            single_entry = False
        else:
            single_entry = True


        new_point = []
        for i,res in enumerate(result):
            if(i==0):
                plt.scatter(res,p,color="r",s = 1.5)
                pass
            if(i==1 and not single_entry):
                plt.scatter(res,p,color="b",s = 1.5)

```

```
            results.append((p,result))

        plt.xlabel("r")
        plt.ylabel("p")
        plt.title("band plot for the Binomial distribution (N="+str(N)+") - central interval")
        plt.xlim([0,N])
        plt.ylim(([0,1]))
        plt.show()

        return results



        def confi_interval(neyman_constr,r):
        """
        finds the confidence interval given the plot data of a neyman construction
        for a given success rate r

        returns: p1,p2 the boundaries of the probability interval,
        and worst (minimum) cummulative probability obtained for this interval
        """

        my_p_list = [] #list to store all p values for r
        cummulate = [] #list of cummulative probabilities

        for result in neyman_constr:
            #print(result)
            if(r >= result[1][0] and r <= result[1][1]):
                my_p_list.append(result[0])
                cummulate.append(result[1][-1])


        return min(my_p_list), max(my_p_list), min(cummulate)
```

C Code for smallest interval

```
def smallest_interval(pdist,mode,N,p,alpha):
    """
    Calculates the smallest interval of a given probability distribution distribution pdist
    parameters:

    pdist   pdist(r,N,p) probability distribution to be observed
    mode    mode of the distribution
    N       number of trials
    p       probability parameter
    alpha   confidence level

    returns:

    r1      left boundary of smallest confidence interval
    r2      right boundary of smallest confidence interval
    F       cummulative probability (numerical value)
    """


    cummulative_prob = pdist(mode,N,p) #cummulative probability

    if(cummulative_prob >= 1-alpha):
        return mode,mode,cummulative_prob
    else:


        interval = [mode]
        ranked_probs = [pdist(r,N,p) for r in range(N)] #list with the ranked probabilities.
        #the index is the rank
```

```
30
31          z_ranked_probs = zip(range(N),ranked_probs) #zip the probabilities and r values
32
33          ranked_probs = list(sorted(z_ranked_probs,key = lambda x: x[1])) #convert to numpy array and sort to get ranking correct
34          #sorting key argument is the probability
35
36          #accumulate more r values to the interval until the cummulative
37          #probability goes to alpha/2
38          while((cummulative_prob < (1-alpha)) and (len(ranked_probs) > 0)):
39              new = ranked_probs.pop()
40              cummulative_prob += new[1]
41              interval.append(new[0])
42
43          #check for special case of equal ranking
44          if(len(ranked_probs) > 0):
45              check = ranked_probs.pop()
46
47              if(new[1] == check[1]):
48                  cummulative_prob += check[1]
49                  interval.append(check[0])
50
51
52          return min(interval),max(interval),cummulative_prob
```

D Code for central interval

```
1   def central_interval(pdist,N,p,alpha):
2       """
3       Calculates the central interval for a given probability distribution pdist
4       parameters:
5
6       pdist   pdist(r,N,p) probability distribution to be observed
7       N       number of trials
8       p       probability parameter
9       alpha   confidence level
10
11
12      returns:
13
14      r1      lower boundary of the central confidence interval
15      r2      upper boundary of the central confidence interval
16      cummulative probability (numerical value)
17      """
18
19      if(pdist(0,N,p) > alpha/2):
20          r1 = 0
21          sumval_1 = pdist(0,N,p)
22
23      else:
24
25          r_vals = []
26          for r in range(N+1):
27              sumval = 0
28              for i in range(r+1):
29                  sumval += pdist(i,N,p)
30
31              if(sumval <= alpha/2):
32                  r_vals.append(r)
33
34          r1 = max(r_vals) +1
35
36          sumval_1 = sumval
37
38      if(pdist(N,N,p) > alpha/2):
39          r2 = N
```

```
40            sumval = pdist(N,N,p)
41        else:
42
43            r_vals = []
44            for r in range(N+1):
45                sumval = 0
46                for i in range(r,N+1):
47                    #print("i",i)
48                    sumval += pdist(i,N,p)
49                if(sumval <= alpha/2):
50                    r_vals.append(r)
51
52            #case of just one point in the central interval
53            if(len(r_vals)> 0):
54                r2 = min(r_vals)-1
55            else:
56                r2 = r1
57
58        cummulative_prob = sumval_1+sumval
59
60        #return the central interval boundaries
61        return r1,r2,cummulative_prob
62
```

Poisson distribution

Thomas Baldauf

Winter semester 2017/18

1 Introduction

The poisson distribution is very useful in physics. When a detector measures a certain number of counts, e.g. photons from an astronomical light source, we do not know the number of ”trials”, i.e. how many particles are impinging in the detector. We just know the ”success rate”, i.e. how many particles were counted by the detector. However, we can assume that the number of trials is large for sufficiently bright sources. Another application would be the Large Hadron Collider, where particles pass through an accelerator with huge rates.
We got to know the Poisson distribution as a limit of the Binomial Distribution

$$P(n|N,p) = \frac{N!}{n!(N-n)!} p^n (1-p)^{N-n} \tag{1}$$

where we introduce an event frequency $\nu = Np$ and let N go to infinity, such that

$$\frac{N!}{(N-n)!} \to N^n \tag{2}$$

The Poisson distribution is

$$P(n|\nu) = \frac{e^{-\nu}\nu^n}{n!} \tag{3}$$

The Poisson distribution has expectation $\mathrm{E}[n] = \nu$ and also variance $\mathrm{Var}[n] = \nu$. The mode is

$$n^* = \lfloor \nu \rfloor, \nu \notin \mathbb{Z} \tag{4}$$

$$n_1^* \nu, n_2^* = \nu - 1, \nu \in \mathbb{Z}. \tag{5}$$

There exist two modes if ν is an integer.
The distribution can also be derived by looking at a constant rate process and integrating over all possible times at which events can occur (see lecture script).

1.1 Neyman confidence interval

We have already seen an example for a band plot of data following a Binomial distribution. But we can also find such a band plot for the Poisson distribution. Like for Binomial data, we construct the Neyman Confidence level. Again, we have to take care that we define an appropriate confidence interval (e.g. central or smallest). We can resemble a result from the lecture (figure 1) using code A.

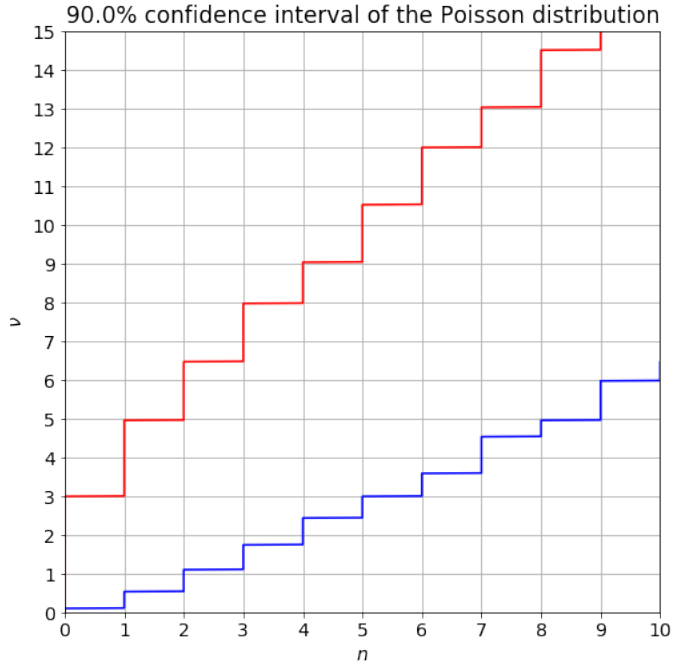


Figure 1: 90% Central interval constructed with Neyman construction for the Poisson distribution. If we observe a certain n_D, we can just read out the confidence interval for ν along the vertical axis.

1.2 Feldman-Cousins interval

The Feldman-Cousins interval is alsmost the same like the smallest or central interval. The probabilities are ranked according to a certain scheme and then accumulated until a desired probability content is reached. However, the Feldman-Cousins interval is somewhat special as it uses the ranking

$$r = \frac{P(n|\mu = \lambda + \nu)}{P(n|\hat{\mu})} \tag{6}$$

where $\hat{\mu}$ is an optimal value, say, the mode.
We have to take care with band plots and because we might get the impression that ”poorer experiments produce stronger results”, for example.

2 Tasks

2.1 Standard deviation

Consider the function $f(x) = \frac{1}{2}\exp(-|x|)$ for $-\infty < x < \infty$.

1. Find the mean and standard deviation of x
2. Compare the standard deviation with the FWHM (Full Width Half Maximum)
3. What probability is contained in the ± 1 standard deviation interval around the peak?

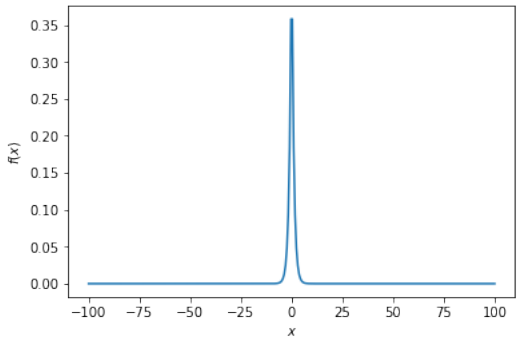


Figure 2: Plot of the function $f(x)$.

1. The mean is the expectation of x, so

$$\mu = \mathrm{E}[x] = \int_{-\infty}^{\infty} x f(x)\,\mathrm{d}\,x = \int_{-\infty}^{\infty} x\frac{1}{2}\exp(-|x|)\,\mathrm{d}\,x \tag{7}$$

The integral can be split into a part on the negative x axis and a part on the positive x axis. As we can already see from the plot, the two integrals vanish.

$$\int x e^{x}\,\mathrm{d}\,x = e^{x}(x-1) \tag{8}$$

We have to be careful inserting the infinite limits: The exponential function drops faster than any polynomial. Thus, we get

$$\int_{-\infty}^{0} x e^{x}\,\mathrm{d}\,x + \int_{0}^{\infty} x e^{-x}\,\mathrm{d}\,x = -1 + 1 = 0 \tag{9}$$

And thus

$$\mu = \mathrm{E}[x] = 0. \tag{10}$$

The variation is

$$\mathrm{Var}[x] = \mathrm{E}[x^2] - \mathrm{E}[x]^2 \tag{11}$$

we can use the integral (repeated partial integration)

$$\int x^2 e^x \,\mathrm{d}\, x = -e^{-x}(x^2 + 2x + 2) \tag{12}$$

to calculate

$$\mathrm{E}[x^2] = \frac{1}{2}\int_{-\infty}^{\infty} x^2 e^{-|x|} \,\mathrm{d}\, x = \frac{1}{2}\int_{-\infty}^{0} x^2 e^{x} \,\mathrm{d}\, x + \frac{1}{2}\int_{0}^{\infty} x^2 e^{-x} \,\mathrm{d}\, x \tag{13}$$

$$= -\int_{0}^{\infty} x^2 e^{-x} \,\mathrm{d}\, x = 2 \tag{14}$$

Such that

$$\sigma = \sqrt{\mathrm{Var}} = \sqrt{2} \approx 1.41 \tag{15}$$

2. The hal f maximum defined as $f(x_{\mathsf{FWHM}}) = \frac{\max f}{2}$. The maximum of $f(x)$ is defined where the derivative is zero. Unfortunately the function is not differentiable at the origin, where it takes its maximum, as we can see from the plot. We conclude from the monotony of the function that the maximum is at the origin. That is

$$f(0) = \frac{1}{2}\exp 0 = \frac{1}{2} \tag{16}$$

The half of the maximum is $1/4$, and this value is reached for

$$f(x) = 1/4 \Leftrightarrow \frac{1}{4} = \frac{1}{2}\exp(-|x|) \tag{17}$$

$$\Rightarrow x = \pm \ln(2) \tag{18}$$

The FWHM is therefore $2\ln(2) \approx 1.37$. This is almos the same value as the standard deviation. In comparison to the standard deviation:

$$\left|\frac{|\mathsf{FWHM} - \sigma|}{\sigma}\right| \approx \left|\frac{1.37 - 1.41}{1.41}\right| \approx 0.0197 \approx 2\% \tag{19}$$

3. We integrate

$$\int_{-\sqrt{2}}^{\sqrt{2}} \frac{1}{2} e^{-|x|} \, \mathrm{d}\, x = 1 - e^{-\sqrt{2}} \approx 0.76 = 76\% \tag{20}$$

2.2 Poisson model

9 events are observed in an experiment modeled with a Poisson probability distribution.

1. What is the 95% probability lower limit on the Poisson expectation value ν? Take a flat prior for your calculations.
2. What is the 68% confidence level interval for ν using Neyman construction and the smallest interval definition?

1. In the lecture, we discussed Bayes' rule, using a Poisson likelihood and a flat prior, $P_0 = C$. The posterior probability is again a Poisson distribution, but with ν as a parameter:

$$P(\nu|n) = \frac{P(n|\nu) P_0(\nu)}{\int_0^\infty P(n|\nu) P_0(\nu) \, \mathrm{d}\, \nu} = \frac{\nu^n e^{-\nu} P_0(\nu)}{\int_0^\infty \nu^n e^{-\nu} P_0(\nu) \, \mathrm{d}\, \nu} \approx \frac{e^{-\nu} \nu^n}{n!} \tag{21}$$

This holds for very small C, i.e. very broad priors. The mode of the posterior is at $\nu^* = n$. We can estimate the 95% probability lower/upper limit by integrating the posterior probability up to that value:

$$0.95 = F(\nu|n=9) \Rightarrow \int_0^{\nu_{95}} \frac{e^{-\nu} \nu^9}{9!} \, \mathrm{d}\, \nu = 0.95 \tag{22}$$

The cummulative probability is

$$F(\nu|n) = 1 - e^{-\nu} \sum_{i=0}^{n} \nu^i / i! \tag{23}$$

Solving this equation numerically yields

$$\nu_{95} \approx 15.7 \tag{24}$$

So for all values $\nu < 15.7$ we are within the 95% confidence level interval. The lower limit is

$$0.95 = F(\nu|n=9) \Rightarrow \int_{\nu_{95}}^{\infty} \frac{e^{-\nu} \nu^9}{9!} \, \mathrm{d}\, \nu = 0.95 \tag{25}$$

This yields 5.4

2. We use Python code A to do the dirty work for us. The output of the smallest interval Neyman construction is shown in figure 3. As we can read from the graph, the confidence interval for ν is where the values are still included within a vertical slice at $n = 9$, i.e. from 6.5 to 13.3. Fortunately, this is consistent with our upper boundary estimate from the previous task.

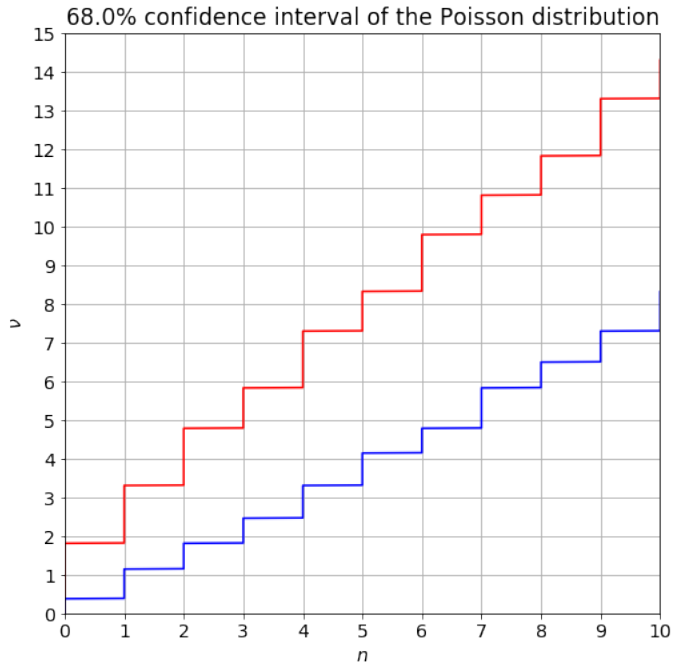


Figure 3: Neyman construction for 68% smallest interval.

2.3 Poisson model with background

Repeat the previous exercise, assuming you had a known background of 3.2 events.

- Find the Feldman-Cousins 68% Confidence Level interval
- Find the Neyman 68% Confidence Level interval
- Find the 68% Credible interval for ν

A result from the lecture was that for two different Poisson processes, e.g. for background and signal, the probability distribution is

$$P(n|\nu_S, \nu_B) = \sum_{n_S=0}^{n} P(n_S|\nu_S) P(n - n_S|\nu_B) \tag{26}$$

Evaluating the sum leads to the expression

$$P(n|\nu_B, \nu_S) = \frac{e^{-(\nu_S+\nu_B)}(\nu_s+\nu_B)^n}{n!} \tag{27}$$

That is the probability of observing n events, given that both processes are present at a time.

1. The 68% Feldman-Cousins interval was plotted using code B. It is shown in table 1. If we estimate ν from our previous calculations, we can easily insert λ, the additional background, and get a confidence interval of around $(4, 10)$.

n	$P(n,u)$	$\hat{\mu}$	r	$rank$	$F_R(n\|u)$
0	0.00075	4.0	0.04076	15	0.99272
1	0.00538	4.0	0.07337	14	0.99197
2	0.01935	4.0	0.13207	12	0.97798
3	0.04644	4.0	0.23773	10	0.94187
4	0.0836	4.0	0.42791	7	0.81476
5	0.12038	5.0	0.68606	5	0.65413
6	0.14446	6.0	0.89936	3	0.42677
7	0.14859	7.0	0.9972	1	0.14859
8	0.13373	8.0	0.95802	2	0.28231
9	0.10698	9.0	0.81197	4	0.53375
10	0.07703	10.0	0.61567	6	0.73116
11	0.05042	11.0	0.42233	8	0.86518
12	0.03025	12.0	0.2645	9	0.89543
13	0.01675	13.0	0.15239	11	0.95863
14	0.00862	14.0	0.0813	13	0.98659
15	0.00414	15.0	0.04038	16	0.99685
16	0.00186	16.0	0.01876	17	0.99871
17	0.00079	17.0	0.00819	18	0.9995
18	0.00032	18.0	0.00337	19	0.99982
19	0.00012	19.0	0.00131	20	0.99994

Table 1: Feldman-Cousin table for $\nu = 7, \lambda = 3.2$

2. The 68% Neyman smallest interval is shown in figure 4. The lines are shifted downwards, i.e., for a certain signal from the source (not noise), we have higher count rates in general, which makes sense, as the noise is always an additional signal.

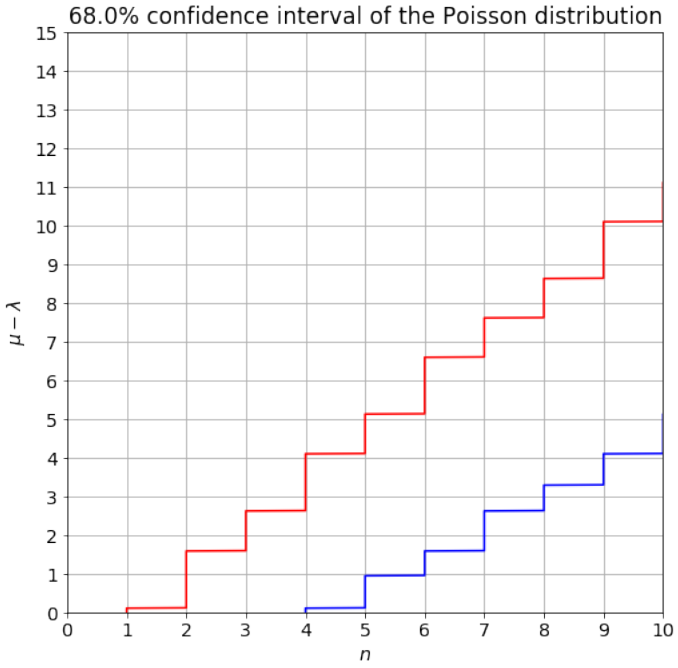


Figure 4: 68% Neyman band plot, with additional noise level $\lambda = 3.2$.

For $n = 9$, the lower boundary for ν is around 3.3 and the upper boundary is around 10.1

3. The 68% Credible interval is about $(4, 9)$.

2.4 Binned Poisson probability

In this problem, we look at the relationship between an unbinned likelihood and a binned Poisson probability. We start with a one dimensional density $f(x|\lambda)$ depending on a parameter λ and defined and anormalized in a range $[a, b]$. n events are measured with x values, $x_i, i = 1, \ldots, n$. The unbinned likelihood is defined as the product of the desities

$$\mathcal{L} = \prod_{i=1}^{n} f(x_i|\lambda). \tag{28}$$

Now we consider that the interval $[a, b]$ is divided into K subintervals (bins). Take for the expectation in bin j

$$v_j = \int_{\Delta_j} f(x|\lambda)\,\mathrm{d}\,x \tag{29}$$

where the integral is over the x range in interval j, which is denoted as Δ_j. Define the probability of the data as the product of the Poisson probabilities in each bin. We consider the limit $K \to \infty$ and, if no two measurements have exactly the same value of x, then each bin will have either $n_j = 0$ or $n_j = 1$ events. Show that this leads to

$$\lim_{K\to\infty} \prod_{j=1}^{K} \frac{e^{-\nu_j} \nu_j^{n_j}}{n_j!} \sim \prod_{i=1}^{n} f(x_i|\lambda)\Delta \tag{30}$$

where Δ is the size of the interval in x assumed fixed for all j. I.e., the unbinned likelihood is proportional to the limit of the product of Poisson probabilities for an infinitely fine binning.

We can start with the left hand side

$$\lim_{K\to\infty} \frac{e^{-\nu_j} \nu_j^{n_j}}{n_j!} = \lim_{K\to\infty} \frac{e^{-\sum_{j=1}^{K} \nu_j} \prod_{j=1}^{K} \nu_j^{n_j}}{1} = \lim_{K\to\infty} e^{-\sum_{j=1}^{K} \nu_j} \prod_{j=1}^{K} \nu_j^{n_j} \tag{31}$$

where we used $\prod_{j=1}^{K} n_j! = 1$ because $0! = 1! = 1$, and we used the rule for products of exponential functions. We can write a new index

$$\{i\} = \{j|n_j = 1\}, |\{i\}| = n \tag{32}$$

with n elements. Then the term $\nu_j^{n_j}$ becomes ν_i if $j \in \{i\}$ or 1 otherwise. We obtain

$$\lim_{K\to\infty} e^{-\sum_{j=1}^{K} \nu_j} \prod_{i=1}^{n} \nu_i. \tag{33}$$

We can write the limit into the argument of the exponential function, and therefore

$$\lim_{K\to\infty} \sum_{j=1}^{K} \int_{\Delta_j} f(x|\lambda)dx = 1 \tag{34}$$

where we assume that f is normalized to 1. Then, equation 33 becomes

$$\lim_{K\to\infty} \frac{1}{e} \prod_{i=1}^{n} \int_{\Delta_i} f(x_i|\lambda)dx. \tag{35}$$

As the inteval Δ_j goes to zero with $K \to \infty$, we can write the integral just as $f(x_i|\lambda)\Delta_i$, so

$$\lim_{K\to\infty} \frac{1}{e} \prod_{i=1}^{n} \int_{\Delta_i} f(x_i|\lambda)dx = \frac{1}{e} \prod_{i=1}^{n} f(x_i|\lambda)\Delta \tag{36}$$

where we substituted $\Delta_i = \Delta$. That is the right hand side of equation 30. End of proof.

2.5 Thinned Poisson process

We consider a thinned Poisson process. Here we have a random number of occurences, N, distributed according to a Poisson distribution with mean ν. Each of the N occurences, X_n, can take on values of 1, with probability p, or 0, with probability $(1-p)$. We want to derive the probability distribution for

$$X = \sum_{n=1}^{N} X_n. \tag{37}$$

Show that the probability distribution is given by

$$P(X) = \frac{e^{-\nu p}(\nu p)^X}{X!} \tag{38}$$

This scenario can be seen as e.g. a source which emits particles with constant rate, but there is a process between the source and the detector that absorbs the particles with a probability $(1-p)$, such that the detector does not count the particles, though emitted. If we know the nature of the process, i.e. its probability p, we can still make assumptions about the source, by including this probability. Let us classify two types of particles: those which are detected (d) and those which are lost (l). It can be expressed as a conditional probability:

$$P(n(d) = X) = \frac{P(n(l) = N - X, n(d) = X)}{P(n(l) = N - X | n(d) = X)} \tag{39}$$

This is the probability for $N - x$ lost particles, and X detected particles, divided by the probability that $N - x$ particles are lost, if we already know that X particles are detected. This is

$$P(n = X) = \frac{e^{-\nu l}\frac{(\nu l)^{N-X}}{(N-X)!} \cdot e^{-\nu(p-l)}\frac{(\nu(p-l)^X)}{X!}}{\binom{N}{X}\left(\frac{l}{p}\right)^{N-X}\left(1-\left(\frac{l}{p}\right)\right)^X} \tag{40}$$

$$= \frac{e^{-\nu p}(\nu p)^N}{N!} \tag{41}$$

where p and $p-l$ represent the probabilities of a particle to be emitted, and detected (i.e. emitted and passed the "absorption" process in between).

A Code for the Neyman band plot

```
def Neyman_table(mu,verbose=False):
    """
    Neyman construction table for (fixed) mu
    verbose = True prints a table with important data
    """
    table = []

    F = 0

    for n in n_range:
        F+= poisson(mu,n) #cummulative probability
        table.append([n,poisson(mu,n),F])

    table = list(sorted(table,key= lambda x: -x[1])) #sort by rank = probability

    #append rank and cummulative probability to table

    F = 0
    for i in range(len(table)):
        table[i].append(int(i)+1) #rank
        F += table[i][1] #sum up probabilities
        table[i].append(F) #add cummulative probability to table


    table = list(sorted(table,key= lambda x: x[0])) #sort by n again


    #print results
    if(verbose):
        print_table(table,"| $n$ | $P(n,\nu)$  | $F(n|\nu)$ | $rank$ | $F_R(n|\nu)$")

    #return the table as list of lists
    return table




def Neyman_central_interval(mu,alpha):
    """
    returns the Neyman constructed smallest interval for the given parameters.
    return values: n1,n2,F (cummulative probability)
    alpha is the confidence level
    """

    table = Neyman_table(mu,verbose=False) #create a Feldman-Cousins table

    #search for the smallest interval
    table = list(sorted(table,key= lambda x: x[3])) #sort by rank

    n_vals = [table[0][0]]     #start with n_start

    i = 0
    F = table[i][4]

    while(i+1 < len(table) and F <= 1-alpha):
        i+= 1
        F = table[i][4]
        n_vals.append(table[i][0])

    i+= 1

    n_start = min(n_vals)
    n_stop = max(n_vals)

    return n_start,n_stop,F

```

```
67
68  def Neyman_smallest_interval(mu,alpha):
69      """
70      returns the Neyman constructed smallest interval for the given parameters.
71      return values: n1,n2,F (cummulative probability)
72      alpha is the confidence level
73      """
74
75      table = Neyman_table(mu,verbose=False) #create a Feldman-Cousins table
76
77      #search for the smallest interval
78      table = list(sorted(table,key= lambda x: x[3])) #sort by rank
79
80      n_vals = [table[0][0]]     #start with n_start
81
82      i = 0
83      F = table[i][4]
84
85      while(i+1 < len(table) and F <= 1-alpha):
86          i+= 1
87          F = table[i][4]
88          n_vals.append(table[i][0])
89
90      i+= 1
91
92      n_start = min(n_vals)
93      n_stop = max(n_vals)
94
95      return n_start,n_stop,F
96
97
98  def band_plot_Neyman(alpha,n_plot_range=[0,16],mu_plot_range=[0,10]):
99      """
100     creates a band plot with the confidence intervals of mu and the confidence level alpha
101     for a poisson distribution with fixed background parameter lambda
102     """
103
104
105     #list of points to be plotted
106     left_points = []
107     right_points = []
108     m_vals = []
109
110
111     #iterate through various N
112     for m in mu_range:
113
114         n_min,n_max,F = Neyman_smallest_interval(m,alpha)
115
116         #append the boundaries to the points to be plotted
117         left_points.append(n_min)
118         right_points.append(n_max)
119         m_vals.append(m)
120
121
122     plt.figure(figsize=(8,8))
123     plt.plot(left_points,m_vals,color="r")
124     plt.plot(right_points,m_vals,color="b")
125     plt.grid()
126     plt.xlabel("$n$")
127
128     plt.ylabel("$\mu$")
129     plt.ylabel(r"$\nu$")
130
131     plt.xticks(np.arange(max(n_range)+1))
132     plt.yticks(np.arange(max(mu_range)+1))
133
134     plt.xlim(n_plot_range)
135     plt.ylim(mu_plot_range)
```

```
136
137         plt.title(str(100*(1-alpha))+"% confidence interval of the Poisson distribution")
138         plt.show()
139
140         #return plot data
141         return left_points,right_points,m_vals
142
143
144     alpha = 1-.68
145
146     n_plot_range = [0,10]
147     mu_plot_range = [0,15]
148     left_point,right_points,m_vals = band_plot_Neyman(alpha,n_plot_range,mu_plot_range)
```

B Code for the Feldman-Cousins table

```
1   def poisson(mu,n):
2       """ The poisson distribution P(n,mu)"""
3       return (np.exp(-mu)*(mu**n))/(fac(n))
4
5   def poisson_mode(mu):
6       """ Calculates the mode of a poisson distribution"""
7       if(mu %1 != 0):
8           return int(np.floor(mu))
9       else:
10          return max(0,np.floor(mu-1))
11
12  def Feldman_Cousins_table(mu,lambd,verbose=False):
13      """
14      Feldman_cousins construction table for (fixed) mu  + lambda (noise)
15      verbose = True prints a table with important data
16      """
17      table = []
18
19      F = 0
20
21      for n in n_range:
22          mu_hat = np.round(max(mu-lambd,poisson_mode(n+1)),1)
23          r = poisson(mu,n)/poisson(mu_hat,n)#Feldman-cousins ranking
24          F+= poisson(mu-lambd,n) #cummulative probability
25          table.append([n,poisson(mu,n),mu_hat,r])
26
27      table = list(sorted(table,key= lambda x: -x[3])) #sort by rank = probability
28
29      #append rank and cummulative probability to table
30
31      F = 0
32      for i in range(len(table)):
33          table[i].append(int(i)+1) #rank
34          F += table[i][1] #sum up probabilities
35          table[i].append(F) #add cummulative probability to table
36
37      table = list(sorted(table,key= lambda x: x[0])) #sort by n again
38
39
40      #print results
41      if(verbose):
42          print_table(table,"| $n$ | $P(n,\nu)$  |$\hat{\mu}$  | $r$ |  $rank$ | $F_R(n|\nu)$")
43
44      #return the table as list of lists
45      return table
46
47
48  def Cousins_smallest_interval(mu,lambd,alpha):
49      """
```

```
50        returns the Feldman-Cousins constructed smallest interval for the given parameters.
51        return values: n1,n2,F (cummulative probability)
52        alpha is the confidence level
53        """
54
55        table = Feldman_Cousins_table(mu,lambd,verbose=False) #create a Feldman-Cousins table
56
57        #search for the smallest interval
58        table = list(sorted(table,key= lambda x: x[3])) #sort by rank
59
60        n_vals = [table[0][0]]      #start with n_start
61
62        i = 0
63        F = table[i][4]
64
65        while(i+1 < len(table) and F <= 1-alpha):
66            i+= 1
67            F = table[i][4]
68            n_vals.append(table[i][0])
69
70        i+= 1
71
72        n_start = min(n_vals)
73        n_stop = max(n_vals)
74
75        return n_start,n_stop,F
```

Gaussian Probability Distribution Function

Thomas Baldauf

Winter semester 2017/18

1 Introduction

We discovered that the Gaussian distribution can be expressed as the limit of the Binomial distribtion, as well as the Poisson distribution. We also saw Gauss' derivation of the Gauss distribution, which was based on the idea of finding a function $\psi(x_i - \mu)$ which gives a maximum for

$$f(x|\mu) = \prod_{i=1}^{n} \phi(x_i - \mu) \tag{1}$$

where μ is the mean value of the values x_i,

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{2}$$

We showed that the Gauss probability distribution fulfils this criterion. The Gauss distribution is defined as

$$\mathcal{G}(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x-\mu)^2}{\sigma^2}\right) \tag{3}$$

in one dimension. We discovered also some basic properties, like the probability content of 68.8% within one standard deviation. We saw that we can easily handle the sum of Gauss distributed data, and add errors from Gauss distributed quantities.

1.1 Multivariate Gauss function

We defined the multivariate Gauss distribution for some (generally correlated) variables $x_1, \ldots x_n$ as

$$P(x_1, \ldots, x_n) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \tag{4}$$

where Σ is called the covariance matrix. The covariance is defined as

$$\begin{aligned} \mathrm{cov}[x_i, x_j] &= \mathrm{E}[(x_i - \mathrm{E}[x_i])(x_j - \mathrm{E}[x_j])]_{P(x_i, x_j, \dots)} && (5) \\ &= \mathrm{E}[x_i x_j]_{P(x_i, x_j)} - \mathrm{E}[x_i]_{P_{x_i}(x_i)} \, \mathrm{E}[x_j]_{P_{x_j}(x_j)} && (6) \\ \Rightarrow \Sigma_{ij} &= \mathrm{cov}[x_i, x_j]. && (7) \end{aligned}$$

In this notation, $P_{x_k}(x_k)$ denotes the distribution P marginalized to coordinate k. We obtain the variance as the sum of covariances

$$\mathrm{Var}(x) = \mathrm{Var}(\sum_i x_i) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{cov}(x_i, x_j) = \sum_{i=1}^{n} \mathrm{Var}(x_i) + 2 \sum_{i<j}^{n} \mathrm{cov}(x_i, x_j) \tag{8}$$

We define a correlation coefficient ρ as

$$\rho_{xy} = \mathrm{cov}(x, y)/(\sigma_x \sigma_y) \tag{9}$$

and we get for Gaussian distributed variables

$$\mathrm{Var}(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} \sigma_i \sigma_j = \sum_{i=1}^{n} \sigma_i^2 + 2 \sum_{i<j}^{n} \rho_{ij} \sigma_i \sigma_j \tag{10}$$

1.2 Central Limit Theorem

The central limit theorem (CLT) states that, for "well-behaving" probability density functions, where higher moments exist and are finite, the distribution of the mean values of a quantity x is approximately Gaussian if x is measured n times and n becomes large. We define

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{11}$$

The expectation value of $\bar{x}$ is called μ,

$$\mathrm{E}[\bar{x}] = \mu, \tag{12}$$

and we further look at the probability distribution of

$$z = \bar{x} - \mu = \sum_{i=1}^{n} \frac{x_i - \mu}{n} \tag{13}$$

In the limit of large n, we then get

$$P(z) = \frac{1}{\sqrt{2\pi}\sigma_z} \exp(-\frac{1}{2}\frac{z^2}{\sigma_z^2}) \tag{14}$$

where $\sigma_z = \sigma_x/n$. This result can be obtained by looking at the characteristic function of $P(z)$ and expanding it up to quadratic contributions.
If we shift z by μ and if z is just the sum of n independent quantities $z' = \sum_i^n x_i$, we get

$$P(z') = \frac{1}{\sqrt{2\pi}\sigma_{z'}} \exp\left(-\frac{1}{2}\frac{(z-n\mu)^2}{\sigma_{z'}^2}\right) \tag{15}$$

where $\sigma_{z'} = \sqrt{n}\sigma_x$.
Note that here we assme that these terms do not "explode", which means that higher moments exist and are finite.

1.3 Frequentist Analysis with Gaussian distributed variables

We got to know the Neyman and Feldman-Cousins construction. For Gaussian distributed data, we can run into "non-intuitive results" using the Neyman construction for confidence level intervals. We can think of examples where we see that there is no μ for which the data lies within the $1-\alpha$ confidence interval of the Neyman construction. However, we can go a different way, and defined the Feldman-Cousins r-value

$$r = \frac{\mathcal{G}(x|\mu,\sigma)}{\mathcal{G}(x|\hat{\mu})} = \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right), x \geq 0 \tag{16}$$

and

$$r = \exp(x\mu/\sigma^2 - \mu^2/2\sigma^2), x < 0 \tag{17}$$

$\hat{mu}$ denotes the mode of $\mathcal{G}(x|\mu,\sigma)$, and we put the boundary $\mu \geq 0$, such that for $x < 0$, $\hat{\mu} = 0$.

1.4 Bayesian Analysis with Gaussian distributed variables

If we look at the distribution of μ of a Gauss pdf using Bayes' Theorem, we get, assuming a flat prior for μ,

$$P(\mu|x,\sigma) = \frac{P(x|\mu,\sigma)P_0(\mu)}{\int P(x|\mu,\sigma)P_0(\mu)\,\mathrm{d}\,\mu} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{18}$$

The distribution for μ is again Gaussian. This is useful if we take measurements with a certain resolution σ and want to perform a Bayesian analysis on the meausred value μ. If we consider different σ_i for different meausrements i, the distribution of μ stays Gaussian, and we can express it in terms of the weighted sum of the measured values, where the weighting factor is the inverse squared variance. Like always in Bayesian analysis, we have to take care which prior we chose. In case of doubt, a flat prior can be taken.

2 Tasks

2.1 Central Limit Theorem for Exponential Distribution

In this problem, you try out the Central Limit Theorem (CLT) for a case where the conditions under which it was derived apply, and a case under which the conditions do not apply.

(a) In this exercise, try out the CLT on the exponential distribution. First, derive what parameters of a Gauss distribution you would expect from the mean of n samples taken from the exponential distribution

$$p(x) = \lambda \exp(-\lambda x). \tag{19}$$

Then, try out the CLT for at least 3 different choices n and λ and discuss the results. To generate random numbers according to the exponential distribution, you can use

$$x = -\frac{\ln(U)}{\lambda} \tag{20}$$

where U is a uniformly distributed random number between $[0, 1)$.

(b) Now try out the CLT for the Cauchy distribution.

$$f(x) = \frac{1}{\pi\gamma} \frac{\gamma^2}{(x - x_0)^2 + \gamma^2} \tag{21}$$

Argue why the CLT is not expected to hold for the Cauchy distribution. You can generate random numbers from the Cauchy distribution by setting

$$x = \gamma \tan(\pi U - \pi/2) + x_0 \tag{22}$$

try $x_0 = 25$ and $\gamma = 3$ and plot the distribution for x. Now take 100 samples and plot the distribution mean. Discuss the results.

(a) We calculate the expected value and the standard deviation for the exponential distribution.

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} x\lambda \exp(-\lambda x)\,\mathrm{d}\,x = 0 + \int_{-\infty}^{\infty} \exp(-\lambda x) = \frac{1}{\lambda} \tag{23}$$

which was solved using partial integration, and

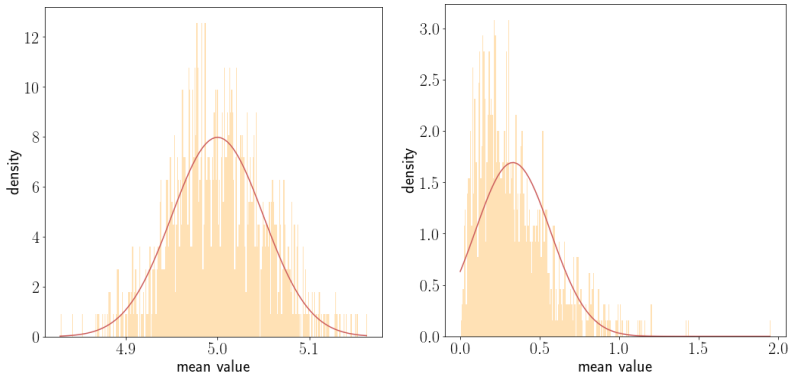
$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} x^2\lambda \exp(-\lambda x)\,\mathrm{d}\,x = \int_{-\infty}^{\infty} 2x \exp(-\lambda x)\,\mathrm{d}\,x = \int_{-\infty}^{\infty} \frac{2}{\lambda} \exp(-\lambda x)\,\mathrm{d}\,x = \frac{2}{\lambda^2} \tag{24}$$

which can be obtained by two-fold partial integration. Then, we get

$$\mathrm{Var} = \mathbb{E}[x^2] - (\mathbb{E}[x])^2 = \frac{2}{\lambda^2} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2} \tag{25}$$

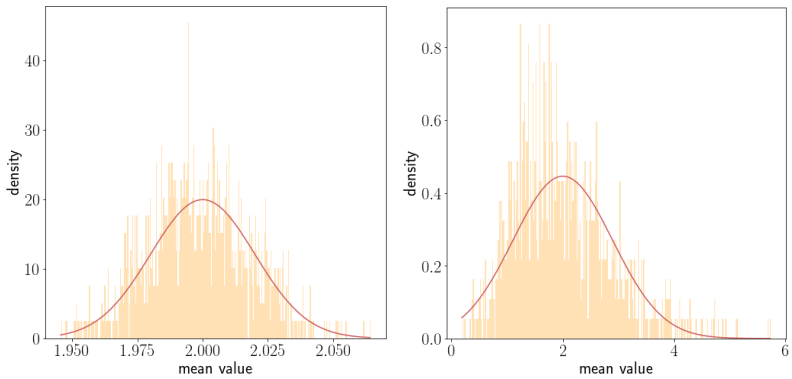
Therefore, we would expect a Gauss distribution with mean $\mu = 1/\lambda$ and standard deviation $\sigma = 1/\lambda$. For n samples, the mean becomes the standard deviation becomes $\sigma_n^2 = (1/\lambda^2) \cdot (1/N)$. We can write a small python program to plot all the data for given input parameters. The results are shown in figure 1. The data tables given in the figure show that the mean and standard deviation have almost no deviation from their theoretical value from CLT if we go to a large number of random draws. The plot of the Gauss curve ”fits” the data reasonably well. In contrast, where only two or five random draws are taken per experiment, the CLT starts to deviate more strongly from the histogram.

left (i) ensemble size 1000 , sample size 10000, $\lambda = 0.2$, 300 bins /
right (ii) ensemble size 1000, sample size 2, $\lambda = 3$, 300 bins



(i) $\mu_{\text{CLT}} = 5.0, \mu_e = 5.0011, \sigma_{\text{CLT}} = 0.05, \sigma_e = 0.0517$
(ii) $\mu_{\text{CLT}} = 1/3, \mu_e = 0.3347, \sigma_{\text{CLT}} = 0.2357, \sigma_e = 0.2334$

left (iii) ensemble size 1000, sample size 10000, $\lambda = 0.5$, 300 bins
right (iv) ensemble size 1000, sample size 5, $\lambda = 0.5$, 300 bins



(iii) $\mu_{\text{CLT}} = 2.0, \mu_e = 1.998, \sigma_{\text{CLT}} = 0.02, \sigma_e = 0.0195$
(iv) $\mu_{\text{CLT}} = 2.0, \mu_e = 1.998, \sigma_{\text{CLT}} = 0.02, \sigma_e = 0.0195$

Figure 1: Comparison of a randomly picked ensemble dataset from the exponential distribution (orange histogramm) with the Gauss distribution (red line) for different parameters. The theoretical values from the Central Limit Theorem (substript CLT) and from the ensemble (subscript e) are given below the plots. Mind the different scales on the horizontal axis.

(b) The Cauchy distribution has no mean and standard deviation (i.e. μ and σ do not exist). This arises from the fact that the characteristic function of the Cauchy distribution is not differentiable at the origin. Thus, we do not expect the CLT to hold. Figure 2 shows a plot of the Cauchy

distribution. The Cauchy distribution has a sharp peak and long tailed-out side wings. These are resembled in the histogram plot of the sample means.

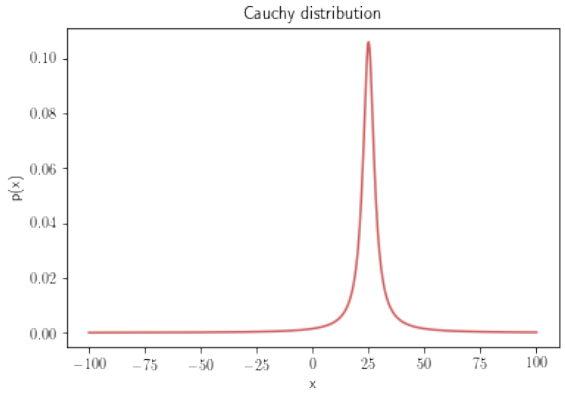


Figure 2: A plot of the Cauchy distribution for $\gamma = 3, x_0 = 25$.

Figure 3 (left side) shows the distribution of the mean for three sample runs. The values are calculated for $x_0 = 25, \gamma = 3, n = 100$, an ensemble size of 1000 (the number of bin is again 300). Unlike the exponential distribution, the Cauchy distribution does not come closer to a Gauss shape when n is increased. For an $n = 500$, we get a similar histogram like $n = 2000$ and $n = 100$ right side). If we draw more samples, then the long tails of the distribution start to fill up (see figure 3 right side). Also, the peak becomes sharper. The Cauchy shape manifests more and more.

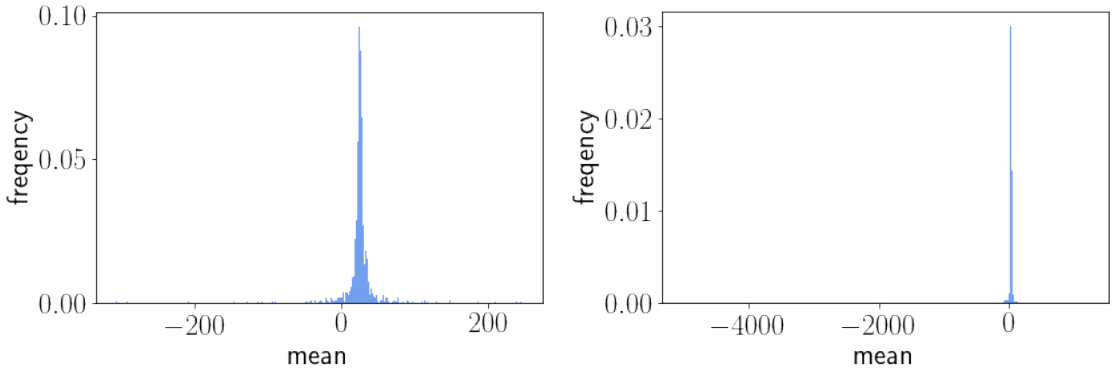


Figure 3: Histogram for $n = 100$ (left) and $n = 500$ (right). The more samples are drawn from the distribution, the more the Cauchy shape manifests.

2.2 Bivariate Gauss Distribution

With a plotting program, draw contours of the bivariate Gauss function

$$P(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right)\right) \tag{26}$$

a) $\mu_x = 0, \mu_y = 0, \sigma_x = 1, \sigma_y = 1, \rho_{xy} = 0$
b) $\mu_x = 1, \mu_y = 2, \sigma_x = 1, \sigma_y = 1, \rho_{xy} = 0.7$
c) $\mu_x = 1, \mu_y = -2, \sigma_x = 1, \sigma_y = 2, \rho_{xy} = -0.7$

Python with matplotlib was chosen as a plotting program. Results are shown in figure 4. The plots show that a correlation/anticorrelation can be easily seen by a tilt ellipse shape of the bivariate Gaussian.

2.3 Bivariate Gauss II

Show that the pdf can be written in the form

$$P(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right)\right) \tag{27}$$

Show that for $z = x - y$ and for x, y following the bivariate Gaussian distribution, the resulting distribution for z is a Gaussian probability distributtion with

$$\mu_z = \mu_x - \mu_y \tag{28}$$
$$\sigma_z^2 = \sigma_x^2 + \sigma_y^2 - 2\rho\sigma_x\sigma_y \tag{29}$$

(a) We have

$$P(x_1, x_2, \dots x_n) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{30}$$

That makes sense as we require a quadratic form in the argument of the exponential function. We now have the special two-dimensional case, where

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} \Rightarrow |\Sigma| = \sigma_x^2\sigma_y^2 - \sigma_{xy}^2 \tag{31}$$

With the definition of the correlation

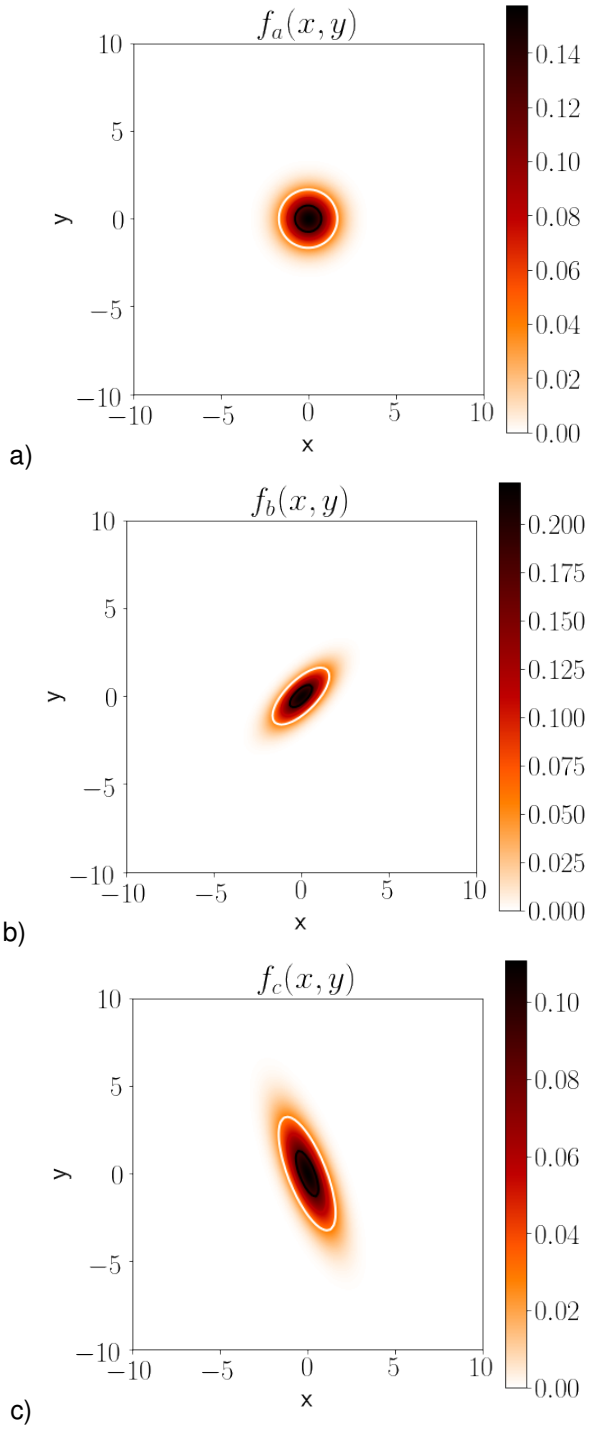


Figure 4: Bivariate Gauss functions for different parameters.

$$\rho_{xy} = \mathrm{cov}(x,y)/(\sigma_x\sigma_y) = \Sigma_{xy}/(\sigma_x\sigma_y) \Rightarrow \Sigma_{xy} = \rho_{xy}\sigma_x\sigma_y \Rightarrow \sigma_{xy}/(\sigma_x\sigma_y) = \rho_{xy} \tag{32}$$

we can re-write 30:

$$P(x,y) = \frac{1}{(2\pi)^{2/2}\sqrt{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2}} \exp\left(-\frac{1}{2}\frac{1}{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2}(x-\mu)^T \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}(x-\mu)\right) \tag{33}$$

We can pull out the factor $\sigma_x\sigma_y$

$$P(x,y) = \frac{1}{2\pi\sqrt{\sigma_x^2\sigma_y^2}\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2}\frac{1}{\sigma_x^2\sigma_y^2(1-\rho^2)}\left(x^2\sigma_y^2 + y^2\sigma_x^2 - 2xy\sigma_{xy}\right)\right) \tag{34}$$

This results in the expression

$$P(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right)\right) \tag{35}$$

(b) Let x denote the sum of correlated variables $x_i, i = 1, \ldots, n$. We then have

$$x = \sum_i x_i \tag{36}$$

$$\mathbb{E}[x] = \sum_i E[x_i] \tag{37}$$

$$\mathrm{Var}[x] = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{cov}(x_i, x_j) \tag{38}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \rho_{ij}\sigma_i\sigma_j \tag{39}$$

$$= \sum_{i=1}^{n} \sigma_i^2 + 2\sum_{i<j}^{n} \rho_{ij}\sigma_i\sigma_j \tag{40}$$

If we have only two variables, we can write

$$z = x - y = x + (-y) \tag{41}$$

$$\mathbb{E}[z] = \mathbb{E}[x] + \mathbb{E}[-y] = \mathbb{E}[x] - \mathbb{E}[y] \Leftrightarrow \boldsymbol{\mu_z = \mu_x - \mu_y} \tag{42}$$

$$\mathrm{Var}[z] = \sigma_x^2 + \sigma_{-y}^2 + 2\rho_{x,-y}\sigma_x\sigma_{-y} \tag{43}$$

We have a look at the definition of the covariance:

$$\mathrm{Cov}[x,-y] = \mathbb{E}[(x-\mathbb{E}[x])(-y-\mathbb{E}[-y])] = \mathbb{E}[-(x-\mathbb{E}[x])(y-\mathbb{E}[y])] = -\mathrm{Cov}[x,y] \tag{44}$$

$$\Leftrightarrow \rho_{xy} = -\rho_{x,-y} \tag{45}$$

Also, x is completely correlated to itself, and so is y ($\Rightarrow \rho_{ii} = 1$). It is left to show that $\sigma_{-y} = \sigma_y$ and we are done:

$$\sigma_{-y}^2 = \mathbb{E}[(-y)^2] - \mathbb{E}[-y]^2 = \mathbb{E}[y^2] - (-\mathbb{E}[y])^2 = (\mathbb{E}[y^2] - \mathbb{E}[y]^2) = \sigma_y^2 \tag{46}$$

Now, putting everything together, we obtain

$$\mathrm{Var}[z = x - y] = \sigma_x^2 + \sigma_y^2 - 2\rho\sigma_x\sigma_{-y} \tag{47}$$

2.4 Convolution of Gaussians

Suppose you have a true distribution around the value x.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left(-\frac{(x-x_0)^2}{2\sigma_x^2}\right) \tag{48}$$

and the measured quantity, y follows a Gaussian distribution around the value x.

$$P(y|x) = \frac{1}{\sqrt{2\pi}\sigma_y} \exp\left(-\frac{(y-y_0)^2}{2\sigma_y^2}\right) \tag{49}$$

What is the predicted distribution for the observed quantity y?

Like spoilered in the title of this exercise, the predicted distribution for the observed quantity y is the convolution of f with $P(y|x)$. This can be expressed as a multiplication in Fourier space, i.e. the multiplication of the characteristic functions0:

$$f(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} \,\mathrm{d}\, x = \tag{50}$$

$$f(\omega) = \frac{1}{\sqrt{2\pi}\sigma_x} \int_{-\infty}^{\infty} \exp\left(-\frac{(x-x_0)^2}{2\sigma_x^2} - i\omega x\right) \mathrm{d}\, x \tag{51}$$

Now we introduce $u = x - x_0$, $x = u + x_0$ and $du = dx$.

$$f(\omega) = \frac{1}{\sqrt{2\pi}\sigma_x} \int_{-\infty}^{\infty} \exp\left(-\frac{u^2}{2\sigma_x^2} - i\omega(u+x_0)\right) \mathrm{d}\, u = \frac{\exp(-i\omega x_0)}{\sqrt{2\pi}\sigma_x} \int_{-\infty}^{\infty} \exp\left(-\frac{u^2}{2\sigma_x^2}\right) \exp\left(-i\omega u\right) \mathrm{d}\, u \tag{52}$$

$$= A \int_{-\infty}^{\infty} \exp\left(\alpha u^2 + \beta u\right) \mathrm{d}\, u \tag{53}$$

where we defined $A = \exp(-i\omega x_0)/(\sqrt{2\pi}\sigma_x)$, $\alpha = -1/2\sigma_x^2$, $\beta = -\omega$. Now we can use the known integral

$$I = \int_{-\infty}^{\infty} \exp(ax^2 + ibx)\,\mathrm{d}\,x = \frac{\sqrt{\pi}}{\sqrt{-a}} \exp(b^2/4a) \tag{54}$$

and substitue α and β. This yields

$$f(\omega) = \frac{\exp(-i\omega x_0)}{\sqrt{2\pi}\sigma_x} \frac{\sqrt{\pi}}{\sqrt{1/2\sigma_x^2}} \exp\left(\omega^2/(-2/\sigma_x^2)\right) = \exp(-i\omega x_0) \exp\left(-\sigma_x^2\omega^2/2\right) \tag{55}$$

We also get

$$P(\omega) = \exp(-i\omega y_0) \exp(-\sigma_y^2\omega^2/2) \tag{56}$$

Then

$$\begin{aligned} f(\omega)P(\omega) &= \exp(-i\omega x_0 + y_0) \exp(-(\sigma_x^2 + \sigma_y^2)\omega^2/2) \\ &= \exp\left(-i\omega(x_0 + y_0) - (\sigma_x^2 + \sigma_y^2)\omega^2/2\right) \end{aligned} \tag{57}$$

Now, we have to back-transform this expression into real space.

$$g(y) = \mathcal{F}^{-1}\{f(\omega)P(\omega)\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(\omega)P(\omega)e^{i\omega y}\,\mathrm{d}\,\omega \tag{58}$$

This yields

$$\begin{aligned} g(y) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-i(x_0 + y_0 - y)\omega - (\sigma_x^2 + \sigma_y^2)\omega^2/2\right) \\ &= \tilde{A} \int_{-\infty}^{\infty} \exp\left(i\tilde{\beta}\omega + \tilde{\alpha}\omega^2\right) \mathrm{d}\,\omega \end{aligned} \tag{59}$$

where $\tilde{A} = 1/\sqrt{2\pi}$, $\tilde{\alpha} = -(\sigma_x^2 + \sigma_y^2)/2$, $\tilde{\beta} = -(x_0 + y_0 - y)$. Again, we can use the known integral I, which yields:

$$\begin{aligned} g(y) &= \tilde{A}\frac{\sqrt{\pi}}{\sqrt{-\tilde{\alpha}}} \exp(\tilde{\beta}^2/4\tilde{\alpha}) = \tilde{A}\frac{\sqrt{2\pi}}{\sqrt{\sigma_x^2 + \sigma_y^2}} \exp\left((x_0 + y_0 - y)^2 \cdot \frac{1}{4}\frac{-2}{\sigma_x^2 + \sigma_y^2}\right) \\ &= \frac{1}{\sqrt{\sigma_x^2 + \sigma_y^2}} \exp\left(-\frac{1}{2}\frac{(y - x_0 - y_0)^2}{\sigma_x^2 + \sigma_y^2}\right) \end{aligned} \tag{60}$$

Now, we can see that

$$\tilde{\sigma}^2 = \sigma_x^2 + \sigma_y^2 \tag{61}$$

$$\tilde{\mu} = x_0 + y_0 = x_0 + x \tag{62}$$

Where $y_0 = x$ as we measure x. Because the characteristic function is equivalent to the Fourier transform, up to a factor, $1/\sqrt{2\pi}$, we can multiply $g(y)$ with this factor. This yields:

$$G(y) = \frac{1}{\sqrt{2\pi}\tilde{\sigma}} \exp\left(-\frac{1}{2}\frac{(y-\tilde{\mu})^2}{\tilde{\sigma}^2}\right) \tag{63}$$

so the distribution $G(y)$ for y is again a Gaussian distribution.

A Code for CLT of the exponential distribution

```
np.random.seed(42)

def p(x,l):
    """pdf of the exponential distribuiton with argument x and parameter lambda"""
    if(x >= 0):
        return l*np.exp(-l*x)
    else:
        return 0

def gauss(sigma,z):
    """
    the gauss distribution with standard deviation sigma and argument z
    """
    A = 1/(np.sqrt(2*np.pi)*sigma)
    return A*np.exp(-0.5*(z**2)/(sigma**2))


def plot_hist(N,sample_size,lambd,n_bins=100):
    """
    plots a histogram of exponentially distributed values

    N = number of trials/samples
    sample_size = sample size
    lambd = parameter
    n_bins = number of histogram bins

    Output: Table with statistics and histogram plot
    """
    mean_vals = [] #end result
    np.random.seed(420)

    #repeat experiment N times
    for n in range(N):

        #generate random vairable
        U_vals = [random.random() for i in range(sample_size)]

        x_vals = [-(np.log(U))/lambd for U in U_vals] #draw random x with poisson probability

        x_bar = np.mean(x_vals)
        mean_vals.append(x_bar) #append to end result


    #---- Statistics
    actual_mean = np.mean(mean_vals)
    actual_var = np.var(mean_vals)
    actual_sigma = np.sqrt(actual_var)

    theo_var = ((1/lambd**2))*(1/sample_size)
    theo_sigma = (1/lambd)*(1/np.sqrt(sample_size))
    theo_mean = 1/lambd

    table_data = [["sample set",actual_mean,actual_var,actual_sigma],["theoretical",theo_mean,theo_var,theo_sigma]]
    print_table(table_data,"| \t|$\mu$ \t| Var|\t $\sigma$")
    #----

    plt.figure(figsize=(8,8))


    #HISTOGRAM

    values, bins = np.histogram(mean_vals,bins=n_bins) #calculate the histogram
    area = sum(np.diff(bins)*values) #area under histogram
    hist_vals = np.divide(mean_vals,area) #normalize histogram data
    heights,bins,_  = plt.hist(mean_vals,n_bins,weights=[1/area for i in range(len(mean_vals))],color="navajowhite",alpha=.9,label=
    #plot histogram
```

```
67
68      #SHIFT HISTOGRAM BY HALF BINSIZE!!!
69
70      #GAUSS CURVE
71
72      z_vals = (mean_vals - np.full(N,theo_mean)) #define z
73      cont_mean = np.linspace(min(mean_vals),max(mean_vals),100) #make continuous interval for gauss plot
74      cont_z = np.linspace(min(z_vals),max(z_vals),100)
75      plt.plot(cont_mean,[gauss(theo_sigma,z) for z in cont_z],color="indianred",label="Gauss model")    #plot Gauss model
76
77      plt.xlabel("mean value")
78      plt.ylabel("density")
79      #plt.legend(loc=5)
80      plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
81      plt.show()
82
83
84      #Q-Q plot
85      quantile_plot(mean_vals)
86
```

B Code for CLT of the Cauchy distribution

```
1       import random as random
2       from scipy.optimize import curve_fit
3       x_vals = []
4
5
6       def gaus(x,a,x0,sigma):
7           """
8           The normal distribution"""
9           return a*np.exp(-(x-x0)**2/(2*sigma**2))
10
11
12      def plot_hist_cauchy(N,sample_size,x0,gamma,n_bins=300):
13          """
14          Plots a histogram of random draws with size 'sample_size', N experiments and parameters x_0, gamma.
15          Binsize is 300 (default)
16          """
17
18          plt.figure()
19
20          means = []
21
22          #N samples of sample size "sample_size"
23          for n in range(N):
24              np.random.seed(1338) #new random seed
25              x_vals = [gamma*(np.tan(np.pi*random.random()-np.pi/2))+x0 for n in range(sample_size)] #random draw
26
27              mean = np.mean(x_vals)
28              means.append(mean)
29
30          #Plot histogram
31          plt.hist(means,bins=n_bins,normed=True,color="cornflowerblue",alpha=.9) #plot histogram
32
33          plt.xlabel("mean")
34          plt.ylabel("freqency")
35
36          print("mean ",np.mean(means))
37
38          plt.show()
39
40  plot_hist_cauchy(1000,100,25,3,n_bins=300) #N, sample_size, x0, gamma
41  plot_hist_cauchy(1000,500,25,3,n_bins=300) #N, sample_size, x0, gamma
```

Model Fitting and Model Selection

Thomas Baldauf

Winter semester 2017/18

1 Introduction

1.1 Bayesian Analysis for Binomial Data: Detector Efficiency

We want to study a model $f(x|\lambda)$, where x is a measurable quantity and λ represents the model parameters. For a binomial process, we can have the outcomes "success" and "failure", and for N trials we obtain a success rate r, which depends on the choice of the model parameters λ. We can measure then x and "learn" the model parameters λ from the measured data by cross-checking it with the predicted success rate (see figure 1).

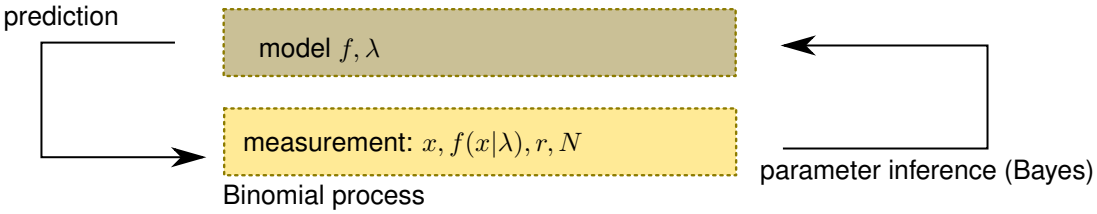


Figure 1: Schematic concept of finding a model for binomial data measured in a detector.

Once obtained, we can use the data to conclude the probabilities of the parameter using Bayes' theorem:

$$P(\lambda|\{r\},\{N\}) = \frac{P(\{r\}|\{N\},\lambda)P_0(\lambda)}{\int P(\{r\}|\{N\},\lambda)P_0(\lambda)\,\mathrm{d}\,\lambda} \tag{1}$$

where

$$P(\{r\}|\{N\},\lambda) = P(\{r\}|\{N\},\{f(x|\lambda)\}) \tag{2}$$

Now, we know $P(\{r\}|\{N\},\{f(x|\lambda)\})$, but we do not know the prior $P_0(\lambda)$. We have to assume flat priors if we have no clue about the distribution of λ. If we think we know λ already pretty well,

we can choose a Gaussian prior. Another problem is that the integral in the denominator becomes intractable for a large number of model parameters and data points. For only two model parameters, the calculation still works out pretty well.

1.2 Detailed example: particle counter

A specific example is the response of a (particle) detector, which measures a certain count rate r when a certain intensity (rate N) comes from a radiation source. This signal r depends on the energy $E \equiv x$ of the photons impinging on the detector. This is a binomial process as the detector can either count or not count an impinging particle. We want to model the detector efficiency function with a sigmoid

$$\epsilon(E|A, E_0) = \frac{1}{1 + \exp(-A(E - E_0))} \tag{3}$$

So if we chose $f(x|\lambda) = \epsilon(E|A, E_0)$, we get

$$P(\{r\}|\{N\}, \{f(x|\lambda)\}) = \prod_{i=1}^{k} \binom{N_i}{r_i} \left(f(x_i|\lambda)^{r_i}(1 - f(x_i|\lambda))^{N_i - r_i}\right) \tag{4}$$

$$= P(\{r\}|\{N\}, \{\epsilon(E|A, E_0)\}) = \prod_{i=1}^{k} \binom{N_i}{r_i} \epsilon(E_i|A, E_0)^{r_i}(1 - \epsilon(E_i|A, E_0))^{r_i - N_i}$$

$$= \prod_{i=1}^{k} \binom{N_i}{r_i} \left(1 + e^{-A(E_i - E_0)}\right)^{-r_i} \left(1 + e^{A(E_i - E_0)}\right)^{r_i - N_i} \tag{5}$$

1.2.1 Bayesian Analysis

We want the prior probability to be Gaussian, i.e.

$$P_0(E_0) = \mathcal{G}(E_0|\mu_E, \sigma_E^2) = \frac{1}{\sqrt{2\pi}\sigma_E} \exp\left(-\frac{1}{2\sigma_E^2}(E_0 - \mu_E)^2\right) \tag{6}$$

$$P_0(A) = \mathcal{G}(A|\mu_A, \sigma_A^2) = \frac{1}{\sqrt{2\pi}\sigma_A} \exp\left(-\frac{1}{2\sigma_A^2}(A - \mu_A)^2\right) \tag{7}$$

$$P_0(A, E_0) = P_0(E_0)P_0(A) = \frac{1}{2\pi\sigma_E\sigma_A} \exp\left(-\frac{1}{2\sigma_E^2}(E_0 - \mu_E)^2 - \frac{1}{2\sigma_A^2}(A - \mu_A)^2\right)$$

$$= \frac{1}{2\pi} \frac{1}{\sqrt{\det \Sigma}} \exp\left(-\frac{1}{2}(\lambda - \mu)^T \Sigma^{-1} (\lambda - \mu)\right) \tag{8}$$

where

$$\Sigma = \begin{pmatrix} \sigma_E^2 & 0 \\ 0 & \sigma_A^2 \end{pmatrix}, \ \mu = \begin{pmatrix} \mu_E \\ \mu_A \end{pmatrix}, \ \lambda = \begin{pmatrix} E_0 \\ A \end{pmatrix} \tag{9}$$

Energy (E_i)	Trials (N_i)	Successes (r_i)
0.5	100	0
1.0	100	4
1.5	100	22
2.0	100	55
2.5	100	80
3.0	1000	987
3.5	1000	995
4.0	1000	998

Table 1: Example data from the lecture.

In the last step, we formulated the prior distribution as a multivariate (bivariate) Gauss distribution on E_0 and A. This means that we assume E_0 and A to be non-correlated. This assumption can be cross-checked by the shape of the posterior distribution, which should have vertical and horizontal principle component axes (see figure 2). The conjugate prior of a normal distribution is again a normal distribution.
So for the posterior distribution, we get

$$
\begin{aligned}
&P(A, E_0|\{r\}, \{N\}, \Sigma, \mu) \\
&= \frac{\prod_{i=1}^{k} \binom{N_i}{r_i} \left(1+e^{-A(E_i-E_0)}\right)^{-r_i} \left(1+e^{+A(E_i-E_0)}\right)^{r_i-N_i} \frac{1}{2\pi \det \sigma} \exp\left(-\frac{1}{2}(\lambda-\mu)^T\Sigma^{-1}(\lambda-\mu)\right)}{\int_A \int_{E_0} \prod_{i=1}^{k} \binom{N_i}{r_i} \left(1+e^{-A(E_i-E_0)}\right)^{-r_i} \left(1+e^{+A(E_i-E_0)}\right)^{r_i-N_i} \frac{1}{2\pi \det \sigma} \exp\left(-\frac{1}{2}(\lambda-\mu)^T\Sigma^{-1}(\lambda-\mu)\right) \mathrm{d}\, A \,\mathrm{d}\, E_0} \\
&= \frac{\prod_{i=1}^{k} \binom{N_i}{r_i} \left(1+e^{-A(E_i-E_0)}\right)^{-r_i} \left(1+e^{+A(E_i-E_0)}\right)^{r_i-N_i} \exp\left(-\frac{1}{2}(\lambda-\mu)^T\Sigma^{-1}(\lambda-\mu)\right)}{\int_A \int_{E_0} \prod_{i=1}^{k} \binom{N_i}{r_i} \left(1+e^{-A(E_i-E_0)}\right)^{-r_i} \left(1+e^{+A(E_i-E_0)}\right)^{r_i-N_i} \exp\left(-\frac{1}{2}(\lambda-\mu)^T\Sigma^{-1}(\lambda-\mu)\right) \mathrm{d}\, A \,\mathrm{d}\, E_0}
\end{aligned}
\tag{10}
$$

This expression still depends on the covariance Σ and the mean μ of the prior probability. For some example data (see table 1.2.1), we determined the prior parameters $\sigma_A = 0.5, \mu_A = 3, \sigma_E = 0.3, \mu_E = 2.0$. With these parameters, we obtain the posterior probability distribution plotted in 2, which has been reconstructed from the lecture notes using code **??**.
Once obtained the posterior probabilitty, we need to determine the smallest intervals to get a valid interval where we accept the model parameters. This can be acchieved by sorting all 2D grid points by probability content and then accumulating the probability point by point, beginning with the mode (A^*, E^*).

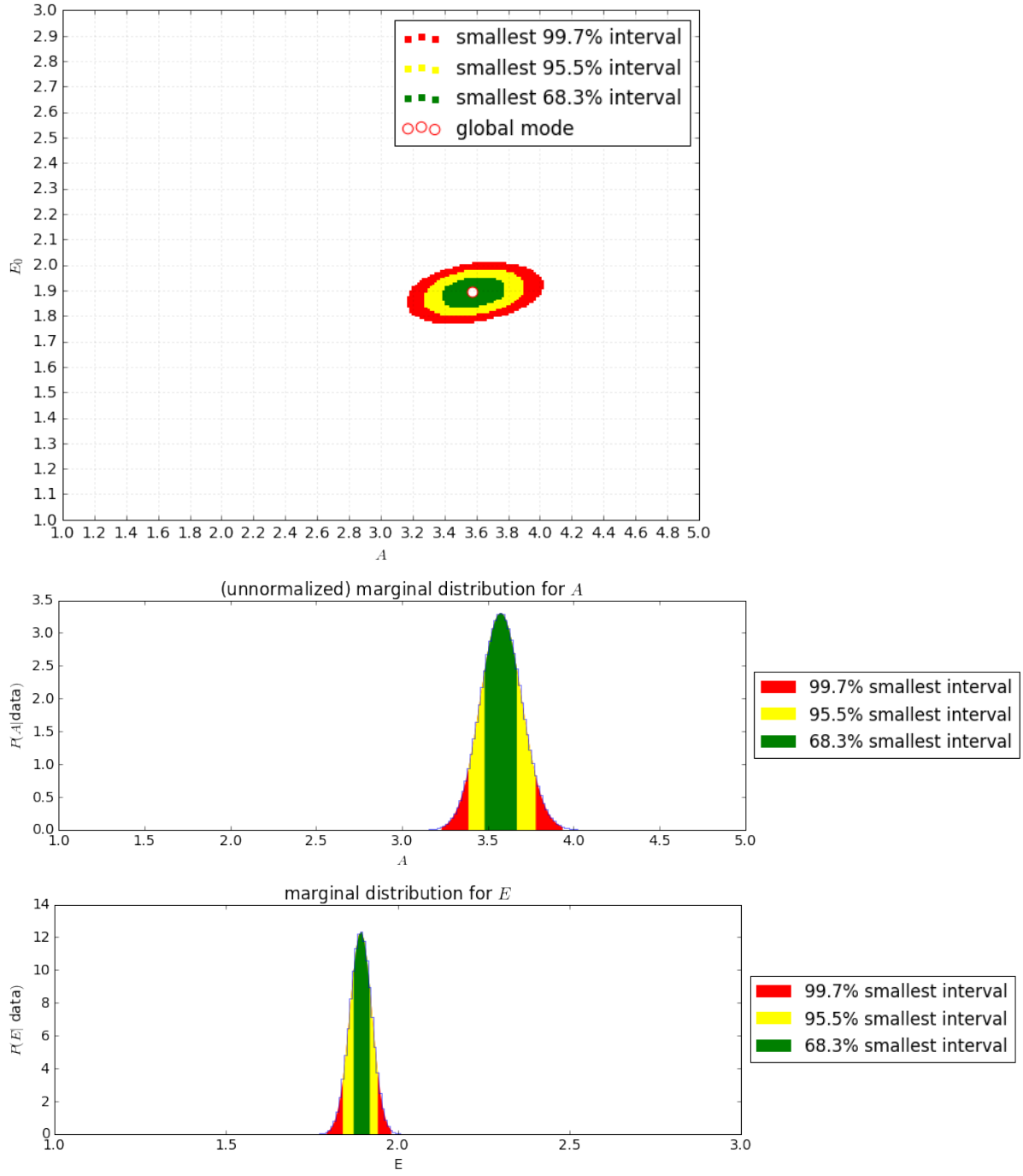


Figure 2: Top: Two-dimensional probability distribution, obtained from the data in table 1.2.1 and using the prior estimates from the lecture. The colors indicate the value of the posterior probability. We can see from the orientation of the multivariate Gauss that A and E_0 are only negligably correlated. Bottom: marginalized distributions from the 2d-posterior. The modes are $A^* = 3.586$ and $E^* = 1.889$.

1.2.2 Frequentist Analysis

We choose the following test statistic for our data with k measurements:

$$\xi(\{r_i\};\lambda) = \prod_{i=1}^{k} \binom{N_i}{r_i} p_i(\lambda)^{r_i}(1-p_i(\lambda))^{N_i-r_i} \tag{11}$$

Which has the form of a binomial distribution. For numerical stability, we work with the $\log\xi$ function:

$$\log\xi(\{r_i\}) = \sum_{i=1}^{k} \left(\log\binom{N_i}{r_i} + r_i \log(p_i(\lambda)) + (N_i - r_i)\log(1-p_i(\lambda)) \right) \tag{12}$$

The probability distribution of our test statistics for fixed parameters λ is

$$P(\xi|\lambda) \sum_{\{r_i\}} P(\{r_i\}) \mathcal{I}(\xi = \xi(\{r_i\};\lambda)) \tag{13}$$

where $\mathcal{I}$ is the indicator function and is 1 if the argument is satisfied, 0 if not. In the case of the sigmoid function, this depends on a threshold value of 0.5, if the argument of the sigmoid is positive, we count it as a success, and the function value is above 0.5. Otherwise, the function value is below 0.5, and we do not count the corresponding trial as a success. Python code B evaluates the test statistics and determines the confidence interval. We perform three steps:

1. We calculate the test statistics $\log\xi$ for a randomly created set (ensemble) of success rates $\{r_i\}_{i=1}^{n}$, where the ensemble size n is chosen sufficiently large, i.e. we have an adequate number of randomly generated possible experiments. From this, obtain a grid plot with all values of $\log\xi$ on the grid

2. We store the values of $\log\xi$ in a sorted array. From this, we obtain a one-dimensional plot of all $\log\xi$ values and their relative frequencies. We determine confidence intervals (values of $\log\xi$ for which a fraction of $1-\alpha$ experiments lie below). Mind that the logarithmic values are negative and the ”best” values ξ^2 are the lowest.

3. We check wether the ξ^2-value of the data lies within that range.

We try an ensemble size $n = 1000$ and a grid size of $N^2 = 32 \times 32$. Mind that the number of samples to be calculated explodes with increasing grid size or increasing n ($\mathcal{O}(m^3), m = n * N^2$). .

1.3 The χ^2 test statistics

The χ^2 distribution is a special test statistics. It is defined as

$$\chi^2 = \sum_i \frac{(y_i - f(x_i|\lambda))^2}{w_i^2} \tag{14}$$

where $f(x|\lambda)$ is the model function we want to perform the test for. So, χ^2 is nothing else than the weighted sum of the difference between model and measurement. The weights are often chosen to

be the variance. Note that by minimizing χ^2, we can minimize the error between $y(x)$ and $f(x|\lambda)$, and by using λ as a free parameter, we can obtain an optimal "fitting value" for λ.
Often, we can assume that the data is Gauss distributed with a certain standard deviation σ. Then the probability of our measurement y is

$$P(y) = \mathcal{G}(y|f(x|\lambda), \sigma(x|\lambda)) \tag{15}$$

and χ^2 becomes

$$\chi^2 = \sum_i \frac{(y_i - f(x_i|\lambda))^2}{\sigma_i^2} \tag{16}$$

where $\sigma_i = \sigma(x_i|\lambda)$. In the lecture, we found by calculating the Jacobian $\left|\frac{\mathrm{d}\,\chi^2}{\mathrm{d}\,y}\right|$, we can obtain the probability distribution $P(\chi^2)$. It is

$$P(\chi^2) = \frac{1}{\sqrt{2\pi\chi^2}} \exp(-\chi^2/2). \tag{17}$$

Some of its properties are derived in the tasks. We call the χ^2 distribution for Gaussian distributed data canonical χ^2 distribution.

1.4 Equivalence of χ^2 minimization and Bayesian Analysis

In the lecture, we saw that, under the assumption of flat priors, the posterior probability of Gaussian distributed data is

$$P(\mu|D) = \frac{1}{\sqrt{2\pi}\sigma_\mu} \exp\left(-\frac{1}{2}\frac{(\mu - \hat{\mu})^2}{\sigma_\mu^2}\right) \tag{18}$$

Here, μ is some quantity we want to measure, and $\hat{\mu}$ is its true value. The measurements of μ deviate from this true value, namely with standard deviation σ_μ. This is proportional to the canonical distribution of χ^2:

$$P(\mu|D) \sim \exp(-\chi^2/2) \tag{19}$$

For Bayesian analysis, we get unscertainties in the interval

$$\mu \in [\hat{\mu} - N\sigma_\mu, \hat{\mu} + N\sigma_\mu] \tag{20}$$

which corresponds to

$$\chi^2 \in \left[\chi^2_{\min}, \chi^2_{\min} + N^2\right] \tag{21}$$

In a nutshell: Bayesian analysis with flat priors and χ^2 minimization are equivalent for Gaussian distributed data.
Now we want to use this revelation for model fitting. For the likelihood of some model parameters λ given the Gaussian distributed data, we can write

$$P(D|\lambda) = \left[\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i}\right] \exp\left(-\frac{1}{2}\frac{y_i - f(x_i|\lambda)^2}{\sigma_i^2}\right) \tag{22}$$

With flat priors, the posterior distribution becomes

$$P(\lambda|D) = \frac{1}{(2\pi)^{K/2}|\Sigma|} \exp\left(-\frac{1}{2}(\lambda - \hat{\lambda})^T \Sigma^{-1} (\lambda - \hat{\lambda})\right) \tag{23}$$

$\hat{\lambda}$ are the expectation values of the model parameters and

$$(\Sigma^{-1})_{ij} = -\frac{\partial^2 \ln P(\lambda|D)}{\partial\lambda_i \partial\lambda_j} \tag{24}$$

1.5 Linear Models: A special Case

We want to fit some data with the function

$$f(x|a_0, a_1, a_2) = a_0 + a_1 x + a_2 x^2 \tag{25}$$

Which corresponds to our general model where we defined f to be of the form

$$f(x|\lambda) = \sum_{k=0}^{K} \lambda_k g_k(x) \tag{26}$$

where $\lambda_k = a_k$, $g_k(x) = x^k$ and $K = 2$ in this case. We have two assumptions for our model:

1. linear dependence of our model function on the parameters (not necessarily on the data)
2. fixed Gaussian uncertainties

Under these assumtions, we can write f as a sum in g:

$$f(x_i|\lambda) = \sum_{l=1}^{K} \lambda_l g_l(x_i) \tag{27}$$

and finally have the condition

$$Y = M\lambda \tag{28}$$

where

$$Y_k = \sum_{i=1}^{n} \frac{C_{ik}}{\sigma_i^2} y_i \tag{29}$$

$$M_{kl} = \sum_{i=1}^{n} \frac{C_{ik} C_{il}}{\sigma_i^2} \tag{30}$$

$$C_{ik} = g_k(x_i) \tag{31}$$

where n is the number of data points. We then obtain the parameters of maximum posterior probability via

$$\Lambda = M^{-1} Y \tag{32}$$

Matrix inversion is a crucial step if M is not diagonal. We can now implement an algorithm to find $\lambda = (a_0, a_1, a_2)$ in our specific example. We can see that in equation 23, Σ is then given by

$$(\Sigma^1)_{ij} = \frac{1}{2} \frac{\partial^2 \chi^2}{\partial \lambda_i \lambda_j} = \frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_i a_j} = M_{ij} \tag{33}$$

2 Particle detector: Homework example

We now repeat exactly the same steps from the introductory section to a new ”homework” dataset. Follow the steps in the script to fit a Sigmoid function to the data in table 2.

Energy (E_i)	Trials (N_i)	Successes (r_i)
0.5	100	0
1.0	100	4
1.5	100	22
2.0	100	55
2.5	100	80
3.0	100	97
3.5	100	99
4.0	100	99

Table 2: Success rate measured with a detector at different energies with a number of 100 trials per measurement.

a) Find the posterior probability distribution for the parameters (A, E_0).

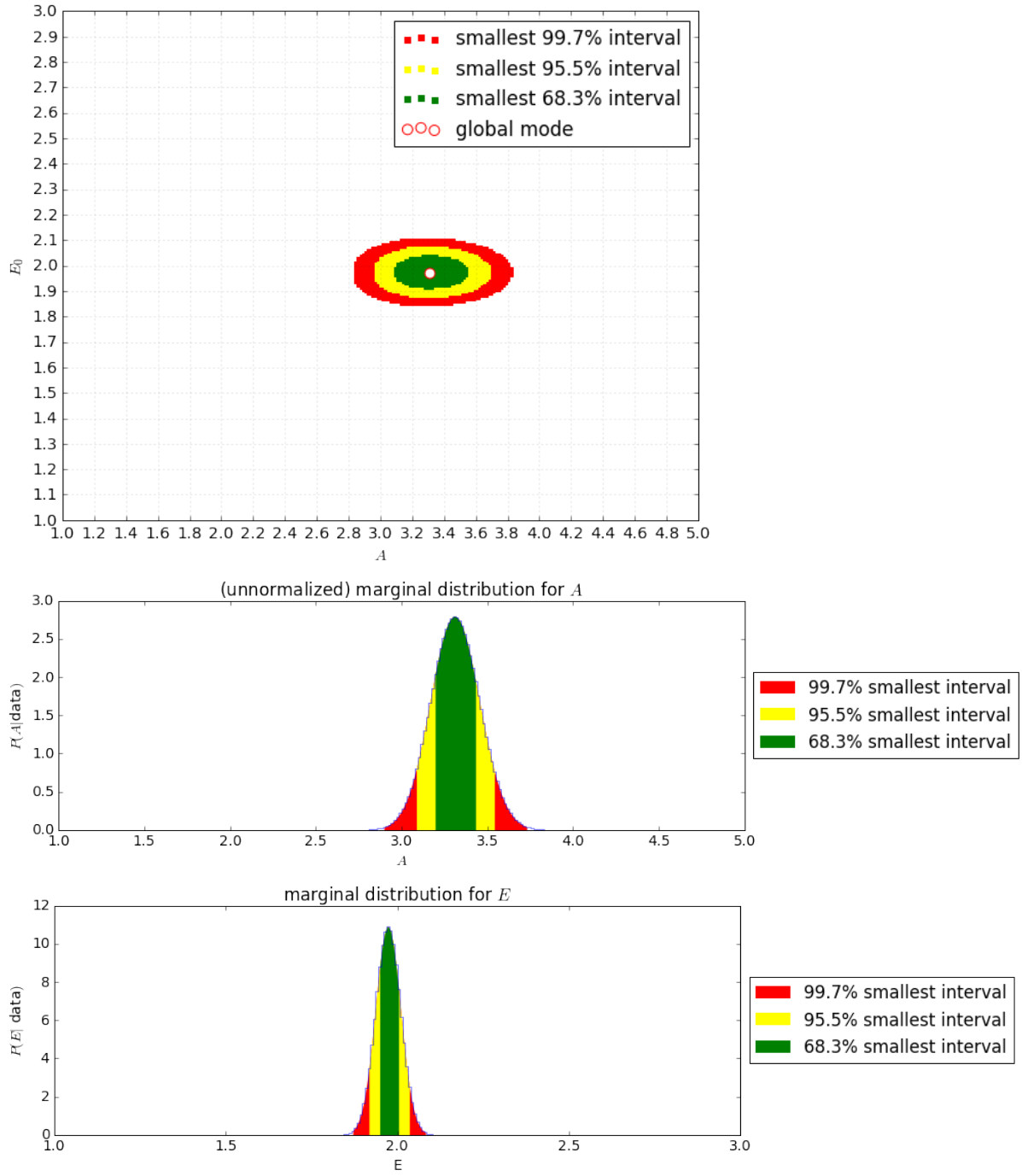


Figure 3: Posterior distribution for the homework example.

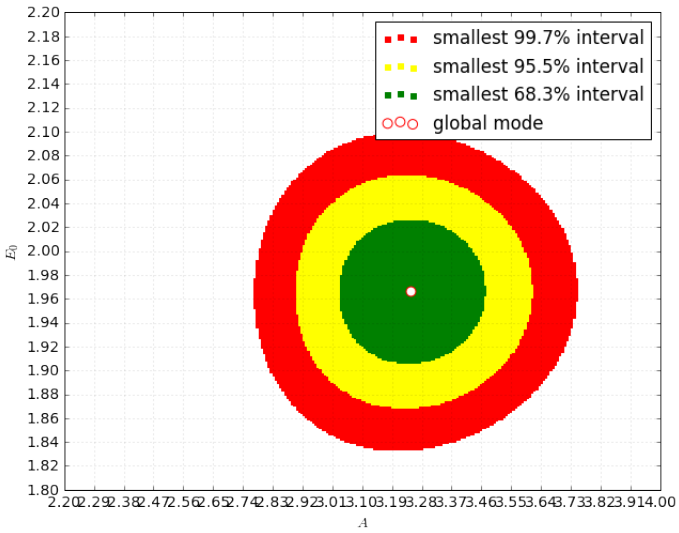


Figure 4: Zoom-in to the posterior distribution for the homework example.

The posterior distribution was created with the same python script (code B) like in the introductory section. The output for the new dataset is visualized in figure 3. The mode is at $(A^*, E_0^*) \approx (3.3, 1.9)$. We can also "zoom" a bit in, and get a more detailed plot of the confidence region (see figure 4). For the prior, a value of $\mu = (3.4, 1.9)$ was satisfying with $\sigma = (.5, .5)$.

b) Define a suitable test statistic and find the frequentist $60\ \%$ Confidence Level region for (A,E_0).

We are going to use the same test statistics like discussed in the introductory section (see equation 12). Going through exactly the same steps leads to the distribution of $\log \xi$ like depicted in figure 5. The smallest interval was used to determine the confidence interval of ξ. First, some test experiments are run (an ensemble is generated), and the corresponding success rate is noted down. Afterwards, a histogram for the ξ values is generated and compared with the ξ value of the data. Depending on wether the data lies within the corresponding confidence level, the pixel belonging to the point in parameter space is colored appropriately.

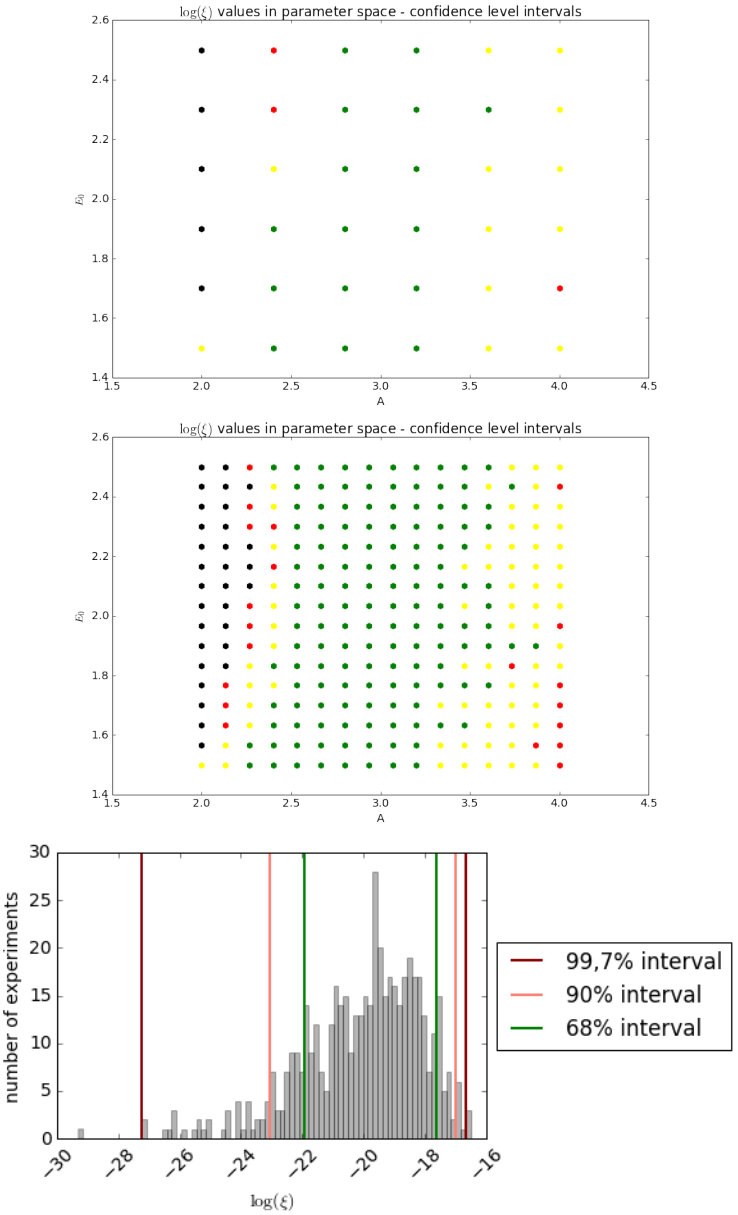


Figure 5: Top: Plots of the ξ distribution with the corresponding confidence intervals for each grid point. Bottom: distribution of ξ for a chosen gridpoint. $\log \xi_{\text{Data}}$ was -15.8.

Te confidence intervals were constructed using the smallest interval. The colors in the plots indicate the different confidence levels. We can recognize a certain shape, but it seems like for a grid size of 16×16 and a sample size of 800, we still get fairly poor results. The confidence levels also depend strongly on ξ_{Data}. However, the green interval for example is enough for a rough estimation and is in

agreement with the Bayesian analysis.

2.1 A different model

Repeat the analysis of the data in the previous problem with the function

$$\epsilon(E) = \sin(A(E - E_0)) \tag{34}$$

1. Find the posterior probability distribution for the parameters (A, E_0)
2. Find the 68% CL region for (A, E_0)
3. discuss the results

1. The posterior probability can be computed with the same script as in the previous exercise. The marginalized probabilities are shown in figure 6. The mode is at $(A^*, E^*) \approx (1.4, 2.2)$. For the prior, $\mu = (2.0, 2.0)$ was used with a standard deviation of $(.2, .5)$.
2. Also the frequentist part is equivalent to the previous exercise, results are shown in 7 for one grid point. The $\log \xi$ values are positive, which is an indicator for a poor fit.

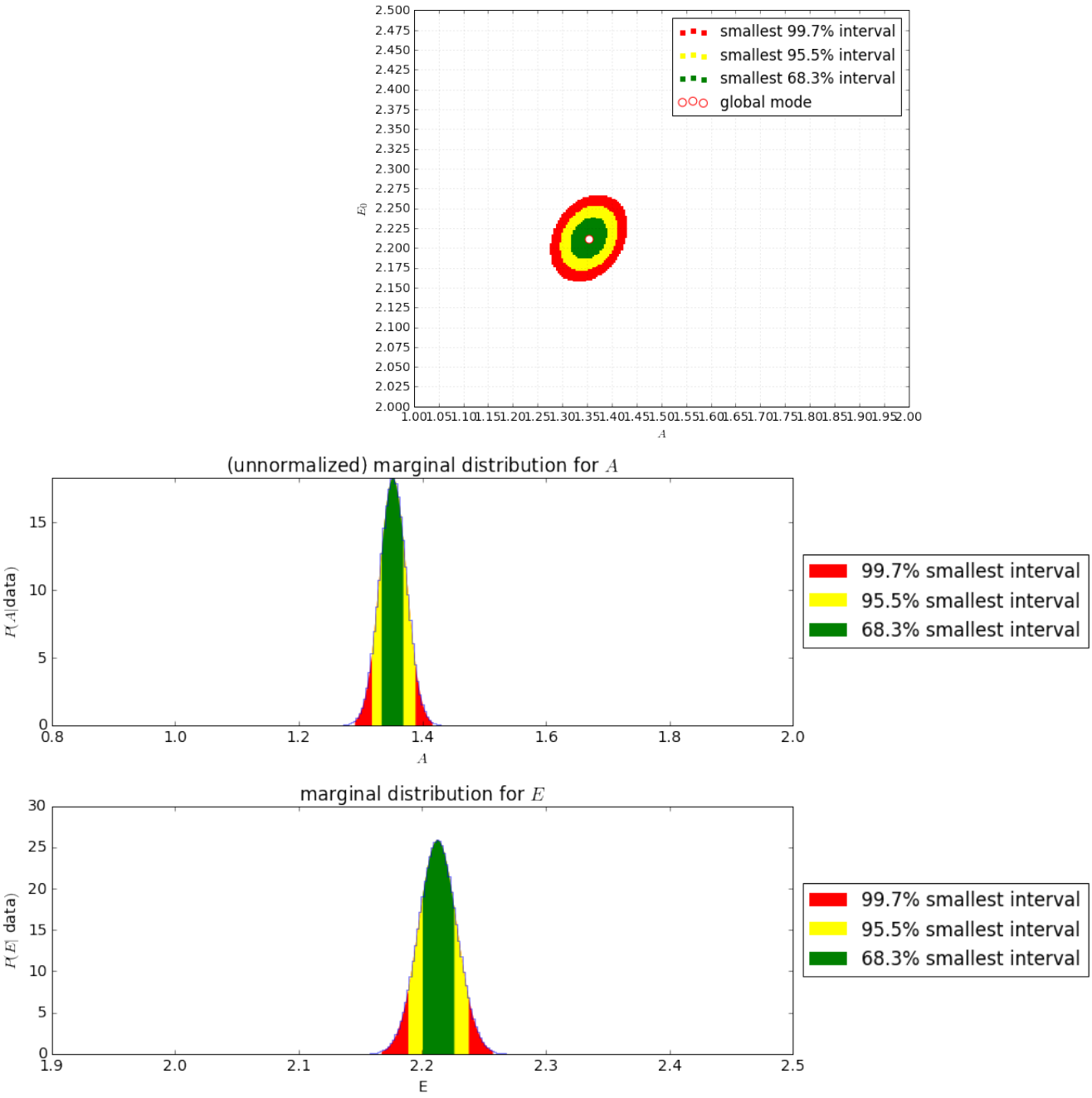


Figure 6: Posterior probability for the $\sin$ model of the data.

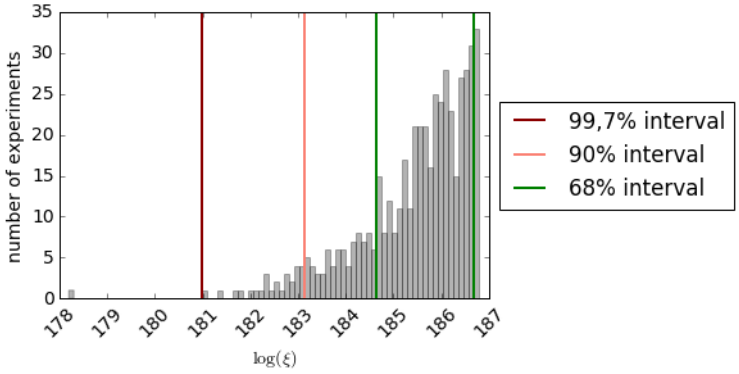


Figure 7: Extract from the frequentist analysis of the $\sin$ model.

2.2 Mean, variance and mode of the χ^2 distribution

Derive the mean, variance and mode for the χ^2 distribution for one data point.

We start with the mean, which is the expectation of χ^2.

$$\mathrm{E}[\chi^2] = \int_0^\infty \chi^2 f(\chi^2)\,\mathrm{d}\,\chi^2 = \int_0^\infty u f(u) du \tag{35}$$

where we abbreviate $u := \chi^2$. Here, f denotes the χ^2 pdf, given by

$$\begin{aligned} P(\chi^2) &= \frac{1}{\sqrt{2\pi\chi^2}} \exp(-\chi^2/2) \\ \Leftrightarrow P(u) &= \frac{1}{\sqrt{2\pi u}} \exp(-u/2) \end{aligned} \tag{36}$$

Thus,

$$\mathrm{E}[u] = \int_0^\infty u \frac{1}{\sqrt{2\pi u}} \exp(-u/2)\,\mathrm{d}\,u = \frac{1}{\sqrt{2\pi}} \int_0^\infty u^{1/2} \exp(-u/2)\,\mathrm{d}\,u \tag{37}$$

At this point, we can make use of the Gamma function, which is defined as

$$\Gamma(z) = \int_0^\infty t^{z-1} \exp(-t) \tag{38}$$

with $\Gamma(1/2) = \sqrt{\pi},\ \Gamma(3/2) = \sqrt{\pi}/2,\ \Gamma(5/2) = 3\sqrt{\pi}/4$. We can write $v = u/2 \Rightarrow 2\,\mathrm{d}\,v = \mathrm{d}\,u$:

$$\begin{aligned} \mathrm{E}[u] &= \frac{1}{\sqrt{2\pi}} \int_0^\infty \sqrt{2} v^{1/2} \exp(-v) 2\,\mathrm{d}\,v = \frac{2}{\sqrt{\pi}} \int_0^\infty v^{3/2-1} \exp(-v)\,\mathrm{d}\,v \\ &= 2\Gamma(3/2)/\sqrt{\pi} = 1 \end{aligned} \tag{39}$$

The variance is defined as

$$\mathrm{Var}[\chi^2] = \mathrm{E}[(\chi^2)^2] - (\mathrm{E}[\chi^2])^2 = \mathrm{E}[u^2] - E[u]^2 \tag{40}$$

We already know $\mathrm{E}[u] = 1$, and

$$\begin{aligned}\mathrm{E}[u^2] &= \frac{1}{\sqrt{2\pi}} \int_0^\infty u^{3/2} \exp(-u/2)\, \mathrm{d}\, u = \frac{2}{\sqrt{\pi}} \int_0^\infty v^{3/2} \exp(-v)\, \mathrm{d}\, v \\ &= \frac{2}{\sqrt{\pi}} \Gamma(5/2) = \frac{2}{\sqrt{\pi}} \frac{3\sqrt{\pi}}{4} = 3 \end{aligned} \tag{41}$$

So finally,

$$\mathrm{Var}[\chi^2] = 3 - 1^2 = 2. \tag{42}$$

The mode is the value for χ^2 where $P(\chi^2)$ is maximum. This can be done by plotting the function, and seeing it is at $\chi^2 = 0$, or by arguing like follows: The function is monotonically decreasing. That is because its derivative is negative

$$\frac{\partial P(\chi^2)}{\partial(\chi^2)} = \frac{\partial P(u)}{\partial u} = \frac{\partial}{\partial u} \left(\frac{1}{\sqrt{2\pi u}} \exp(-u/2) \right) \tag{43}$$

$$= \frac{1}{\sqrt{2\pi}} \left(-\frac{1}{2} u^{-3/2} \exp(-u/2) - \frac{1}{2} u^{-1/2} \exp(-u/2) \right) < 0. \tag{44}$$

Because it is a decreasing function, $P(\chi^2)$ takes its maximum at the boundary of its domain, which lies at $(\chi^2)^* = 0$. Summary:

- $\mathrm{E}[\chi^2] = 1$
- $\mathrm{Var}[\chi^2] = 2$
- $(\chi^2)^* = 0$

3 Model with Gauss probability distribution

Meausrements of a cross section for nuclear reactions yields the folloing data

θ (degrees)	30	45	90	120	150
cross section	11	13	17	17	14
error	1.5	1.0	2.0	2.0	1.5

The units of cross section are $10^{-30} cm^2/sr$. Assume the quoted errors correspond to one Gaussian standard deviation. The assumed model has the form

$$\sigma(\theta) = A + B \cos\theta + C \cos(\theta^2) \tag{45}$$

1. Set up the equation for the posterior probability density assuming flat priors for the parameters A, B, C
2. What are the values of A, B, C at the mode of the posterior pdf?

1. As the data is Gaussian distributed, the form of the posterior is just the expression 23, i.e. a multivariate Gaussian in three dimensions (for three parameters A, B, C). We can set $f(x|\lambda) = \sigma(\theta|A, B, C)$. The covariance matrix of the posterior is given by the minimal χ^2 value via the relation in equation 33. The linear model has the functions $g_l(\theta) = \cos(\theta^l)$. The error is given in the data table.

   We can o solve for M by performing a χ^2 on the computer. The output is

$$Y = [32.61, 5.91, 15.87] \tag{46}$$

$$M^{-1} = \begin{pmatrix} 0.81 & -0.01 & -0.68 \\ -0.01 & 1.05 & -0.43 \\ -0.68 & -0.43 & 1.41 \end{pmatrix} \tag{47}$$

$$\Lambda = [15.36, -1.08, -2.57] \tag{48}$$

2. We can plot the posterior distribution on a discrete grid in parameter space and see where the mode lies. Figure 8 shows the marginalized posterior probabilities in one and two dimensions. Note that the normalization of the posterior probability plays no role for finging the mode.

   Figure 9 shows the resulting fit with the data points and the model curve.

The implementation of the computer program (Python) is attached in A.

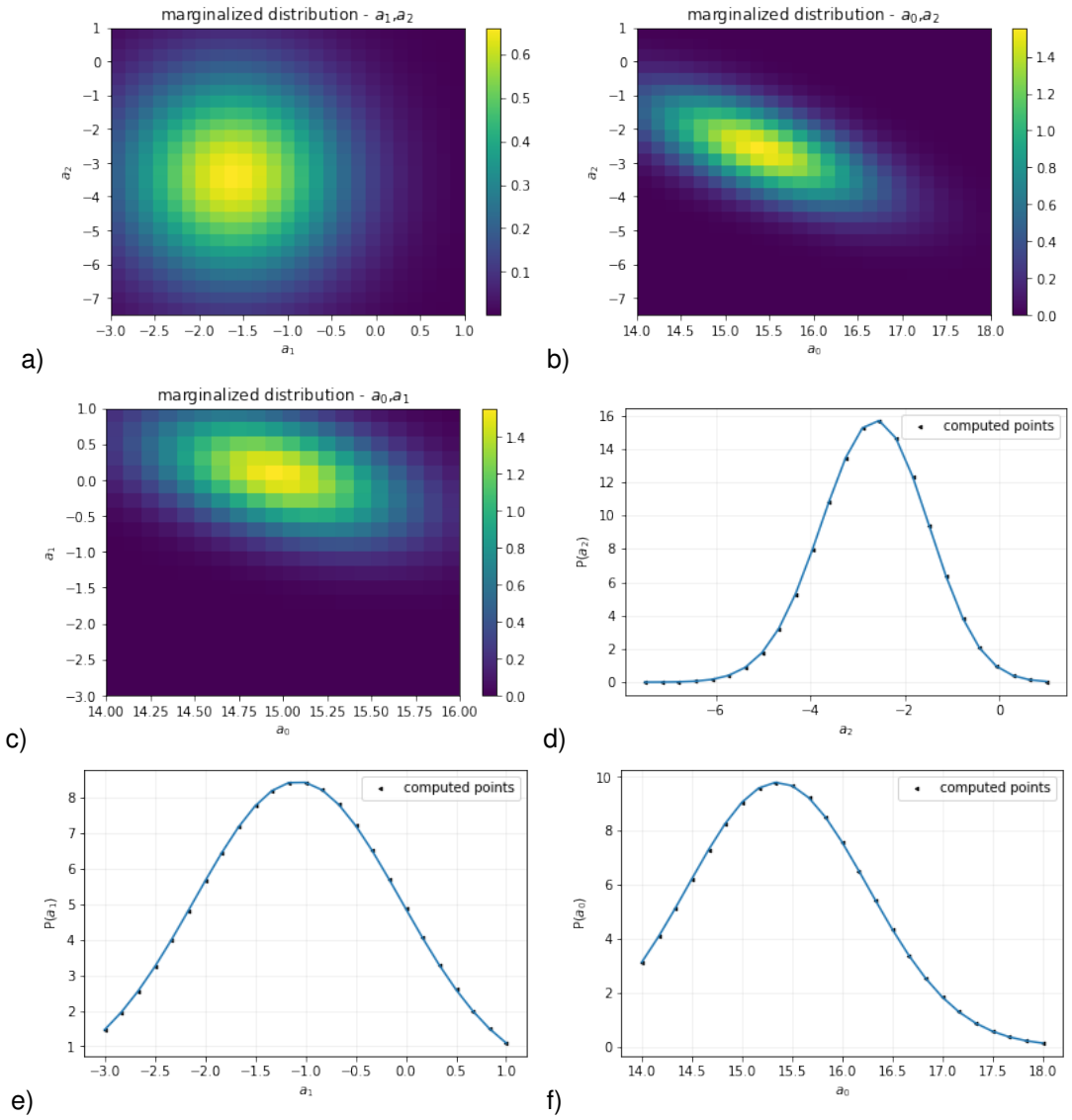


Figure 8: Marginalized posterior probability. The graphs are shown to better see the position of the mode of the posterior.

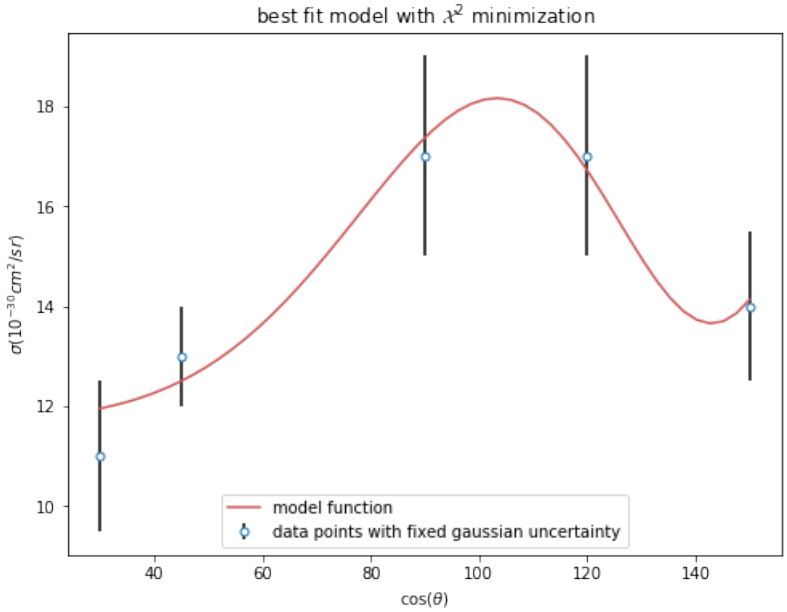


Figure 9: Given data (dots) and the resulting fit from the parameters at the mode of the posterior pdf (line).

3.1 Breit-Wigner model with Gaussian data

Analyze the following data set (table 3) assuming that the data can be modeled using a Gauss probability distribution where all data have the same uncertainty given by $\sigma = 4$. Try the two models:

- quadratic,representing background only

$$f(x|A,B,C) = A + Bx + Cx^2 \tag{49}$$

- quadratic + Breit-Wigner representing background+signal

$$f(x|A,B,C,x_0,\Gamma) = A + Bx + Cx^2 + \frac{D}{(x-x_0)^2 + \Gamma^2} \tag{50}$$

1. Perform a chi-squared minimization fit, and find the best values of the parameters as well as the covariance matrix for the parameters. What is the p-value of the fits?
2. Perform a Bayesian fit assuming flat priors for the paramters. Find the best values for the parameters as well as uncertainties based on the marginalized probability distributions. What is the Bayes Factor for the two models?

x	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
y	11.3	19.9	24.9	31.1	37.2	36.0	59.1	77.2	96.0
x	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
y	90.3	72.2	89.9	91.0	102.0	109.7	116.0	126.6	139.8

Table 3: Data for the Breit-Wigner+background model.

Frist, we perform a χ^2 minimization fit for the background model. We have then

$$f(x|A,B,C) = A + Bx + Cx^2 \tag{51}$$

as a linear model, like discussed before. The computer can easily output the corresponding values:

$$Y = [83.14, 54.48, 39.77] \tag{52}$$

$$M^{-1} = \begin{pmatrix} 15.3 & -61.1 & 51.6 \\ -61.1 & 286.3 & -260.1 \\ 51.6 & -260.1 & 247.7 \end{pmatrix} \tag{53}$$

$$\Lambda = [-7.27, 173.47, -28.88] \tag{54}$$

This is a very fast algorithm, compared to the Bayesian analysis. We can directly output the resulting fit (see figure 10).

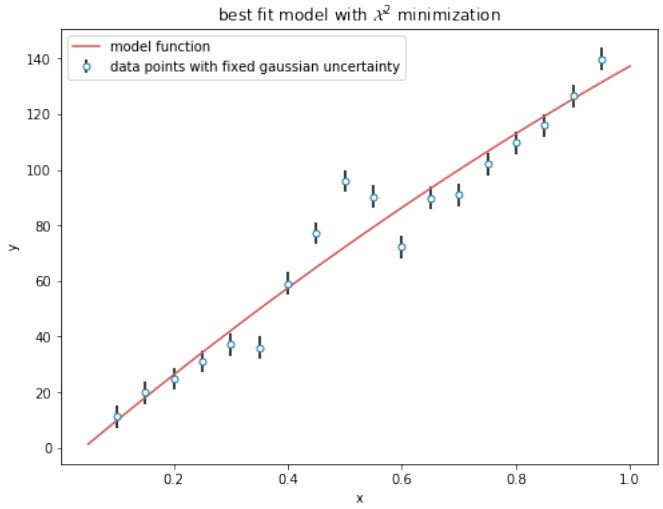


Figure 10: Fit of the background only, using χ^2 minimization.

In a Bayesian analysis, we can estimate the parameter uncertainties of the linear model with the following formula

$$P(\lambda|D) = \frac{1}{(2\pi)^{K/2}|M^{-1}|} \exp\left(-\frac{1}{2}(\lambda-\hat{\lambda})^t M (\lambda-\hat{\lambda})\right) \tag{55}$$

where we insert the values for M and λ from the χ^2 minimization result. Te resulting posterior distribution depends on three parameters and is therefore three-dimensional. But marginalized distributions can be plotted as 2d maps, as well as one-dimensional marginalized distributions.

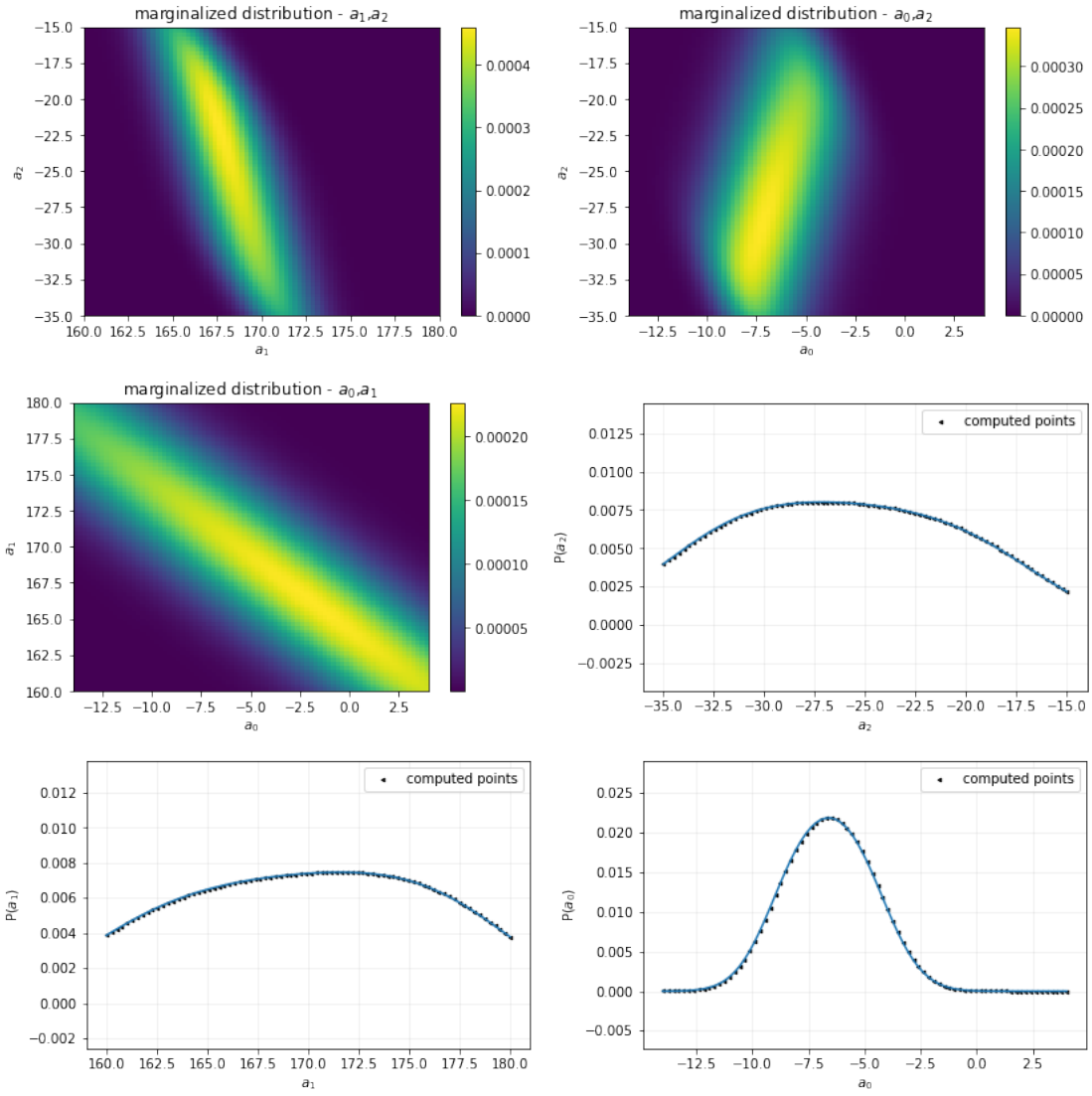


Figure 11: Marginalized distributions of the posterior probability (flat Prior) for the model with background only. The 1D-marginalizations are plotted near the mode. a_0, a_1, a_2 correspond to A, B, C.

From the posterior probability, we can also calculate uncertainties for the model paramters. Obtaining the χ^2 distribution from the ”grid scan” through parameter space, we can determine the confidence intervals. The p-value is the integrated tail from the point χ^2_{exp} of the Data to the maximum χ^2 value.

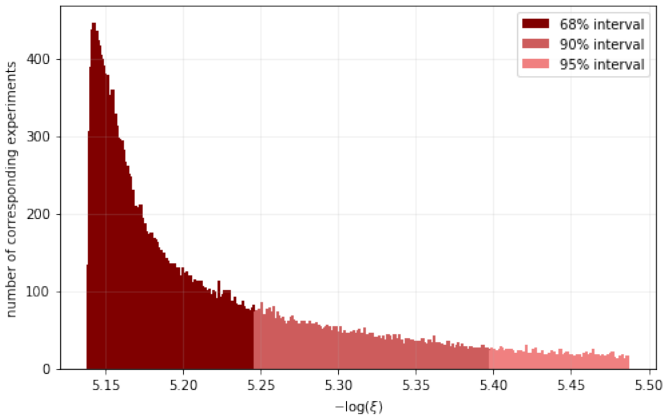


Figure 12: Plot of $-\log \xi$ (negative log test statistics) for the fits of the background only. The shape of the histogram is almost exponential. Confidence levels are colored in red. The histogram includes the data of all gridpoints. χ^2_{Data} was 5.13.

Now, we can also try the background plus signal

$$f(x|A,B,C,x_0,\Gamma) = A + Bx + Cx^2 + \frac{D}{(x-x_0)^2+\Gamma^2}. \tag{56}$$

The model is linear only in the parameters A, B, C, D, and not in x_0, Γ. Unlike the model for the background, which was linear, we do not expect a Gaussian shaped posterior distribution. Thus, it is no longer necessary/useful to talk about the covariance matrix, as it corresponds to a Gaussian approximation. Nevertheless, we can run an optimization algorithm on the value of χ^2, and we can obtain optimal values for the parameters anyway. In Python (numpy), there are several algorithms implemented. It turned out to be strongly dependent on the initial conditions and the optimization method wether the optimization works or not. In the lecture, an optimization method using derivatives was proposed. There are other algorithms, such as the Powell method, which does not rely on building gradients of the objective. Figure 13 shows the results for the χ^2 optimization curve fit. Although some of the algorithms did not converge, we have satisfying results for the Powell method.

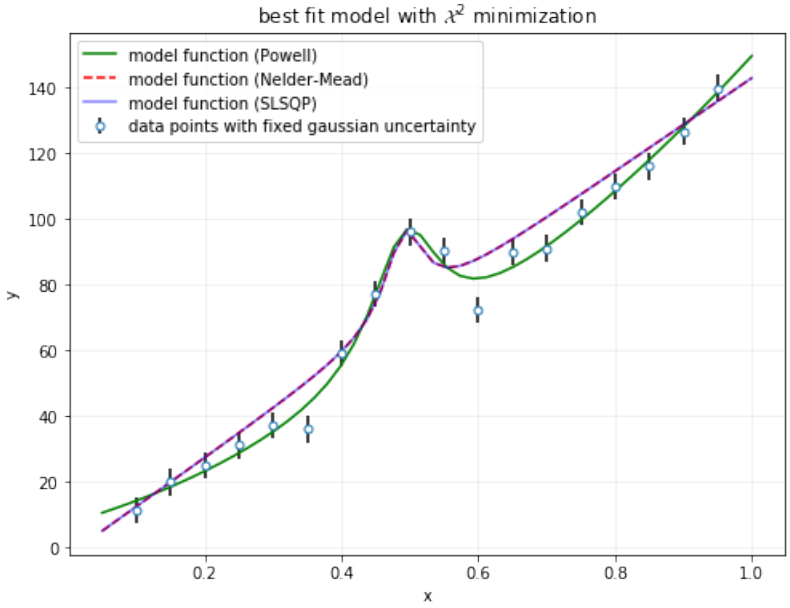


Figure 13: Different fit results for the χ^2 minimization fit of the model with background+Breit-Wigner signal. The green curve (Powell method) converged nicely.

The output of the optimization algorithm is

$$\Lambda_{\text{opt}} = [6.68365028e+00, 5.68686683e+01, 8.54396994e+01, 1.55742316e-01, 4.93324163e-01, 6.16283777e-02] \tag{57}$$

the first entries of Λ_{opt} represent the paramters A, B, C, D, the second last parameter is x_0 and the last parameter is Γ.
A full Bayesian treatment was also performed, though it takes very long time to compute on a standard computer. A very large parameter space was chosen, and a very fine grid. Using the fact that the distribution of χ^2 is almost exponential, we can calculate the smallest interval by just summing up from the smallest value of χ^2 upwards to get the confidence intervals. The 2D and 1D marginalized probability distributions are shown in figures 14 and 15. Here, $a_0, a_1, a_2, a_3, a_4, a_5$ correspond to A, B, C, D, x_0, Γ. Some of the parameters show high correlation and look like a multivariate Gaussian, whilst others exhibit exotic shapes. χ^2_{Data} was determined 18.97 and lies within the 68% confidence interval. The p-value is 0.35. The Bayes factor is defined as

$$K = \frac{P(\lambda_1|\text{model 1})}{P(\lambda 2|\text{model 2})} = \frac{\int P(\lambda_1|\text{model 1}) P(D|\lambda_1, \text{model 1})\, \mathrm{d}\,\lambda_1}{\int P(\lambda_2|\text{model 2}) P(D|\lambda_2, \text{model 2})\, \mathrm{d}\,\lambda_2} \tag{58}$$

The Bayes factor (in this case 10^{11}) of the two models lies far beyond 10^2, so it is a significant improvement to take the model with the peak instead of only the background. We can just pick the

"likelihood ratio" of the two models (which is at the best parameter point), it is approximately 20. However, to determine if this is a significant change is rather critical.

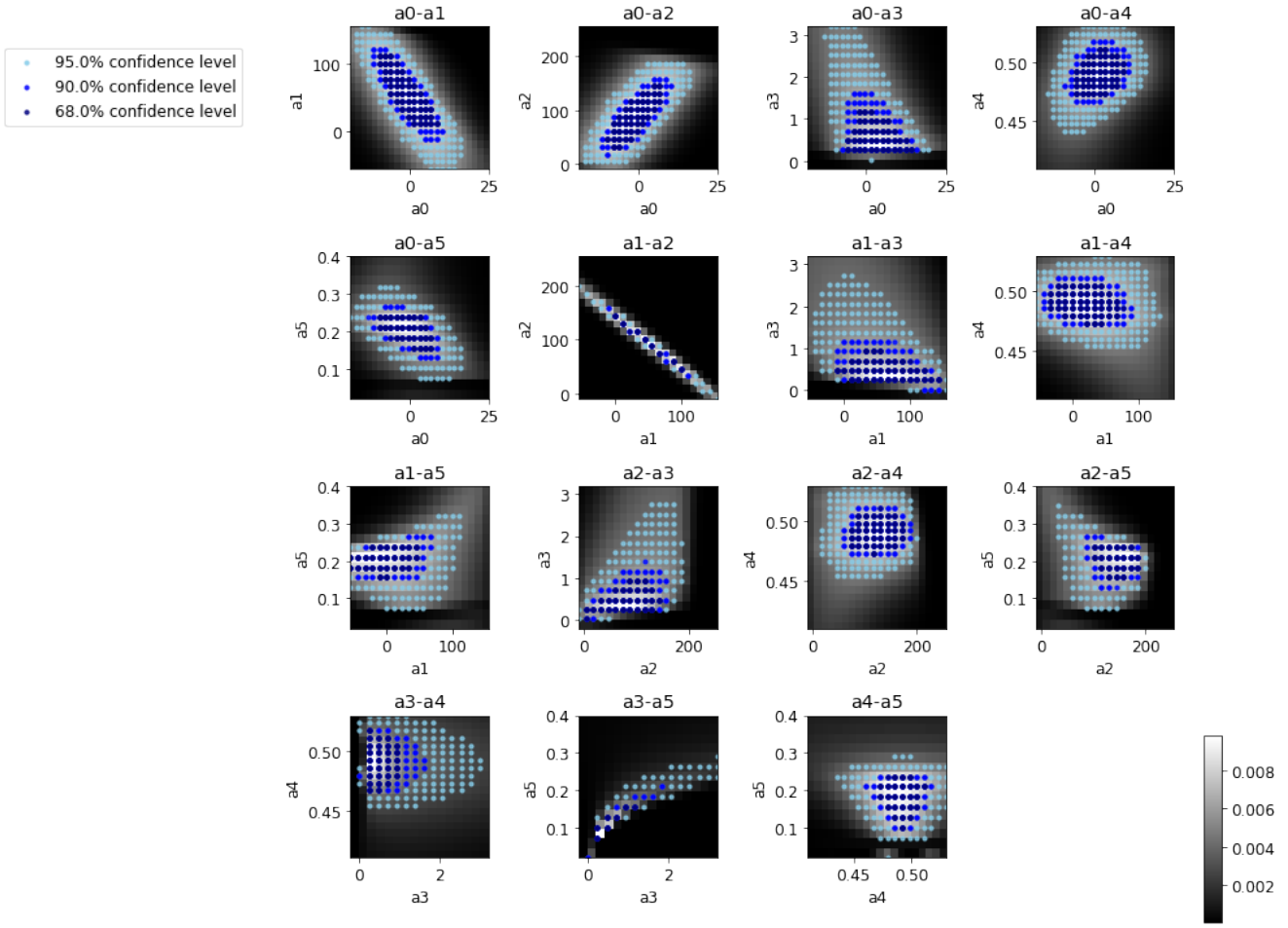


Figure 14: Posterior probability (2D marginalization) for the model with background + Breit-Wigner peak. The colored dots mark the obtained confidence intervals. $a_0, a_1, a_2, a_3, a_4, a_5$ correspond to A, B, C, D, x_0, Γ.

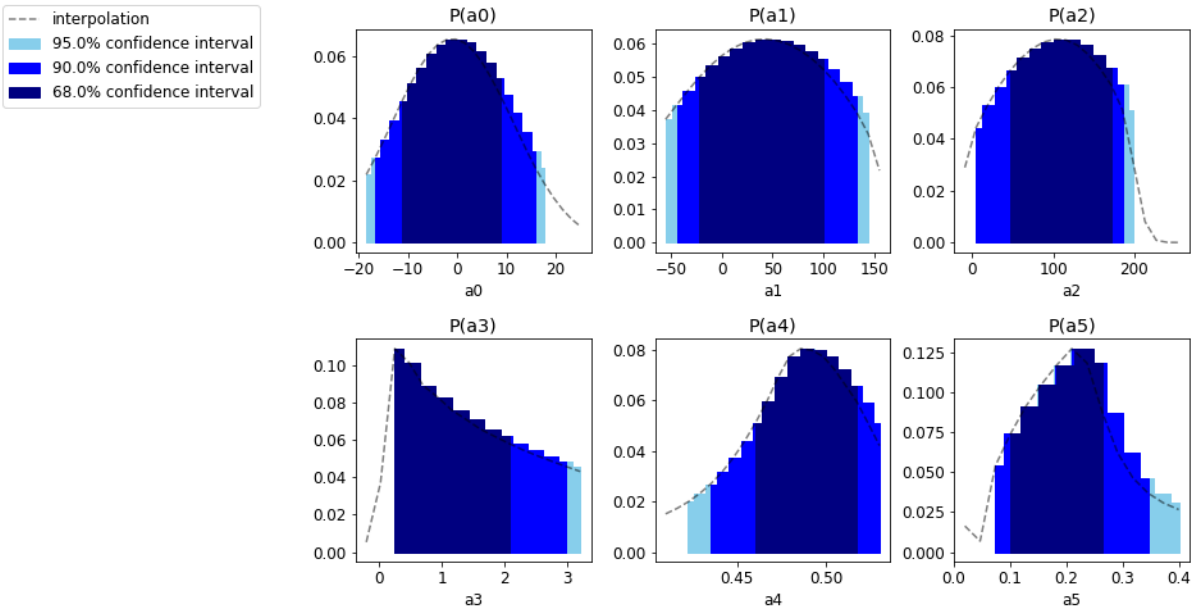


Figure 15: Posterior probability (1D marginalization) for the model with background + Breit-Wigner peak. The colored areas mark the obtained confidence intervals. $a_0, a_1, a_2, a_3, a_4, a_5$ correspond to A, B, C, D, x_0, Γ.

A Code for the Chi-squared minimization for linear models

```
import numpy as np
from IPython.display import display, Markdown, Latex
import matplotlib.pyplot as plt
import itertools

import scipy.stats
import time
import progressbar
from functools import partial
import itertools

from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
from matplotlib import cm


def lambda_MAP(x,y,sigma,g,K):
    """Calculates the parameter vector lambda, for a linear model with fixed Gaussian uncertainties!
    given data input x,y and fixed uncertainties sigma_i (ndarray of dimension n)
    and kernel functions g
    K = degree of the linear model (number of free model parameters)
    """

    n = len(x)

    #compute Y vector
    Y = np.empty(K)

    for k in range(K):
        z = 0
        for i in range(n):
            z += g(x[i],k)/(sigma[i]**2)*y[i]

        Y[k] = z

    print("Y",Y)

    #compute M matrix
    M = np.empty((K,K))

    for k in range(K):
        for l in range(K):
            m = 0
            for i in range(n):
                m += g(x[i],k)*g(x[i],l)/(sigma[i]**2)

            M[k,l] = m

    print("M = ",np.round(M))

    #lambda = M^{-1}Y

    Minv = np.linalg.inv(M)
    print("M^{-1} = ",Minv)

    L = np.dot(Minv.T,Y)

    print("parameters with maximum likelihood: ",L)

    return M,L



def g_cos(x,k):
    """returns the cosine of x^k"""
    if(k==0):
```

```
            return 1
        else:
            return np.cos((x*np.pi/180)**k)




    #calculate the parameters with highest likelihood for linear model with polynomial basis functions and 3
    #free parameters
    M_hat,lambd_hat = lambda_MAP(x,y,sigma,g_cos,3)


    #generate a fit function
    def pred_y(L,g,x):
        """generates predicted y values fit function, given the parameter vector L and a basis function g
        and evaluates it for the input values x (ndarray)"""

        K = len(L)

        return [np.sum([L[k]*g(x[i],k) for k in range(K)]) for i in range(len(x))]



    plt.figure(figsize=(8,6))

    #plot the data
    plt.errorbar(x,y,sigma,label="data points with fixed gaussian uncertainty",marker="o",markerfacecolor="w",markersize=5,linestyle="N

    #predict y values from model
    x_pred = np.linspace(min(x)-.05,max(x)+.05)
    y_pred = pred_y(lambd_hat,g_cos,x_pred)

    plt.plot(x_pred,y_pred,"-",color="indianred",label="model function")


    plt.legend(loc=8)
    plt.title(r"best fit model with $\mathcal{X}^2$ minimization")
    plt.xlabel(r"$\cos(\theta)$")
    plt.ylabel("$\sigma (10^{-30} cm^2/sr)$")
    plt.show()


    def parameter_uncert(l,M,l_hat):
        """Computes the probability of the parameters given the covariance matrix M, lambda hat  of the data
        given a certain position l
        """
        K = len(l_hat)

        Minv = np.linalg.inv(M)

        pre = 1/((2*np.pi)**(K/2)*np.linalg.det(Minv)) #pre factor

        #posterior distribution has multivariate gaussian form
        #P = pre*np.exp(-0.5*np.dot((l-l_hat).T,np.dot(M,(l-l_hat))))
        P = pre*np.exp(-.5*np.dot((l-l_hat).T,np.matmul(M,(l-l_hat))))

        return P


    def subarray(a,maxdepth,depth=0,l=[]):
        """Flattens an mgrid object to an array of n-dimensional coordinate vectors
        """
        #print("depth",depth)
        if(depth == 0):
            l = []
            print(l)
        #print("a,depth",a,depth)
        if(depth == maxdepth):
```

```
            #print("a",a,depth)
            l.append(a)
            return a
        else:
            for i in a:
                subarray(i,maxdepth,depth+1,l) #determine sub-items for each entry

        return l


    #== RUN ONLY IN ONE CELL TOGHETHER WITH THE CODE BELOW!
    #== DEFINE PARAMETER DOMAIN HERE
    step_size=60j

    mgrid = []
    mgrid = np.lib.index_tricks.nd_grid() #create multi-dimensional meshgrid

    param_space = mgrid[14:18:25j,-3:1:25j,-7.5:1:25j].T  #multidimensional meshgrid start:stop:step'j'
    coord_names = ["$a_0$","$a_1$","$a_2$"]

    flat_param_space = subarray(param_space,3)  #flatten mgrid
    l = []
    dim = np.shape(param_space)[-1] #dimensions = number of free parameters

    print("dim",dim)

    print("shape of parameters space", np.shape(param_space))
    print("shape of flat param space",np.shape(flat_param_space))




    min_vals = [min([i[j] for i in flat_param_space]) for j in range(dim)]
    print("min values",min_vals)

    max_vals = [max([i[j] for i in flat_param_space]) for j in range(dim)]
    print("max values",max_vals)

    print("computing, please wait...")
    start = time.time()

    #CRUCIAL STEP, CALCULATE THE DISTRIBUTION OVER WHOLE N-DIMENSIONAL PARAMETER SPACE
    #be careful, scales exponentially with number of dimensions

    part_parameter_uncert = partial(parameter_uncert,M=M_hat,l_hat=lambd_hat)
    uncert = list(map(part_parameter_uncert,flat_param_space))
    uncert = np.reshape(uncert,np.shape(param_space)[:-1])

    #in three dimensions
    #uncert = [[[parameter_uncert([i,j,k],M,lambd) for i in a0_range] for j in a1_range] for k in a2_range]

    delta = time.time()-start


    print("done")
    print("total runtime",delta,"s")


    def without(l,i):
        """returns an input array l without entry i"""
        out = []
        for j in l:
            if(j != i):
                out.append(j)
        return out

    def find(l,e):
        """finds element e in list l and returns its index. return -1 if not in list"""
        for ind,j in enumerate(l):
```

```
205             if(e==j):
206                 return ind
207         return -1
208 
209 
210     def margin_2d(dist,i,interpol="none"):
211         """performs marginalization on three-dimensional probability distribution
212         until 2d probability is left and plots the result
213         interpol (str) sets the interpolation mode (default none)
214 
215         coordinate to leave out: i
216 
217         returns ndarray with marginalized distribution"""
218 
219         plt.figure()
220         marginalized = np.sum(dist,axis=i) #marginalize
221 
222         coord1,coord2 = without(range(3),i) #two marignalized coordinates
223         name1 = coord_names[coord1]
224         name2 = coord_names[coord2]
225 
226         ext = [min_vals[coord1],max_vals[coord1],min_vals[coord2],max_vals[coord2]] #extent
227 
228         plt.figure()
229         plt.imshow(marginalized,origin="lower",interpolation=interpol,extent = ext,aspect="auto") #plot
230         plt.title("marginalized distribution - "+name1+","+name2)
231         plt.xlabel(name1)
232         plt.ylabel(name2)
233         plt.colorbar()
234         plt.show()
235 
236 
237         #for c in [coord1,coord2]:
238         #    margin_1d(marginalized,i,c)
239         return marginalized
240 
241     def margin_1d(dist,i1,i2):
242         """
243         perform a marginalization on two-dimensional probability distribution until 1d probability
244         is left and plots the result
245 
246         i = coordinate to be marginalized (0,1)
247         c = coordinate index (for labeling)
248 
249         returns marginalized probability as numpy array
250         """
251 
252         plt.figure()
253 
254         marginalized = np.sum(dist.T,axis=i1) #marginalized probability distribution
255 
256         second_marg_index = find(without(range(3),i1),i2)
257         print("second marg index",second_marg_index)
258 
259         i = without(without(range(3),i1),i2)[0]
260         print("coordinate",i)
261         marginalized2 = np.sum(marginalized, axis=second_marg_index)
262 
263         print("sum",i1,second_marg_index)
264 
265         #print("marginalized2",marginalized2)
266         #print("mean",np.mean(marginalized2))
267         #print("fwhm",np.fwhm(marginalized2))
268 
269         ext = np.linspace(min_vals[i],max_vals[i],len(marginalized2))
270 
271         print("global mode",ext[np.argmax(marginalized2)])
272         print("global min",ext[np.argmin(marginalized2)])
273 
```

```
274
275         plt.grid(alpha=.2)
276
277         plt.scatter(ext,marginalized2,marker="<",s=6,color="k",label="computed points")
278         plt.legend()
279         plt.plot(ext,marginalized2)
280         plt.xlabel(coord_names[i])
281         plt.ylabel("P("+coord_names[i]+")")
282         plt.show()
283
284
285     #Marginal distribution is derived in Bishop p.88
286     #RESULT: sigma_ii for marginalized coordinate i
287     #..
288
289     inv = np.linalg.inv(M_hat)
290     sigm_0 = np.sqrt(inv[0,0])
291     sigm_1 = np.sqrt(inv[1,1])
292     sigm_2 = np.sqrt(inv[2,2])
293
294
295     mu_0 = lambd_hat[0]
296     mu_1 = lambd_hat[1]
297     mu_2 = lambd_hat[2]
298
299
300     data = [[0,1,2],[sigm_0,sigm_1,sigm_2],[mu_0,mu_1,mu_2]]
301     print("Standard deviation and mean")
302     print_table(data,header="|$i$ | $\sigma_i$ | $\mu_i = a_i $")
303
304     #marginalized distributions -1d
305     margin0 = margin_1d(uncert,0,1)
306     margin0 = margin_1d(uncert,2,0)
307     margin0 = margin_1d(uncert,1,2)
308
309     #marginalized distributions - 2d
310     margin0 = margin_2d(uncert,0)
311     margin1 = margin_2d(uncert,1)
312     margin2 = margin_2d(uncert,2)
```

B Code for the Bayesian and Frequentist analysis

```
1   def efficiency(E,E_0,A):
2       """Computes the efficiency of the detector given an energy E,
3       and detector parameters E_0, A"""
4
5       return np.sin(A*(E-E_0))
6       #return 1/(1+np.exp(-A*(E-E_0)))
7
8
9   E_0 = 1.9
10  A =  3.4
11
12  E_range = np.linspace(min(x),max(x),100,endpoint=True)
13
14  #plot data points
15  plt.scatter(x,[efficiency(i,E_0,A) for i in x],label="data points")
16
17  #plot sigmoid model
18  plt.plot(E_range,[efficiency(e,E_0,A) for e in E_range],"--",color="r",label="sigmoid model function")
19
20  #plt.legend()
21
22  plt.title("sigmoid detector efficiency for $E_0 = 2, A= 3$")
23  plt.xlabel("Energy E")
```

```
24  plt.ylabel("Efficiency $\epsilon$")
25  plt.show()
26
27
28  def posterior(r,N,E,A,E_0,sigma_A,sigma_E,mu_a,mu_E):
29      """Calculates the (unnormalized!) posterior distribution
30      given a set of success rates r and a set of trial numbers N
31
32      Parameters:
33
34      r, k-dimensional array (ndarray)
35      N, k-dimensional array (ndarray)
36      E, k-dimensional array (ndarray)
37
38      sigma_A,sigma_E,mu_a,mu_E,E_0,A_0 are numbers, parameters
39
40      Returns:
41
42      P(A,E_0) =  posterior distribution of the parameters A, E_0 as defined in lecture
43      (which is a number)
44      """
45
46      posterior = 1
47
48      for i in range(len(r)):
49          #prefactor
50          pre =  binom(N[i], r[i])*(1+np.exp(-A*(E[i]-E_0)))**(-r[i])*(1+np.exp(A*(E[i]-E_0)))**(r[i]-N[i])
51          posterior = posterior*pre*np.exp((-1/(2*sigma_A**2))*(A-mu_A)**2-(1/(2*sigma_E**2))*(E_0-mu_E)**2)
52
53      return posterior
54
55
56  def posterior_sin(r,N,E,A,E_0,sigma_A,sigma_E,mu_a,mu_E):
57      """Calculates the (unnormalized!) posterior distribution
58      given a set of success rates r and a set of trial numbers N
59
60      Parameters:
61
62      r, k-dimensional array (ndarray)
63      N, k-dimensional array (ndarray)
64      E, k-dimensional array (ndarray)
65
66      sigma_A,sigma_E,mu_a,mu_E,E_0,A_0 are numbers, parameters
67
68      Returns:
69
70      P(A,E_0) =  posterior distribution of the parameters A, E_0 as defined in lecture
71      (which is a number)
72      """
73
74      posterior = 1
75
76      for i in range(len(r)):
77          #prefactor
78          pre =  binom(N[i], r[i])*(np.sin(A*(E[i]-E_0)))**r[i]*(1-np.sin(A*(E[i]-E_0)))**(N[i]-r[i])
79          posterior = posterior*pre*np.exp((-1/(2*sigma_A**2))*(A-mu_A)**2-(1/(2*sigma_E**2))*(E_0-mu_E)**2)
80
81      return posterior
82
83
84  mu_A = 2
85  sigma_A = .5
86
87  mu_E = 2
88  sigma_E = .2
89
90  #posterior parameter space to be computed
91  n = 256 #number of data points
92  A = np.linspace(1.8,2.2,n)
```

```
93  E_0 = np.linspace(2.0,2.2,n)
94
95
96  #2D posterior distribution
97  import time
98  import progressbar
99
100 post_dist = np.empty((len(A),len(E_0))) #empty array for posterior distribution
101
102 bar = progressbar.ProgressBar()
103
104 #iterate through parameter space
105 for i in bar(range(len(A))):
106     for j in range(len(E_0)):
107
108         post_dist[i,j] = posterior_sin(r,N,x,A[i],E_0[j],sigma_A,sigma_E,mu_A,mu_E) #fill array with values
109
110 #normalize
111 s = np.sum(post_dist)
112 post_dist = post_dist.T/s
113
114 #counter-check
115 print("sum after normalizing:",np.sum(post_dist))
116
117 import matplotlib
118
119 #smallest interval... store values as [p,x,y]
120
121 matplotlib.rcParams.update({'font.size': 14})
122 plt.figure(figsize=(10,8))
123
124 def mark_smallest_interval(ax,alpha,color="blue",plotglobalmode=False):
125     values = []
126
127     for i in range(np.shape(post_dist)[0]):
128         for j in range(np.shape(post_dist)[1]):
129             values.append([post_dist[i,j],A[j],E_0[i]])
130
131
132     values = sorted(values,key= lambda x: x[0])
133
134     cum = 0
135     cum_vals_x = []
136     cum_vals_y = []
137
138     #pop values...
139     while(cum < alpha):
140         nxt = values.pop()
141         cum += nxt[0]
142         cum_vals_x.append(nxt[1])
143         cum_vals_y.append(nxt[2])
144
145
146     ax.scatter(cum_vals_x,cum_vals_y,s=25,color=color,marker=",",label="smallest "+ str(np.round(100*alpha,3))+"% interval")
147     if(plotglobalmode):
148         ax.scatter(cum_vals_x[0],cum_vals_y[0],s=80,facecolors='white', edgecolors='r',color="linen",label="global mode")#global mo
149
150
151
152 mark_smallest_interval(plt.gca(),.997,"red",False)
153 mark_smallest_interval(plt.gca(),.955,"yellow",False)
154 mark_smallest_interval(plt.gca(),.683,"green",True)
155
156
157 #plt.imshow(post_dist,extent = [min(A),max(A),min(E_0),max(E_0)],origin="lower",cmap="gray_r",interpolation="kaiser",alpha=.9)
158
159 plt.xticks(np.arange(min(A),max(A)+.01,(max(A)-min(A))/20.0))
160 plt.yticks(np.arange(min(E_0),max(E_0)+.01,(max(E_0)-min(E_0))/20.0))
161 plt.grid(alpha=.3)
```

```
162
163 plt.xlabel("$A$")
164 plt.ylabel("$E_0$")
165
166 plt.legend()
167
168 plt.show()
169
170
171
172 #compute the marginalized probability densiy from the 2D posterior
173
174 def marginal_A(post_dist):
175     """computes the marginal probability density for parameter A = mu_A"""
176     x,y = np.shape(post_dist)
177     #sum over all columns and normalize
178     #marginal = np.sum(post_dist,axis=1)
179
180
181     marginal = []
182     for i in range(x):
183         s = 0
184         for j in range(y):
185                s+= post_dist[j][i]
186         marginal.append(s)
187
188     #norm
189     norm = np.sum([marginal[i]*(A[i]-A[i-1]) for i in range(1,len(A))])
190     marginal = marginal/norm
191
192     return marginal
193
194 def marginal_E(post_dist):
195     """computes the marginal probability density for parameter E = mu_E"""
196
197     #sum over all rows and normalize
198     marginal = np.sum(post_dist.T,axis=0)
199
200     #norm
201     norm = norm = np.sum([marginal[i]*(E_0[i]-E_0[i-1]) for i in range(1,len(E_0))])
202     marginal = marginal/norm
203
204     return marginal
205
206
207 def confi_inter_1d(alpha,x,axis_vals):
208     #find confidence interval
209
210     mysum = np.sum(x)
211
212     my_vals = []
213     for i in range(len(x)):
214         my_vals.append([x[i], axis_vals[i]])
215
216     my_vals = sorted(my_vals)
217
218     #values to return
219     confi_int = []
220     vals = []
221
222     cum = 0
223     while(cum <= alpha*mysum):
224         nxt = my_vals.pop()
225         cum += nxt[0]
226         confi_int.append(nxt[1])
227         vals.append(nxt[0])
228
229     #print(cum)
230
```

```
231         #sort output
232         sorted_confi_vals = [[vals[i],confi_int[i]] for i in range(len(confi_int))]
233         sorted_confi_vals = sorted(sorted_confi_vals,key = lambda x: x[1])
234         confi2 = [i[1] for i in sorted_confi_vals]
235         vals2 = [i[0] for i in sorted_confi_vals]
236 
237 
238         return confi2,vals2 #return confidence interval and probabilty values
239 
240 
241     #posterior marginalized in A
242     plt.figure(figsize=(12,4))
243     mA_post = marginal_A(post_dist)
244     print("A* = ",A[np.argmax(mA_post)])  #print the mode
245 
246 
247     barwidth = 20/(n)
248 
249 
250     #== Confidence intervals
251     confi,vals = confi_inter_1d(.997,mA_post,A)
252     plt.fill_between(confi,vals,color="red",linewidth=0,interpolate=True,label="99.7% smallest interval")
253     #plt.bar(confi,vals,width=barwidth,color="red",edgecolor='red',align="center",linewidth=0.1,label="99.7% smallest interval")
254 
255     confi,vals = confi_inter_1d(.9,mA_post,A)
256     plt.fill_between(confi,vals,color="yellow",linewidth=0,interpolate=True,label="95.5% smallest interval")
257     #plt.bar(confi,vals,width=barwidth,color="yellow",edgecolor='yellow',align="center",linewidth=0.1,label="95.5% smallest interval")
258 
259 
260     confi,vals = confi_inter_1d(.6,mA_post,A)
261     plt.fill_between(confi,vals,color="green",linewidth=0,interpolate=True,label="68.3% smallest interval")
262     #plt.bar(confi,vals,width=barwidth,color="green",edgecolor='green',align="center",linewidth=0.1,label="68.3% smallest interval")
263 
264 
265     #== Plot
266     plt.step(A,mA_post,where='mid',alpha=.5)
267     plt.ylim([0,max(mA_post)])
268 
269     #== Plot labels
270     matplotlib.rcParams.update({'font.size': 14})
271 
272     ax = plt.gca()
273     box = ax.get_position()
274     ax.set_position([box.x0, box.y0, box.width, box.height])
275     plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
276 
277     plt.title("(unnormalized) marginal distribution for $A$")
278     plt.xlabel("$A$")
279     plt.ylabel("$P(A |$data$)$")
280 
281     #plt.grid()
282     plt.show()
283 
284     #posterior marginalized in E_0
285     plt.figure(figsize=(12,4))
286     mE_post = marginal_E(post_dist)
287 
288     print("E* = ",E_0[np.argmax(mE_post)])
289 
290 
291     barwidth = 0.02
292 
293     #== Confidence intervals
294     confi,vals = confi_inter_1d(.997,mE_post,E_0)
295     plt.fill_between(confi,vals,color="red",linewidth=0,interpolate=True,label="99.7% smallest interval")
296     #plt.bar(confi,vals,width=barwidth,color="red",align="center",alpha=.997,label="99.7% smallest interval")
297 
298 
299     confi,vals = confi_inter_1d(.9,mE_post,E_0)
```

```
plt.fill_between(confi,vals,color="yellow",linewidth=0,interpolate=True,label="95.5% smallest interval")
#plt.bar(confi,vals,width=barwidth,color="yellow",align="center",alpha=.955,label="95.5% smallest interval")


confi,vals = confi_inter_1d(.6,mE_post,E_0)
plt.fill_between(confi,vals,color="green",linewidth=0,interpolate=True,label="68.3% smallest interval")
#plt.bar(confi,vals,width=barwidth,color="green",align="center",alpha=.683,label="68.3% smallest interval")

#== Plot
plt.step(E_0,mE_post,where="mid",alpha=.5)


#== Plot labels
matplotlib.rcParams.update({'font.size': 14})
plt.title("marginal distribution for $E$")
plt.xlabel("E")
plt.ylabel("$P(E |$ data$)$")
#plt.grid()
ax = plt.gca()
box = ax.get_position()
ax.set_position([box.x0, box.y0, box.width, box.height])
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()









import random
random.seed(4283)

def P(r,N,f):
    """returns log p (success probability)"""
    results = []
    for i in range(len(r)):
        successes = 0
        for k in range(N[i]):

            prob = f(x[i])
            #print(x[i],prob)
            if(prob >random.random()):
                successes += 1


        results.append(successes)

    return results


A = np.linspace(1.86,2.06,16)
E_0 = np.linspace(2.1,2.2,16)

def log_xi(x,r,N,p):
    """computes test statistic"""
    return np.sum([np.log(binom(N[i],r[i])) + r[i]*np.log(max(1e-7,p(x[i]))) + (N[i]-r[i])*np.log(1-p(x[i])) for i in range(len(r))])




n= 500
Xi = np.empty((len(A),len(E_0),n))

bar = progressbar.ProgressBar() #progress bar

i = 0
tot = len(A)*len(E_0)
```

```
369
370 for a,a_i in enumerate(A):
371     for b,e_i in enumerate(E_0):
372
373         i += 1
374         bar.update(100*i/tot)
375
376         for k in range(n):
377             P_vals = P(r,N,lambda x: efficiency(x,e_i,a_i))
378             xi = log_xi(x,P_vals,N,lambda x: efficiency(x,e_i,a_i))
379             Xi[a,b,k] = xi
380
381
382
383 def draw_confi_hist(x,n_bins=80):
384     """draws a histogram with confidence intervals"""
385
386     #== histogram of the xi values
387
388     #compute histogram with numpy
389     hist = np.histogram(x,bins=n_bins,normed=True)
390     #plt.bar(hist[1][:-1],hist[0],width=100) #doesnt look so nice...better use matplotlib
391
392     #== Compute smallest confidence level interval
393
394     #99.7 % smallest confidence interval
395     confi_int,confi_vals = confi_inter_1d(.997,hist[0],hist[1][:-1])
396     mi99,ma99 = min(confi_int),max(confi_int)
397     print("99,7%",mi99,ma99)
398     plt.gca().axvline(mi99,color="darkred",linewidth=2,label="99,7% interval")
399     plt.gca().axvline(ma99,color="darkred",linewidth=2)
400     #plt.bar(confi_int,confi_vals,width=90,color="red")
401
402     #90 % smallest confidence interval
403     confi_int,confi_vals = confi_inter_1d(.9,hist[0],hist[1][:-1])
404     mi90,ma90 = min(confi_int),max(confi_int)
405     print("90%",mi90,ma90)
406     plt.gca().axvline(mi90,color="salmon",linewidth=2,label="90% interval")
407     plt.gca().axvline(ma90,color="salmon",linewidth=2)
408     #plt.bar(confi_int,confi_vals,width=90,color="yellow")
409
410     #68 % smallest confidence interval
411     confi_int,confi_vals = confi_inter_1d(.68,hist[0],hist[1][:-1])
412     mi68,ma68 = min(confi_int),max(confi_int)
413     print("68%",mi68,ma68)
414     plt.gca().axvline(mi68,color="green",linewidth=2,label="68% interval")
415     plt.gca().axvline(ma68,color="green",linewidth=2)
416     #plt.bar(confi_int,confi_vals,width=90,color="green")
417
418     confi_intervals = [[mi68,ma68],[mi90,ma90],[mi99,ma99]] #confidence interval boundaries
419
420     #plot histogram nicely
421     plt.hist(x,n_bins,alpha=.3,color="k",align='mid') #alpha = opacity of plot
422
423     plt.title(r"Histogram for $\xi$ grid points"+"\n")
424     plt.xlabel(r"$\log(\xi)$")
425     plt.xticks(rotation=45)
426
427     plt.ylabel(r"number of experiments")
428
429     ax = plt.gca()
430     box = ax.get_position()
431     ax.set_position([box.x0, box.y0, box.width, box.height])
432     plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
433
434     plt.show()
435
436 plt.figure()
437 plt.imshow(np.average(Xi,axis=2),interpolation="none")
```

```
438 plt.colorbar()
439 plt.show()
440
441
442 def plot_scatter(X):#== flatten the log xi array to 1D
443
444     #X_flat = X.flatten()
445     #print("total number of xi values:",np.shape(X_flat))
446
447     plt.figure(figsize=(12,8))
448     n_bins = 100
449
450
451     for i in range(np.shape(X)[0]):
452         for j in range(np.shape(X)[1]):
453
454             #== histogram of the xi values
455             n_bins = 80 #number of bins
456
457             #compute histogram with numpy
458             hist = np.histogram(X[i,j],bins=n_bins,normed=False)
459
460             #plt.bar(hist[1][:-1],hist[0],width=100) #doesnt look so nice...better use matplotlib
461
462             #== Compute smallest confidence level interval
463
464             #99.7 % smallest confidence interval
465             confi_int,confi_vals = confi_inter_1d(.997,hist[0],hist[1][:-1])
466             mi99,ma99 = min(confi_int),max(confi_int)
467
468             #90 % smallest confidence interval
469             confi_int,confi_vals = confi_inter_1d(.9,hist[0],hist[1][:-1])
470             mi90,ma90 = min(confi_int),max(confi_int)
471
472             #68 % smallest confidence interval
473             confi_int,confi_vals = confi_inter_1d(.68,hist[0],hist[1][:-1])
474             mi68,ma68 = min(confi_int),max(confi_int)
475
476             confi_intervals = [[mi68,ma68],[mi90,ma90],[mi99,ma99]] #confidence interval boundaries
477
478             if(xi_data >= mi68 and xi_data <= ma68): #xi of the data lies within the 68% interval?
479                 plt.scatter(A[i],E_0[j],s=50,color="green",marker="h",label="grid points")
480             elif(xi_data >= mi90 and xi_data <= ma90): #xi of the data lies within the 90% interval?
481                 plt.scatter(A[i],E_0[j],s=50,color="yellow",marker="h",label="grid points")
482             elif(xi_data >= mi99 and xi_data <= ma99): #xi of the data lies within the 99% interavl?
483                 plt.scatter(A[i],E_0[j],s=50,color="red",marker="h",label="grid points")
484             else:
485                 plt.scatter(A[i],E_0[j],s=50,color="black",marker="h",label="grid points")
486
487
488     plt.title(r"$\log(\xi)$ values in parameter space - confidence level intervals")
489     plt.xlabel("A")
490     plt.ylabel("$E_0$")
491     plt.show()
492
```

Maximum Likelihood Estimation

Thomas Baldauf

Winter semester 2017/18

1 Introduction

1.1 Motivation from Bayes' Theorem

We want to describe a physical observation (data D) with a model (parameters λ). From Bayes' theorem, the posterior probability $P(\lambda|D)$ is

$$P(\lambda|D) \sim P(D|\lambda)P_0(\lambda), \tag{1}$$

where the proportionality changes into an equality considering the normalization condition $P(D) = \int P(\lambda|D)\,\mathrm{d}\,\lambda = 1$. $P(D|\lambda)$ is called the likelihood of the data D given the parameters λ, and $P_0(\lambda)$ is the prior assigned to the parameters. If we use a flat (i.e. constant) prior, the posterior distribution is proportional to the likelihood of the data. We can abbreviate

$$\mathcal{L}(\lambda) := P(D|\lambda) \tag{2}$$

The likelihood represents the probability of the data. However, it is not normalized. The mode of the likelihood function $\mathcal{L}$ gives us a value for the most likely model parameters $\lambda_{\mathsf{MLE}} \equiv \lambda^*$.
It is important to understand that, in principle, we have three possibilities:

- observe the likelihood $P(D|\lambda)$ (maximum likelihood estimate, MLE)
- include a prior and look at the posterior distribution $P(\lambda|D)$ (maximum a-posteriori estimate, MAP)
- include the proportionality factor and perform a full Bayesian analysis

note that for the first option (MLE), we just need the data and the model parameters as an input. The second option (MAP) contains prior information and the last option gives us a complete distribution of the model parameters given the data, but also demands prior information, as well as large computation power. In Machine Learning, ways have been found to approximate the posterior probability using the observed data and unobserved variables (variational Bayes). The expectation maximization (EM) algorithm approximates the MAP estimate. But even if finding an approximate prior is possible, it still includes unobserved quantities. Estimating the maximum likelihood only uses the aquired data.

This is also in contrast to a frequentist analysis, where all (theoretically) possible experimental results are considered.
So if we have a fixed model with parameters λ which can take all values with equal probability, MLE is the right way to go.

1.2 Parameter uncertainties in likelihood analysis

The maximum likelihood estimator λ^* is just the mode of the likelihood function $\mathcal{L}$, and thus a single value. However, we do not know anything about the uncertainty $\Delta\lambda$ of this value. For a very sharp likelihood function, we expect $\Delta\lambda$ to be rather small, whereas we expect it to be large for a less sharp likelihood function. We can approximate the likelihood near its maximum by calculating its second derivative. If we assume a Gausssian shape of $\mathcal{L}$, we can set $\Delta\lambda$ to be the standard deviation of a normalized Gaussian, i.e.

$$\Delta\lambda = \left[\frac{\int(\lambda-\lambda^*)^2\mathcal{L}\,\mathrm{d}\,\lambda}{\int\mathcal{L}(\lambda)\,\mathrm{d}\,\lambda}\right]^{1/2} \tag{3}$$

If $\mathcal{L}$ is the result of N individual observations $\{X_i\}_{i=1}^N$, define

$$\mathcal{L}(\lambda) = P(D|\lambda) = \prod_{i=1}^{N} P(X_i|\lambda). \tag{4}$$

The log-likelihood is

$$\ln\mathcal{L}(\lambda) = L_N(\lambda) = \frac{1}{N}\sum_{i=1}^{N}\ln P(X_i|\lambda) \tag{5}$$

The law of large numbers tells us that for $N >> 1$,

$$\lim_{N\to\infty} L_N(\lambda) = L(\lambda) = \int \ln P(X_i|\lambda)P(x|\theta_0)dx = \mathrm{E}[\ln P(X|\lambda)]_{\lambda_0} \tag{6}$$

where λ_0 is the mode of $L(\lambda)$, i.e. some true value of the parameters that describes reality best.

1.3 Approximation of the likelihood using the Fischer information

We do not know the optimal model parameters λ. However, we know there must be a true value λ_0 which describes the physics behind our model best. We need to somehow measure how far our choice of λ is away from the true value λ_0. The Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(p||q)$ of two probability distributions p, q describes the information gain when q is used instead of p. We want to minimize the convex KL-divergence to get the optimal parameter $\hat{\lambda}$. In computer science, there exist various optimization algorithms. It makes also sense to use the KL-divergence as a measure of how well the model fits, as we wish to gain knowledge (i.e. information) about the data from our model. The KL-divergence is defined as the difference of the expected log likelihoods of two distributions

$$
\begin{aligned}
D_{\mathrm{KL}}(p(D)||q(D)) &:= \int_{-\infty}^{\infty} p(x) \ln\left(\frac{p(x)}{q(x)}\right) \mathrm{d}\, x \\
&= \mathrm{E}_{x\sim p}\left[\ln(p(x))\right] - \mathrm{E}_{x\sim p}\left[\ln(q(x))\right]
\end{aligned} \tag{7}
$$

We then optimize

$$
\begin{aligned}
D_{\mathrm{KL}}(p(D|\lambda)||p(D|\lambda_0) &= \mathrm{E}_{x\sim p(x|\lambda_0)}\left[\ln(p(x|\lambda))\right] - \mathrm{E}_{x\sim p(x|\lambda_0)}\left[\ln(p(x|\lambda_0))\right] \\
&= \mathrm{E}_x\left[\frac{\ln p(x|\lambda)}{\ln p(x|\lambda_0)}\right]_{\lambda_0}
\end{aligned} \tag{8}
$$

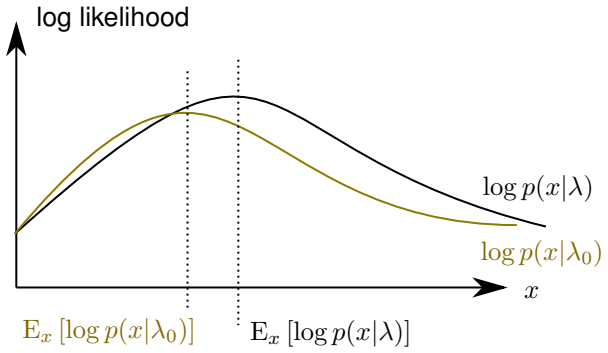


Figure 1: Illustration of the Kullback-Leibler divergence.

If the two distributions are very similar, we can approximate the KL-divergence in second order:

$$
\begin{aligned}
D_{\mathrm{KL}}(p(D|\lambda)||p(D|\lambda_0)) \approx & D_{\mathrm{KL}}(p(D|\lambda_0)||p(D|\lambda_0)) \\
& + (\lambda - \lambda_0)\nabla_\lambda D_{\mathrm{KL}}(p(D|\lambda)||p(D|\lambda_0)) \\
& + \frac{1}{2}\Delta\lambda_i \Delta\lambda_k \, \mathrm{H}_{ij}(D_{\mathrm{KL}}(p(D|\lambda)||p(D|\lambda_0)))_{\lambda_0} \\
& = \frac{1}{2}\Delta\lambda_i \Delta\lambda_j \, \mathrm{H}_{ij} \left(D_{\mathrm{KL}}(p(D|\lambda)||p(D|\lambda_0))\right)_{\lambda_0}
\end{aligned} \tag{9}
$$

where H is the Hessian of the KL-divergence, and the Einstein convention was used. The first term is zero from the definition, and the second term vanishes because we require a minimum.
We can now insert the definition of the KL-divergence and obtain

$$
D_{\mathrm{KL}}(p(D|\lambda)||p(D|\lambda_0) \approx \frac{1}{2}\Delta\lambda_i \lambda_j \, \mathrm{H}_{ij} \left(\mathrm{E}_{x\sim p(x|\lambda_0)}\left[\ln\left(\frac{p(x|\lambda)}{p(x|\lambda_0)}\right)\right]\right)_{\lambda_0} \tag{10}
$$

A more sloppy expression in terms of the parameter vector λ would be

$$D_{\mathrm{KL}} \approx \frac{1}{2}\Delta\lambda^2 \,\mathrm{E}_{x\sim p(x|\lambda_0)}\left[\frac{\partial^2}{\partial\lambda^2}\ln\left(\frac{p(x|\lambda)}{p(x|\lambda_0)}\right)\right]_{\lambda_0}$$
$$\sim \mathrm{E}\left[\frac{\partial^2}{\partial\lambda^2}\ln p(x|\lambda)\right]_{\lambda_0} - \mathrm{E}\left[\frac{\partial^2}{\partial\lambda^2}\ln p(x|\lambda_0)\right]_{\lambda_0} = I(\lambda_0) - I(\lambda) \tag{11}$$

In the last step, we used the definition

$$I(\theta_0) := -\,\mathrm{E}\left[\left(\frac{\partial^2 \ln P(X|\theta)}{\partial\theta^2}\right)_{\theta=\theta_0}\right]_{\theta_0} \tag{12}$$

where the minus sign comes from the fact that the information is conventionally defined as the negative log probability. I is called the Fisher information. It can actually be shown that minimizing the difference of the log likelihoods is equivalent to the minimization of the difference of their expectations.

1.4 Asymptotic properties of maximum likelihood estimators

If the KL-divergence becomes zero, the Fischer information for λ and λ_0 are equal. It does not necessarily mean that $\lambda = \lambda_0$, but we showed in the lecture, using the mean value theorem, that the distribution of $\lambda - \lambda_0$ is Gaussian :

$$P(\hat{\lambda} - \lambda_0) = \mathcal{N}(0, \frac{1}{NI(\theta_0)}) \tag{13}$$

where $\hat{\lambda}$ is the mode of $L_N(\lambda)$, and λ_0 is the mode of $L(\lambda)$. The latter statement holds under the assumption that the probability density function under consideration is "well-behaving", such that the central limit theorem can be applied.
The asymptotic properties ($N \to \infty$) fof a maximum likelihood esimator are

- consistent estimator:

$$\hat{\lambda} \to \lambda_0, \quad \lambda_0 \text{ true value} \tag{14}$$

- asymptotic normality:

$$\sqrt{N}(\hat{\lambda} - \lambda_0) \xrightarrow{d} \mathcal{N}(0, \sigma^2_{\lambda_0}), \quad \sigma_{\lambda_0} = \sqrt{\frac{1}{NI(\lambda_0)}} \approx \left[-\frac{\partial^2 \ln \mathcal{L}(\lambda)}{\partial\lambda^2}\right]^{-1/2} \tag{15}$$

2 The Family of Bernoulli Distributions

The family of Bernoulli distributions have the probability density $P(x|p) = p^x(1-p)^{1-x}$.
a) Calculate the Fischer information $I(p) = -\,\mathrm{E}\left[\frac{\partial^2 \ln P(x|p)}{\partial p^2}\right]$.

The second partial derivative of the logarithm of the probability density $P(x|p)$ with respect to p is

$$\begin{aligned}\frac{\partial^2}{\partial p^2}\ln P(x|p) &= \partial_p^2\left(x\ln p-(1-x)\ln(1-p)\right)\\ &= \partial_p\left(\frac{x}{p}-\frac{1-x}{1-p}\right) = -\frac{x}{p^2}-\frac{1-x}{(1-p)^2}\end{aligned} \tag{16}$$

The Fischer Information of the family of Bernoulli distributions is then

$$\begin{aligned}I(p) = -\,\mathrm{E}\left[\frac{\partial^2\ln P(x|p)}{\partial p^2}\right] &= \mathrm{E}\left[\frac{x}{p^2}+\frac{1-x}{(1-p)^2}\right] &(17)\\ &= \frac{\mathrm{E}[x]}{p^2}+\frac{1-\mathrm{E}[x]}{(1-p)^2} = \frac{1}{p}+\frac{1-p}{(1-p)^2}\\ &\overset{1-p\geq 0}{=} \frac{1-p}{p(1-p)}+\frac{p}{p(1-p)} = \frac{1}{p(1-p)} &(18)\end{aligned}$$

For N independent samples, we have

$$P(X|p) = \prod_{i=1}^{N} p^{x_i}(1-p)^{1-x_i} \tag{19}$$

$$\begin{aligned}\frac{\partial^2}{\partial p^2}\ln P(X|p) &= \partial_p^2\Big(\sum_{i=1}^{N}x_i\ln p-\Big(N-\sum_{i=1}^{N}\Big)\ln(1-p)\Big)\\ \Rightarrow I(p) = -\,\mathrm{E}\left[\frac{\partial^2\ln P(X|p)}{\partial p^2}\right] &= \frac{\sum_{i=1}^{N}\mathrm{E}[x_i]}{p^2}+\frac{N-\sum_{i=1}^{N}\mathrm{E}[x_i]}{(1-p)^2}\\ &= \frac{N}{p}+\frac{N-Np}{(1-p)^2} = \frac{N}{p(1-p)}\end{aligned} \tag{20}$$

b) What is the maximum likelihood estimator for p?
To find the maximum likelihood estimator, we set the derivative of the log likelihood to zero:

$$\begin{aligned}\frac{x}{\hat{p}}-\frac{1-x}{1-\hat{p}} &\overset{!}{=} 0\\ \Rightarrow &(1-\hat{p})x-\hat{p}(1-x) = 0 \Leftrightarrow \hat{p} = x.\end{aligned} \tag{21}$$

For N independent samples

$$\frac{\sum_{i=1}^{N} x_i}{\hat{p}} - \frac{N - \sum_{i=1}^{N} x_i}{(1-\hat{p})} = 0$$
$$\Leftrightarrow \sum_{i=1}^{N} x_i - p\sum_{i=1}^{N} x_i - \hat{p}N + \hat{p}\sum_{i=1}^{N} x_i = 0$$
$$\Leftrightarrow \hat{p} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{22}$$

We see that our result stays the same, even if we do not know the outcome of every single Bernoulli experiment, but just the total success rate.

c) What is the expected distribution for $\hat{p} - p_0$?
Like discussed in the introduction section, the expected distribution is Gaussian:

$$P(\hat{p} - p_0) = \mathcal{N}(0, \frac{1}{NI(p_0)}) \tag{23}$$

where $I(p_0)$ is the Fisher information calculated above.

2.1 The family of exponential distributions

The family of exponential distributions have pdf $P(x|\lambda) = \lambda e^{-\lambda x}$, $x \geq 0$.
a) Generate $n = 2, 10, 100$ values of x using $x = -\ln U$ where U is a uniformly distributed random number between 0 and 1. Repeat this for 1000 experiments and plot the distribution of the maximum likelihood estimator, $\hat{\lambda}$ (note that the true value in this case is $\lambda_0 = 1$).

The samples are drawn using the python code A. The output is visualized in figure 2. For $N = 2$ values, the histogram looks still very asymmetric. For $N = 100$ valus, it already exhibits a sharper, more Gaussian-like peak.

Find the MLE estimator from your generated data.

Looking at the log likelihood:

$$\ln P(X|\lambda) = N \ln \lambda - \lambda \sum_{i=1}^{N} x_i \tag{24}$$
$$\partial_\lambda \ln P(X|\lambda) = \frac{N}{\lambda} - \sum_{i=1}^{N} x_i = 0 \Leftrightarrow \hat{\lambda} = N / \sum_{i=1}^{N} x_i \tag{25}$$

The maximum likelihood estimator is just the inverse of the mean value of the sampled values. The mean values of the generated samples are

μ_2	μ_{10}	μ_{100}
0.9914	1.0077	1.0028

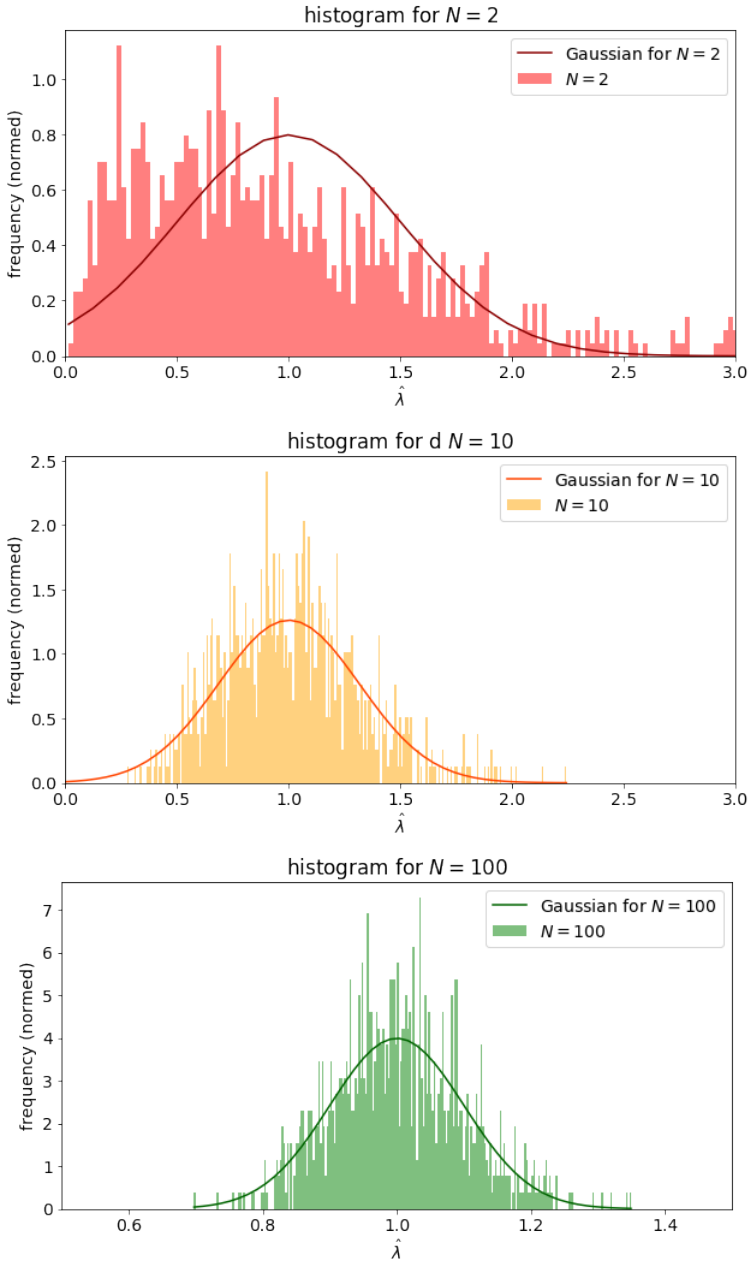


Figure 2: Top: Histogram of randomly generatd values from the exponential distribution, using a sample size $N = 2, 10, 100$, respectively. Bottom: histogram for $N = 1000$. The continuous lines show the Gaussian approximations of the distributions $P(\hat{\lambda} - \lambda_0)$.

The mean values (and thus their inverse values) seem to converge to 1. That is the true value λ_0 given in the task.

Compare the distributions you found for the MLE to the expectation from the Law of Large Numbers and CLT (see lecture notes) and discuss.

We again expect a Gaussian distribution

$$P(\hat{\lambda} - \lambda_0) = \mathcal{N}(0, \frac{1}{NI(\lambda_0)}). \tag{26}$$

The Fisher information of the exponential distribution is

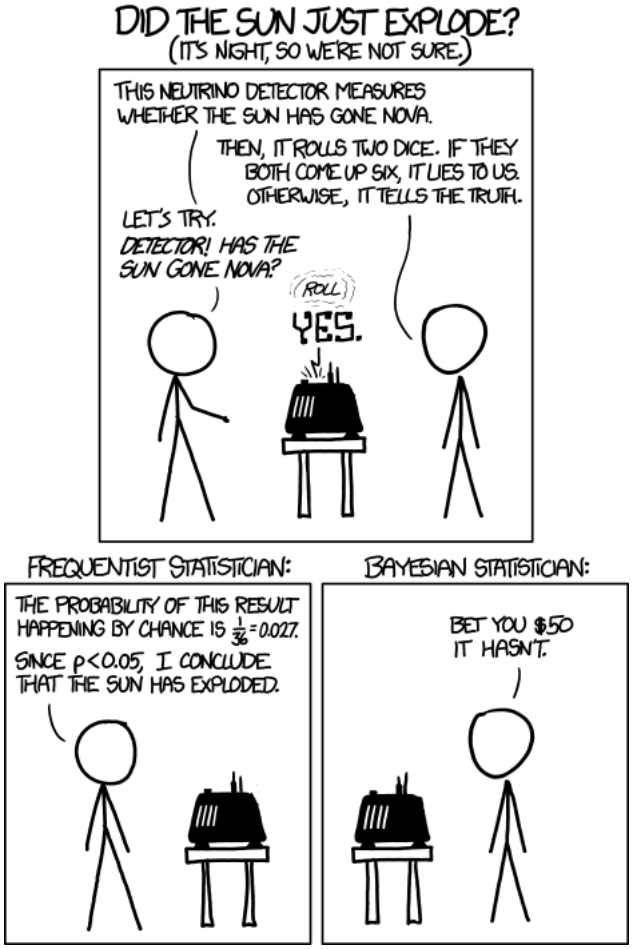
$$\begin{aligned} I(P) &= -\,\mathrm{E}\left[\frac{\partial^2 \ln P(x|\lambda)}{\partial p^2}\right]_{\lambda_0} = -\partial_\lambda \left(\frac{N}{\lambda} - \sum_{i=1}^{N} x_i\right)_{\lambda_0} \\ &= N/\lambda_0^2 \qquad (27) \\ \Rightarrow \sigma &= \sqrt{\frac{1}{\frac{N}{\lambda_0^2}}} = \frac{\lambda_0}{\sqrt{N}} = \frac{1}{\sqrt{N}} \qquad (28) \end{aligned}$$

The resulting Gaussians are plotted in figure 2 as continuous lines. As expected, with increasing N, they resemble the histogram better and better. Note that for the plots, the histograms were normed in order to represent probability densities.

A Sampling from exponential distribution

```python
import numpy as np

def exp_dist(x,l):
    """Computes the exponential distribution given x and
    parameter lambda(l)"""

    return l*np.ex(-l*x)

#generate 2, 10, 100 random numbers U in (0,1)
N = 1000

means2 = np.empty(N)
means10 = np.empty(N)
means100 = np.empty(N)

for i in range(N):

    rand2 = -np.log(np.random.ranf(2)) #random float in [0,1)
    rand10 = -np.log(np.random.ranf(10))
    rand100 = -np.log(np.random.ranf(100))

    means2[i] = (np.mean(rand2))
    means10[i] = (np.mean(rand10))
    means100[i] = (np.mean(rand100))
```

B Bayes vs. Frequentist


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY.
DETECTOR! HAS THE SUN GONE NOVA?
ROLL
YES.
FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS 1/36 = 0.027.
SINCE p<0.05, I CONCLUDE THAT THE SUN HAS EXPLODED.
BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.