



LECTURE 4: INTERVAL ESTIMATION

# STATISTICAL ANALYSIS IN EXPERIMENTAL PARTICLE PHYSICS

---

Kai-Feng Chen

National Taiwan University

# INTERVAL ESTIMATION

---

- The general goal of interval estimation — with a given **probability  $\beta$** , we want to find the range

$$\theta_a \leq \theta \leq \theta_b$$

which can contain the true value of the parameter of interest  $\theta_0$ .

- One can choose a large probability  $\beta$  (e.g. 90% or 99%), such that the interval indeed contains the true value.
- One can choose  $\beta=68.3\%$  or  $95.5\%$ , and derives the corresponding “errors” for “ $1\sigma$ ” and “ $2\sigma$ ”, although this is obviously only works for Normal distribution.
- Surely this estimate depends on the definitions of the probability used here. Thus the meaning of the interval, will be rather different for the **Bayesian method** and **frequentist method**.

# INTERVAL ESTIMATION (CONT.)

---

- Different methods of interval estimation which are usually considered:
  - **Normal theory interval estimation**, a text-book/elementary frequentist method. It is an asymptotic theory, which only works when the estimates are approximately Gaussian distributed (*very often the case?*).
  - **Likelihood-based method**, which is used by Minitab and already touched during the last lecture!
  - **Neyman construction**, which is an exact frequentist method. It was developed by Jerzy Neyman et al around 1930.
  - **Bayesian interval estimation**, again, is based on the Bayes Theorem. It can be treated as a relatively straightforward(?) extension of the Bayesian point estimation. One has to take care of the prior problems as usual.

 Let's start with Bayesian method this time!

# INTERVAL ESTIMATION: BAYESIAN

---

- Remember: one can claim that all the knowledge about the parameter is already **summarized in the posterior density  $P(\theta|X)$**  in the Bayesian parameter estimation.
- To compute an interval  $[\theta_L, \theta^U]$  which contains a given probability  $\beta$ , one has to find two points that fulfill the condition:

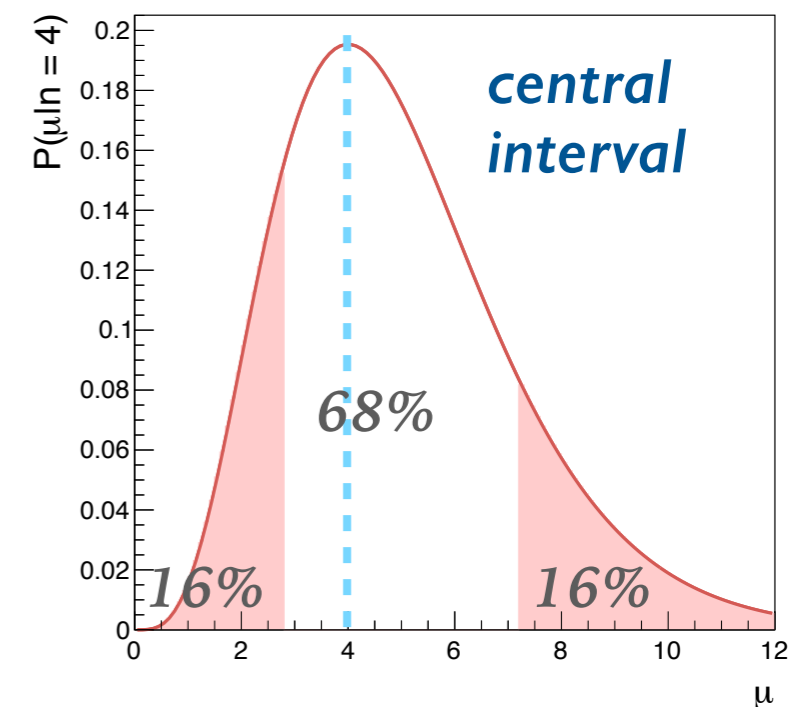
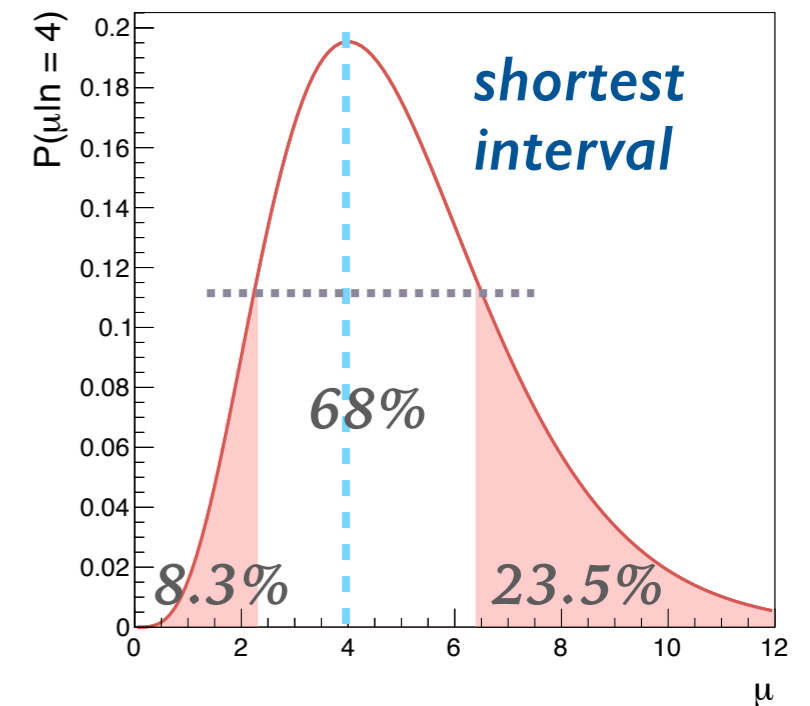
$$\int_{\theta_L}^{\theta^U} P(\theta|X) d\theta = \beta$$

- The usual choice of  $\beta$  either 68.3% or 90%. This represents the degree of belief that the true value of  $\theta$  lies within the given range.
- A Bayesian interval with probability  $\beta$  is named as a **credible interval**, while the frequentist version is called the **confidence interval**.



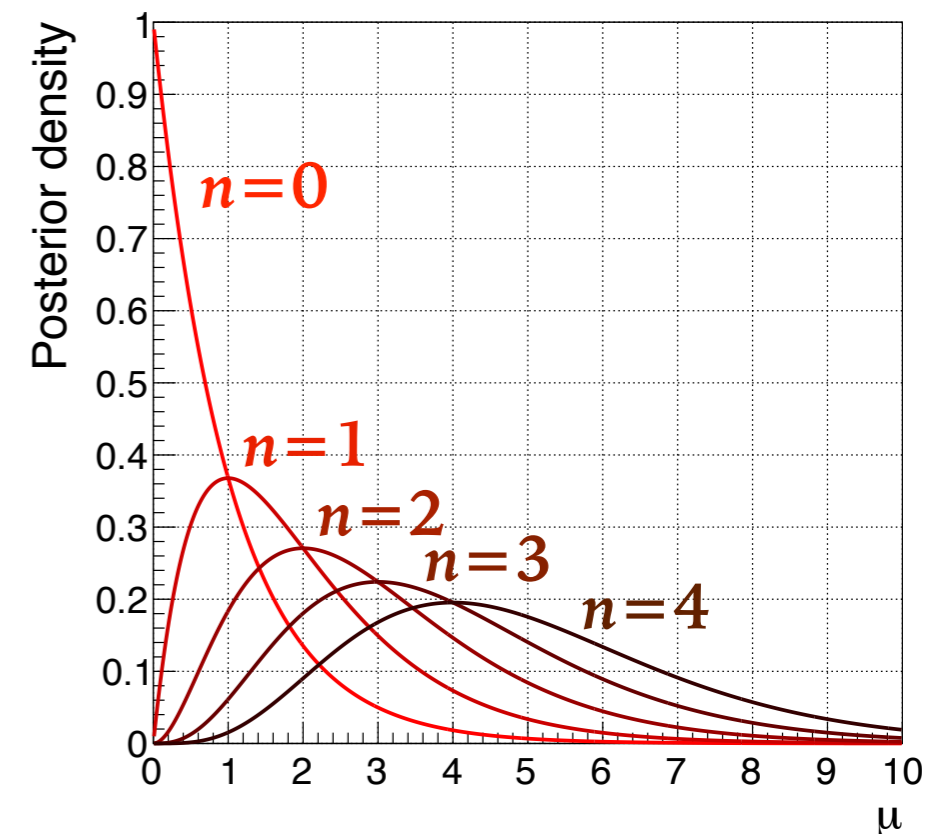
# INTERVAL ESTIMATION: BAYESIAN (CONT.)

- Just with the given probability  $\beta$ , the construction of the credible interval is not unique. It is common to impose additional choice:
- **Shortest interval:** integrate over the region with highest posterior density, until it reaches the given  $\beta$ . This interval is not invariant under parameter transformation.
  - **Central interval:** such that the integral of central part is  $\beta$  and in each side is  $(1-\beta)/2$ . Central interval is invariant under parameter transformation.
  - **One-sided interval:** usually for obtaining an upper limit, especially the case when  $\theta$  is near the end of the allowed region. One-sided intervals are invariant.



# NEAR THE PHYSICAL BOUNDARIES

- As a good feature: the **prior is always zero in the non-physical region** in the Bayesian method, this drives **the credible interval must be in the allowed region**.
- However a measurement near the edge of the physical region **will be biased toward the (*interior*) physically allowed region**.
  - In general this is a result since the **credible interval represents the belief** itself.
  - But this also means that now we cannot distinguish the information from the actual measurement or from the prior.



Bayesian posterior  
for Poisson with  
uniform prior

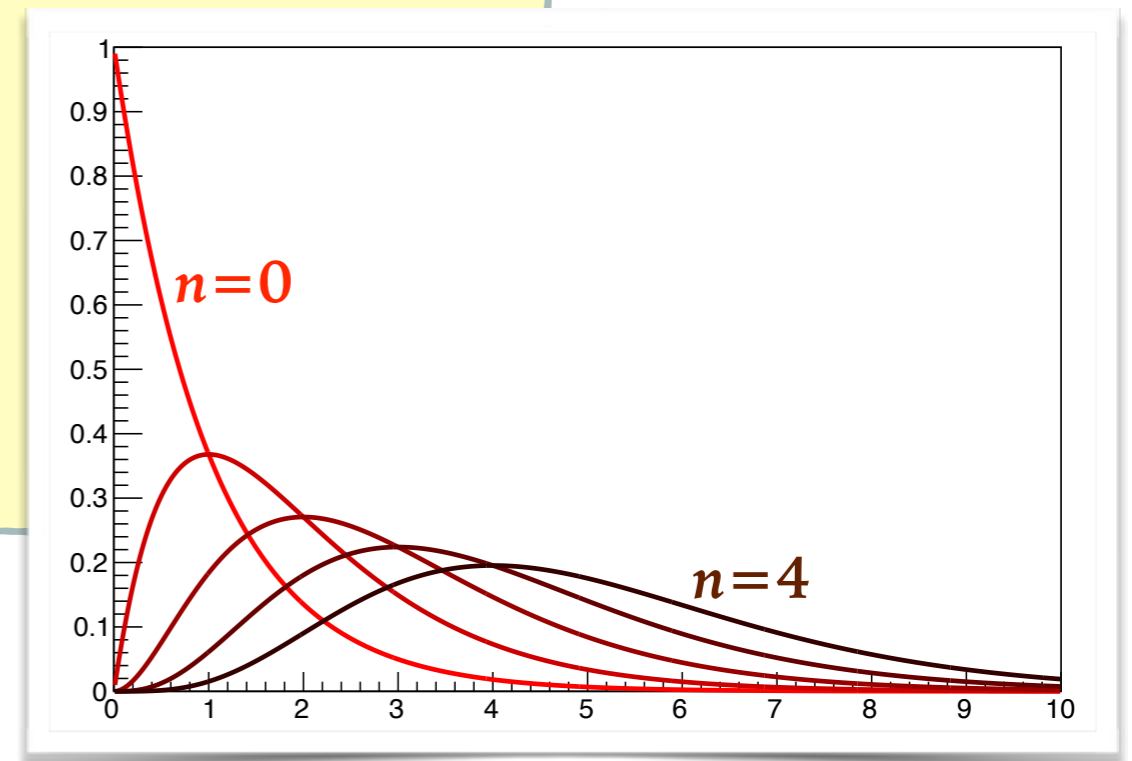
# EXAMPLE: POISSON POSTERIOR WITH UNIFORM PRIOR

- Here we provide you an example code to produce the Bayesian posterior distributions for Poisson with uniform prior, with observed  $n=0,1,2,3,4$ :

example\_01.cc

```
TH2D *frame = new TH2D("frame", "", 10, 0., 10., 10, 0., 1.0);
frame->SetStats(false);
frame->Draw();

for (int n=0; n<=4; n++) {
    TH1D *P = new TH1D(Form("P%d", n), "", 500, 0., 10.);
    for (int i=1; i<=P->GetNbinsX(); i++) {
        double mu = P->GetBinCenter(i);
        P->SetBinContent(i,
            TMath::Poisson(n, mu));
    }
    P->SetLineWidth(3);
    P->SetLineColor(kRed+n);
    P->Draw("csame");
}
```



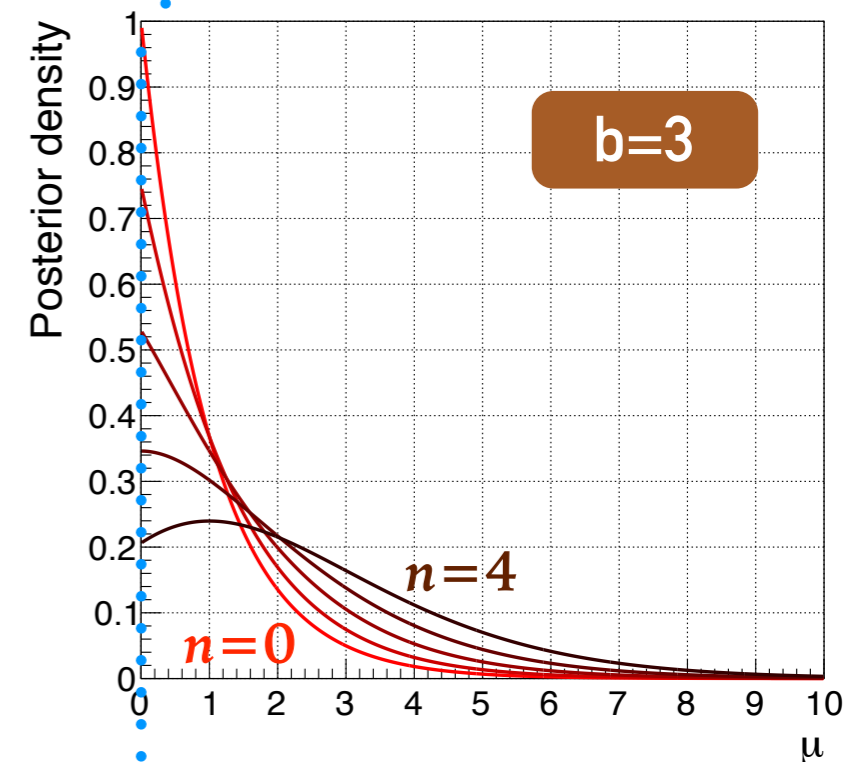
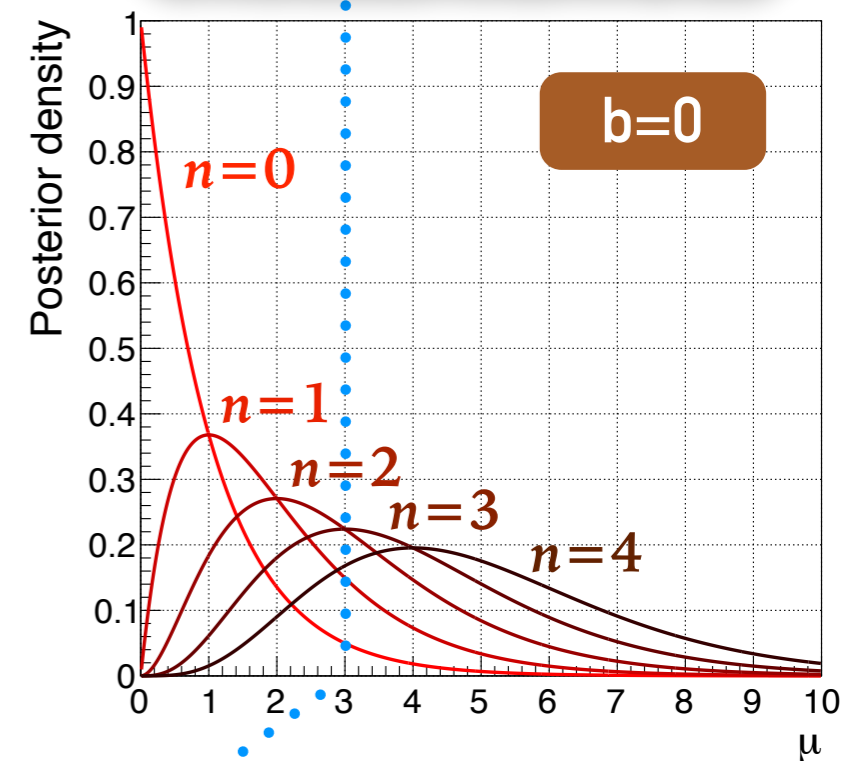
# POISSON WITH BACKGROUND

One can separate the following two cases:

- **Exactly known background expectation:** the amount of background  $b$  is known and is fixed. e.g.
  - Expected  $b=3.0$ , observed  $n=0$
  - Expected  $b=2.5$ , observed  $n=8$
- **Background expectation is measured with some uncertainty:** the amount of background  $b$  has an error associated with it. e.g.
  - Expected  $b=3.2 \pm 1.4$

In this case  $b$  should be treated as a **nuisance parameter**.

Bayesian posterior for Poisson with uniform prior





# EXPECTED BACKGROUND AS NUISANCE PARAMETER

---

- Everything has a probability distribution in the Bayesian framework, including the **nuisance parameters**!
- For example the distribution of background is described by a PDF  $P(\mathbf{b})$ . In the calculation of the **posterior density**, one has to integrate over the nuisance parameters, e.g.

$$P(\theta|X) = \int_{\mathbf{b}} \frac{P(X|\theta, \mathbf{b})P(\theta)}{P(X)} P(\mathbf{b})d\mathbf{b}$$

- This calculation might be heavy in terms of computing. However if background  $\mathbf{b}$  is exactly known as  $\mathbf{b}_0$ , the  $P(\mathbf{b})$  is just a delta function and the integration becomes trivial.

$$P(\theta|X) = \int_{\mathbf{b}} \frac{P(X|\theta, \mathbf{b})P(\theta)}{P(X)} \delta(\mathbf{b} - \mathbf{b}_0)d\mathbf{b} = \frac{P(X|\theta, \mathbf{b}_0)P(\theta)}{P(X)}$$

# EXAMPLE: POISSON POSTERIOR WITH NUISANCE

- Here are an example of calculating the Poisson posterior probability density with a nuisance parameter for background and is constrained by a Gaussian model.
- To make our life easier, the code is based on `RooStats::BayesianCalculator`. Surely you can also do all the calculation by yourself!

partial example\_02.cc

```
using namespace RooFit;  
using namespace RooStats;
```

```
RooRealVar x("x", "dummy obs", 0., 1.);  
RooUniform pdf_x("pdf_x", "dummy pdf of x", x);  
  
RooRealVar s("s", "# of signal", 0., 10.);  
RooRealVar b("b", "# of background", 3.2, 0., 10.);  
RooAddPdf pdf_splusb("pdf_splusb", "total PDF",  
    RooArgList(pdf_x, pdf_x), RooArgList(s, b));
```

Build a dummy extended s+b PDF for further use

```
RooUniform prior_s("prior_s", "prior for signal", s);  
RooGaussian prior_b("prior_b", "prior for background",  
    b, RooConst(3.2), RooConst(1.4));
```

times  
background prior  
(=constrained model)

```
RooProdPdf model("model", "constrained model", pdf_splusb, prior_b);
```

# EXAMPLE: POISSON POSTERIOR WITH NUISANCE (CONT.)

partial example\_02.cc

```
Rooworkspace wspace("wspace");  
ModelConfig cfg(&wspace);  
cfg.SetPdf(model);  
cfg.SetParametersOfInterest(s);  
cfg.SetPriorPdf(prior_s);  
cfg.SetNuisanceParameters(b);
```

```
Root *frame = s.frame();
```

```
for (int n=0; n<=4; n++) {
```

```
  RooRealVar nevt("nevt", "# of observed event", n);
```

```
  RooDataSet data("data", "data", RooArgSet(x, nevt), WeightVar("nevt"));
```

```
  data.add(RooArgSet(x), n);
```

```
  BayesianCalculator bcalc(data, cfg);
```

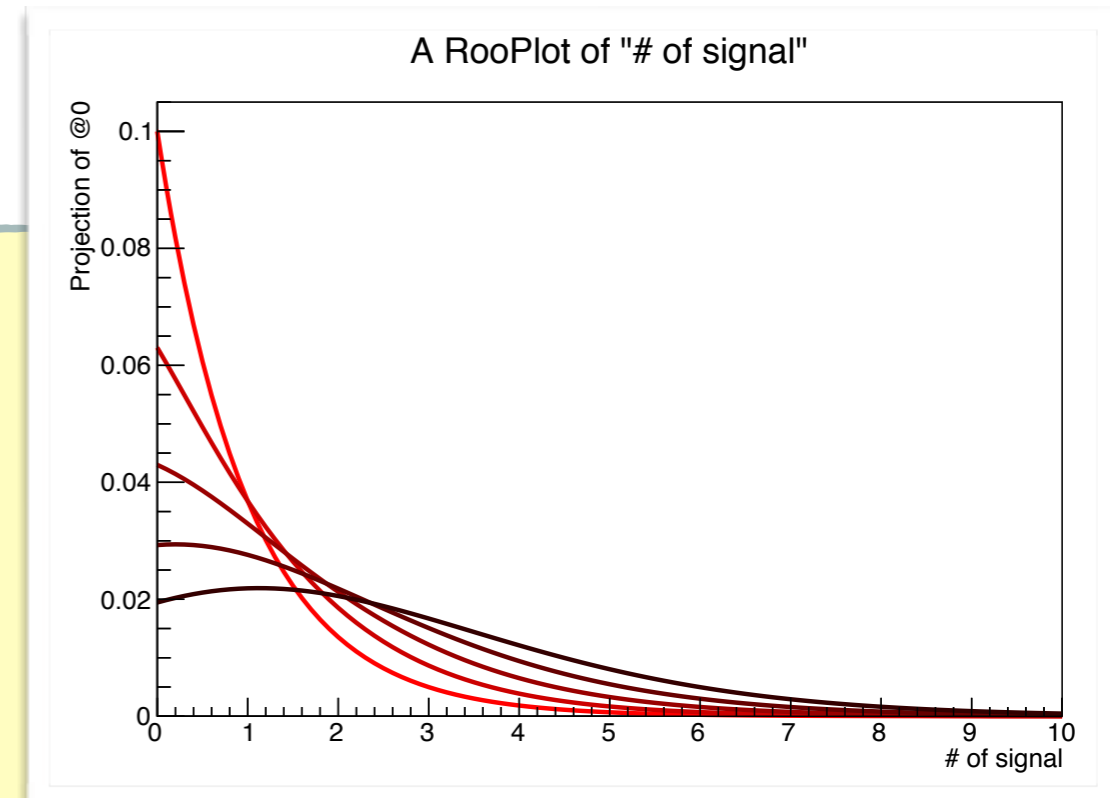
```
  RooAbsPdf *posterior = bcalc.GetPosteriorPdf();
```

```
  posterior->plotOn(frame, LineColor(kRed+n), LineWidth(3));
```

```
}
```

```
frame->Draw();
```

put in the  
needed stuff!



weighted data  
set with dummy  
observable

# POISSON WITH KNOWN BACKGROUND

- For the case of Poisson with known background and a **uniform prior**, the posterior density can be expressed as

$$P(\mu|n) \propto \frac{(\mu + b)^n}{n!} e^{-(\mu+b)}$$

- Let's practice the calculation of the 90% upper limit:

Observed	0	1	2	3
bkg = 0.0	2.30	3.89	5.32	6.68
0.5	2.30	3.51	4.84	6.18
1.0	2.30	3.27	4.44	5.71
2.0	2.30	2.99	3.88	4.93
3.0	2.30	2.84	3.52	4.36

You may find the results looks quite reasonable, no matter with or without the background. In particular **when n=0, the limits decouple from the background.**

- However a **uniform prior PDF  $P(\mu)$**  cannot represent the belief:

$$\int_a^b P(\mu) d\mu = 0$$

this will happen for any finite  $[a,b]$ , since  $P(\mu)$  has to be normalized in  $[0, \infty]$

# THEN... TRY SOMETHING ELSE?

---

- Since the uniform prior has such a “brief” problem, let’s examine the case of Jeffreys priors.
- Recall: Jeffreys priors are derived to be invariant under coordinate transformations.
- In particular **the prior  $1/\mu$**  is scale-invariant; it could represent the belief, since it goes to zero at infinity. But this does not work either since it goes to infinity when  $\mu=0$ .

*Bayesian 90% upper limit with  $1/\mu$  prior*

Observed	0	1	2	3
bkg = 0.0	0.00	2.30	3.89	5.32
0.5	0.00	0.00	0.00	0.00
1.0	0.00	0.00	0.00	0.00
2.0	0.00	0.00	0.00	0.00
3.0	0.00	0.00	0.00	0.00

The upper limits goes to zero since the posterior density goes to infinity when  $\mu$  approaches zero.



# THEN...TRY SOMETHING ELSE? (CONT.)

- There is another Jeffreys prior that minimizes the Fisher information contained in the prior:

$$P(\mu) = 1/\sqrt{\mu}$$

but obviously this does not solve the divergences problem with background.

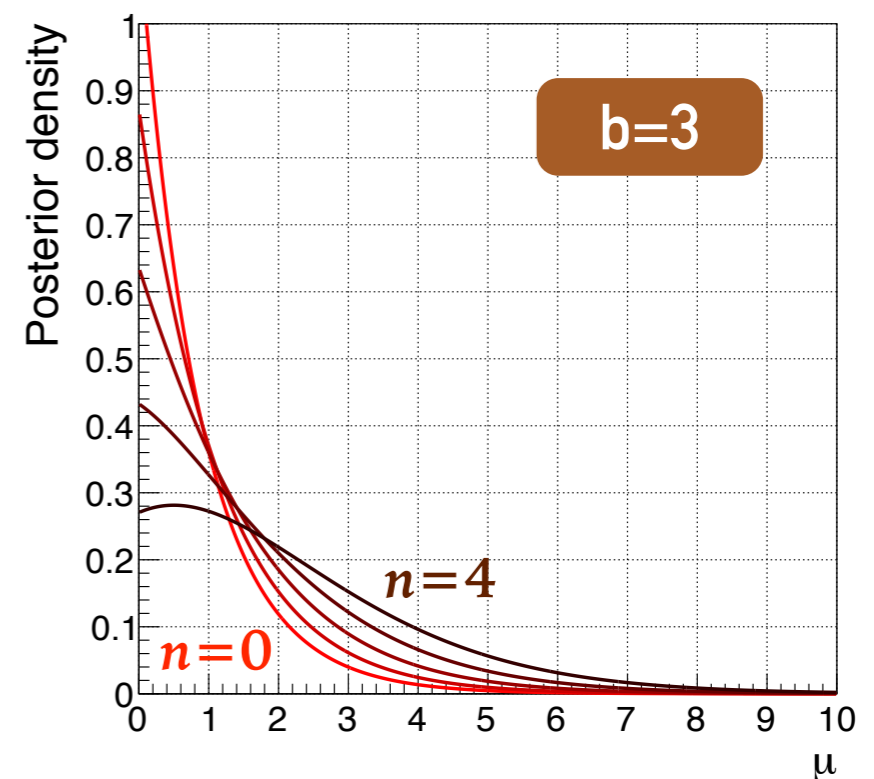
- The suggested Jeffreys prior to solve the case of Poisson with the expected background  $b$  is

$$P(\mu) = 1/\sqrt{\mu + b}$$

but this also means **the prior has to depend on the background level.**

*Bayesian 90% upper limit with prior  $1/\sqrt{\mu + b}$*

Observed	0	1	2	3
bkg = 0.0	1.35	3.13	4.62	6.01
0.5	1.81	2.88	4.17	5.52
1.0	1.92	2.76	3.84	5.07
2.0	2.03	2.63	3.41	4.38
3.0	2.08	2.55	3.16	3.92



# EXAMPLE: POISSON WITH KNOWN BACKGROUND

- Here are an example of calculating the upper limit table for the case of Poisson with **known background** and a **uniform prior**. Again we are using `RooStats::BayesianCalculator` here.

partial example\_03.cc

```
RooMsgService::instance().setGlobalKillBelow(RooFit::FATAL);
```

```
TCanvas *c1 = new TCanvas("c1", "", 800, 600);
```

```
c1->SetMargin(0.05, 0.05, 0.22, 0.05);
```

```
c1->Divide(4, 4, 0., 0.);
```

```
printf("\nObserved    0    1    2    3\n");
```

```
for (int b_set=0; b_set<4; b_set++) {
```

```
    printf("bkg = %d  ", b_set);
```

```
    for (int n_set=0; n_set<4; n_set++) {
```

```
        RooRealVar x("x", "dummy obs", 0., 1.);
```

```
        RooUniform pdf_x("pdf_x", "dummy pdf of x", x);
```

```
        RooRealVar s("s", "# of signal", 1E-5, 15.);
```

```
        RooRealVar b("b", "# of background", b_set);
```

```
        RooAddPdf model("model", "total PDF",
```

```
            RooArgList(pdf_x, pdf_x), RooArgList(s, b));
```

```
        RooUniform prior_s("prior_s", "prior for signal", s);
```

Build a dummy  
extended PDF

# EXAMPLE: POISSON WITH KNOWN BACKGROUND (II)

partial example\_03.cc

```
Rooworkspace wspace("wspace");
ModelConfig cfg(&wspace);
cfg.SetPdf(model);
cfg.SetParametersOfInterest(s);
cfg.SetPriorPdf(prior_s);
```

```
RoorealVar nevt("# of event", n_set);
RooDataSet data("data", "data",
    RooArgSet(x, nevt), WeightVar("nevt"));
data.add(RooArgSet(x), n_set);
```

```
BayesianCalculator bcalc(data, cfg);
bcalc.SetLeftSideTailFraction(0.);
bcalc.SetConfidenceLevel(0.9);
SimpleInterval* interval = bcalc.GetInterval();
printf("%.2f ", interval->UpperLimit());
```

```
c1->cd(b_set*4+n_set+1);
RooPlot *frame = bcalc.GetPosteriorPlot();
frame->Draw();
```

```
}
printf("\n");
```

```
}
```

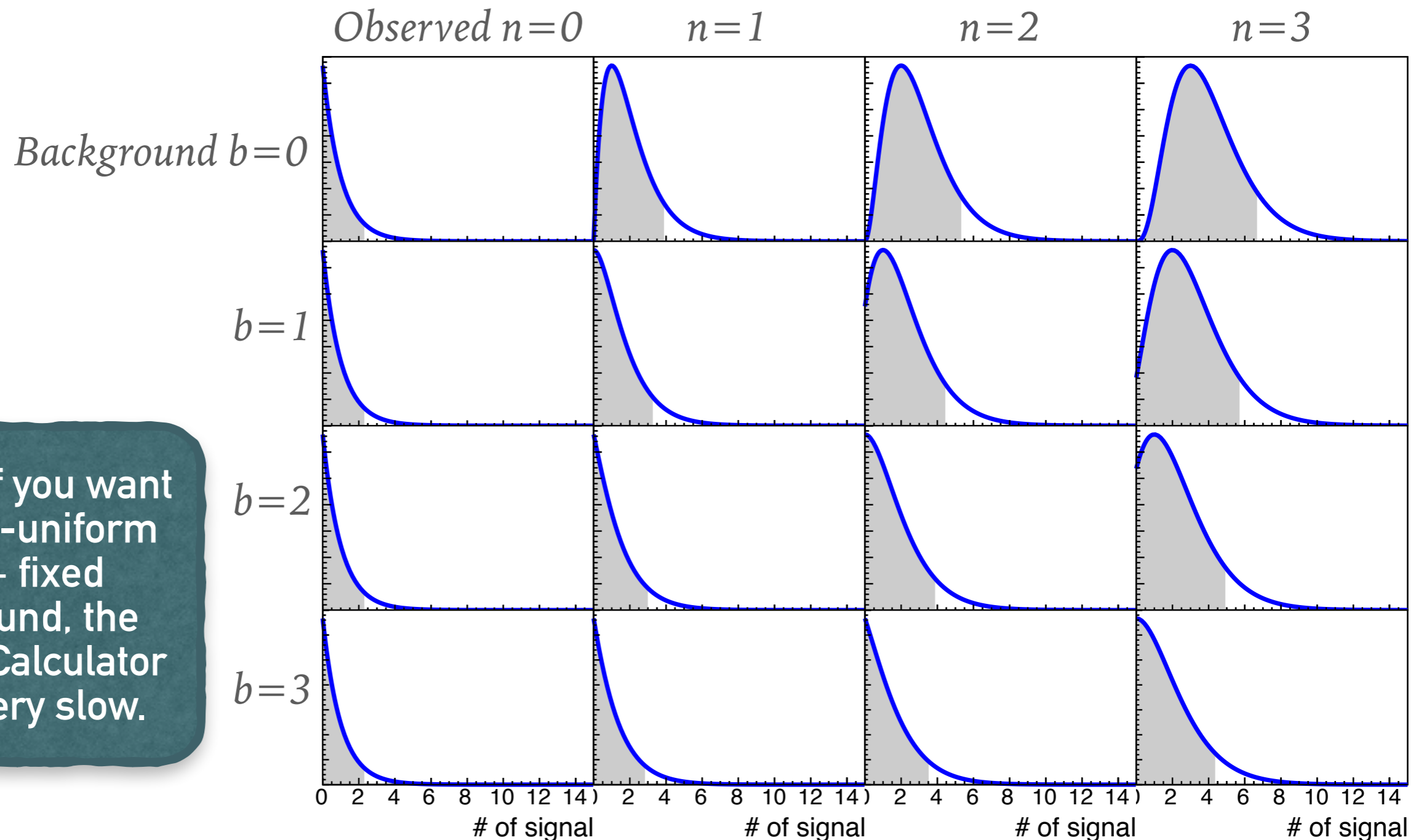
Observed	0	1	2	3
bkg = 0	2.30	3.89	5.32	6.68
bkg = 1	2.30	3.27	4.44	5.71
bkg = 2	2.30	2.99	3.88	4.93
bkg = 3	2.30	2.84	3.52	4.36

Ask the calculator to evaluate 1-side upper limit

You are encouraged to play with different priors & adding background nuisance PDF!

# EXAMPLE: POISSON WITH KNOWN BACKGROUND (III)

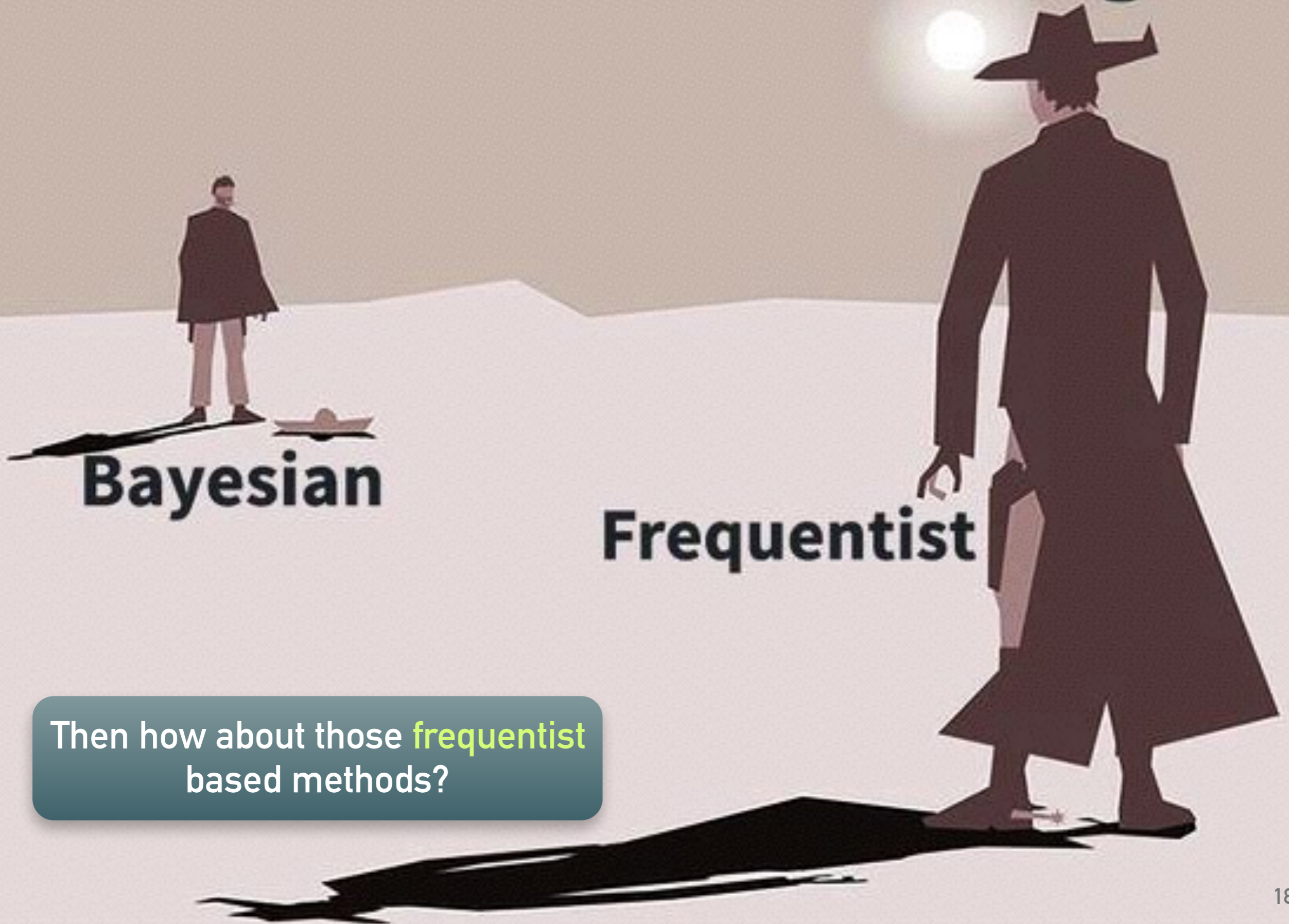
- Also get the posterior density for each configuration:



Remark: if you want to try non-uniform prior + fixed background, the BayesianCalculator will be very slow.



# Mathematics Debate Night



Then how about those frequentist based methods?



# INTERVAL ESTIMATION: FREQUENTIST

---

- The central problem: given a probability  $\beta$ , find the optimal range  $[\theta_a, \theta_b]$  in the space of  $\theta$  and:

$$P(\theta_a \leq \theta_0 \leq \theta_b) = \beta$$

where  $\theta_0$  is the true value of  $\theta$ . The interval  $(\theta_a, \theta_b)$  is a **confidence interval**.

- A method gives an intervals  $(\theta_a, \theta_b)$  satisfying the equation above is said to have the **property of coverage**. Note: the  $(\theta_a, \theta_b)$  are random variables,  $\theta_0$  is not.
- If an interval does not hold the property of coverage, it cannot be a confidence interval in fact. Although one can still consider **approximate confidence intervals**, which have only approximate coverage.
  - **Over-coverage** means when  $P > \beta$ .
  - **Under-coverage** means when  $P < \beta$  (*bad!*)

# NORMAL THEORY INTERVAL ESTIMATION

---

- A data  $X$  is sampling from a Gaussian distribution  $N(\mu, \sigma)$ .
- The relation between the given probability and the interval is straightforward when both the mean  $\mu$  and variance  $\sigma^2$  are known:

$$\beta = P(a \leq X \leq b) = \int_a^b N(X; \mu, \sigma) dX$$

- **When  $\mu$  is unknown**, the probability content of the interval  $[a, b]$  cannot be calculated anymore. But it is possible to evaluate the probability  $\beta$  where  $X$  lies in some interval **relative to its unknown mean**, e.g.  **$[\mu+c, \mu+d]$** , with a simple transformation  $Y = (X-\mu)/\sigma$ :

$$\beta = P(\mu + c \leq X \leq \mu + d) = \int_{\mu+c}^{\mu+d} N(X; \mu, \sigma) dX = \int_{c/\sigma}^{d/\sigma} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}Y^2\right) dY$$


*(no  $\mu$  here!)*

- Re-arrange the inequalities inside the probability gives:

$$\beta = P(\mu + c \leq X \leq \mu + d) = P(X - d \leq \mu \leq X - c)$$

# NORMAL THEORY INTERVAL ESTIMATION (II)

---

- This is working since (by shifting the calculation of probability  $\beta$  around the given data  $X$ ):
  -  The data are distributed according to Gaussian, and the Gaussian PDF is **symmetric** in  $X$  and  $\mu$ , since it is a function only of  $(X-\mu)^2$ .
  - We have assumed that the integration of both variables can be carried out in both directions, as far as they go, i.e., there is **no colliding of physical boundaries**.
- According to the theory of point estimation discussed in the previous lecture, both of the above bullets are true for **the maximum likelihood and least squares estimators**, asymptotically (valid when  $N \rightarrow \infty$ ).
- So we already have **an asymptotically working theory of interval estimation!**

The extension to multiple variables is also straightforward!

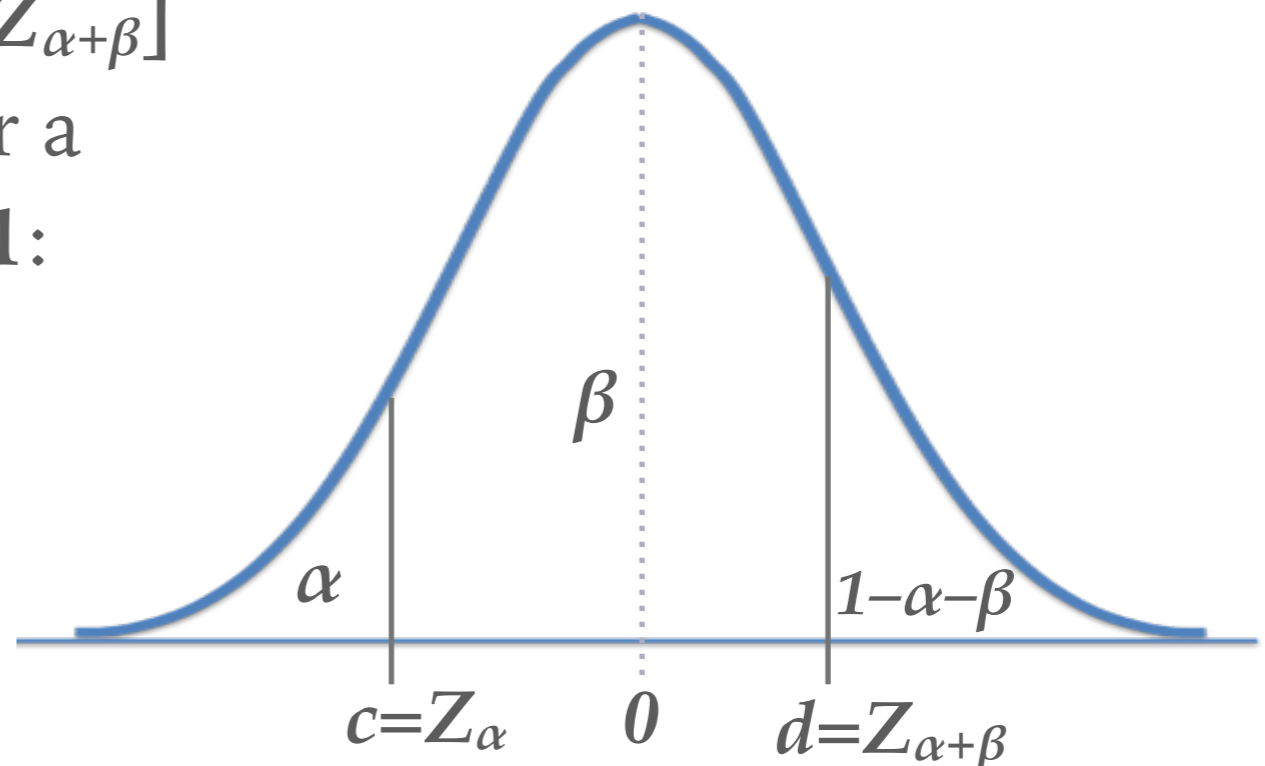
# NORMAL THEORY INTERVAL ESTIMATION (III)

---

- This is something very similar to what we already discussed when introducing Gaussian distribution itself!
- Given a random variable  $X$  with Gaussian PDF  $f(X)$  and the cumulative distribution  $F(X)$ , the “ $\alpha$ -point”  $X_\alpha$  is defined by

$$\int_{-\infty}^{X_\alpha} f(X) dX = F(X_\alpha) = \alpha$$

- The interval  $[c,d]$  is just  $[Z_\alpha, Z_{\alpha+\beta}]$  obviously, as shown below for a Gaussian PDF with  $\mu=0, \sigma=1$ :



# NORMAL THEORY INTERVAL ESTIMATION (IV)

---

- For a given value of  $\beta$ , there are many possible choice of value  $\alpha$ , and hence there are multiple intervals as a result.
- Surely the most common choice is  $\alpha = (1-\beta)/2$ , which gives the **central interval**, and is symmetric around zero.
- This is what we have seen before, for a standard Normal distribution:

$\beta=(1-\alpha)/2$	$Z_\alpha$	$Z_{\alpha+\beta}$
0.6827	-1.00	+1.00
0.9000	-1.65	+1.65
0.9500	-1.96	+1.96
0.9545	-2.00	+2.00
0.9900	-2.58	+2.58
0.9973	-3.00	+3.00



# INTERVAL IN MULTIPLE VARIABLES

---

- In the case of more than one dimension, the Gaussian PDF can be expressed as (*note we have discussed this in the earlier lecture once!*):

$$f(X) \propto \exp \left[ -\frac{1}{2} (X - \mu)^T \cdot V^{-1} \cdot (X - \mu) \right] = \exp \left( -\frac{1}{2} Q \right)$$

- The quantity  $Q = (X - \mu)^T V^{-1} (X - \mu)$  is the **covariance form** of  $X$ , and it follows a  **$\chi^2$  distribution with  $k$  degrees of freedom**. This means  $Q$  is not really dependent on  $\mu$ , we can use  $\chi^2$  distribution to define the  $\beta$  point ( $K_\beta^2$ ):

$$P[Q(X, \mu) \leq K_\beta^2] = \beta$$

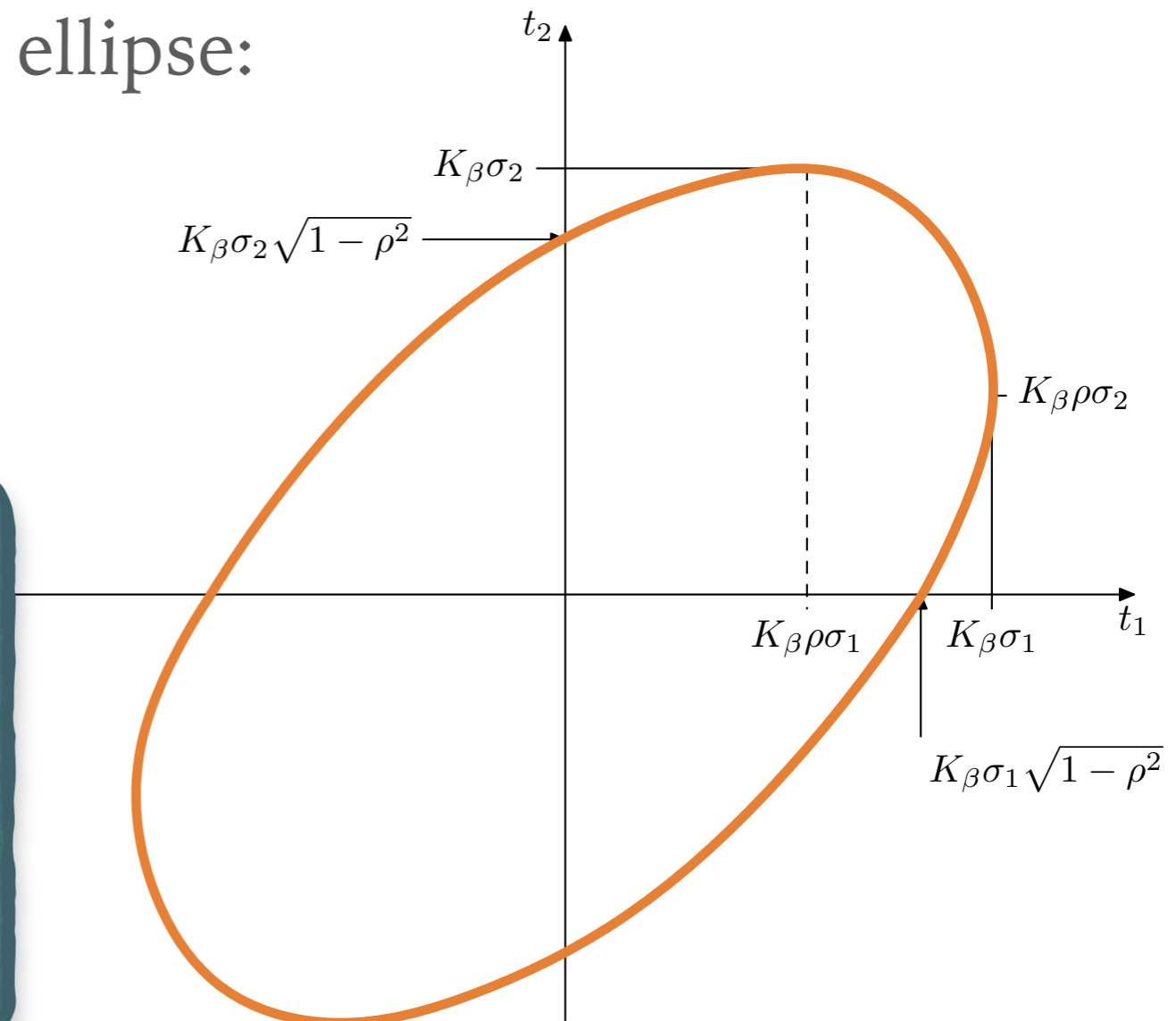
- Thus the confidence interval becomes a **confidence region** with probability content  $\beta$ , defined by  $Q \leq K_\beta^2$ .

# CONFIDENCE REGION IN 2D

- Consider two Normally distributed variables  $(t_1, t_2)$  with the corresponding covariance matrix  $V$  (which contains the correlation parameter  $\rho$ ).
- This is just the standard error ellipse:

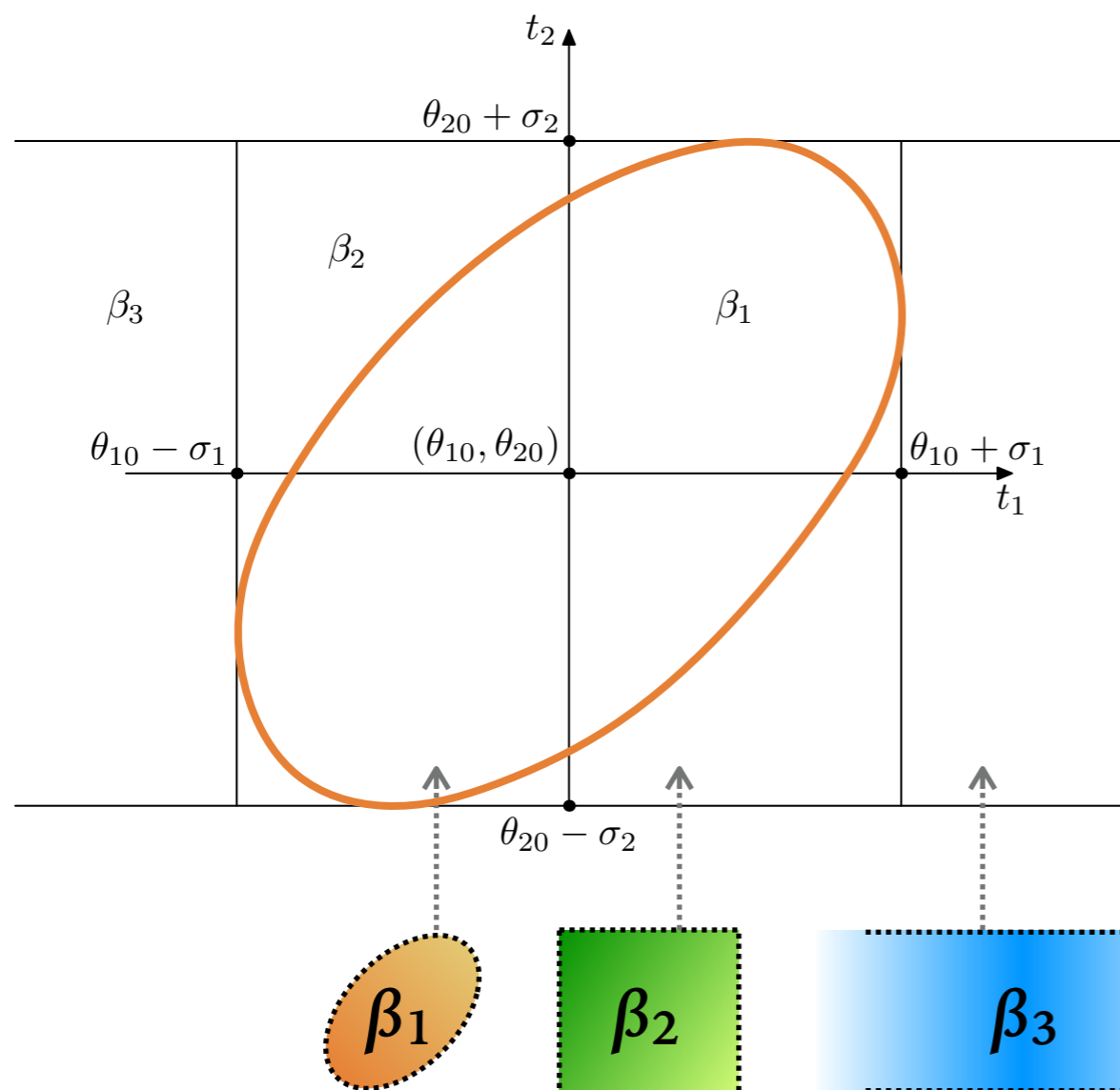
$$V = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

- If  $\rho$  is positive (negative), the major axis of the ellipse has a slope of  $+1$  ( $-1$ ).
- If  $\rho=0$ , the axes of the ellipse coincide with the coordinate axes.
- If  $\rho=1$ , the ellipse degenerates to a diagonal line.



# CONFIDENCE REGION IN 2D (CONT.)

- For two Gaussian-distributed variables, the probability contents of the three regions for different values of  $K_\beta$  and different correlation  $\rho$ .



	$K_\beta=1$	$K_\beta=2$	$K_\beta=3$
$\beta_1$	0.393	0.865	0.989
$\beta_2$	$\rho=0.00$	0.466	0.911
	$\rho=0.50$	0.498	0.917
	$\rho=0.80$	0.561	0.929
	$\rho=0.90$	0.596	0.936
	$\rho=0.95$	0.622	0.941
	$\rho=1.00$	0.683	0.954
$\beta_3$	0.683	0.954	0.997

*$\beta_1$  &  $\beta_3$  are independent of  $\rho$ !*

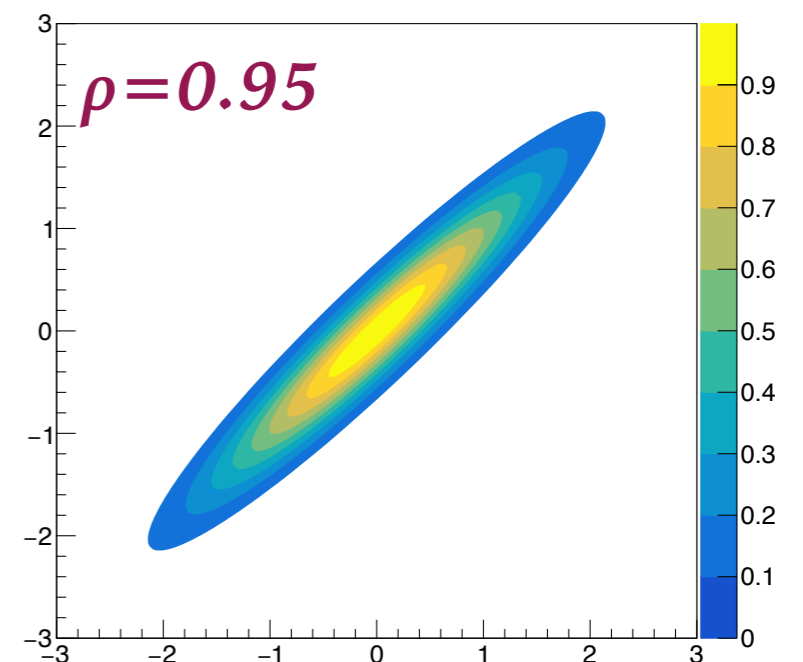
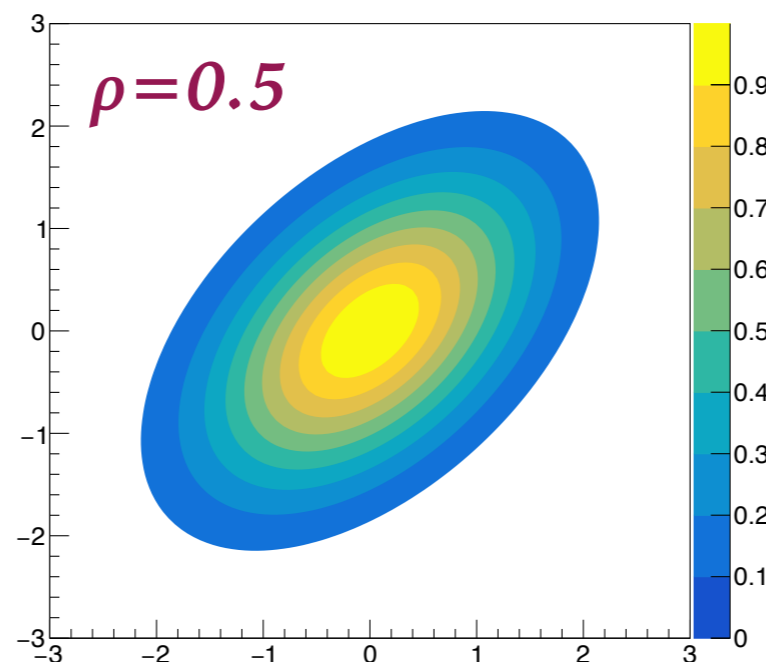
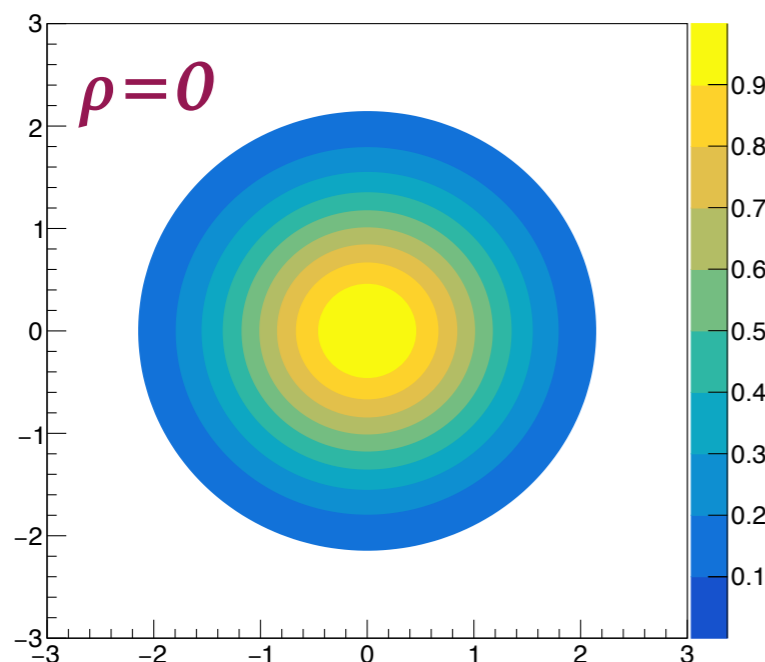
# EXAMPLE: DRAWING THE CORRELATED 2D GAUSSIAN

- Here we try to draw the contours with a 2D correlated Gaussian distribution, as a straightforward demonstration of the correlation  $\rho$ !

example\_04.cc

```
TF2 *func = new TF2("func", "exp(-0.5*(x*x+y*y-2*[0]*x*y)/(1-[0]*[0]))", -3, 3, -3, 3);  
func->SetParameter(0, 0.9); ← Put the  $\rho$  value here!
```

```
func->SetContour(10);  
func->SetNpx(100);  
func->SetNpy(100);  
func->Draw("cont0z");
```



# LIKELIHOOD-BASED CONFIDENCE INTERVALS

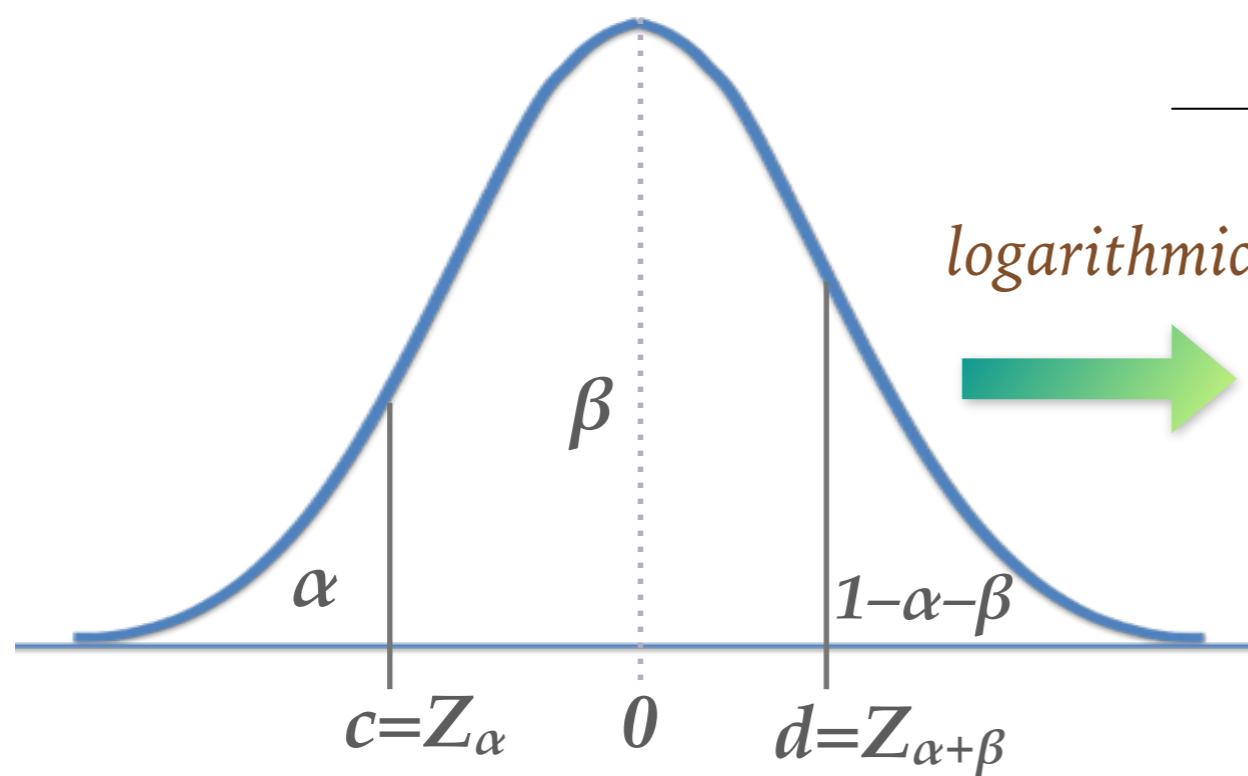
---

- Now we quickly revisit the method we already touched in the previous lecture: **likelihood-based confidence intervals**. It is kind of intermediate method between the Normal theory intervals (*discussed above!*) and the exact frequentist method (*to be discussed afterwards*).
- This method was suggested by D. Hudson, a statistician working at CERN in 1964 — it was implemented already in Minuit since 1966! At the time, the properties of this method were not fully understood and only works for simple (single-parameter) problems.
- This is exactly the “**MINOS**” method, as the Minuit command which calculates this confidence interval. Statisticians started to study it for the multi-parameter case, ie. the **profile likelihood method**. It turns out to have a good coverage!

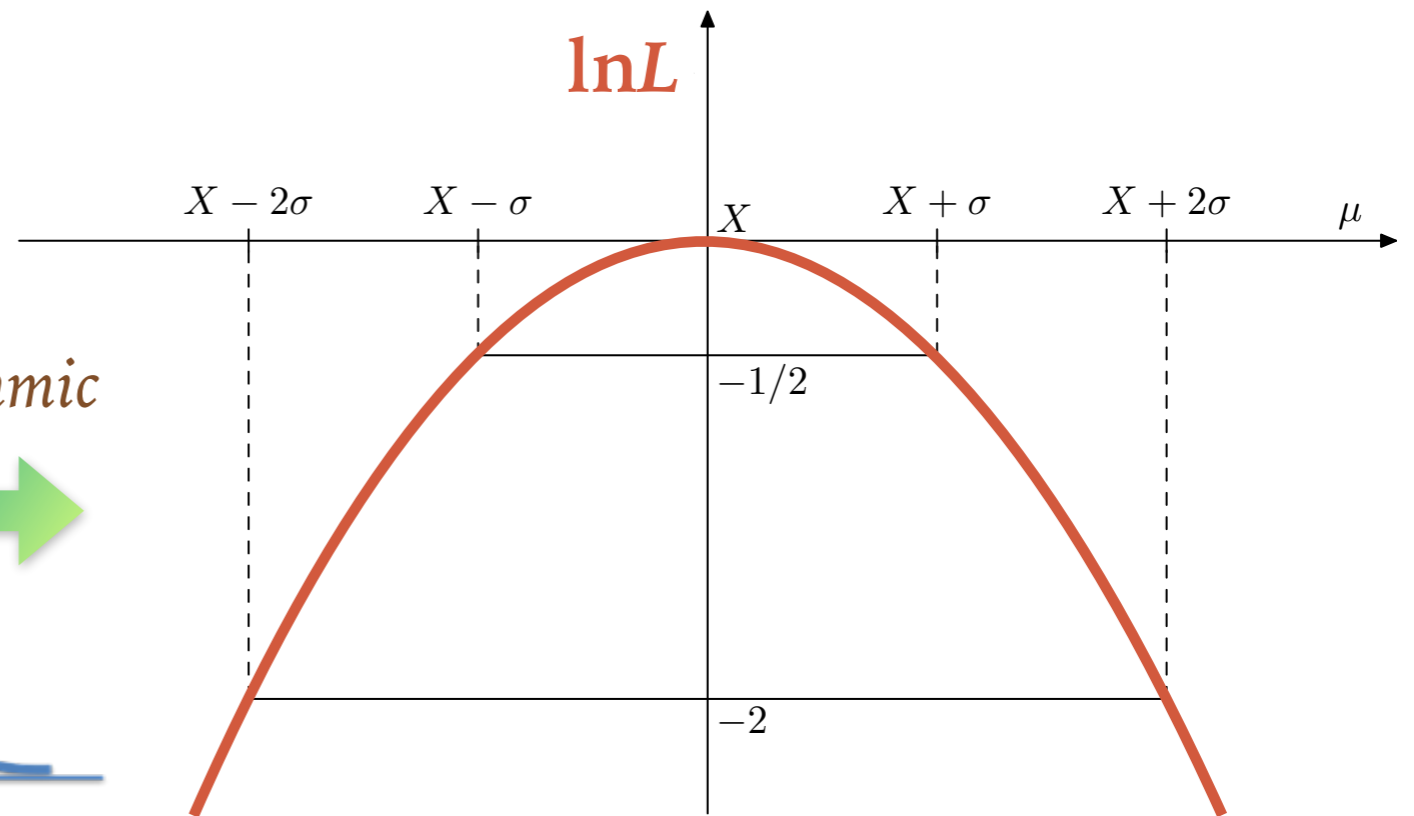


# LIKELIHOOD-BASED CONFIDENCE INTERVALS (II)

- Recall the Normal Theory, we have showed how to convert the PDF into a likelihood function. Now if one takes the logarithm of the likelihood function, the Gaussian becomes a parabola.



Gaussian distributed  $N(\mu, \sigma)$ .



Log-likelihood function for Gaussian  $X$ , distributed  $N(\mu, \sigma)$ .

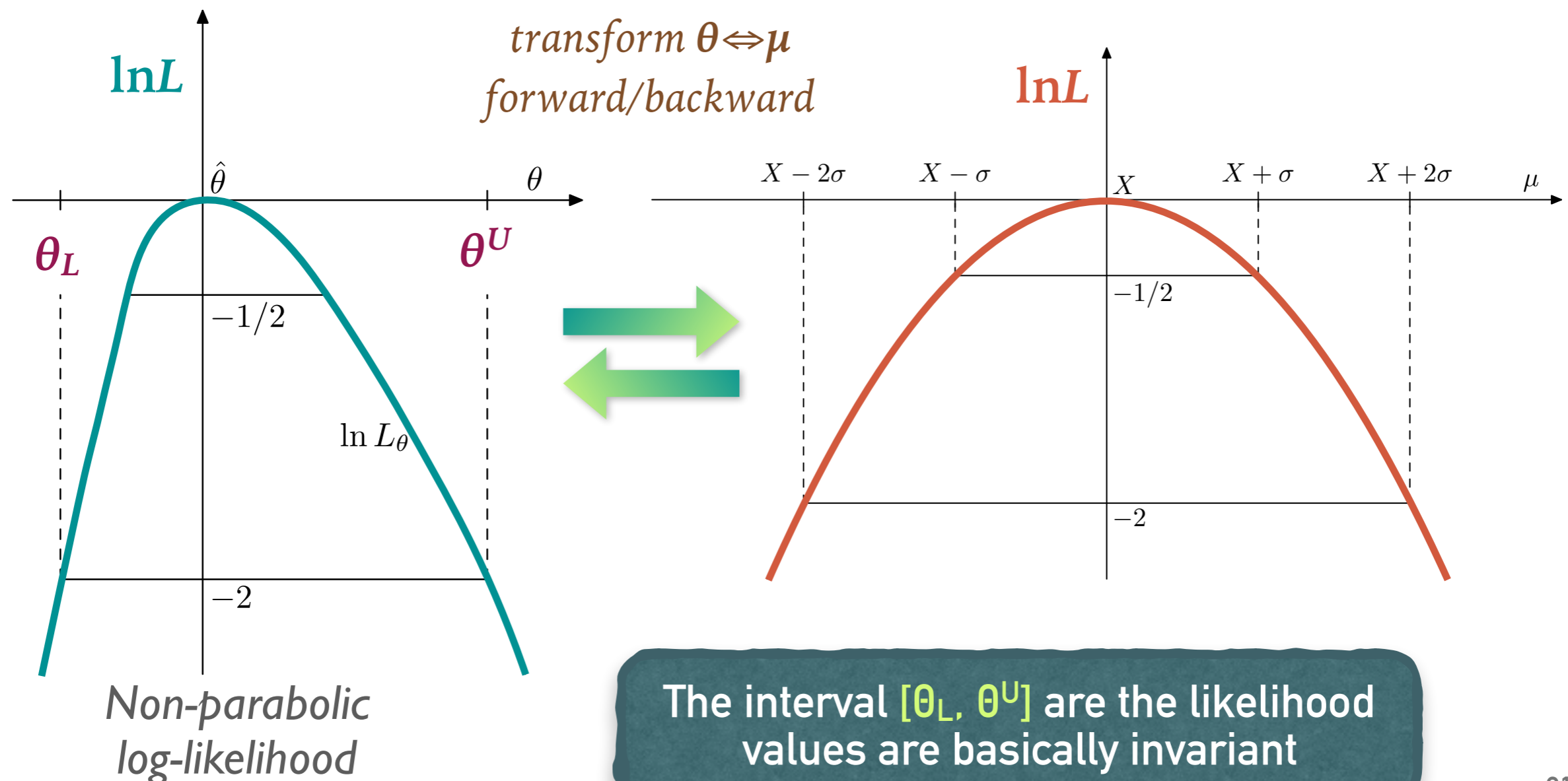
# LIKELIHOOD-BASED CONFIDENCE INTERVALS (III)

---

- When the data are Gaussian-distributed, the Normal Theory can be applied. In this case the confidence intervals can be calculated easily, the log-likelihood would have an exact parabolic shape as shown before.
- Then how about an inverse case: **a parabolic log-likelihood function also implies that the Normal theory is applicable!**
  - Generally if the log-likelihood function is **not parabolic**, it can be (*mostly always*) transformed to a parabola by a **transformation** of the parameter  $\Rightarrow$  Normal theory applied.
  - However, the values of the likelihood are actually invariant under transformation, it is not necessary to find what is exactly the transformation, if we just need the likelihood itself.

# LIKELIHOOD-BASED CONFIDENCE INTERVALS (IV)

- Since the likelihood values are invariant, one only needs to catch the parameter values when  $2\ln L = 2\ln L_{\max} - 1$ , for the standard one-sigma confidence interval.

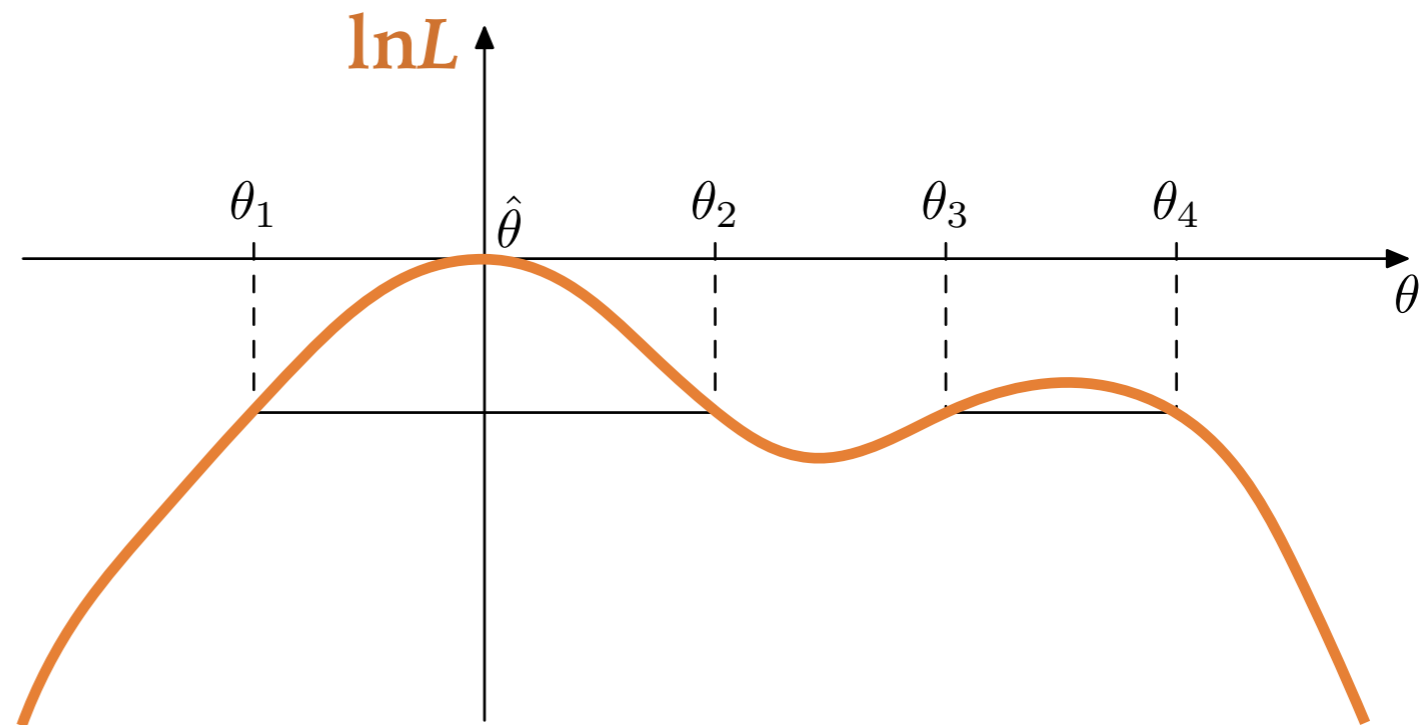


# LIKELIHOOD-BASED CONFIDENCE INTERVALS (V)

---

- In fact this method does not have good coverage for the simplest problem, i.e. Poisson distribution with very few events and no background, or some cases with “pathological” log-likelihood function, e.g.

You will run into the problem of **local minimum** in this case.

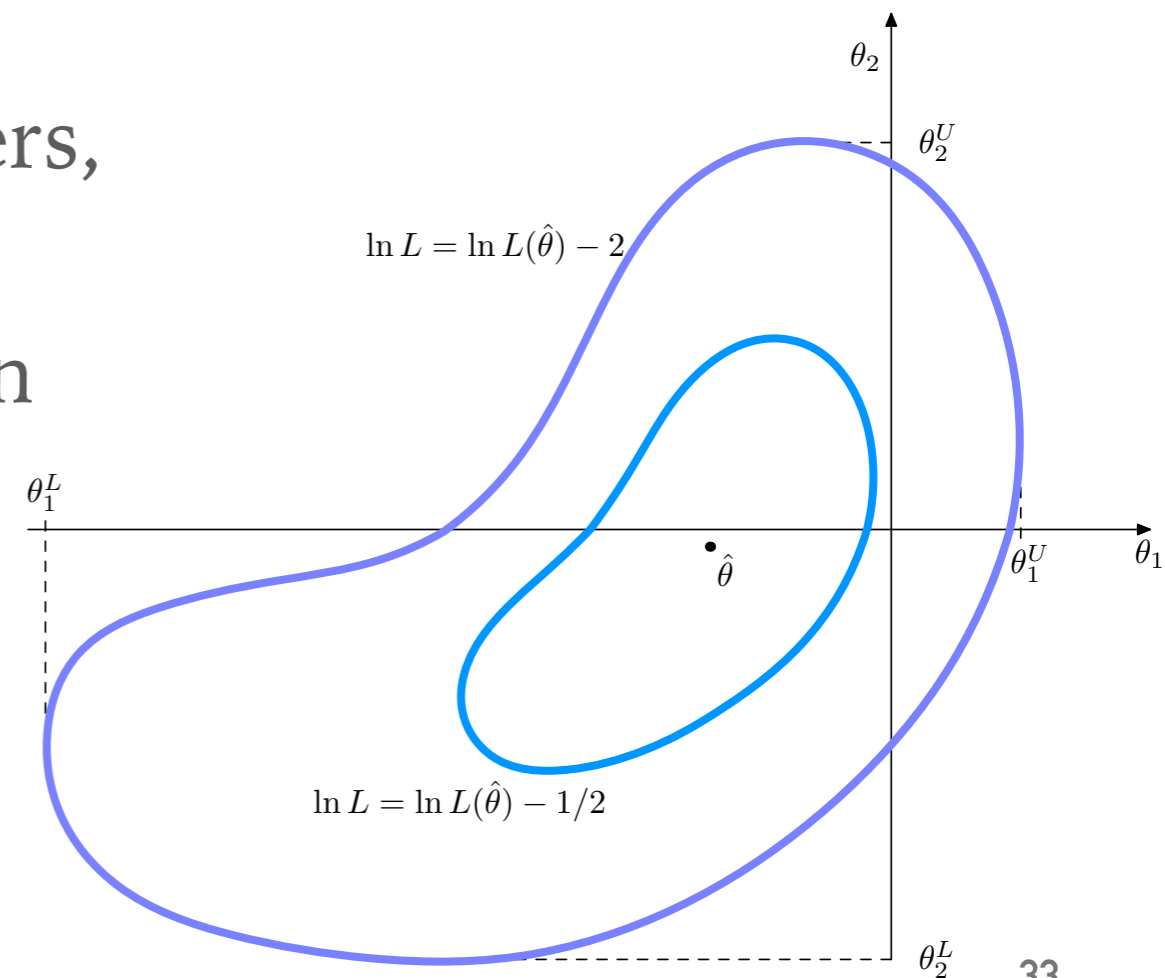


- However this method turns out to have very good coverage for large numbers of parameters, so it is good for handling the **nuisance parameters**.

# THE “MINOS” COMMAND

---

- The command MINOS in Minuit finds the intersection of the log-likelihood function with  $2\ln L/L_{\max}=1$ . This gives an *asymmetric confidence interval* in general case.
- The Normal theory always gives a *symmetric interval* around the best the estimated value.
- When there are multiple parameters, the MINOS error is calculated by maximizing the likelihood function with respect to all other floated parameters, a.k.a. the *profile likelihood method*.





# NEYMAN CONSTRUCTION OF CONFIDENCE INTERVALS

---

- Polish mathematician Jerzy Neyman, together with Egon Pearson produced two major contributions in statistics:
  - **The Neyman construction of confidence intervals**
  - **The Neyman-Pearson Test** (*to be discussed in the next lecture*)
- As for finding the exact frequentist intervals, the first important step is to work in the right space:  
**P(data | hypothesis)** — *with one axis for data, and another one for hypotheses*
  - In fact trying to place “true values” and “measured values” on the same axis is not a good approach, since hypotheses and data are living in different spaces.

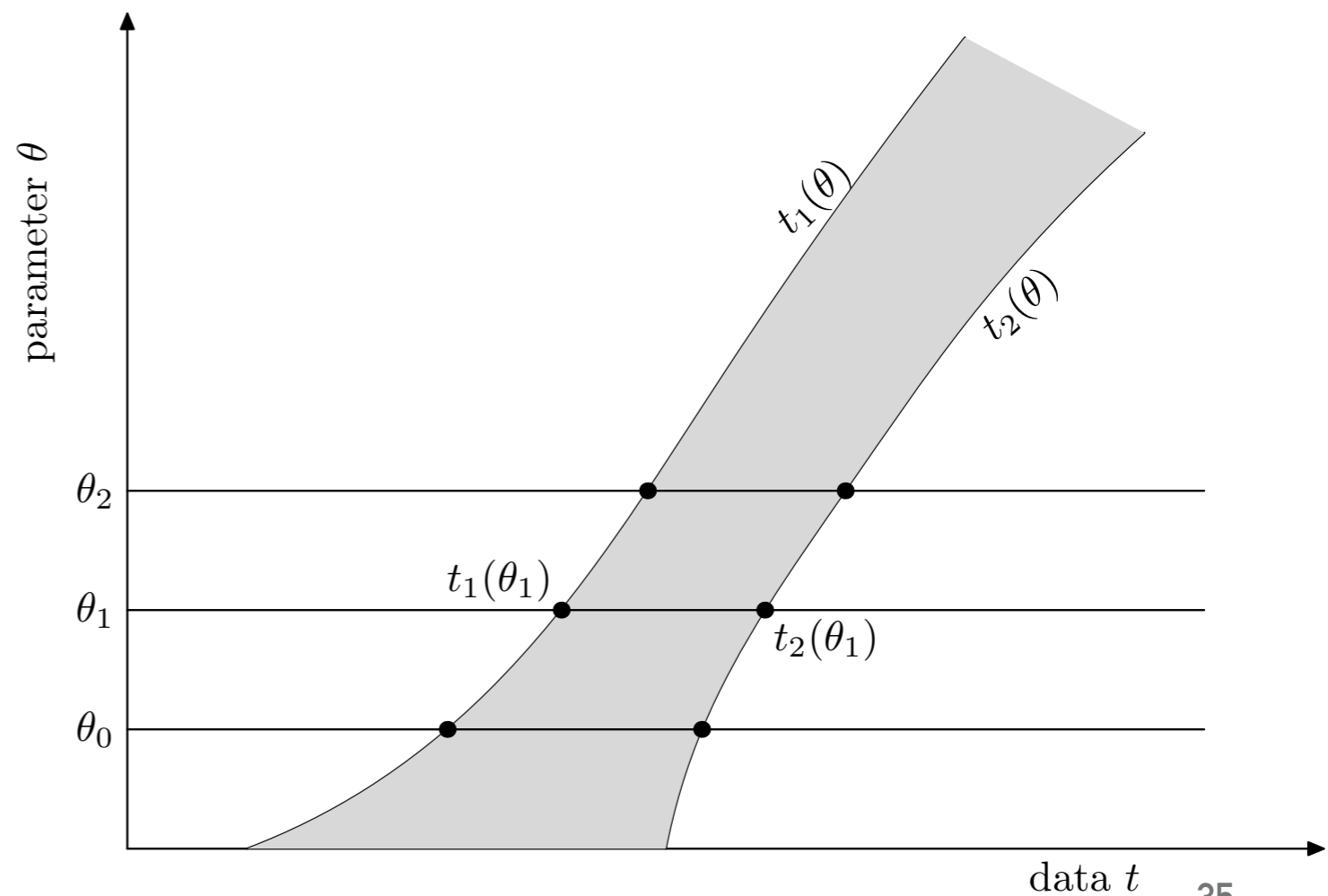
# NEYMAN CONSTRUCTION

- Construct the confidence belt horizontally — **for each hypothetical value of  $\theta$** , the points  $t_1(\theta)$  and  $t_2(\theta)$  are determined such that:

$$\int_{t_1}^{t_2} f(t|\theta) dt = \beta \quad \text{where } f(t|\theta) \text{ is the PDF.}$$

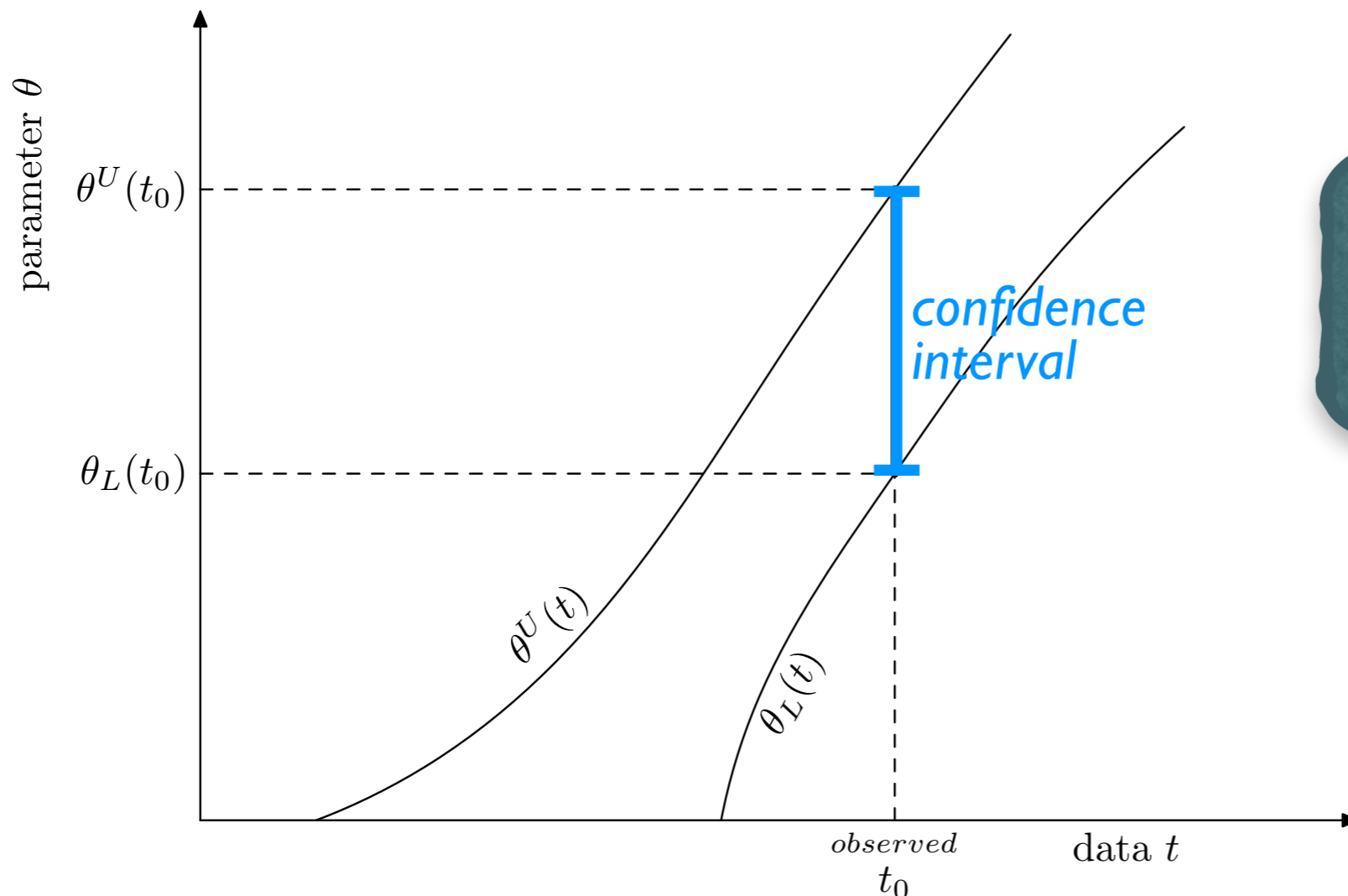
- Typical choices of  $\beta$  are 68.3% or 90%.
- Repeating the calculation for  $t_1(\theta)$  and  $t_2(\theta)$  and draw the confidence belt:

Remark: the solutions of  $t_1$  and  $t_2$  are not unique for each  $\theta$ ; here we just assume they can be obtained/defined using some way!  
(e.g. **central interval**)



# NEYMAN CONSTRUCTION (II)

- Now the two curves of  $t_1(\theta)$  and  $t_2(\theta)$  are re-labelled as  $\theta(t)$ , then the **confidence interval** can be read *vertically*.

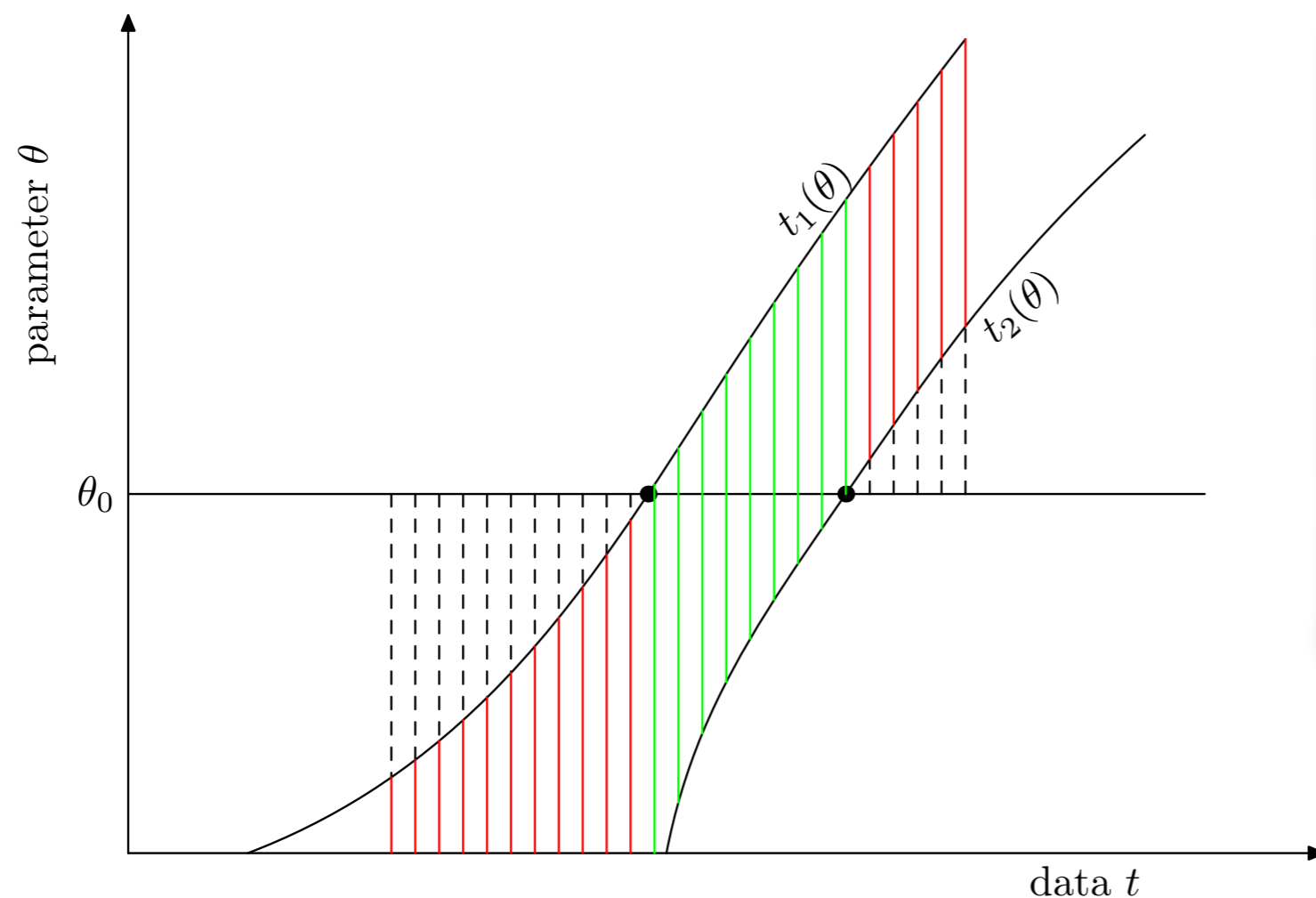


For given observed data  $t_0$ , the confidence interval is then  $[\theta_L(t_0), \theta^U(t_0)]$ .

*Question: how could we claim the coverage, e.g.  $P(\theta_L < \theta_{\text{true}} < \theta^U) = \beta$  ?*

# NEYMAN CONSTRUCTION (III)

- Suppose the true value is  $\theta_0$ . Depending on the observed data, one could get the intervals either indicated as **red**, or as **green** vertical lines:



Only the **green confidence intervals** cover the true value; **red intervals** are not.

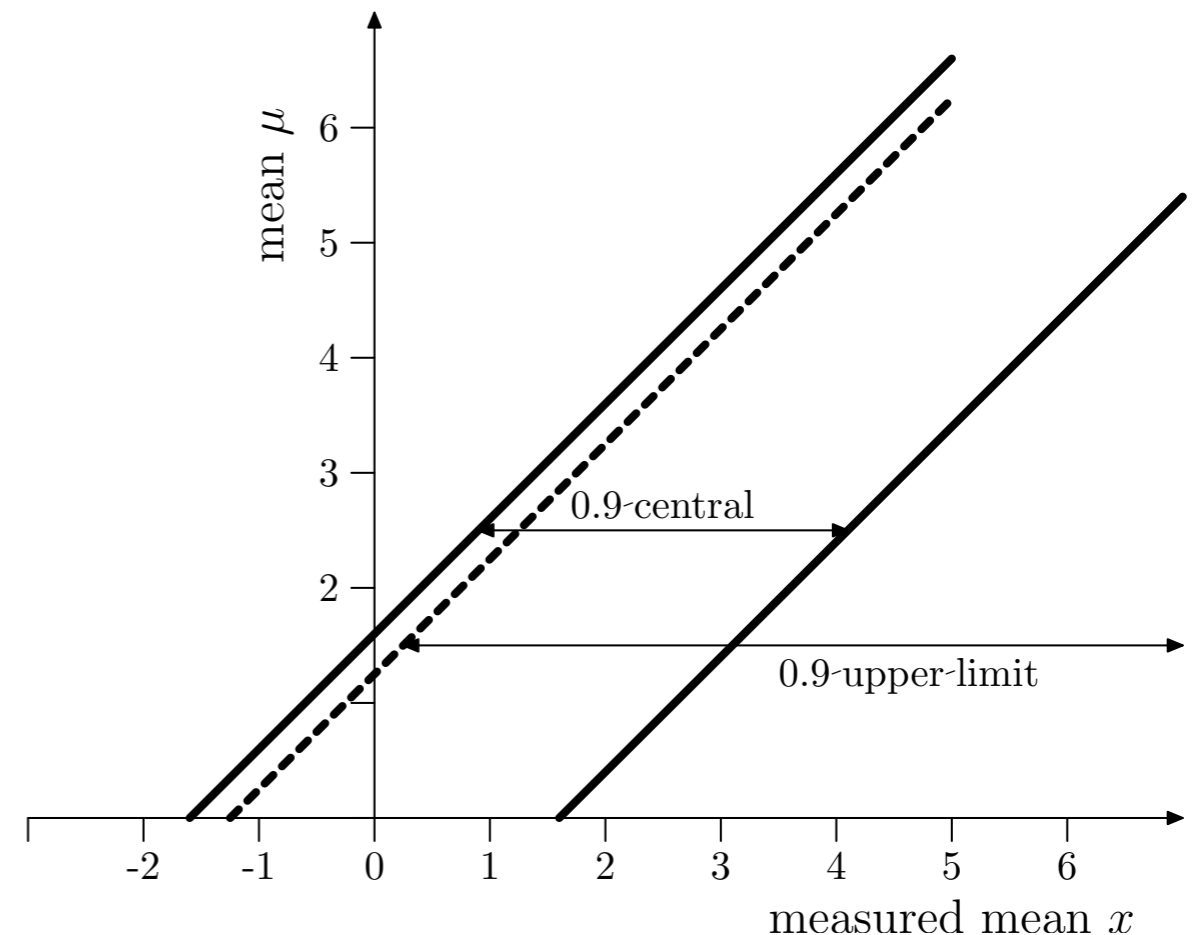
However, by definition  
 $P(t_1(\theta) < \text{data} < t_2(\theta)) = \beta$

The chance of getting a **green interval** is just  $\beta$ .

*i.e. for any value of  $\theta_0$ ,*  
 $P(\theta_L < \theta_0 < \theta^U) = \beta$

# CHOICES OF INTERVAL

- As mentioned earlier, there are multiple choices of interval, even for a fixed probability  $\beta$ . The most common choice is the central interval, but it is also very common to discuss “one-side” only: **upper limit**.
- In particular if the parameter has a physical boundary (e.g. *cannot be negative or so*).
- Figure shows the confidence belts for a Gaussian measurement with unknown  $\mu$ , and known  $\sigma=1$ .



the **central confidence belt**  
 $P(\theta_L > \theta_0) = P(\theta_0 > \theta^U) = (1 - \beta) / 2$   
versus the **upper limit**  
 $P(\theta_L < \theta_0) = \beta$  or  $P(\theta_0 < \theta^U) = \beta$



# CONFIDENCE INTERVALS FOR DISCRETE DATA

---

- Reminder: Frequentist description of the confidence interval with a coverage probability  $\beta$ :

$$\int_{t_1}^{t_2} f(t|\theta) dt = \beta$$

- For the case discrete observable, one has to sum over the cases to fulfill the coverage:

$$\sum_{i=L}^U P(t_i|\theta) \geq \beta$$

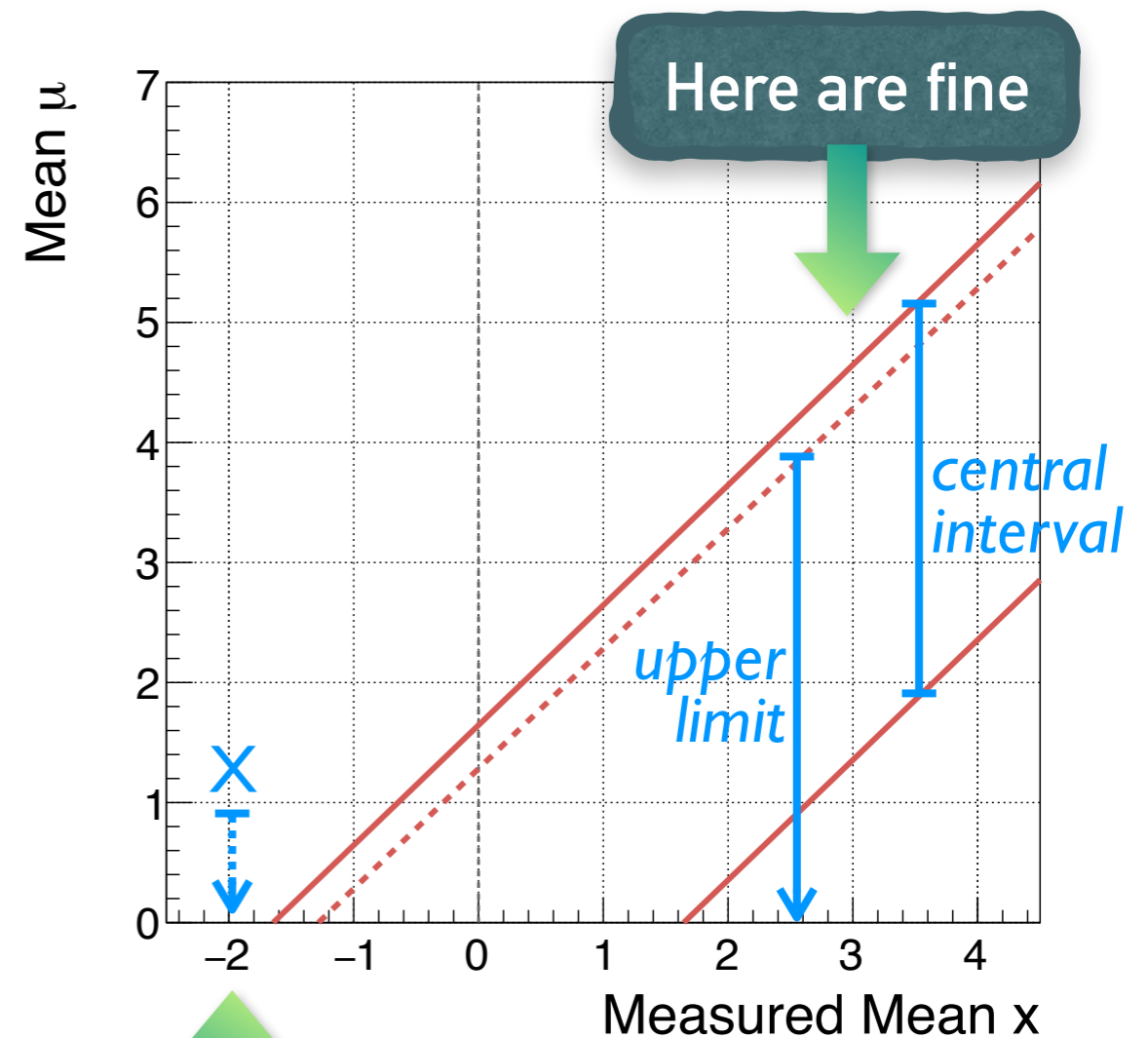
- A slightly over-coverage may happen when adding discrete probabilities.
- Note the **over-coverage** case ( $P > \beta$ ), it just loses the statistic power and results more conservative intervals, while the **under-coverage** case ( $P < \beta$ ) is something bad and would like to be avoided definitely.

# PROBLEMS IN NEYMAN CONSTRUCTION

- Consider an example of Gaussian distribution (with  $\sigma=1$ ) with a physical boundary of  $\mu > 0$ :

$$P(x|\mu) = \frac{1}{\sqrt{2\pi}} \exp \left[ -\frac{1}{2}(x - \mu)^2 \right]$$

- **Empty interval:** if the measured value of  $x$  is very negative (e.g.  $x=-2$ , due to some possible statistical fluctuations), the resulting confidence interval is an empty set since negative values of  $\mu$  are unphysical.

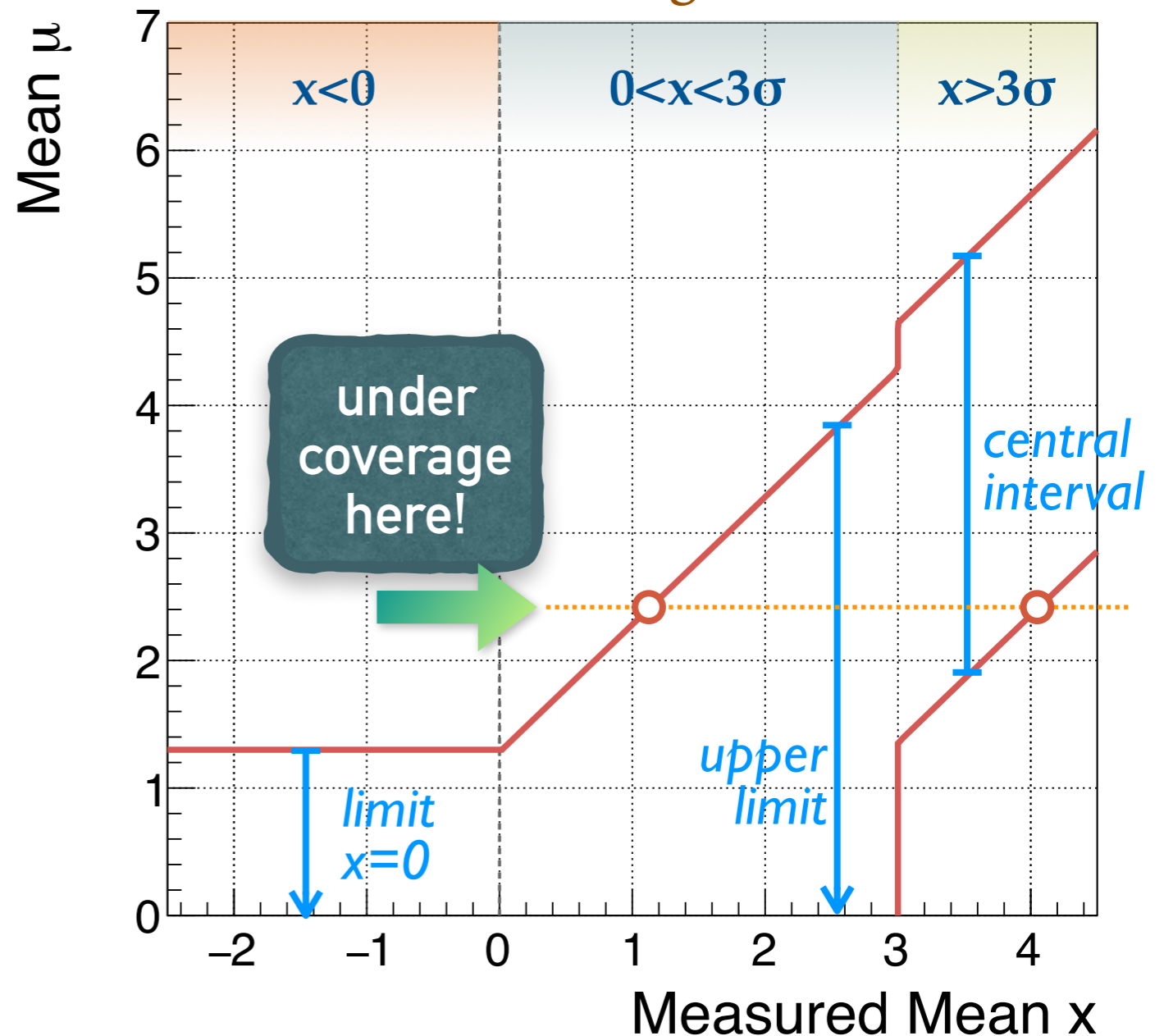


If the measured  $x$  is here, then the mapped interval is empty

# PROBLEMS IN NEYMAN CONSTRUCTION (II)

- **Flip-flop**: physicist tend to decide what to report based on the observed data, e.g.
  - if measured  $x > 3\sigma$ , report the central interval
  - if measured  $x < 3\sigma$ , report the upper limit
  - if measured  $x < 0$  (*unphysical*), report the limit obtained from  $x=0$

*flip-flop choice of interval may lead to under-coverage!*

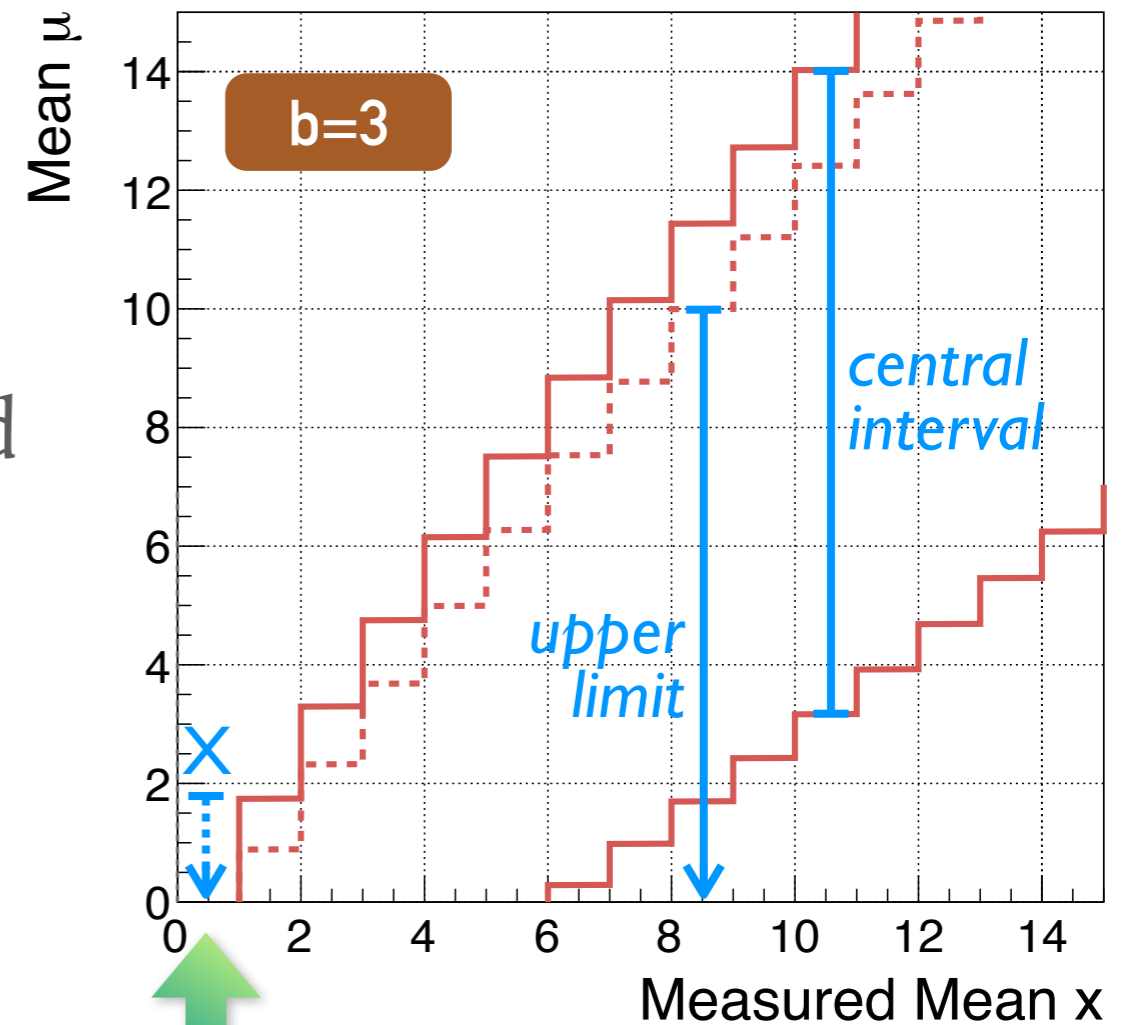


# PROBLEMS IN NEYMAN CONSTRUCTION (III)

- Let's examine another example, Poisson with background  $b=3$ :

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

- **Empty interval**: if the measured  $n=0$ , the resulting confidence interval is an empty set since negative values of  $\mu$  are unphysical.
- **Flip-flop**: choice of interval based on the observed data; lead to under-coverage.



If the measured  $n$  is 0, then the resulting interval is empty

# SOLUTION: THE FELDMAN-COUSINS UNIFIED APPROACH

---

➤ Feldman and Cousins have proposed an unified approach to solve the problems (*flip-flopping and empty intervals due to physical boundary*). See **PRD 57 (1998) 3873**.

➤ This elegant method is to find an ordering principle which automatically gives the intervals with desired properties.

➤ The key idea is the **likelihood ratio ordering** principle:

$$R = \frac{f(x|\theta)}{f(x|\hat{\theta})}$$

where  $f(x|\theta)$  is the likelihood of parameter  $\theta$  given data  $x$ ,

$f(x|\hat{\theta})$  is the maximized likelihood (over  $\hat{\theta}$ ) for given data  $x$ .

➤ Then replace the interval integration by

$$\int_{t_1}^{t_2} f(t|\theta)dt \Rightarrow \int_{R > R_{\min}(\beta)} f(t|\theta)dt = \beta$$

*integrate over  
the space w/ larger  
likelihood ratio R*



# FELDMAN-COUSINS FOR POISSON WITH BACKGROUND


- Let's practice the likelihood ratio ordering: Poisson with background  $b=3$ .
- Start with a target  $\mu$ , and for each possible  $n$ , determine the  $P(n|\mu)$ .
- Evaluate  $\hat{\mu}$  which maximize  $P(n|\mu)$  and the **likelihood ratio**  $R$  is given by  $R = P(n|\mu)/P(n|\hat{\mu})$ .
- Add up  $P(n|\mu)$  according to the  $R$ -ranking, until  $\Sigma P(n|\mu) \geq \beta$
- Repeat this for all  $\mu$ , then extract the interval  $[\mu_L, \mu^U]$  with the same procedure.

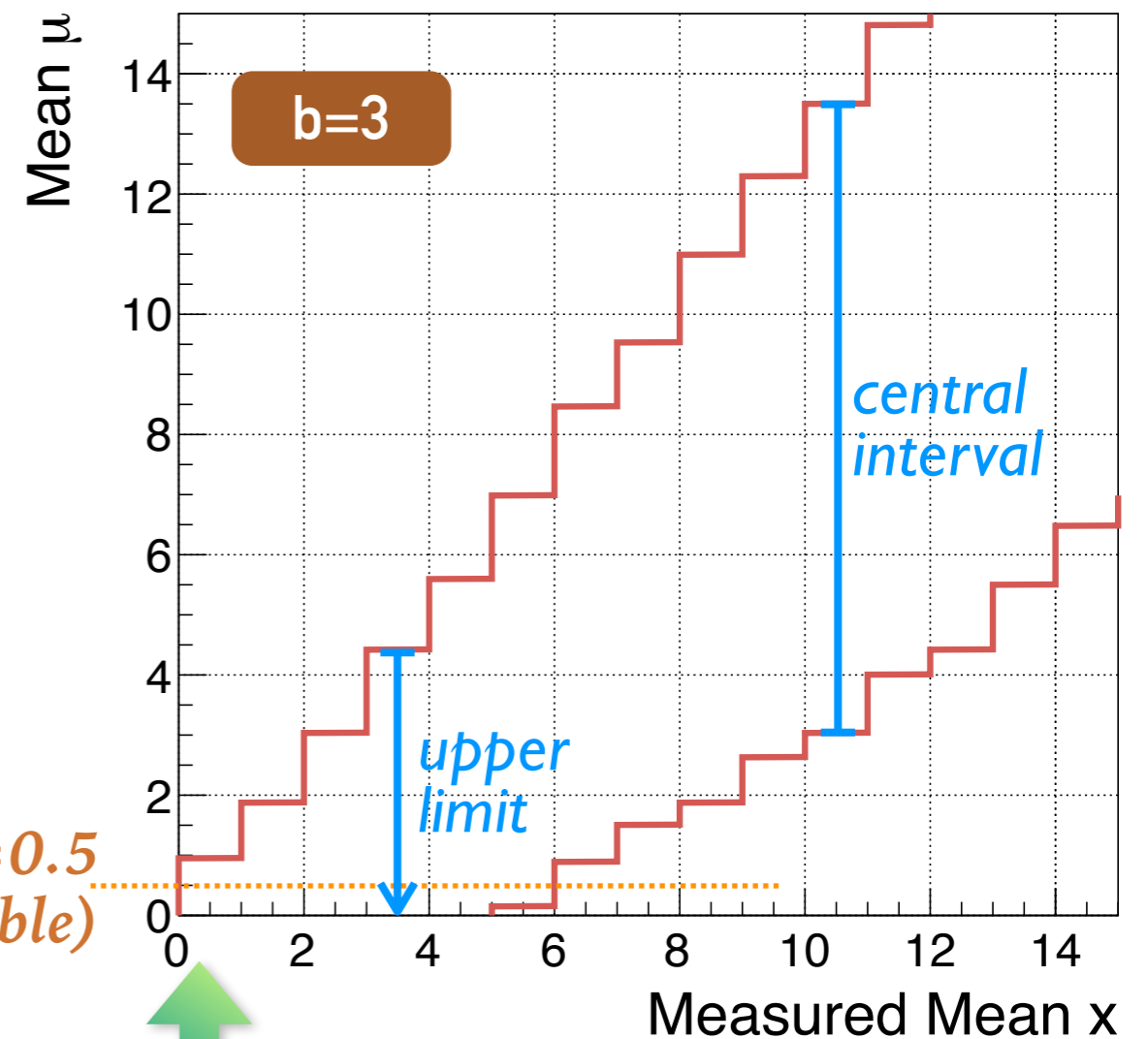
$$P(n|\mu) = \frac{(\mu + 3)^n}{n!} e^{-(\mu+3)}$$

$(\mu=0.5)$

n	P(n μ)	$\hat{\mu}$	P(n  $\hat{\mu}$ )	R	Rank
0	<b>0.030</b>	0	0.050	<b>0.607</b>	<b>6</b>
1	<b>0.106</b>	0	0.149	<b>0.708</b>	<b>5</b>
2	<b>0.185</b>	0	0.224	<b>0.826</b>	<b>3</b>
3	<b>0.216</b>	0	0.224	<b>0.963</b>	<b>2</b>
4	<b>0.189</b>	1	0.195	<b>0.966</b>	<b>1</b>
5	<b>0.132</b>	2	0.175	<b>0.753</b>	<b>4</b>
6	<b>0.077</b>	3	0.161	<b>0.480</b>	<b>7</b>
7	0.039	4	0.149	<b>0.259</b>	
8	0.017	5	0.140	<b>0.121</b>	
9	0.007	6	0.132	<b>0.050</b>	
10	0.002	7	0.125	<b>0.018</b>	
11	0.001	8	0.119	<b>0.006</b>	

# FELDMAN-COUSINS FOR POISSON WITH BACKGROUND (II)

- For each  $\mu$ , apply the likelihood ordering and evaluate the probability content over  $n$ ;
- Repeat for all  $\mu$ , and perform the Neyman construction, then extract the resulting interval according to the observed data.
- With this method there is no empty interval and  no flip-flopping.
- Slightly falling into conservative side due to the discreteness of  $n$ .

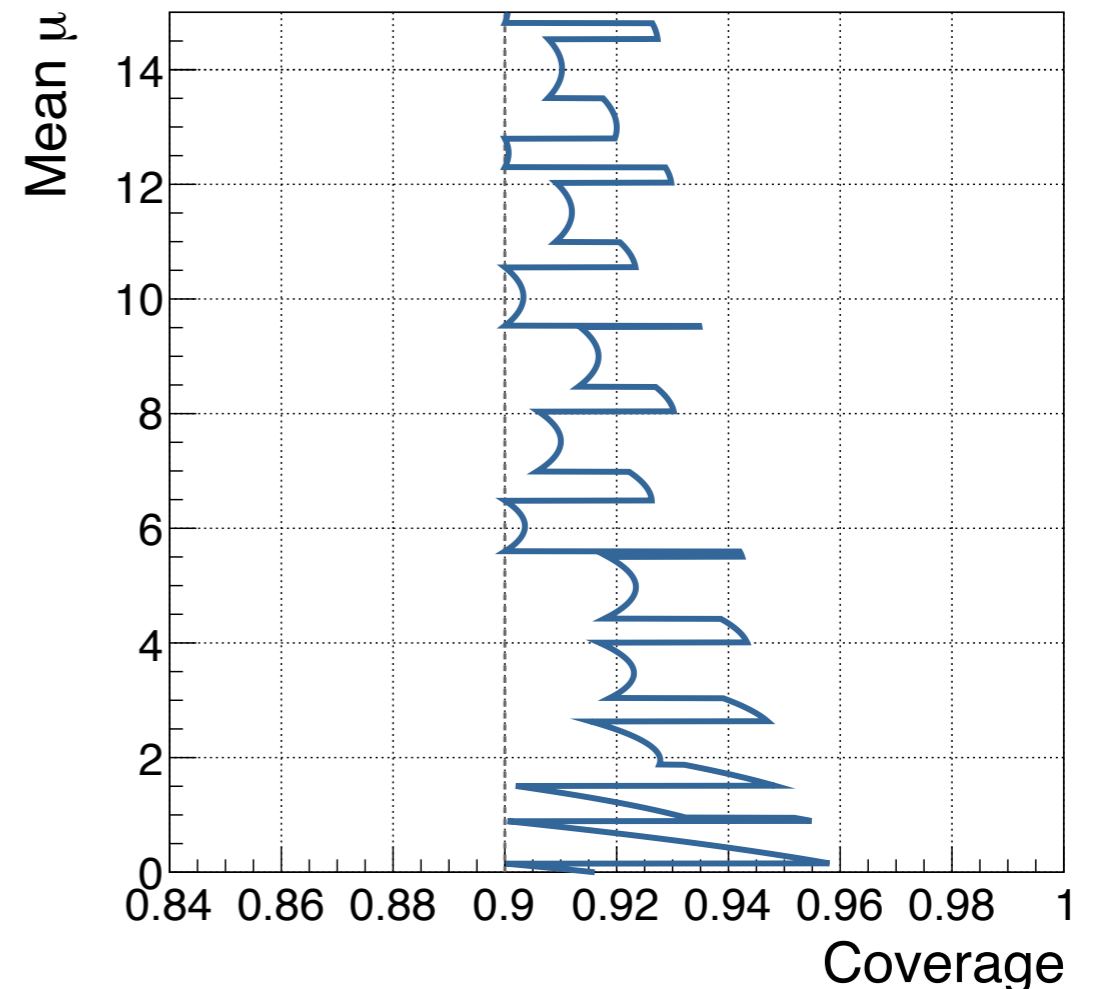


# FELDMAN-COUSINS FOR POISSON WITH BACKGROUND (III)

- The Feldman-Cousins unified approach has had its greatest success when applied to Poisson data, where all previously used methods had some undesirable properties.
- The Feldman-Cousins coverage versus true mean  $\mu$  “Dinosaur Plot”: it shows a good coverage (all above 90% in this Poisson example with background  $b=3$ ). The “teeth” are just due to the discreteness of  $n$ .

*F&C 90%  
Poisson  
upper bound*

Observed	0	1	2	3
bkg = 0.0	2.44			
0.5	1.94	3.86		
1.0	1.61	3.36	4.91	
2.0	1.26	2.53	3.91	5.42
3.0	1.08	1.88	3.04	4.42



# EXAMPLE: F&C INTERVAL FOR POISSON WITH BACKGROUND

---

- Here are an example code for the Feldman-Cousins calculation for the case of Poisson distribution with known background, using the `RooStats::FeldmanCousins` tool here!

partial example\_05.cc

```
using namespace RooFit;
using namespace RooStats;

RooRealVar n("n", "observed yield", 0., 50.);

RooRealVar s("s", "# of signal", 1., 0., 10.);
RooConstVar b("b", "# of background", 3.);
RooAddition mean("mean", "s+b", RooArgList(s, b));
RooPoisson pois("pois", "Poisson PDF", n, mean);

RooDataSet data("data", "data", RooArgSet(n));
n.setVal(3);
data.add(RooArgSet(n));

RooWorkspace *wspace = new RooWorkspace("wspace");
ModelConfig cfg(wspace);
cfg.SetPdf(pois);
cfg.SetParametersOfInterest(s);
cfg.SetObservables(n);
```

↪ set a constant background of 3 events

↪ also observed 3 events!

# EXAMPLE: F&C INTERVAL FOR POISSON WITH BACKGROUND (CONT.)

partial example\_05.cc

```
FeldmanCousins *fc = new FeldmanCousins(data, cfg);
fc->SetConfidenceLevel(0.9);
fc->UseAdaptiveSampling(true);
fc->FluctuateNumDataEntries(false);
fc->SetNBins(100); ← scan over true "s" with 100 bins

PointSetInterval* interval = fc->GetInterval();
printf("F&C interval: [%.2f, %.2f]\n",
       interval->LowerLimit(s), interval->UpperLimit(s));
```

```
NeymanConstruction: Prog: 100/100 total MC = 40 this test stat =
5.56255 s=9.95 [-1e+30, 1.6437] in interval = 0

[#1] INFO:Eval -- 44 points in interval
F&C interval: [0.05, 4.55]
```

- You may find the code gives you somewhat not exactly the same results as we saw in the earlier slides!
- This is due to the fact that the `RooStats::FeldmanCousins` tool is using **Monte Carlo integration**.



# BACK TO THE GAUSSIAN WITH BOUNDARY EXAMPLE

---

- ▶ Let's roll back a little bit to the previous example, Gaussian distribution with physical boundary at zero, and adopt the Feldman-Cousins method on it:

$$P(x|\mu) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x - \mu)^2\right]$$

- ▶ For **each target  $\mu$** , find  $\hat{\mu}$  which maximize  $P(x|\mu)$ :

$$\hat{\mu} = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \quad P(x|\hat{\mu}) = \begin{cases} \exp(-x^2/2)/\sqrt{2\pi} & x < 0 \\ 1/\sqrt{2\pi} & x \geq 0 \end{cases}$$

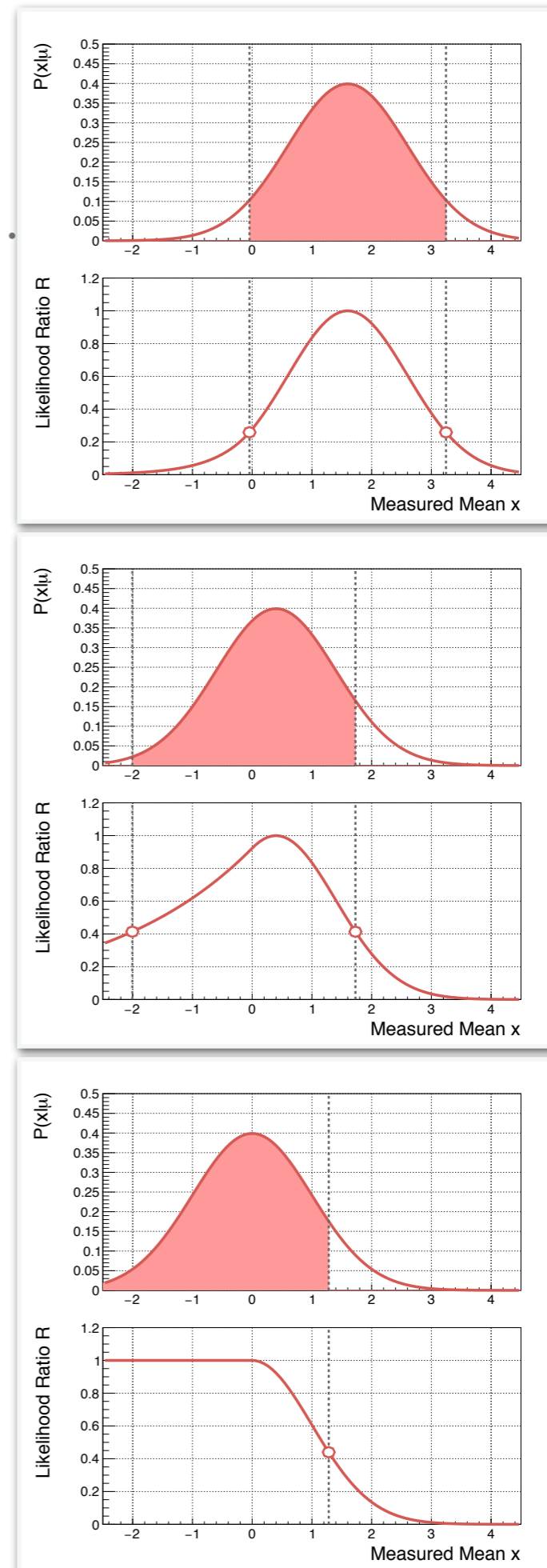
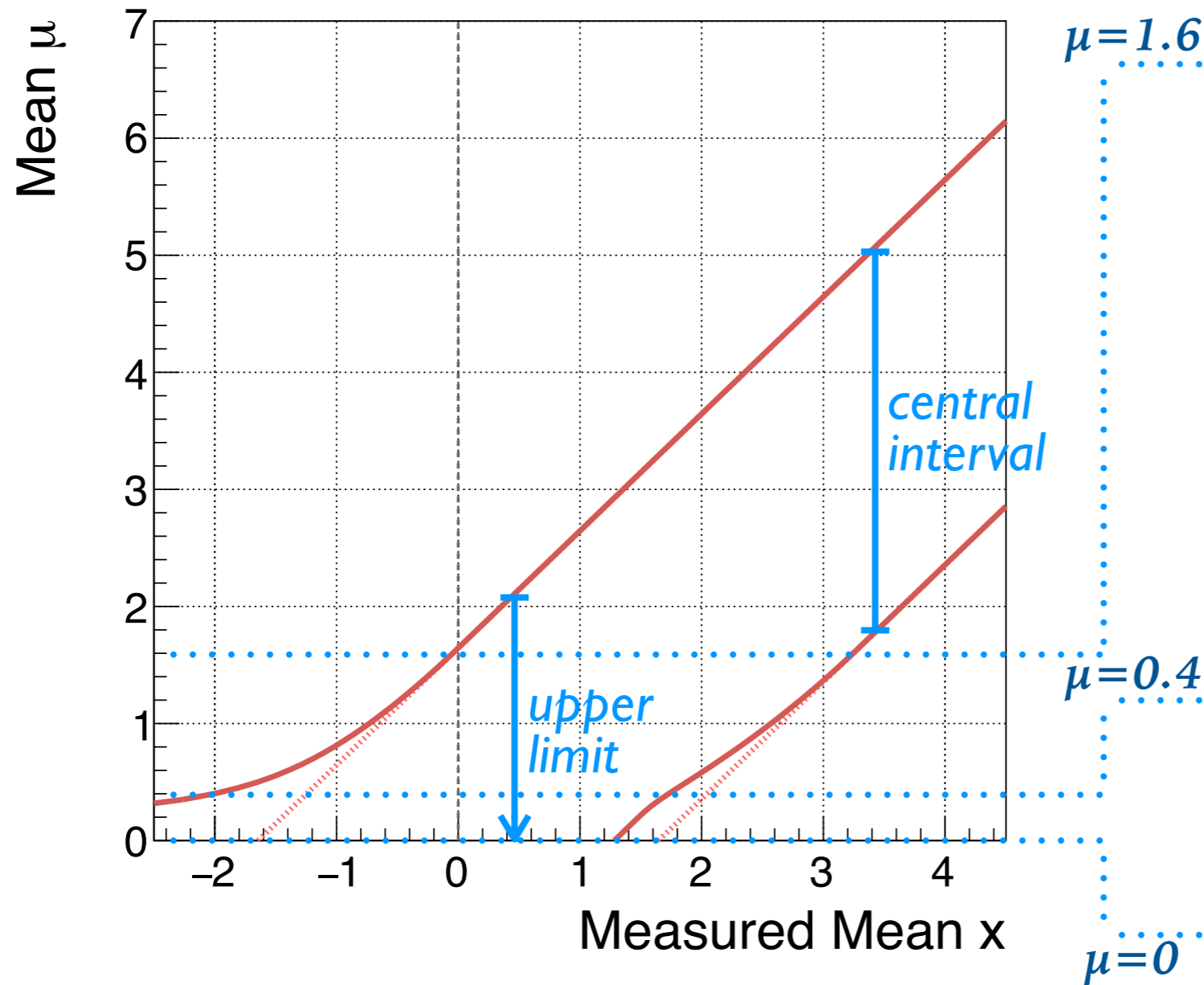
- ▶ The **likelihood ratio  $R$**  can be computed:

$$R = \begin{cases} \exp[-(x - \mu)^2/2] / \exp(-x^2/2) & x < 0 \\ \exp[-(x - \mu)^2/2] & x \geq 0 \end{cases}$$

- ▶ Now integrated over the  $R$ -ranked interval  $[x_1, x_2]$ , where

$$\int_{x_1}^{x_2} P(x|\mu) dx = \beta \quad \text{over the space } R(x) \geq R(x_1) = R(x_2)$$

# GAUSSIAN WITH BOUNDARY



- **Feldman-Cousins** belts of 90% central intervals for a Gaussian measurement.
- No empty intervals and no flip-flopping.

# COMMENT: NUISANCE PARAMETERS TREATMENT

---

- Generally in the Neyman construction, adding nuisance parameters is kind of awkward and easily gives an over-coverage result. This is due to the fact one is requiring the coverage for every possible value of the nuisance parameters (*remember our 2D Normal theory example!*).
- A proper solution has been suggested by Feldman (*which is beyond the F&C paper*), it looks for a coverage in the “worst case” of the nuisance parameter. By modify the ranking **likelihood ratio** as:

*maximizes the numerator*

$$R = \frac{P(x|\theta, \hat{\eta}) P(b|\hat{\eta})}{P(x|\hat{\theta}, \hat{\eta}) P(b|\hat{\eta})}$$

*← nuisance PDF*

$\theta$ : signal parameter

$\eta$ : background (nuisance) parameter

$x$ : a measurement of  $\theta + \eta$

$b$ : a measurement of  $\eta$

*maximizes the denominator as usual*

Then proceed to the usual confidence belt construction.

In fact this is nothing different from making the nuisance parameters “profiled”!

# COMMENT: UNIFIED APPROACH WITH 2 PARAMETERS

---

- The Feldman-Cousins approach does give good statistical properties: tight limit and correct coverage even with 2 parameters. It has been demonstrated already in the original paper.
- However, it is clear that the calculation becomes rather complicated and very difficult to use (*also requires a lot of CPU power*).
- A usual alternative solution to this is to take **profile likelihood** again, i.e. when dealing with the parameter  $X$ , and make another parameter  $Y$  profiled.
- Remark: Bayesian method can be also very problematic with 2D or more. Constructing a proper multidimensional prior will pose a great problem, and hard to be “*uninformative*”.

# F&C CALCULATION FOR A MORE PRACTICAL CASE

---

- As we already pointed out, the computing for Feldman-Cousins approach is rather heavy, in particular if one really wants follow the principle of confidence belt construction:
  - For each value of true  $\theta$ , scanning over the possible measure  $x$ .
  - Integrate the probabilities according to the likelihood ratio ordering until it reaches the desired probability  $\beta$ .
- Even if there is only one parameter of interests, the construction is still in 2D: **true  $\theta$  and measure  $x$** .
- Generally if the model is complicated (*and with multiple nuisance parameters*), a quick integration is almost impossible. In many cases this has to be carried out by Monte Carlo integration (*as you already seen when we are playing with the RooStats tool*).



# F&C CALCULATION FOR A MORE PRACTICAL CASE (II)

---

- On the other hand, we do not really need to calculate for all possible value of measure  $x$ , given you may only want to study the situation for your observed data (*only one set / one measurement!*)
- Based on the likelihood ratio ordering, one can in fact calculate the **confidence level for any point of  $(x, \theta)$**  with **toy Monte Carlo**.
- If we only scan over the true  $\theta$ , only only for the observed data  $x$ , it would be a great reduction of the computing time; this is should produce a similar result as **profile likelihood scan** if everything is close to Normal distribution.
- We will demonstrate how to do this (*without RooStats tool*) in the next example.

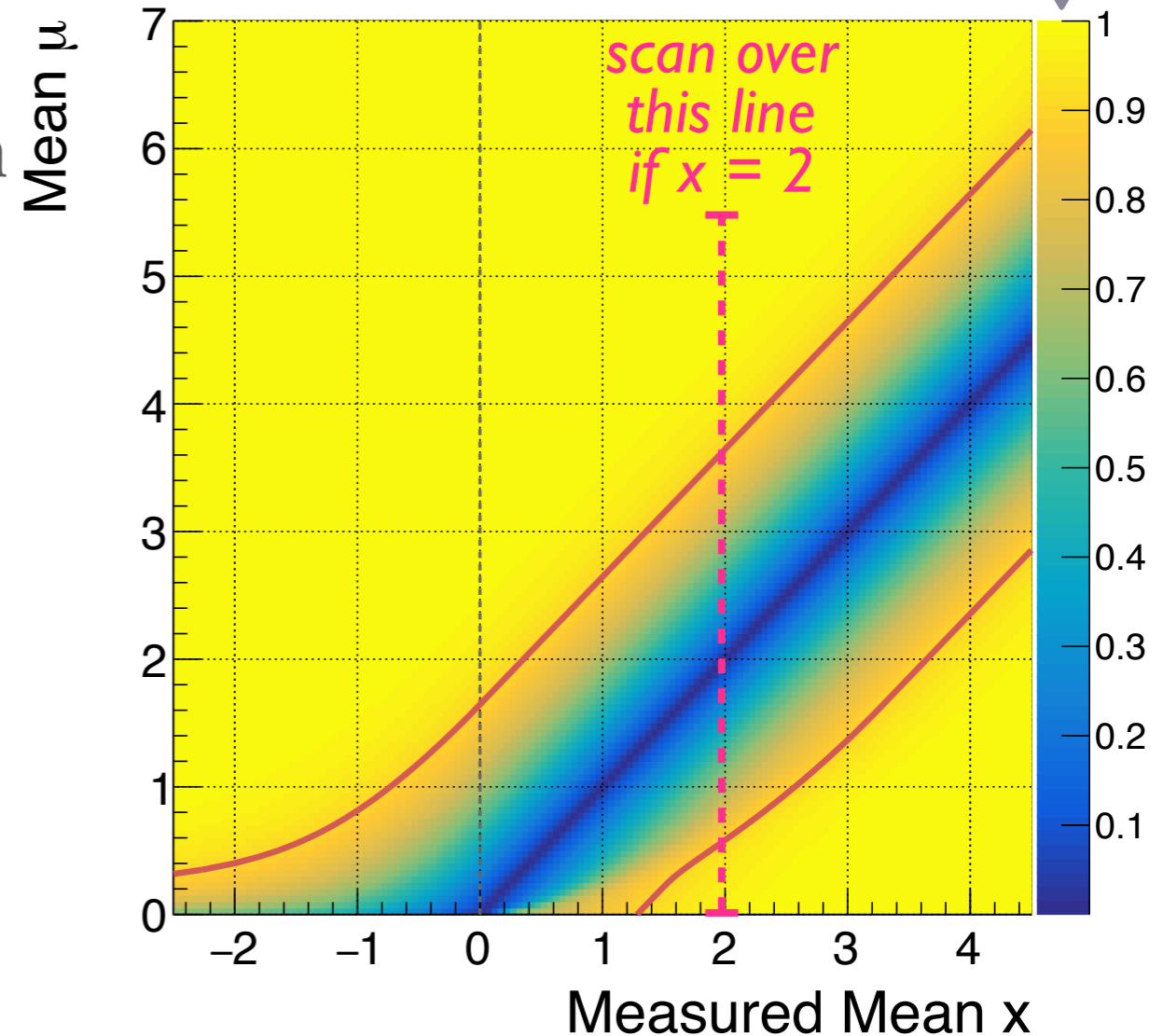
Why no RooStats tool? For easy problem this kind of heavy study is not really needed; for difficult problem it is already beyond the capability of the standard tool...

# F&C CALCULATION FOR A MORE PRACTICAL CASE (III)

- Remember our confidence belt constructed for a Gaussian model with physical boundary.
- You can see the confidence belt we saw earlier is just an extraction for the points with  $\beta=0.9$ .
- For a practical case, one only need to scan over the true  $\mu$  for a single value of measured  $x$ , without doing the full construction of the belt.
- RooStats tool is doing the same thing in fact.

over the space  $R(x) \geq R(x_1) = R(x_2)$

$$\int_{x_1}^{x_2} P(x|\mu) dx = \beta$$



# EXAMPLE: STEP-BY-STEP F&C CALCULATIONS

- Here are a demonstration of adopting Feldman-Cousins approach. Assuming this is the data you received, and already fitted with a very simple model (note: *data file is available on the lecture web!*):

example\_06.cc

```
TFile *fin = new TFile("example_06.root");
TNtupleD* nt = (TNtupleD *)fin->Get("nt");

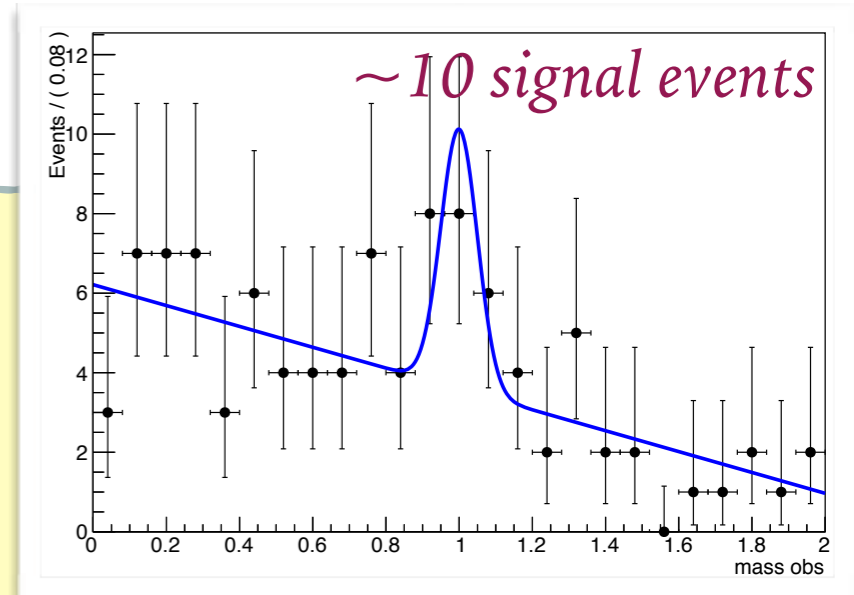
RooRealVar mass("mass", "mass obs", 0., 2.);

RooRealVar mu("mu", "signal mean", 1.0);
RooRealVar sigma("sigma", "signal width", 0.05);
RooGaussian gaus("gaus", "signal PDF", mass, mu, sigma);

RooRealVar slope("slope", "background slope", -0.3, -5., 5.);
RooPolynomial linear("linear", "background PDF", mass, RooArgSet(slope));

RooRealVar ns("ns", "ns", 10, 0., 1000.);
RooRealVar nb("nb", "nb", 90, 0., 1000.);
RooAddPdf model("model", "PDF", RooArgList(gaus, linear), RooArgList(ns, nb));

RooDataSet data("data", "data", nt, RooArgSet(mass));
model.fitTo(data, Minos(true));
```



1	nb	8.97806e+01	1.00129e+01	-9.66491e+00	1.03692e+01
2	ns	<b>1.02353e+01</b>	<b>4.55361e+00</b>	<b>-4.21951e+00</b>	<b>4.90387e+00</b>
3	slope	-4.22156e-01	4.63662e-02	-3.87779e-02	5.51910e-02

- First perform a **profile likelihood scan** (with background part being profiled!), and convert the resulting  $-2\ln L/L_{\max}$  to corresponding confidence level  $\beta$ :

partial example\_07.cc

```
void buildModel(RooWorkspace *wspace) {
    TFile *fin = new TFile("example_06.root");
    TNtupleD* nt = (TNtupleD *)fin->Get("nt");
    . . . . .
    wspace->import(model);
    wspace->import(data);
    delete fin;
}
```

import the model into a RooWorkspace for further use.

Before introducing a heavy F&C calculation, one should start from a lighter **profile likelihood scan** to get a rough idea!

```
void example_07() {
    RooWorkspace *wspace = new RooWorkspace("wspace");
    buildModel(wspace);

    RooFitResult *res0 = wspace->pdf("model")->fitTo(
        *wspace->data("data"), Save(true), Minos(true));

    TH1D *scan_2nll = new TH1D("scan_2nll", "", 101, -0.1, 20.1);
    TH1D *scan_beta = new TH1D("scan_beta", "", 101, -0.1, 20.1);

    for (int i=1; i<=scan_2nll->GetNbinsX(); i++) {
        wspace->var("ns")->setVal(scan_2nll->GetBinCenter(i));
        wspace->var("ns")->setConstant(true);
        RooFitResult *res1 =
            wspace->pdf("model")->fitTo(*wspace->data("data"), Save(true));
        double d2NLL = (res1->minNll()-res0->minNll())*2.;

        scan_2nll->SetBinContent(i, d2NLL);
        scan_beta->SetBinContent(i, 1.-TMath::Prob(d2NLL, 1));

        delete res1;
    }
}
```

Full both the  $-\ln(L/L_{\max})$  & confidence level



```

RooRealVar* best_ns = (RooRealVar*)res0->floatParsFinal().find("ns");
c1->cd(1);
scan_2nll->SetStats(false);
scan_2nll->GetYaxis()->SetTitle("-2ln(L/L_{max})");
scan_2nll->Draw("axis");

box.DrawBox(best_ns->getVal()+best_ns->getErrorLo(),0.,
            best_ns->getVal()+best_ns->getErrorHi(),scan_2nll->GetMaximum());
for(int n=0; n<=2; n++)
    lin.DrawLine(0.,n*n,20.,n*n);

scan_2nll->Draw("csame");

c1->cd(2);
scan_beta->SetStats(false);
scan_beta->GetYaxis()->SetTitle("Confidence Level (#beta)");
scan_beta->Draw("axis");

box.DrawBox(best_ns->getVal()+best_ns->getErrorLo(),0.,
            best_ns->getVal()+best_ns->getErrorHi(),scan_beta->GetMaximum());
for(auto prob: {0.,0.6827,0.9545})
    lin.DrawLine(0.,prob,20.,prob);

scan_beta->Draw("csame");
}

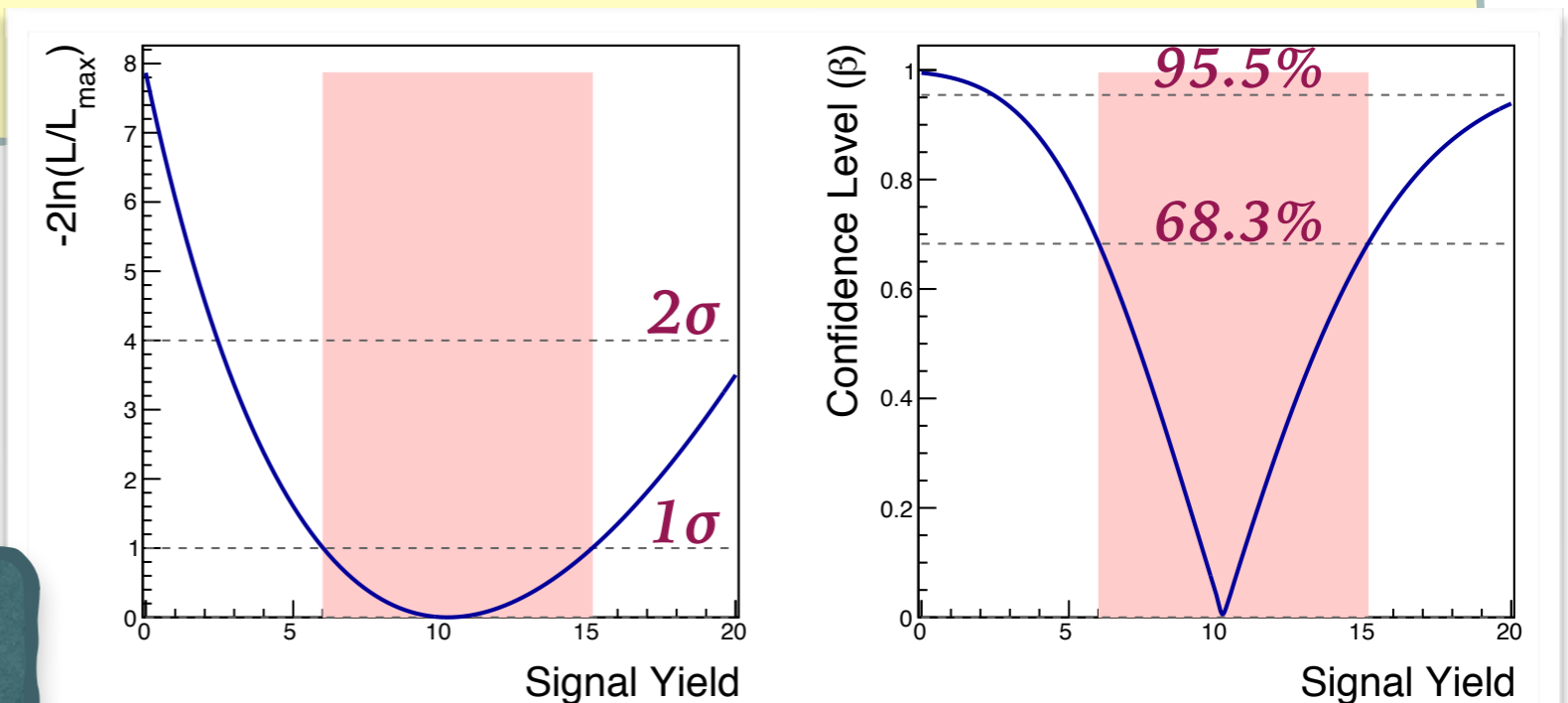
```

partial example\_07.cc

*One can find a good match between MINOS error band, and the 68% C.L. cross points.*



Proceed to **F&C** study now!





# EXAMPLE: STEP-BY-STEP F&C CALCULATIONS (CONT.)

---

- Now let's perform a Feldman-Cousins study for a given target (*true*) signal strength.
- The procedure can be carried out as following:
  - **Fit to data with the signal fixed to the target value**, using this resulting model to generate toy data sets.
  - For each set of toy data, two fits are performed: **one with signal fixed, one with signal floated**. Calculate the difference in the resulting  $-\ln(L)$  value, this actually gives the expected distribution of  $\ln(R)$ , where  $R$  is the **likelihood ratio for F&C ordering**.
  - Perform the same two fits to data and obtain the  $\ln(R)$  for data.
  - The fraction of toy sets which has a  **$\ln(R)$  value greater than the value from data** gives the estimate of confidence level  $\beta$  at the target signal strength and with the observed data.



Example code is given in the next page.

```

double FC_scan(double target_ns = 10., int ntoys = 100)
{
    RooWorkspace *wspace = new RooWorkspace("wspace");
    buildModel(wspace);

    RooFitResult *res0 = wspace->pdf("model")->fitTo(*wspace->data("data"), Save(true));
    wspace->var("ns")->setVal(target_ns);
    wspace->var("ns")->setConstant(true);
    RooFitResult *res1 = wspace->pdf("model")->fitTo(*wspace->data("data"), Save(true));
    double logR_data = res0->minNll()-res1->minNll();

    double beta = 0.;
    for (int idx=0; idx<ntoys; idx++) {
        RooWorkspace* wspace_toy = new RooWorkspace(*wspace);
        RooDataSet *toy = wspace_toy->pdf("model")->generate(*wspace_toy->var("mass"));

        RooFitResult *res1 = wspace_toy->pdf("model")->fitTo(*toy, Save(true));
        wspace_toy->var("ns")->setConstant(false);
        RooFitResult *res0 = wspace_toy->pdf("model")->fitTo(*toy, Save(true));

        double logR = res0->minNll()-res1->minNll();
        if (logR>logR_data) beta += 1.;

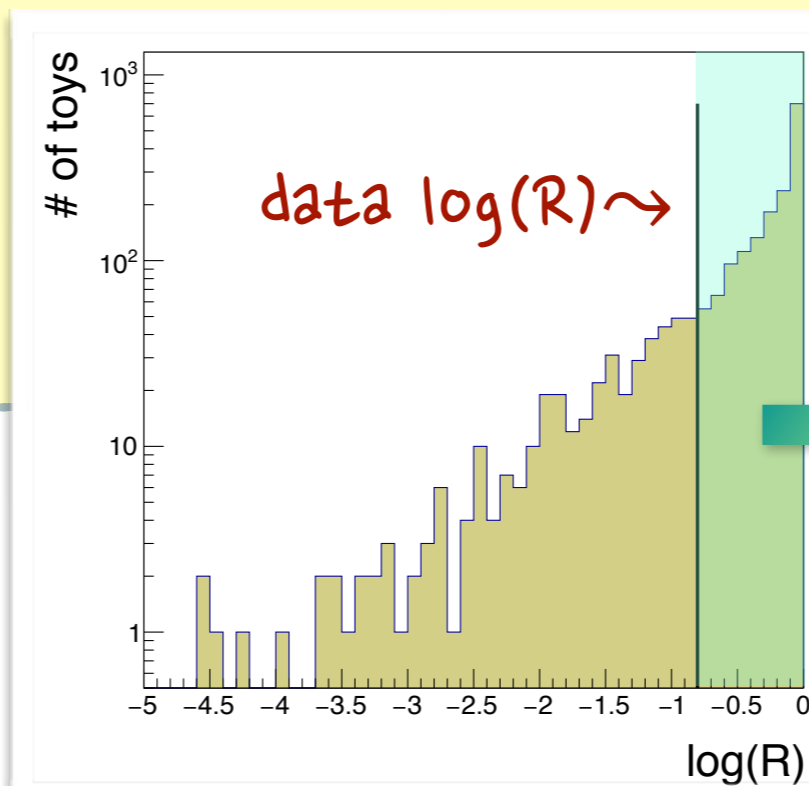
        delete . . .
    }
    beta /= (double)ntoys;

    delete . . .
    return beta;
}

```

produce test statistics w/ toy

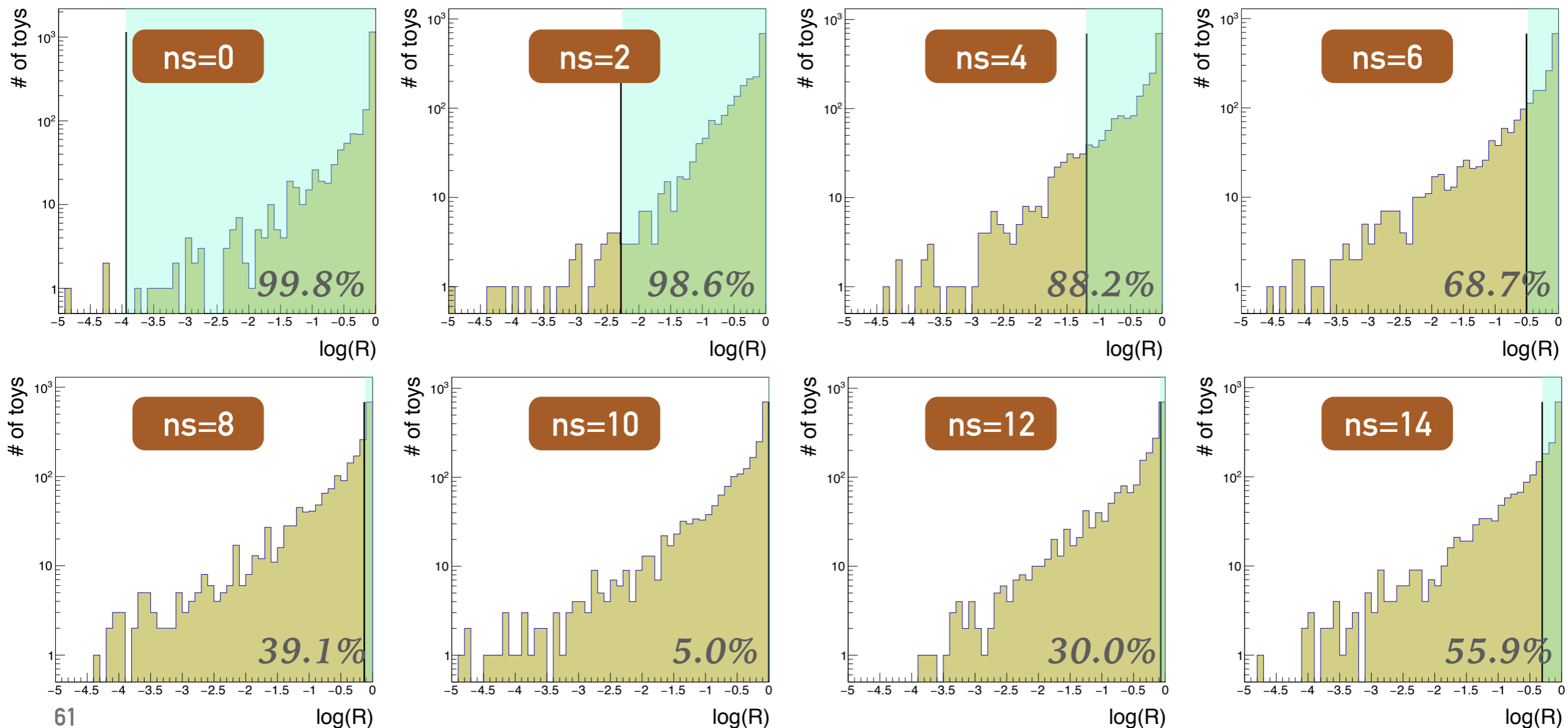
*Test statistics distribution  
for # of target signal = 5*



This region gives an estimation of confidence level.  
e.g. 79% of toys are here, thus  $\beta=0.79$ .

# SCAN OVER TARGET (TRUE) SIGNAL

- This is what one should be able to observe: when the scanned point of signal strength is away from the data fitted value, the probability content increases.



# F&C APPROACH VERSUS PROFILE LIKELIHOOD SCAN

- .....
- Now we can easily calculate the confidence level for any given target signal strength using the F&C approach. The results can be compared with the result of likelihood scan:

partial example\_08.cc

```
void example_08()
{
    RooWorkspace *wspace = new RooWorkspace("wspace");
    buildModel(wspace);

    RooFitResult *res0 = wspace->pdf("model")->fitTo(
        *wspace->data("data"), Save(true), Minos(true));

    TH1D *scan_beta = new TH1D("scan_beta", "", 101, -0.1, 20.1);
    TH1D *scan_fc = new TH1D("scan_fc", "", 101, -0.1, 20.1);

    for (int i=1; i<=scan_beta->GetNbinsX(); i++) {
        wspace->var("ns")->setVal(scan_beta->GetBinCenter(i));
        wspace->var("ns")->setConstant(true);
        RooFitResult *res1 = wspace->pdf("model")->fitTo(
            *wspace->data("data"), Save(true));
        double d2NLL = (res1->minNll()-res0->minNll())*2.;
        scan_beta->SetBinContent(i, 1.-TMath::Prob(d2NLL, 1));

        if ((i%5)==1) {
            double beta = FC_scan(scan_beta->GetBinCenter(i), 100);
            double beta_err = sqrt(beta*(1.-beta)/100);
            scan_fc->SetBinContent(i, beta);
            scan_fc->SetBinError(i, beta_err);
        }
    }
}
```

call the FC scan function  
in steps of 1 event

# F&C VERSUS PROFILE LIKELIHOOD

- Let's do a little bit interpolation to get the confidence intervals from F&C results (*very consistent with MINOS = profile likelihood!*)

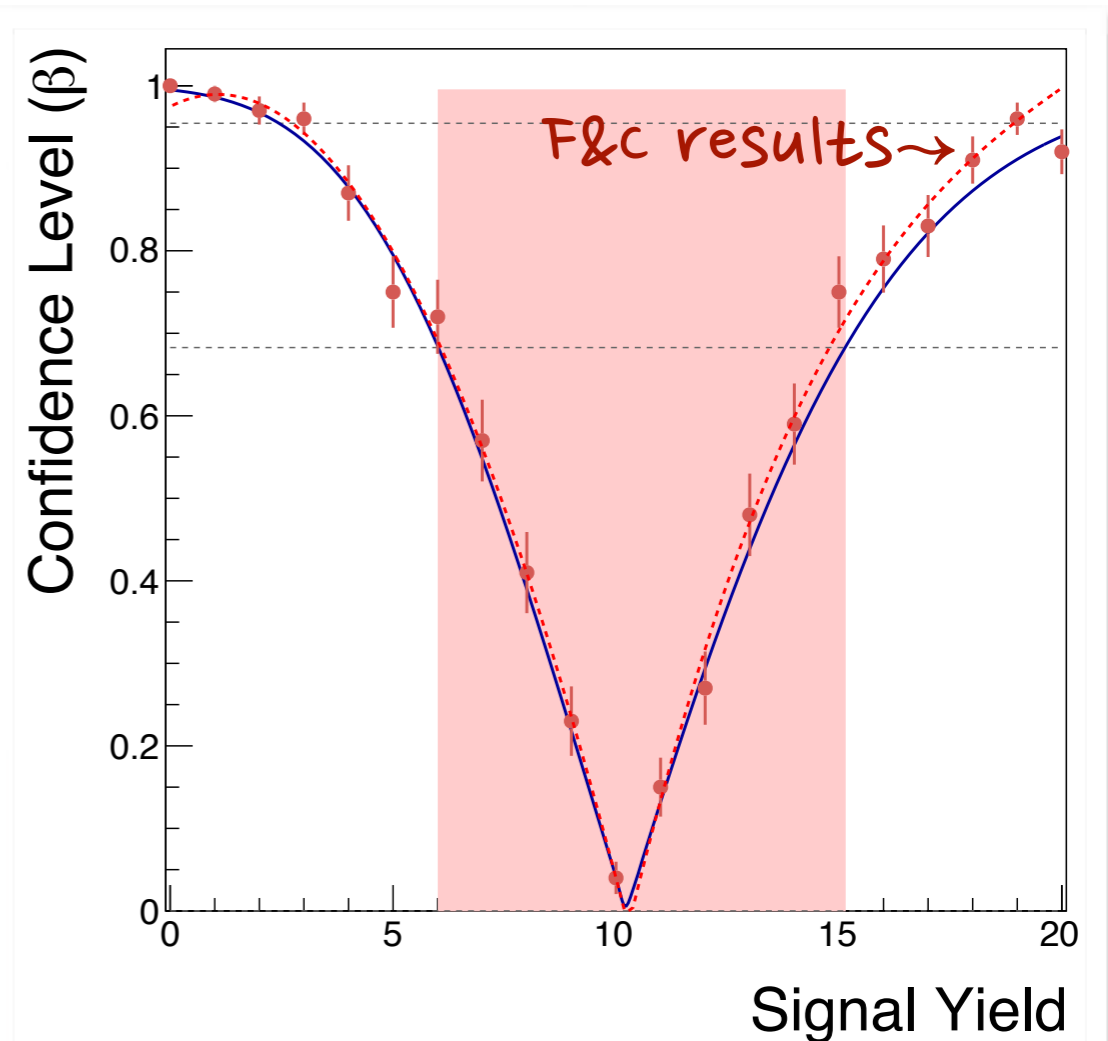
partial example\_08.cc

```
scan_beta->Draw("csame");  
scan_fc->SetMarkerStyle(20);  
scan_fc->Draw("esame");
```

```
TF1 fl("fl", "pol3", 0., best_ns->getVal());  
TF1 fr("fr", "pol3", best_ns->getVal(), 20.);  
scan_fc->Fit("fl", "+R");  
scan_fc->Fit("fr", "+R");
```

```
printf("MINOS 68.3%% interval: [%.2f, %.2f]\n",  
      best_ns->getVal()+best_ns->getErrorLo(),  
      best_ns->getVal()+best_ns->getErrorHi());  
printf("F&C 68.3%% interval: [%.2f, %.2f]\n",  
      fl.GetX(0.6827), fr.GetX(0.6827));
```

```
}
```



↪ fit the scanning points w/ 3rd order polynomial

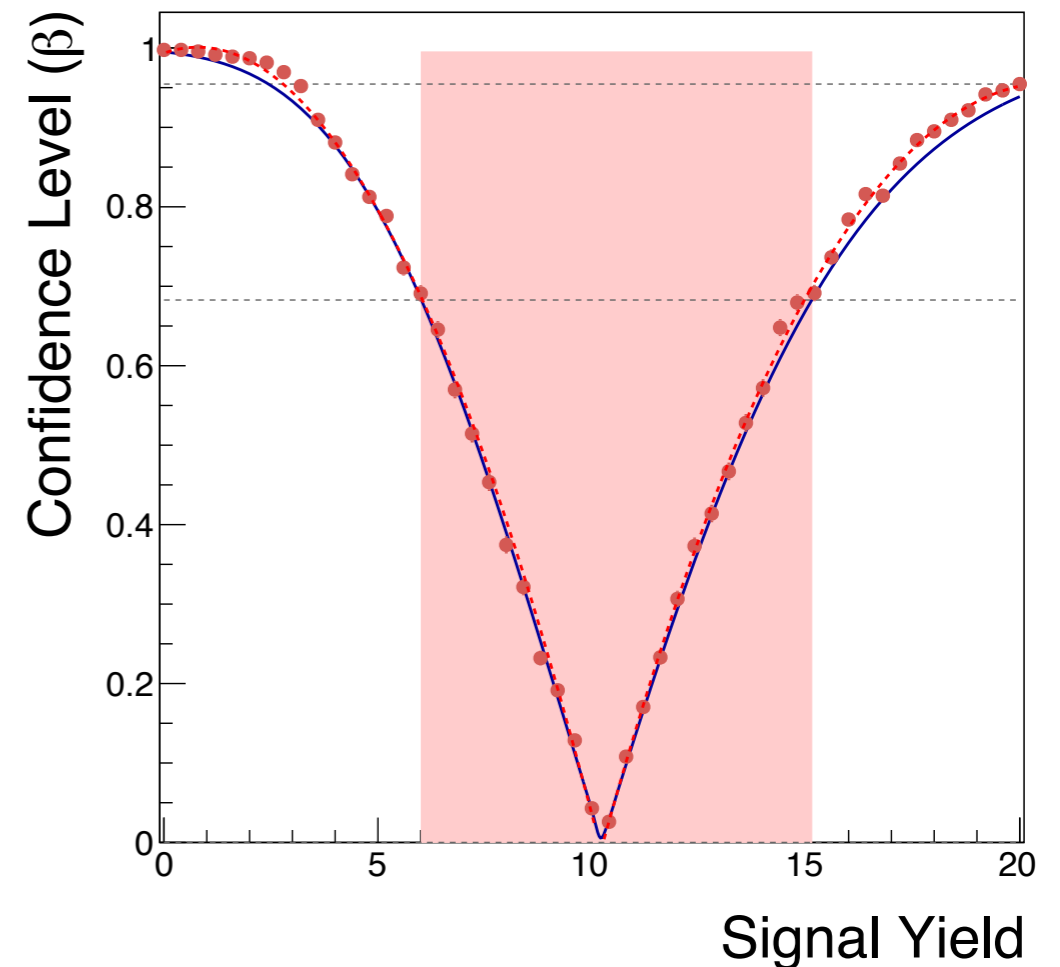
MINOS	68.3% interval:	[6.02, 15.14]
F&C	68.3% interval:	[6.07, 14.79]



# COMMENT: STEP-BY-STEP F&C CALCULATIONS

---

- We have demonstrated how to perform a confidence interval calculation using the unified approach, by ourselves.
- The calculation is heavy even with such a simple model and small data set. The real application will require far more computing power.
- The resulting confidence interval is generally consistent (with only minor deviations) with the profile likelihood scan, which is *asymptotically* valid.



A finer F&C scan with more toys!  
Start to see some deviations from  
likelihood method!

# COMMENT: NUISANCE PARAMETERS IN TOY MC

---

- In the previous example there are two floated nuisance parameters (**nb** and **slope**), in general there is no special treatment but just have them profiled during the generation of the test statistics distribution.
- However if there is any constrained parameter, **the constraint PDF term has to be randomized for each toy data as well**. For example, consider a Gaussian constrained likelihood:

$$L' = L(X|\theta, \lambda) \times P(\lambda|\mu_\lambda, \sigma_\lambda)$$

The model  $P(\lambda)$  is generally not handled by RooFit in the event generation. Thus one has to either randomize the constrained mean ( $\mu_\lambda$ ) for the fitting model or  $\lambda$  value in the generation for each set of toy, according to the constraint PDF.

- Otherwise the toy data will not have the proper statistical behavior!

# COMMENT: ISSUES OF THE UNIFIED APPROACH

---

- As we already see: the Feldman-Cousins unified approach solves the main problems when the parameter close to its physical boundary, within the framework of the classical Neyman construction. It also ensures a proper statistical coverage!
- **However there are some issues still:**
  - constructing the confidence intervals is rather complicated, CPU-intensive calculation are generally required (*e.g. large toy Monte Carlo set generation*);
  - In case of zero observed events, gives better limits for experiments that expect higher background unlike Bayesian method with uniform prior (*although this also happens for the case of non-uniform prior*).

# COMMENT: PROBLEMS WITH FREQUENTIST METHODS

---

- Just using the pure unified approach to estimate the upper limits may give problematic results in the presence of background.
  - In some cases, people found a statistical (*under-*)fluctuation of the background may lead to the exclusion of zero signal, which is unphysical.
    - ➔ Information is not sufficient to discriminate the **b-only** and **s+b** hypotheses.
  - Also, in some of the cases, when adding the channels with low signal sensitivity, may produce upper limits that are worse than without adding them.
- This is the reason why people introduce a modified frequentist method at LEP and LHC: **the CLs method**. Which will be discussed in the next lecture.





Not yet at the end of story!  
We will continue to discuss **hypothesis testing** &  
**upper limit calculation** again in the next lecture!

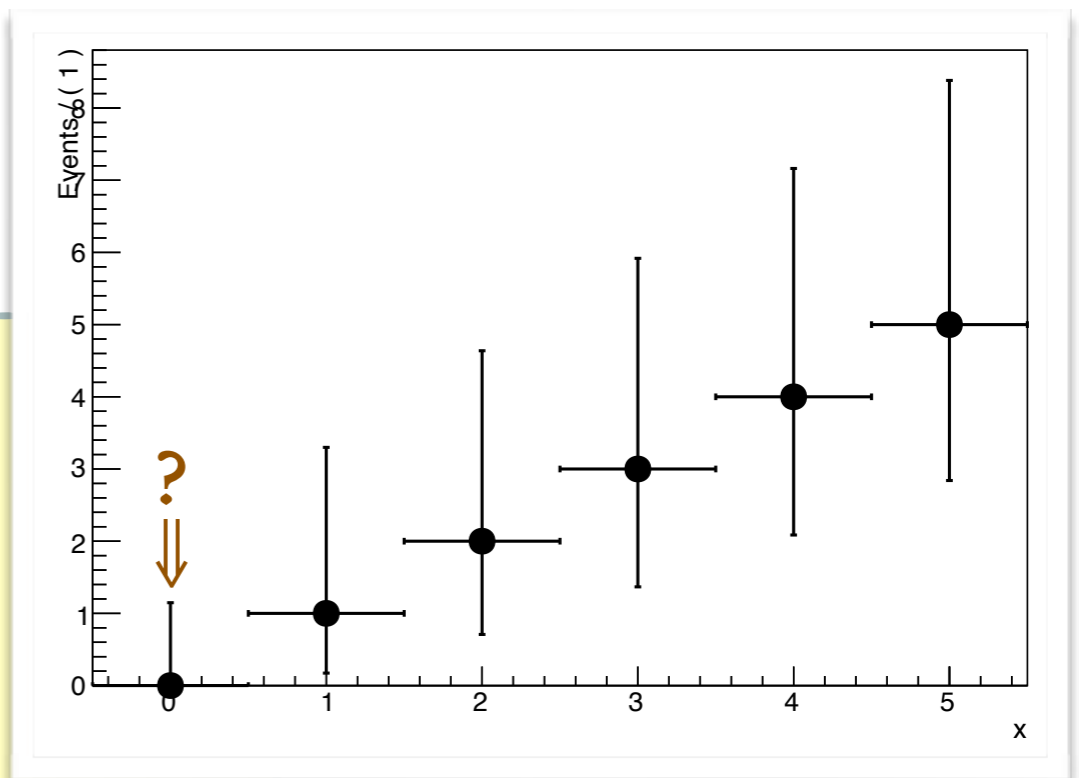


# COMMENT: ROOFIT ASYMMETRIC ERROR BARS?

- You may have notice one thing: RooFit actually put asymmetric errors on data points by default.
- Now we can finally come back to discuss this. **Why/how RooFit obtain those asymmetric errors and put on the figures?**
- An example code to demo:

example\_09.cc

```
TH1D *hist = new TH1D("hist",  
    "test histogram",6,-0.5,5.5);  
for(int i=0;i<6;i++)  
    hist->SetBinContent(i+1,i);  
  
RooRealVar x("x","x",-0.5,5.5);  
RooDataHist data("data","data",x,hist);  
  
RooPlot *frame = x.frame();  
data.plotOn(frame,LineWidth(2),MarkerSize(2.));  
frame->Draw();
```



*error bar for  $n=0$  is very tricky! Will discuss it later.*

# (A) SYMMETRIC ERROR BARS

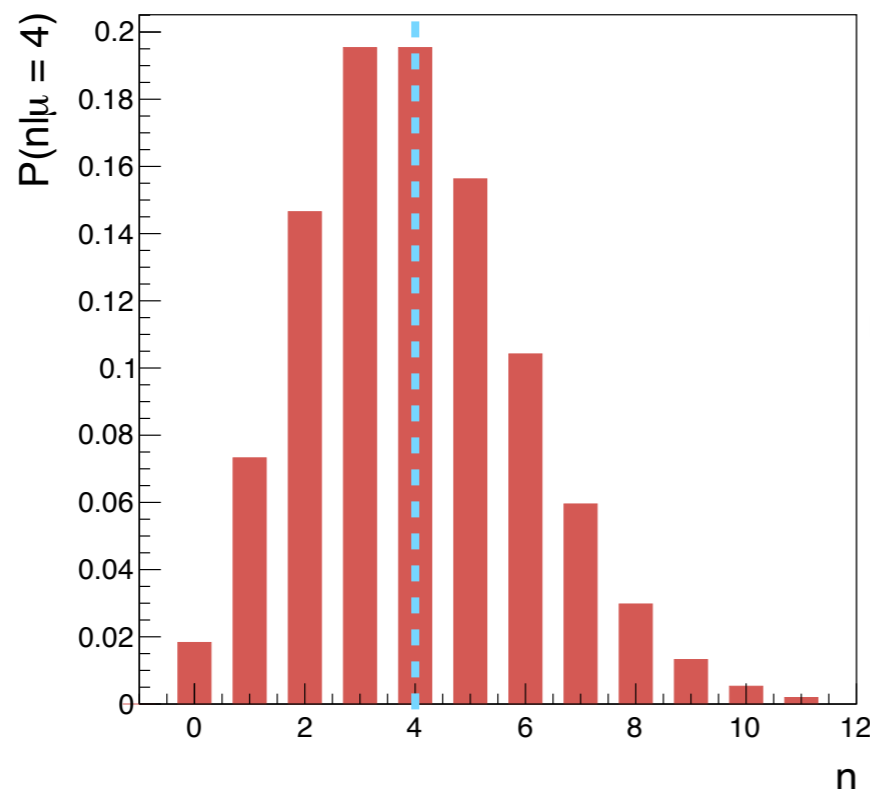
---

- We have to come back to the Poisson distribution:

$$p(n|\mu) = \frac{\mu^n e^{-\mu}}{n!}$$

This gives the probability of expected mean value  $\mu$  and finding  $n$ .

- The usual **square-root-of-n** error is obtained by just setting  $\mu$  to be observed value  $n$ , and take the variance of the distribution. For example, you find  $n=4$ , and set  $\mu=n=4$ :



Given variance is 4, so the “error bar” is  $\pm 2$ . But Poisson distribution is asymmetric...

$$p(4|\mu=2) = 0.090$$

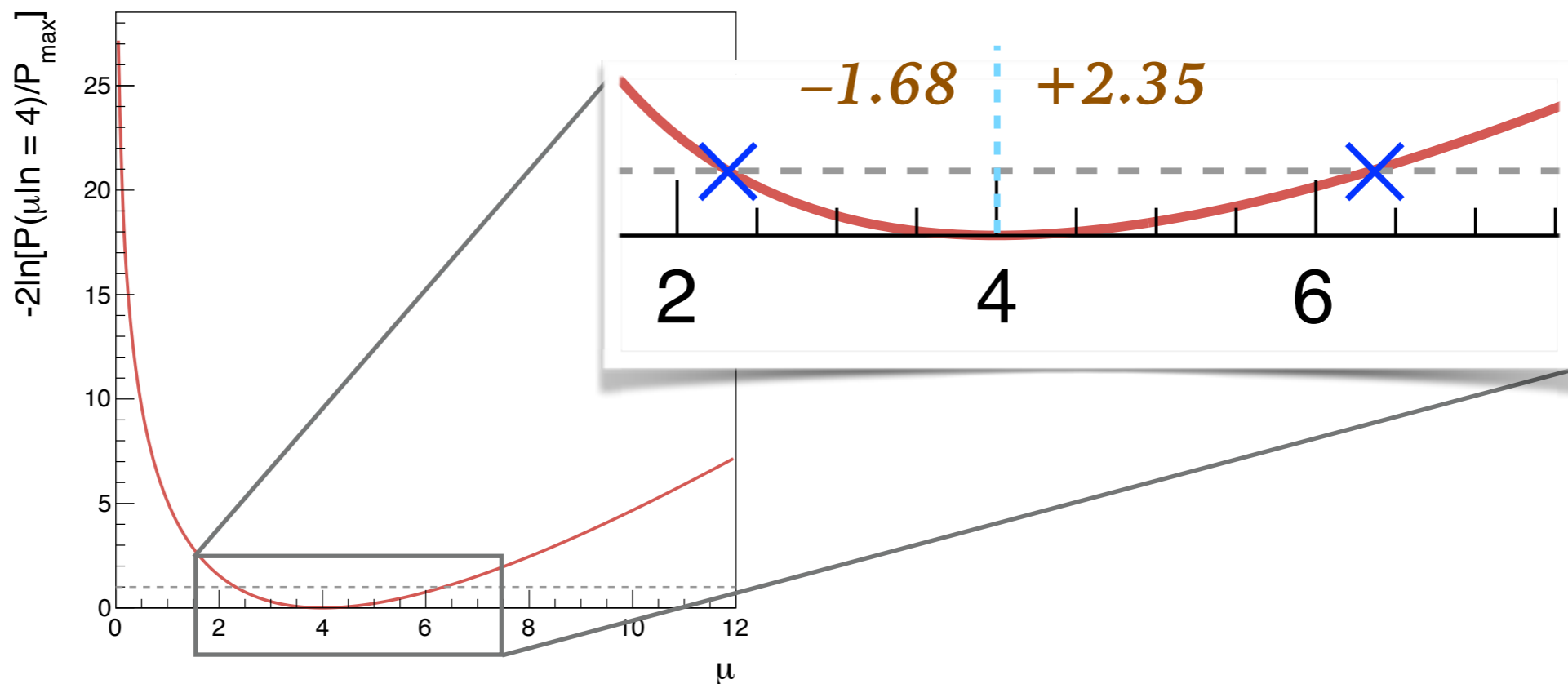
$$p(4|\mu=6) = 0.134$$

# ALTERNATIVE OPTION

- Based on what we discussed earlier in this lecture, inserting the observed  $n$  into the Poisson PDF:

$$L(\mu) = \frac{\mu^n e^{-\mu}}{n!}$$

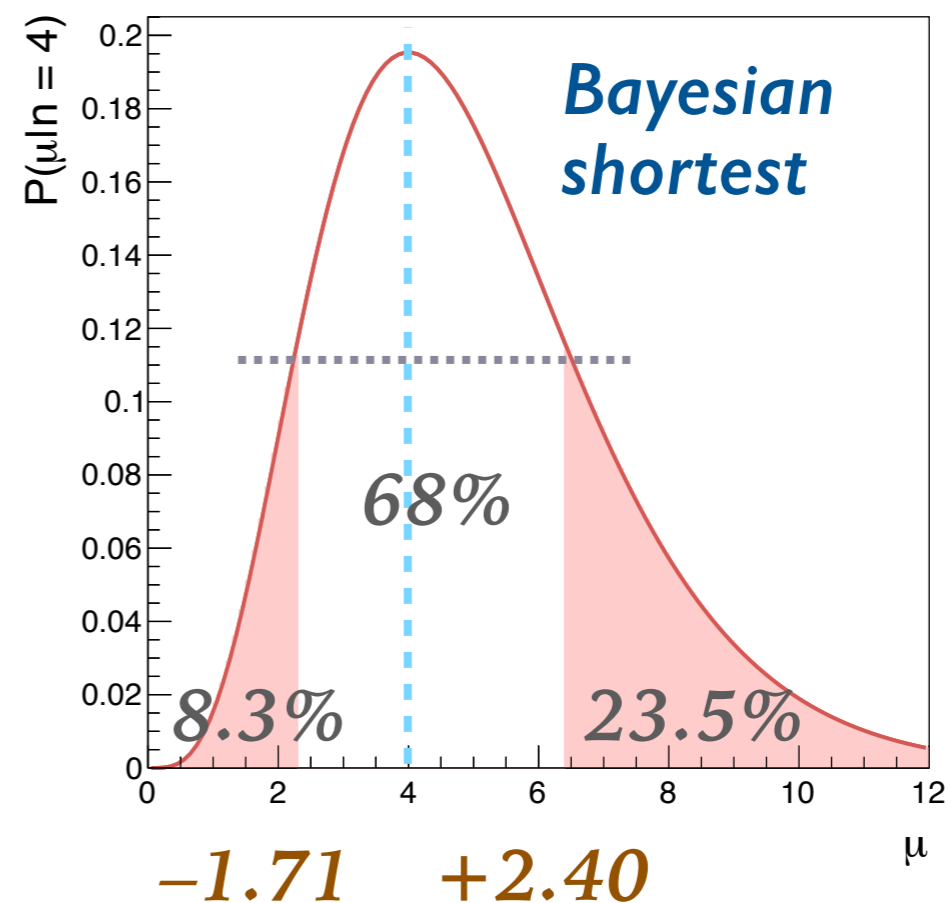
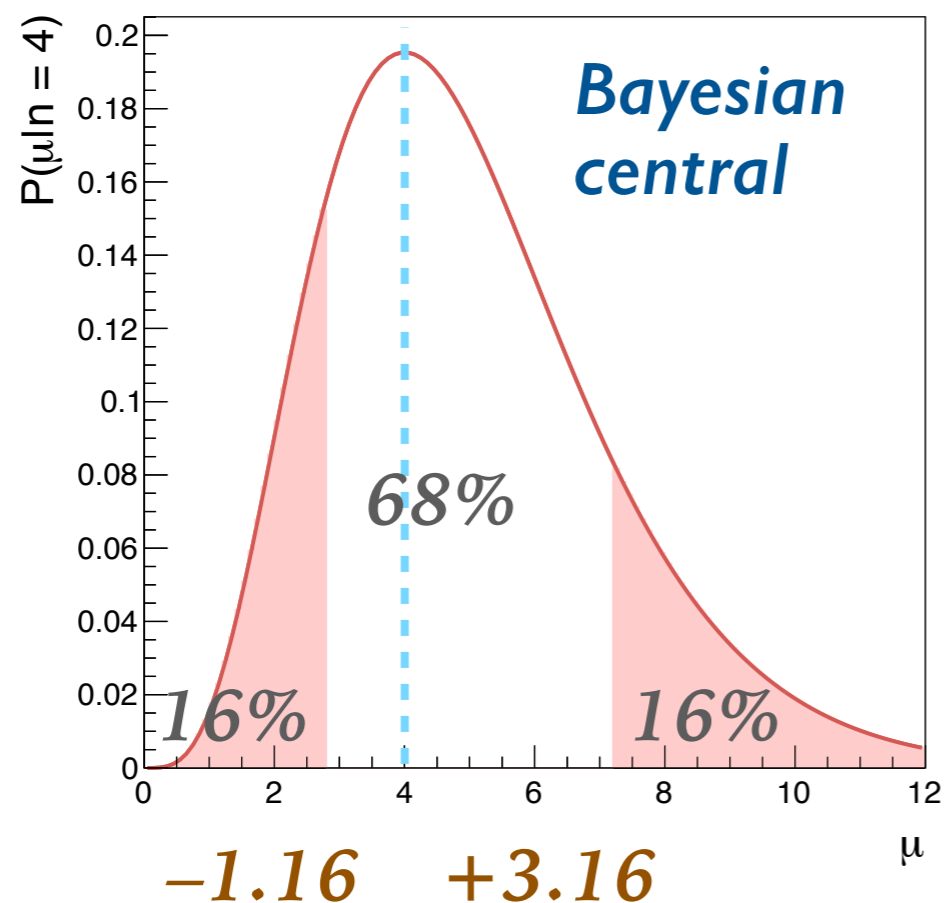
- Basically this is a **likelihood function of  $\mu$** . So let's do a likelihood scan as we introduced earlier in this lecture?



# MORE ALTERNATIVE OPTIONS?

- Well, we have discussed the Bayesian method, let's take the **posterior probability** with an uniform prior, one can at least come up with two more options:

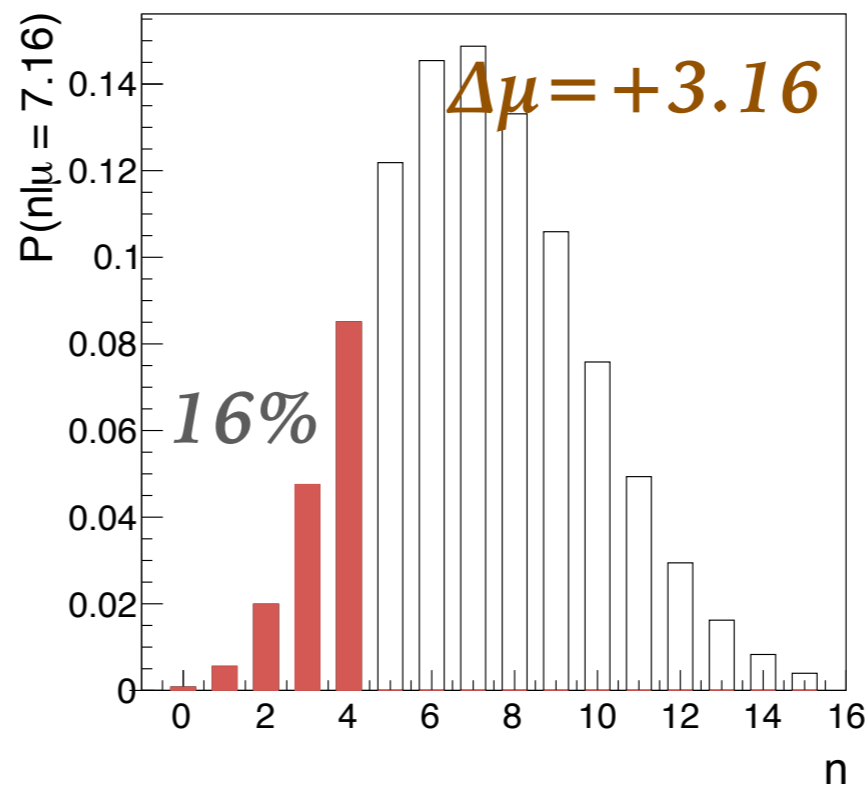
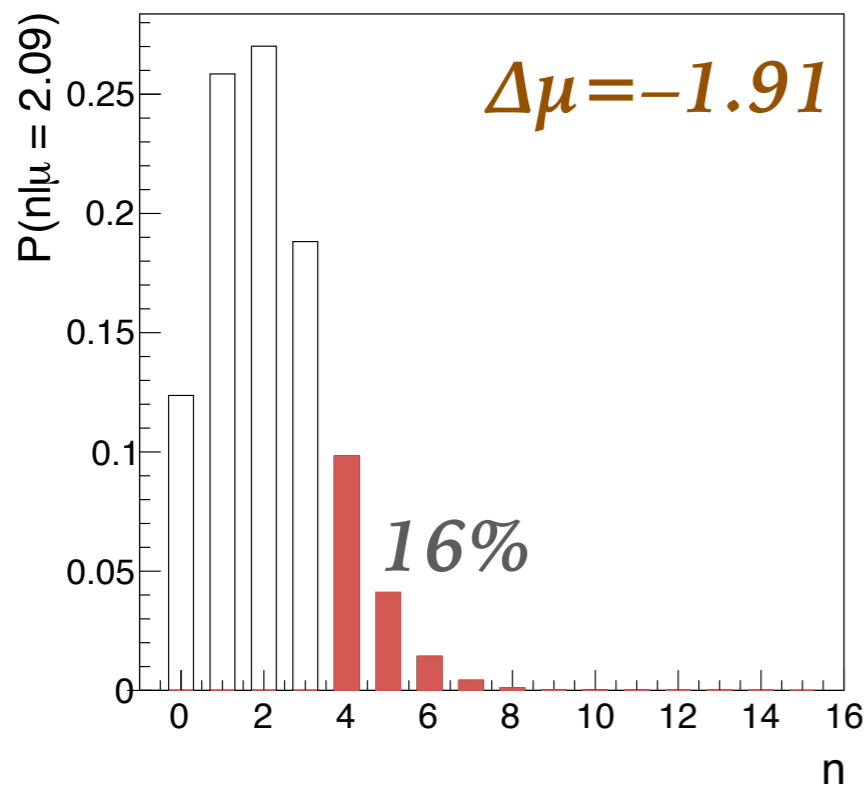
$$p(\mu|n) = \frac{\mu^n e^{-\mu}}{n!}$$



# THE FREQUENTIST APPROACH

---

- One can also adopt the full Neyman construction here, but since we only need to calculate the interval for a fixed given observed  $n$ , this can be done easily — find extreme values of  $\mu$  that are still being compatible with observed yield:
  - As  $\mu > 7.16$ , the probability to observe 4 events (or less) is  $< 16\%$ ;
  - As  $\mu < 2.09$ , the probability to observe 4 events (or more) is  $< 16\%$ .

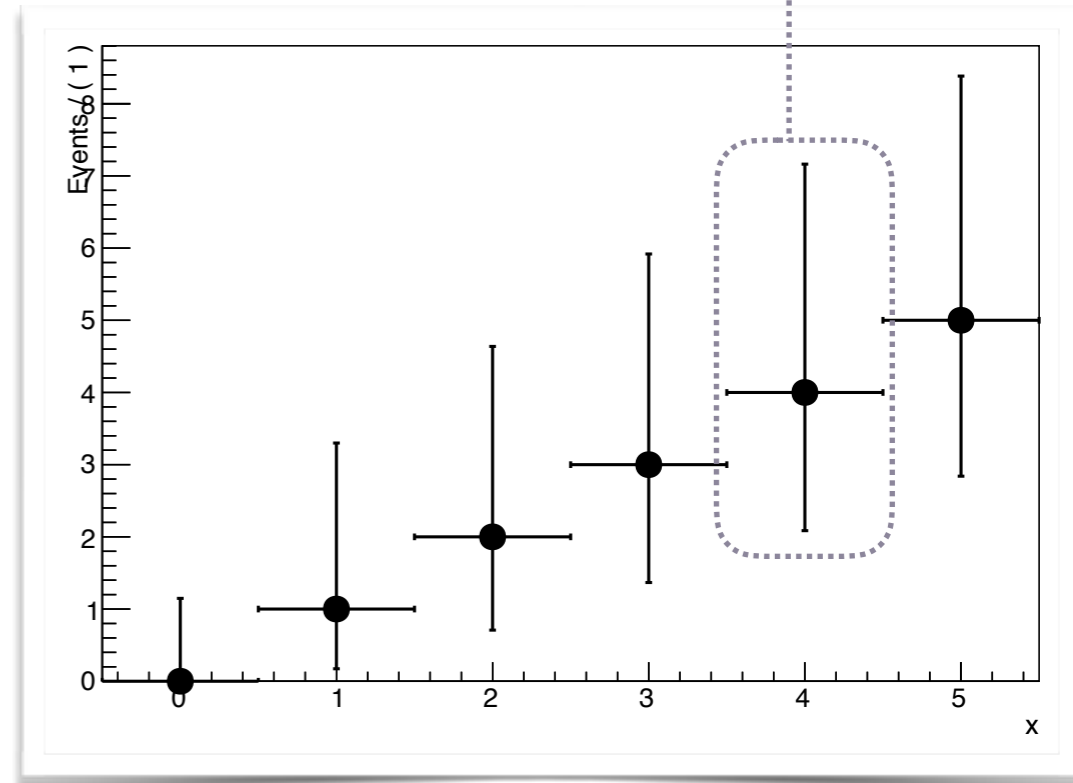
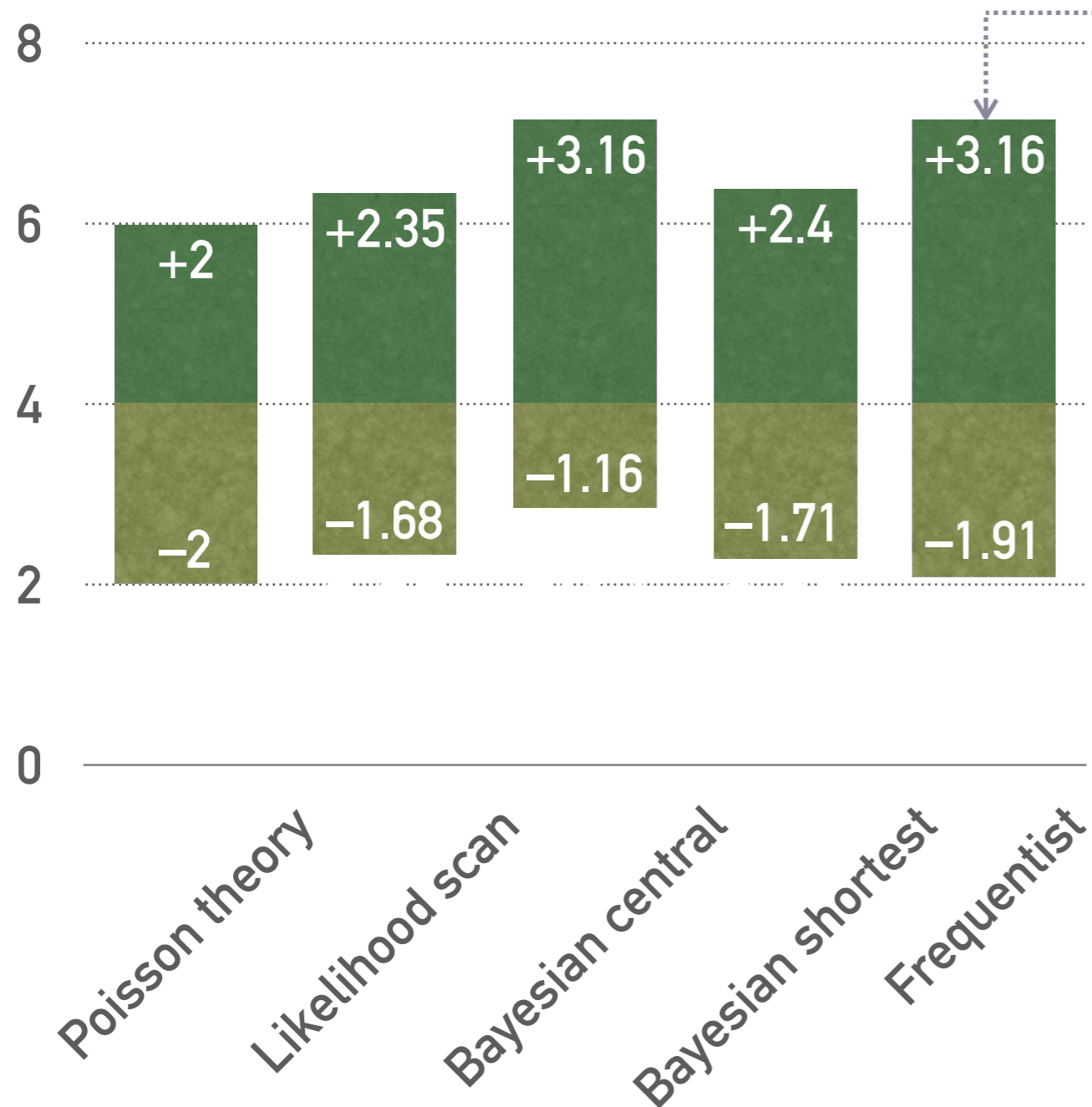


$$p(n|\mu) = \frac{\mu^n e^{-\mu}}{n!}$$



# ROOFIT CHOICE

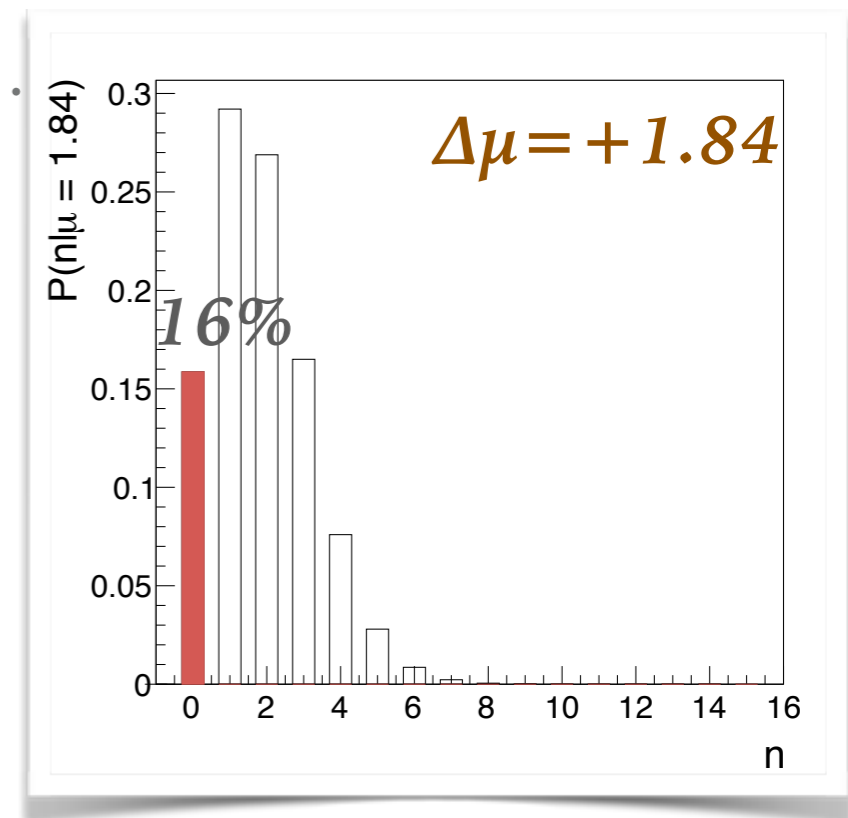
► Let's summarize all these different intervals:



RooFit chooses the **frequentist approach** as the default.

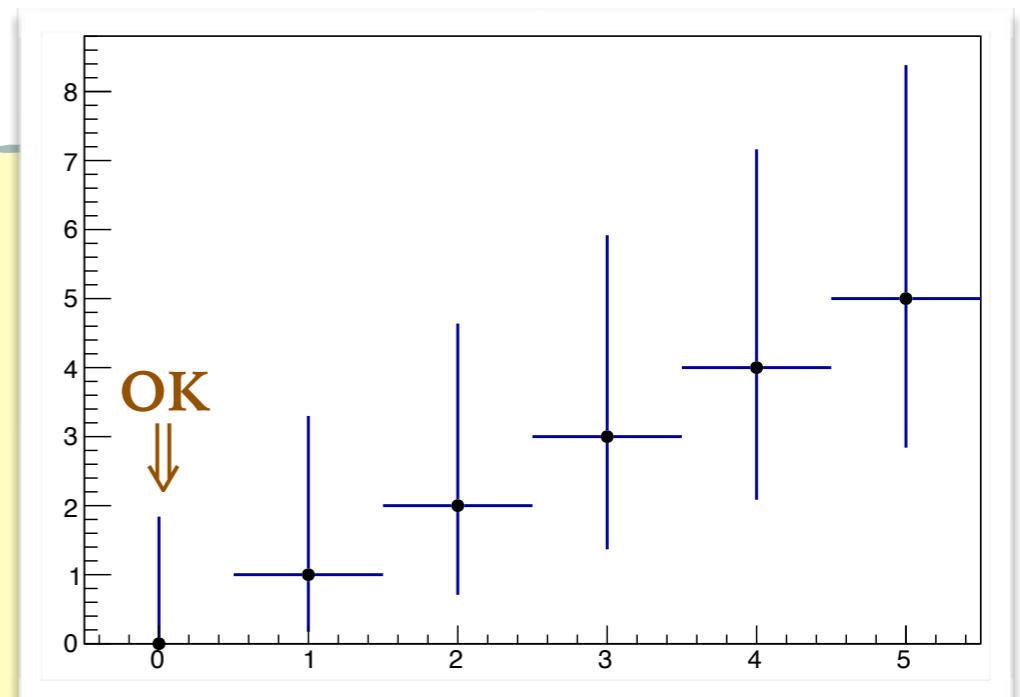
# ROOFIT CHOICE (CONT.)

- But there is an inconsistency for the case of  $n=0$ . This is due to the flip-flopping between 1-side and 2-side intervals.
- RooFit draws a positive error bar of around  $+1.2$ , but it should be in fact around  $+1.8$  if we adopt the same recipe as discussed above.
- The “correct” interval has been implemented within ROOT already.



example\_09a.cc

```
TH1D *hist = new TH1D("hist",  
    "test histogram",6,-0.5,5.5);  
for(int i=0;i<6;i++)  
    hist->SetBinContent(i+1,i);  
  
hist->SetBinErrorOption(TH1::kPoisson);  
hist->SetLineWidth(2);  
hist->SetMarkerStyle(20);  
hist->Draw("e0");
```



# SUMMARY

---

- In this lecture several methods of interval estimation have been discussed, including both Bayesian methods and frequentist methods.
- This should give you a more general picture behind just quoting an error bar from your fitter.
- For the next lecture, we are going to discuss the hypothesis test, and will touch the upper limit calculation again.