Lecture on limits: exercises

1. Bayes law

2. Frequentist limit and Gaussian approximation

3. Bayesian limit

4. Frequentist/Bayesian limits with background

5. Expected limit

6. CL_S limit

7. Limits with systematic uncertainties

8. Combining two channels

Note: most exercises require running RooT macros. For some exercises, the macros have to be modified.

Preliminary versions of the macros are available on the statistics-school virtual machine:

```
/statistics-school/limits/
```

Improved versions of the macros (same functionality, but better/simpler implementation) are available here:

```
http://www.desy.de/~sschmitt/LimitLecture/macros/
```

For downloading, You can use the two commands given below.

Load list of the macros:

```
wget -N -nd www.desy.de/~sschmitt/LimitLecture/macros.list
```

Load the macros using the list:

```
wget -N -nd -i macros.list
```

The macros included in this document are the improved versions.

Exercise 1: Bayes' law, test for a disease.

Test is positive in 99/100 cases if one has the disease

Test is positive in 1/100 cases if no disease

0.1% of the population have the disease

What is the probability to have the disease in case of a positive test?

A positive test

B have disease

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$$

P(B)=0.001 P(~B)=0.999

P(A∩B)=P(A|B)P(B)=0.001*0.99=0.00099 P(A∩~B)=P(A|~B)P(~B)=0.01*0.999=0.00999

P(A)=P(A∩B)+P(A∩~B)=0.01098

P(B|A)=0.99*0.001/0.01098=0.0901

Exercise 2a: calculate the Frequentist limit on μ for $N_{obs}=0$

Use the Poisson probability for $N=0$: $P_\mu(N_{obs}=0) = \exp[-\mu]$

How does the calculation look like for $N_{obs}=1,2,3,...$

Solution for $N_{obs}=0$:

1-CL=$\exp[-\mu]$

$\mu=-\ln[1-CL]=-\ln[0.05]=2.996$

Calculation for $N_{obs}=1,2,3,...$

1-CL=$\exp[-\mu](1+ \mu+ \mu^2/2+ \mu^3/6+...)$

Exercise 2b: calculate the Frequentist limit μ_0 for N_{obs}=2,10,100 and compare to the Gaussian approximation

For the Frequentist limit, use the macro `GetPoissonLimit.C`

```
void GetPoissonLimit(Double_t cl,Int_t n0) {
  /* calculates the limit on the number of events
     given n0 */
  Double_t mu=0.5*TMath::ChisquareQuantile(cl,2*(n0+1));
  cout<<"Limit on mu at "<<100*cl<<"%CL for nObs="<<n0
     <<" (Frequentist method) : "<<mu<<"\n";
}
```

$$CL = \frac{\sum_{N > N_{obs}} \exp(-\mu_0)\mu_0^N}{N!} \quad \text{has the inverse function}$$

$$\mu_0 = \frac{1}{2} \text{TMath::ChisquareQuantile}(CL, 2(N_{obs}+1))$$

For the Gaussian approximation, use the macro `GetGaussLimit.C`

```
void GetGaussLimit(Double_t cl,Int_t n0) {
  /* calculates the limit on the number of events
     given n0 in the Gaussian approximation. */
  Double_t mu=TMath::ErfcInverse(2.*(1.-cl))*TMath::Sqrt(2.*n0)+n0;
  cout<<"Limit on mu at "<<100*cl<<"%CL for nObs="<<n0
     <<" (Gaussian approximation) : "<<mu<<"\n";
}
```

The approximation is $\mu_0 = N_{obs}$ and $\sigma = sqrt(N_{obs})$. The macro uses the function `Tmath::ErfcInverse`

$$(1-CL) = \int_{\mu_0}^{\infty} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) dx \implies \text{TMath::ErfcInverse}(2(1-CL)) = \frac{x-\mu_0}{\sqrt{2}\sigma}$$

Collect the results in the table on page 12

Exercise 3a: calculate the Bayesian limit on μ for $N_{obs}=0$ with „flat prior"

Use Poisson´s law. How does the calculation look like for $N_{obs}=1,2,3,...$?

Compare to the Frequentist calculation from exercise 2.

Solution for $N_{obs}=0$:

1-CL=exp[-μ]

μ=-ln[1-CL]=-ln[0.05]=2.996

Calculation for $N_{obs}=1,2,3,...$

Integrate exp[-μ] μ^ N_{obs}/N_{obs}! over μ then normalize

Exercise 3b: Calculate the Bayesian limit on μ for N_{obs}=2,10,100 with „flat prior" (macro `GetBayesCL.C`). When using the macro, you have to test several values of μ_0 to find the CL=0.95.

Exercise 3c: modify the macro and use a prior distribution $P(\mu)=\mu$

Exercise 3d: modify the macro and use a flat prior for $0<\mu<\mu_{max}$=90, set the prior to zero for other values of μ

Collect the results in the table on page 12, together with the results of exercise 2.

The macro `GetBayesCL.C`

```
Double_t GetBayesPrior(Double_t mu) {
  // prior probability, used below
  Double_t p=1.0;
  return p;
}

void GetBayesCL(Double_t mu0,Int_t nObs) {
  /* calculates the Bayesian probability CL for my<=my0
     given nObs and the prior as defined above. */
  TRandom3 rnd;
  Int_t nTry=30000;
  Double_t sum[2],sum2[2];
  sum[0]=0.; sum[1]=0.; sum2[0]=0.; sum2[1]=0.;
  for(Int_t iTry=0;iTry<nTry;iTry++) {
    // 2*mu is drawn from a chi**2 distribution with ndf=2*(nobs+1)
    Double_t mu=0.5*TMath::ChisquareQuantile(rnd.Uniform(),2*(nObs+1));
    // prior probability is used as weight
    Double_t w = GetBayesPrior(mu);
    Int_t i= (mu>=mu0) ? 1 : 0;
    sum[i] += w;
    sum2[i] += w*w;
  }
  Double_t norm=sum[0]+sum[1];
  Double_t p=sum[1]/norm;
  Double_t e=TMath::Sqrt(sum2[1]*sum[0]*sum[0]+sum2[0]*sum[1]*sum[1])/norm/norm;
  cout<<"CL for nObs="<<nObs<<" and mu="<<mu0<<" is "<<p<<" +/- "<<e<<"\n";
}
```

The integral is calculated using Monte Carlo techniques. Random numbers are generated in the interval [0;1]: `rnd.Uniform()`. These are transformed to the distribution of the likelilhood, using the function: `ChisquareQuantile`. The prior distribution defines weights. These are summed up for two regions: $\mu<\mu_0$ and $\mu>\mu_0$. The normalisation of the posterior is determined from the sum-of-weights over both regions.

Exercise 4: calculate Frequentist/Bayesian limits for a given amount of background.

$$\mu = L(s+b) \quad , \text{where} \quad L=1 \quad \text{and} \quad s \geq 0$$

For the background test the cases b={0.5,2,3.5}. For the number of observed events test the cases N_{obs}={0,2}.

Use the macros from exercise 2 and 3, GetPoissonLimit.C and GetBayesCL.C, modified as needed.

For the Bayesian limit with background, modify the macro to use two arguments s,b instead of the argument mu0. Use a (flat) prior taking into account the physically allowed region of s:

$$P(s)=\begin{cases} 0 \text{ for } s<0 \\ 1 \text{ for } s \geq 0 \end{cases}$$

For the frequentist limit, divide by L and subtract b.

Collect the results in the table on page 12 together with the results from exercise 5,6,7.

For the Frequentist case: what is the meaning of the limit with s<0?

The case N_{obs}=0, b=3.5 gives the limit: s < -0.5 at 95%CL. But the allowed region is s>0.

All models (including the model with s=0) are excluded. Recall Frequentist limit definition:

„The models with s > -0.5 predict the observation with a probability of less than 5%."

Exercise 5: calculate expected limits. Use the macro GetExpectedLimit.C. It calculates the expected limit on the Poisson parameter μ.

For the background test the cases b={0.5,2,3.5}.

Do You have to subtract b from the limit returned by the macro?

Answer: The expected limit is on s+b, so b has to be subtracted.

Collect the results in the table on page 12.

The macro `GetExpectedLimit.C`.

```
void GetExpectedLimit(Double_t cl,Double_t mu) {
  /* calculate the expected limit, given the confidence level
     and the Poisson parameter mu. Note: does not work for very large mu */
  Int_t n0=(int)mu;
  Int_t n1=n0+1;
  Double_t p0=1.0;
  if(mu>0.0) p0=TMath::Exp(n0*TMath::Log(mu)-mu-TMath::LnGamma(n0+1.));
  Double_t p1=p0*mu/n1;
  Double_t sum=0.0;
  while((p0>0.)&&(n0>=0)) {
    sum += p0*0.5*TMath::ChisquareQuantile(cl,2*(n0+1));
    p0 *= n0/mu;
    n0--;
  }
  while(p1>0.0) {
    sum += p1*0.5*TMath::ChisquareQuantile(cl,2*(n1+1));
    n1++;
    p1 *= mu/n1;
  }
  cout<<"expected limit for CL="<<cl*100<<"% and mu="<<mu<<" is "<<sum<<"\n";
}
```

The average is calculated as a weighed sum of all limits for the possible observations, where the weights are taken as the Poisson probabilities.

The sum is split into two parts. The splitting is done at the nearest integer to the Poisson parameter mu.

First, the possible observations n0,n0-1,...,0 are summed up. Then the possible observations n0+1,n0+2,... are summed up. In both cases the summing is stopped as soon as the weight is zero (within the machine accuracy).

Exercise 6: calculate limits using the CL_S method.

For the background test the cases b={0.5,2,3.5}. For the number of observed events test the cases N_{obs}={0,2}.

Use the macro `GetCLs.C`. It calculates the modified Frequentist variable CL_S, given N_{obs} and expected signal, background. You have to try different values of the signal, until the desired CL is reached.

Collect the results in the table on page 12.

Why are the results so similar to the Bayesian case?

CL_S is calculated as a ratio of 2 Poisson sums. The Bayesian limit is calculated as the ratio of two integrals over the parameter μ, and the result is exactly the same (using the identities discussed in the lecture).

The macro `GetCLs.C`

```
void GetCLs(Double_t signal,Double_t bgr,Int_t nobs) {
  /* calculate CLs for the given
     signal, background , nobs. Does not work for high nobs. */
  Double_t cl_s;
  Double_t cl_b=0.0;
  Double_t cl_sb=0.0;
  Double_t lnGamma=0.0;
  Double_t logB=TMath::Log(bgr);
  Double_t logSB=TMath::Log(signal+bgr);
  for(Int_t i=0;i<=nobs;i++) {
    Double_t p_b=TMath::Exp(i*logB-bgr-lnGamma);
    Double_t p_sb=TMath::Exp(i*logSB-signal-bgr-lnGamma);
    cl_b += p_b;
    cl_sb += p_sb;
    lnGamma +=TMath::Log(i+1);
  }
  cl_s=cl_sb/cl_b;
  cout<<"For signal="<<signal<<" and bgr="<<bgr<<" CLS is "<<cl_s<<"\n";
}
```

This macro sums the probabilities for all possible N≤N_{obs}, given background only or background plus signal.

Exercise 7: limits with systematic uncertainties. Use a background error of 50% and a luminosity error of 10%.

For the background test the cases b={0.5,3.5}. For the number of observed events test the cases N_{obs}={0,2}. Calculate each case with/without the errors on the background and/or luminosity.

Collect the results in the table on page 12.

What is the influence of the systematic errors? Which systematic error is most relevant for the limit?

Answer: The systematic errors make the limits worse. The luminosity uncertainty acts on all cases, whereas the background error only affects the cases where the influence of the background on the CL_S limit is high.

The macro `GetCLsSys.C`:

```
void GetCLsSys(Double_t signal,Double_t bgr,Double_t dBgr,Double_t dLumi,Int_t
nobs) {
  /* calculate CLs for the given signal, background, errors
     and observed number of events, using Monte Carlo methods */
   TRandom3 rnd;
  Int_t nTry=30000;
  Double_t nexp_sb=0.0;
  Double_t nexp_b=0.0;
  for(Int_t iTry=0;iTry<nTry;iTry++) {
    /* get Luminosity from truncated Gaussian */
    Double_t l=1.0;
    if(dLumi>0.0) {
      do {
        l=rnd.Gaus(1.0,dLumi);
      } while(l<=0.0);
    }
    /* get Background from truncated Gaussian */
    Double_t b=bgr;
    if(dBgr>0.0) {
      do {
        b=rnd.Gaus(bgr,dBgr);
      } while(b<=0.0);
    }
    Int_t n_b=rnd.Poisson(l*b);
    Int_t n_sb=rnd.Poisson(l*(signal+b));
    if(n_b<=nobs) nexp_b += 1.0;
    if(n_sb<=nobs) nexp_sb += 1.0;
  }
  Double_t cl_s=nexp_sb/nexp_b;
  Double_t dcl_s= TMath::Sqrt(nexp_sb+nexp_b*cl_s*cl_s)/nexp_b;
  cout<<"CLSsys="<<cl_s<<" +/- "<<dcl_s<<" for B="<<bgr<<" +/- "<<dBgr
      <<", L=1 +/- "<<dLumi<<", signal="<<signal<<"\n";
}
```

The macro is discussed in the lecture.

Exercise 8: events are observed in two channels. Calculate limits (CL_S method) on the number of signal events, (a) for each of the two channels alone, (b) using the plain sum of both channels and (c) for the combination of the two channels.

For (a) and (b) use `GetCLs.C` (take care of the efficiency!) or modify the macro below.
For (c), use the macro `GetCLsCombined.C`

Collect the results in the table on page 12.

The macro `GetCLsCombined.C`

```
Double_t GetX(Int_t nChan,Int_t *nobs,Double_t signal,Double_t *eff,Double_t *b)
{
  Double_t x=0.;
  for(Int_t i=0;i<nChan;i++) {
    Double_t s=signal*eff[i];
    x += nobs[i]*s/b[i];
  }
  return x;
}
void GetCLsCombined(Double_t signal) {
  TRandom3 rnd;
  // data to be tested
  static Int_t nObs[2]  ={7    , 2  };
  static Double_t bgr[2]={6.5 ,1.8 };
  static Double_t eff[2]={0.5 ,0.5 };
  Int_t firstChannel=0;
  Int_t lastChannel=1;
  // number of toy experiments
  Int_t nTry=100000;
  // count toy wrt data experiments
  Double_t ndata_sb=0.0;
  Double_t ndata_b=0.0;
  // observed X from data
  Double_t Xobs=GetX(firstChannel,lastChannel,nObs,signal,eff,bgr);
  // toy experiments
  for(Int_t iTry=0;iTry<nTry;iTry++) {
     Int_t n_b[2],n_sb[2];
    for(Int_t i=firstChannel;i<=lastChannel;i++) {
      n_b[i]=rnd.Poisson(bgr[i]);
      n_sb[i]=rnd.Poisson(signal*eff[i]+bgr[i]);
    }
    Double_t X_b=GetX(firstChannel,lastChannel,n_b,signal  ,eff,bgr);
    Double_t X_sb=GetX(firstChannel,lastChannel,n_sb,signal,eff,bgr);
    if(X_b<=Xobs) ndata_b++;
    if(X_sb<=Xobs) ndata_sb++;
  }
  Double_t cls=ndata_sb/ndata_b;
  Double_t
cls_err=TMath::Sqrt(ndata_sb)/ndata_b*TMath::Sqrt(1.+ndata_sb/ndata_b);
  cout<<"CLS(data)="<<cls<<" +/- "<<cls_err<<"\n";
}
```

Probabilities are calculated using MC experiments, similar to the macros discussed earlier.

Results of Exercises 2/3

	$N_{obs}=0$	$N_{obs}=2$	$N_{obs}=10$	$N_{obs}=100$
Exercise 2				
Frequentist limit	3.0	6.3	17.0	118.1
Gauss approx	0.0	4.3	15.2	116.4
Exercise 3				
a/b: Bayesian „flat prior"	3.0	6.3	17.0	118.1
c: Bayesian prior=µ	4.7	7.7	18.2	119.2
d: Bayesian flat prior 0<µ<100	3.0	6.3	17.0	89.7

Results of exercises 4,5,6,7

	b=0.5		b=2		b=3.5	
	$N_{obs}=0$	$N_{obs}=2$	$N_{obs}=0$	$N_{obs}=2$	$N_{obs}=0$	$N_{obs}=2$
Exercise 4						
Bayesian	3.0	5.8	3.0	4.8	2.8	4.3
Frequentist	2.5	5.8	1.0	4.3	-0.5	2.8
Exercise 5						
expected	3.3		4.2		4.9	
Exercise 6						
CL_S	3.0	5.8	3.0	4.8	3.0	4.3
Exercise 7						
$\sigma_b/b=50\%$	3.0	5.8	The calculation for b=2 is not requested here		3.0	4.9
$\sigma_L/L=10\%$	3.1	6.0			3.2	4.6
$\sigma_b/b=50\%$, $\sigma_L/L=10\%$	3.1	6.0			3.1	5.0

Results of exercise 8

	Method	N_{obs}	**bgr**	CLS Limit
(a)	Channel 1	7	6.5	14.8
	Channel 2	2	1.8	9.9
(b)	Add channels	9	8.3	8.2
(c)	Combined w=s/b	(7,2)	(6.5,1.8)	7.3