Workspace

Kenji Hamano

1 PDF and Likelihood

Gaussian signal PDF (on n_s)

$$f_s(x|\theta_s = \mu_x, \sigma_x) = \frac{1}{\sigma_x\sqrt{2\pi}} e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}} \tag{1}$$

Exponential background PDF (on n_b)

$$f_b(x|\theta_b = \lambda) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}} \tag{2}$$

Total (signal + background) PDF on n

$$\begin{aligned} f(x|\mu,\theta) &= f(x|\mu,\mu_b,\theta_s+\theta_b) = f(\mu,\mu_b,\mu_x,\sigma_x,\lambda) && (3)\\ &= \frac{\mu s f_s(x|\mu_x,\sigma_x) + \mu_b b f_b(x|\lambda)}{\mu s + \mu_b b} && (4) \end{aligned}$$

Liklihood

$$L(x|\mu,\theta) = L(x|\mu,\mu_b,\mu_x,\sigma_x,\lambda) = \mathrm{Pois}(n,\langle n\rangle) \prod_{j=1}^{N} f(x_j|\mu,\mu_b,\mu_x,\sigma_x,\lambda) \tag{5}$$

These can be implemented with, for example, C++ without using any other statistical tools.

1.1 RooFit

Using RooFit, the above PDF are:

```
// define variable x
RooRealVar x("x", "x", 0., 5.);

// define signal shape parameters
RooRealVar meanx("meanx", "signal mean", 2.5);          // fixed
RooRealVar sigmax("sigmax", "signal std dev", 0.2);     // fixed
// define signal PDF
RooGaussian signal("signal","gaussian signal PDF", x, meanx, sigmax);

// define background parameters
RooRealVar lambda("lambda", "lambda", 1., 0., 5.);      // floating
RooFormulaVar alpha("alpha", "-1./lambda", RooArgList(lambda));
// define background PDF
RooExponential background("background", "background exponential PDF",
```

```
                                x, alpha);

// define signal yield
RooRealVar s("s", "signal yield", 100);                  // fixed
RooRealVar mu("mu", "signal strength", 1., -10., 10.); // floating
RooFormulaVar mus("mus", "mu*s", RooArgList(mu, s));

// define background yield
RooRealVar b("b", "background yield", 1000);                        // fixed
RooRealVar mub("mub", "background strength", 1., -10., 10.);    // floating
RooFormulaVar mubb("mubb", "mub*b", RooArgList(mub, b));

// add signal + background PDFs to create model (extended)
RooAddPdf model("model", "gaussian plus exponential PDF",
                RooArgList(signal, background), RooArgList(mus, mubb));
```

Similarly, need to code every step of calculation.

1.2 RooFit and Workspace

All of the model information can be contained in so called "workspace".

```
RooWorkspace w("w");

// define variable x
RooRealVar x("x", "x", 0., 5.);

// define signal shape parameters
RooRealVar meanx("meanx", "signal mean", 2.5);          // fixed
RooRealVar sigmax("sigmax", "signal std dev", 0.2);    // fixed
// define signal PDF
RooGaussian signal("signal","gaussian signal PDF",
                                    x, meanx, sigmax);

// define background parameters
RooRealVar lambda("lambda", "slope", 1., 0., 2.);      // floating
RooFormulaVar alpha("alpha", "-1./lambda", RooArgList(lambda));
// define background PDF
RooExponential background("background",
    "background exponential PDF", x, alpha);

// define signal yield
RooRealVar s("s", "signal yield", 100.);                   // fixed
RooRealVar mu("mu", "signal strength", 0., -10., 10.);  // floating
RooFormulaVar mus("mus", "mu*s", RooArgList(mu, s));

// define background yield
```

```
RooRealVar b("b", "background yield", 1000.);                    // fixed
RooRealVar mub("mub", "background strength", 1., -10., 10.);  // floating
RooFormulaVar mubb("mubb", "mub*b", RooArgList(mub, b));

// add signal + background PDFs
RooAddPdf model("model", "gaussian plus exponential PDF",
                RooArgList(signal, background), RooArgList(mus, mubb));

// import everything in workspace
w.import(model);
```

This makes it easier to configure the model:

```
ModelConfig c("config");
c.SetWorkspace(w);
c.SetPdf("model");
c.SetParametersOfInterest("mu");
c.SetNuisanceParameters("mub,lambda");
c.SetObservables("x");

// import config in workspace
w.import(c);
```

2 RooFit + HistFactory

We did simple example so far, but in real analysis, the models are more complicated. There are many different background samples and many uncertainties to be considered. This makes model building very difficult.
For, example, even for simple counting experiment (one bin histograms), the Likelihood is

$$\mathcal{L}(\mu, \sigma^0_{j\in CR}; \delta_s, \delta_j.\delta_i) = \mathrm{Pois}(n|n_T)N(m_{\delta_s}|\delta_s) \prod_{j\in CR} \mathrm{Pois}(n_j|\sigma^0_j) \prod_j N(m_{\delta_j}|\delta_j) \prod_i N(m_{\delta_i}|\delta_i) \tag{6}$$

where indecie are

- j is over backgroud processes.
- $j \in CR$ is a part of j which are measured in control regions.
- i is over systematic effects.

and

- μ: parameter of interest = signal strength.
- n: number of events in the signal region.
- n_T: total number of expected events given by

$$n_T = \sum_l \mu L \sigma_l (1 + \epsilon_l^s \delta_s) \prod_i (1 + \epsilon_{li}^s \delta_i) + \sum_j L \sigma_j^0 (1 + \epsilon_j^b \delta_j) \prod_i (1 + \epsilon_{ji}^s \delta_i) \quad (7)$$

- δ_s: systematic uncertainty on signal cross section σ^s.
- δ_j: systematic uncertainty on effective background cross section σ_j.
- δ_i: systematic uncertainty of the systematic effect i.
- $m_{\delta_s}, m_{\delta_j}, m_{\delta_i}$: auxiliary measurements of corresponding δ. When generating toy MC, the m_δ fluctuate around the value of δ.
- $n_{j \in CR}$: number of events in the CR, wihch is scaled by en extrapolation coefficient τ to estimate the number of events in the signa region.

$$\tau_{j \in CR} = 1 + \epsilon_j \delta_j \quad (8)$$

- σ_l^s: effective cross section of signal in channel l.
- ϵ_l^s: relative uncertainty on σ_l^s.
- σ_j^0: nominal effective cross section for background j.
- ϵ_j^b: relative uncertainty on σ_j^b.
- ϵ_{li}^s: relative change in the effective cross section due to the ith systematic effect on signal channel l.
- ϵ_{ji}^s: relative change in the effective cross section due to the ith systematic effect on background channel j.
- L: integrated luminosity.

HistFactory helps us building models in RooStat workspace.
To use Histfactory, we need:

HistFactorySchema.dtd

The following are actual files used for "Same-sign dilepton analysis".

- 11 background components: WZ, ZZ, ttbarW, ttbarZ, WWjj, Fake, DrellYan, ttbar, WW, Wgamma and DPI.
- 3 or 4 uncertainties for each component.
- 2 signal regions: positive pair and negative pair.

2.1 XML files

The information to build a model is written in 3 xml files:
LikeSign_ee_combine.all.m100.xml : combine pos and neg channels.

```
<!DOCTYPE Combination  SYSTEM 'HistFactorySchema.dtd'>
<Combination OutputFilePrefix="./LikeSign_ee_combine" >
  <Input>./LikeSign_channel_ee_combine.pos.m100.xml</Input>
  <Input>./LikeSign_channel_ee_combine.neg.m100.xml</Input>
  <Measurement Name="example" Lumi="1." LumiRelErr="0.028" >
    <POI>SigXsecOverSM</POI>
  </Measurement>
</Combination>
```

POI (Parameter Of Interest) = μ : signal strength.
Other two files are:
LikeSign_channel_ee_combine.neg.m100.xml : details of negative channel.
LikeSign_channel_ee_combine.pos.m100.xml : details of positive channel.

```
<!DOCTYPE Channel  SYSTEM 'HistFactorySchema.dtd'>
<Channel Name="ee_combine_pos" InputFile="./SignalHists_ee_combine_pos.root" Hist
  <Data HistoName="Data_ee_combine_pos" HistoPath="" />
  <Sample Name="Signal_pos" HistoPath="" HistoName="Signal_ee_combine_pos">
    <OverallSys Name="SignalStat_pos" High="1.034" Low="0.96"/>
    <OverallSys Name="MCCorr_pos" High="1.034" Low="0.96"/>
    <NormFactor Name="SigXsecOverSM" Val="1.0" Low="0." High="2." />
  </Sample>
  <Sample Name="WZ_pos" HistoPath="" NormalizeByTheory="True" HistoName="WZ_ee_c
    <OverallSys Name="WZStat_pos" High="1.03342302448206" Low="0.966576975517936"
    <OverallSys Name="DibosonXsec_pos" High="1.06999529423121" Low="0.9300047057
    <OverallSys Name="MCCorr_pos" High="1.0603303691012" Low="0.939669630898802"/
  </Sample>
  <Sample Name="ZZ_pos" HistoPath="" NormalizeByTheory="True" HistoName="ZZ_ee_c
    <OverallSys Name="ZZStat_pos" High="1.07086228034328" Low="0.929137719656723"
    <OverallSys Name="DibosonXsec_pos" High="1.05002043318349" Low="0.9499795668
    <OverallSys Name="MCCorr_pos" High="1.06031875766244" Low="0.939681242337556"
  </Sample>
  <Sample Name="ttW_pos" HistoPath="" NormalizeByTheory="True" HistoName="ttW_ee
    <OverallSys Name="ttWStat_pos" High="1.04909925821265" Low="0.950900741787354
    <OverallSys Name="ttWXsec_pos" High="1.22006358177323" Low="0.779936418226775
```

```
  <OverallSys Name="MCCorr_pos" High="1.06040268456376" Low="0.939597315436242"
</Sample>
<Sample Name="ttZ_pos" HistoPath="" NormalizeByTheory="True" HistoName="ttZ_ee_
  <OverallSys Name="ttZStat_pos" High="1.09644670050761" Low="0.903553299492386
  <OverallSys Name="ttZXsec_pos" High="1.21996615905245" Low="0.780033840947546
  <OverallSys Name="MCCorr_pos" High="1.06091370558376" Low="0.939086294416244"
</Sample>
<Sample Name="WWjj_pos" HistoPath="" NormalizeByTheory="True" HistoName="WWjj_
  <OverallSys Name="WWjjStat_pos" High="1.0710636321412" Low="0.928936367858802
  <OverallSys Name="WWjjXsec_pos" High="1.5" Low="0.5"/>
  <OverallSys Name="MCCorr_pos" High="1.0604969809568" Low="0.939503019043196"/
</Sample>
<Sample Name="Fake_pos" HistoPath="" NormalizeByTheory="False" HistoName="Fake
  <OverallSys Name="FakeStat_pos" High="1.06232950491725" Low="0.93767049508274
  <OverallSys Name="FakeFactor_pos" High="1.18264482464362" Low="0.817355175356
  <OverallSys Name="MCCorr_pos" High="1.05152973434243" Low="0.948470265657569"
</Sample>
<Sample Name="DrellYan_pos" HistoPath="" NormalizeByTheory="True" HistoName="D
  <OverallSys Name="DrellYanStat_pos" High="1.09672362755651" Low="0.9032763724
  <OverallSys Name="DrellYanXsec_pos" High="1.07000134553283" Low="0.9299986544
  <OverallSys Name="QFlipSF_pos" High="1.09001951022605" Low="0.909980489773951
  <OverallSys Name="MCCorr_pos" High="1.06082144779333" Low="0.939178552206674"
</Sample>
<Sample Name="tt_pos" HistoPath="" NormalizeByTheory="True" HistoName="tt_ee_co
  <OverallSys Name="ttStat_pos" High="1.12163108155408" Low="0.878368918445922"
  <OverallSys Name="ttXsec_pos" High="1.06002800140007" Low="0.93997199859993"/
  <OverallSys Name="QFlipSF_pos" High="1.11527243028818" Low="0.884727569711819
  <OverallSys Name="MCCorr_pos" High="1.06078637265197" Low="0.939213627348034"
</Sample>
<Sample Name="WW_pos" HistoPath="" NormalizeByTheory="True" HistoName="WW_ee_c
  <OverallSys Name="WWStat_pos" High="1.09251336898396" Low="0.907486631016043"
  <OverallSys Name="DibosonXsec_pos" High="1.11996434937611" Low="0.88003565062
  <OverallSys Name="QFlipSF_pos" High="1.10713012477718" Low="0.892869875222816
  <OverallSys Name="MCCorr_pos" High="1.06078431372549" Low="0.93921568627451"/
</Sample>
<Sample Name="Wgamma_pos" HistoPath="" NormalizeByTheory="True" HistoName="Wga
  <OverallSys Name="WgammaStat_pos" High="1.0886199970607" Low="0.9113800029393
  <OverallSys Name="WgammaXsec_pos" High="1.14000881791016" Low="0.859991182089
  <OverallSys Name="QFlipSF_pos" High="1.124063097046" Low="0.875936902954"/>
  <OverallSys Name="MCCorr_pos" High="1.06050066134326" Low="0.939499338656738"
</Sample>
<Sample Name="DPI_pos" HistoPath="" NormalizeByTheory="True" HistoName="DPI_ee
  <OverallSys Name="DPIStat_pos" High="1.11643835616438" Low="0.883561643835616
  <OverallSys Name="DPIXsec_pos" High="2" Low="0"/>
  <OverallSys Name="MCCorr_pos" High="1.06164383561644" Low="0.938356164383562"
</Sample>
```

```
</Channel>
```

Same uncertainty name = correlated uncertainty.
The Fake background is data-driven: NormalizeByTheory="False".

2.2 Input histograms

We need histograms with yields as a input to create workspace. The following example is a simple counting experiment and histograms have only one bin with entry of yields, nSignal_pos, nttZ_pos, nWZ_pos, etc.

```
// create pos histograms which have number of events for each component
TFile *tf_pos = new TFile("SignalHists_ee_combine_pos.root","recreate");
TH1F *hs_pos = new TH1F("Signal_ee_combine_pos","",1,0,1);
hs_pos->Fill(0.5,nSignal_pos);
hs_pos->Write("Signal_ee_combine_pos");

TH1F *hb3_pos = new TH1F("ttZ_ee_combine_pos","",1,0,1);
hb3_pos->Fill(0.5,nttZ_pos);
hb3_pos->Write("ttZ_ee_combine_pos");

TH1F *hb2_pos = new TH1F("WZ_ee_combine_pos","",1,0,1);
hb2_pos->Fill(0.5,nWZ_pos);
hb2_pos->Write("WZ_ee_combine_pos");

TH1F *hb4_pos = new TH1F("Wgamma_ee_combine_pos","",1,0,1);
hb4_pos->Fill(0.5,nWgamma_pos);
hb4_pos->Write("Wgamma_ee_combine_pos");

TH1F *hb5_pos = new TH1F("ttW_ee_combine_pos","",1,0,1);
hb5_pos->Fill(0.5,nttW_pos);
hb5_pos->Write("ttW_ee_combine_pos");

TH1F *hb6_pos = new TH1F("WW_ee_combine_pos","",1,0,1);
hb6_pos->Fill(0.5,nWW_pos);
hb6_pos->Write("WW_ee_combine_pos");

TH1F *hb10_pos = new TH1F("ZZ_ee_combine_pos","",1,0,1);
hb10_pos->Fill(0.5,nZZ_pos);
hb10_pos->Write("ZZ_ee_combine_pos");

TH1F *hb7_pos = new TH1F("Fake_ee_combine_pos","",1,0,1);
hb7_pos->Fill(0.5,nFake_pos);
hb7_pos->Write("Fake_ee_combine_pos");

TH1F *hb1_pos = new TH1F("DrellYan_ee_combine_pos","",1,0,1);
```

```
hb1_pos->Fill(0.5,nDrellYan_pos);
hb1_pos->Write("DrellYan_ee_combine_pos");

TH1F *hb9_pos = new TH1F("WWjj_ee_combine_pos","",1,0,1);
hb9_pos->Fill(0.5,nWWjj_pos);
hb9_pos->Write("WWjj_ee_combine_pos");

TH1F *hb11_pos = new TH1F("DPI_ee_combine_pos","",1,0,1);
hb11_pos->Fill(0.5,nDPI_pos);
hb11_pos->Write("DPI_ee_combine_pos");

TH1F *hb8_pos = new TH1F("tt_ee_combine_pos","",1,0,1);
hb8_pos->Fill(0.5,ntt_pos);
hb8_pos->Write("tt_ee_combine_pos");

TH1F *hd_pos = new TH1F("Data_ee_combine_pos","",1,0,1);
hd_pos->Fill(0.5,nObs_pos);
hd_pos->Write("Data_ee_combine_pos");
tf_pos->Close();

// create neg histograms which has number of events for each component
TFile *tf_neg = new TFile("SignalHists_ee_combine_neg.root","recreate");

< skip negative histogram creation part >
```

SignalHists_ee_combine_pos.root and SignalHists_ee_combine_neg.root will be used to create a workspace.

Shape information

If we want to include shape information such as MET shape, the input histogams are MET histograms instead of one bin histograms.

2.3 Creating workspace

The following command create the workspace.

```
hist2workspace LikeSign_ee_combine.all.m100.xml
```

Control region

The Fake background was estimated from a Fake control region. This background can be treated differently from other background. An example is

LikeSign_channel_mumu.all.m600.CR.xml and LikeSign_channel_mumu.all.m600.SR.xml
Now, the fake background in the LikeSign_channel_mumu.all.m600.SR.xml is

```
<Sample Name="Fake" HistoPath="" NormalizeByTheory="False" HistoName="unitHisto
  <!-- this Extrapolation factor is what 1 event in control region corresponds
  <NormFactor Name="Extrapolation" Val="0.2" Low=".1" High="1.0" />
  <ShapeFactor Name="Bkg2Shape" />
</Sample>
```

and in the LikeSign_channel_mumu.all.m600.CR.xml

```
<!DOCTYPE Channel  SYSTEM 'HistFactorySchema.dtd'>
  <Channel Name="CR_mumu" InputFile="./SignalHists_mumu.root" HistoName="" >
    <Data HistoName="Data_CR_mumu" HistoPath="" />
    <Sample Name="Fake" HistoPath="" NormalizeByTheory="False" HistoName="unitHisto
      <ShapeFactor Name="Bkg2Shape"/>
    </Sample>
  </Channel>
```

So, in this case, we create two histograms Data_CR_mumu for the control region and that is connected to the unitHistogram in the signal region via normalization factor NormFactor.

3 RooStat

RooStat provides many tools to make the following calculation part easier:

```
ProfileLikelihoodTestStat //set up test statistics
ToyMCSampler //set up toys
FrequentistCalculator //get q0 distribution
```

Example macros to use these tools:

```
StandardFrequentistCLsDemo.C
StandardBayesianMCMCDemo.C
OneSidedFrequentistUpperLimitWithBands_asymptotic.C
```

3.1 Macro: StandardFrequentistCLsDemo

This macro does frequenist CLs calculations.

```
void StandardFrequentistCLsDemo(const char * fileName =0,
                                const char * wsName = "combined",
                                const char * modelSBName = "ModelConfig",
                                const char * dataName = "obsData",
```

```
                                int npoints = 10,
                                double poimin = 0,
                                double poimax = 3,
                                int ntoys=1000 )

// read in workspace from the input file
RooWorkspace * w = dynamic_cast<RooWorkspace*>( file->Get(wsName) );

// get models from WS
// get the modelConfig out of the file
ModelConfig* bModel = NULL;
ModelConfig* sbModel = (ModelConfig*) w->obj(modelSBName);

// construct test statistics and set up toys
ProfileLikelihoodTestStat profll(*sbModel->GetPdf());
profll.SetOneSided(1);

TestStatistic * testStat = &profll;

FrequentistCalculator* hc = new FrequentistCalculator(*data, *bModel, *sbModel);

ToyMCSampler *toymcs = (ToyMCSampler*)hc->GetTestStatSampler();
toymcs->SetTestStatistic(testStat);

hc->SetToys(ntoys,ntoys);

// start calculation
const RooArgSet * poiSet = sbModel->GetParametersOfInterest();
RooRealVar *poi = (RooRealVar*)poiSet->first();

// fit the data first
sbModel->GetPdf()->fitTo(*data);
double poihat  = poi->getVal();

// calculate limit
HypoTestInverter calc(*hc);
calc.SetConfidenceLevel(0.95);

calc.UseCLs(true);
calc.SetVerbose(true);

HypoTestInverterResult * r = calc.GetInterval();

// print out results
std::cout << " expected limit (median) " <<  r->GetExpectedUpperLimit(0) << std
std::cout << " expected limit (-1 sig) " << r->GetExpectedUpperLimit(-1) << std
```

```
std::cout << " expected limit (+1 sig) " << r->GetExpectedUpperLimit(1) << std::
std::cout << " expected limit (-2 sig) " << r->GetExpectedUpperLimit(-2) << std
std::cout << " expected limit (+2 sig) " << r->GetExpectedUpperLimit(2) << std::
```

This uses tools such as ProfileLikelihoodTestStat, FrequentistCalculator and HypoTestInverter.

3.2 Run Macro

To do actual calculation, we need a run macro:
LikeSign_ee_combine_2.C :

```
// create pos histograms which have number of events for each component
TFile *tf_pos = new TFile("SignalHists_ee_combine_pos.root","recreate");
TH1F *hs_pos = new TH1F("Signal_ee_combine_pos","",1,0,1);
hs_pos->Fill(0.5,nSignal_pos);
hs_pos->Write("Signal_ee_combine_pos");

TH1F *hb3_pos = new TH1F("ttZ_ee_combine_pos","",1,0,1);
hb3_pos->Fill(0.5,nttZ_pos);
hb3_pos->Write("ttZ_ee_combine_pos");

< skip the rest of histogram creation part >

// create workspace
gSystem->Exec("hist2workspace LikeSign_ee_combine.all.m100.xml");

// execute the macro
StandardFrequentistCLsDemo("LikeSign_ee_combine_combined_example_model.root",
                           "combined","ModelConfig");
```

Then you can get limit results in your log file:

```
The computed upper limit is: 0.982396 +/- 0.0755381
expected limit (median) 1.16432
expected limit (-1 sig) 0.891718
expected limit (+1 sig) 1.55212
expected limit (-2 sig) 0.60671
expected limit (+2 sig) 2.05829
```

3.3 Tutorial

Let us run the limit calculation with HistFactory + RooStat.

```
login to symmetry
$ lsetup root
```

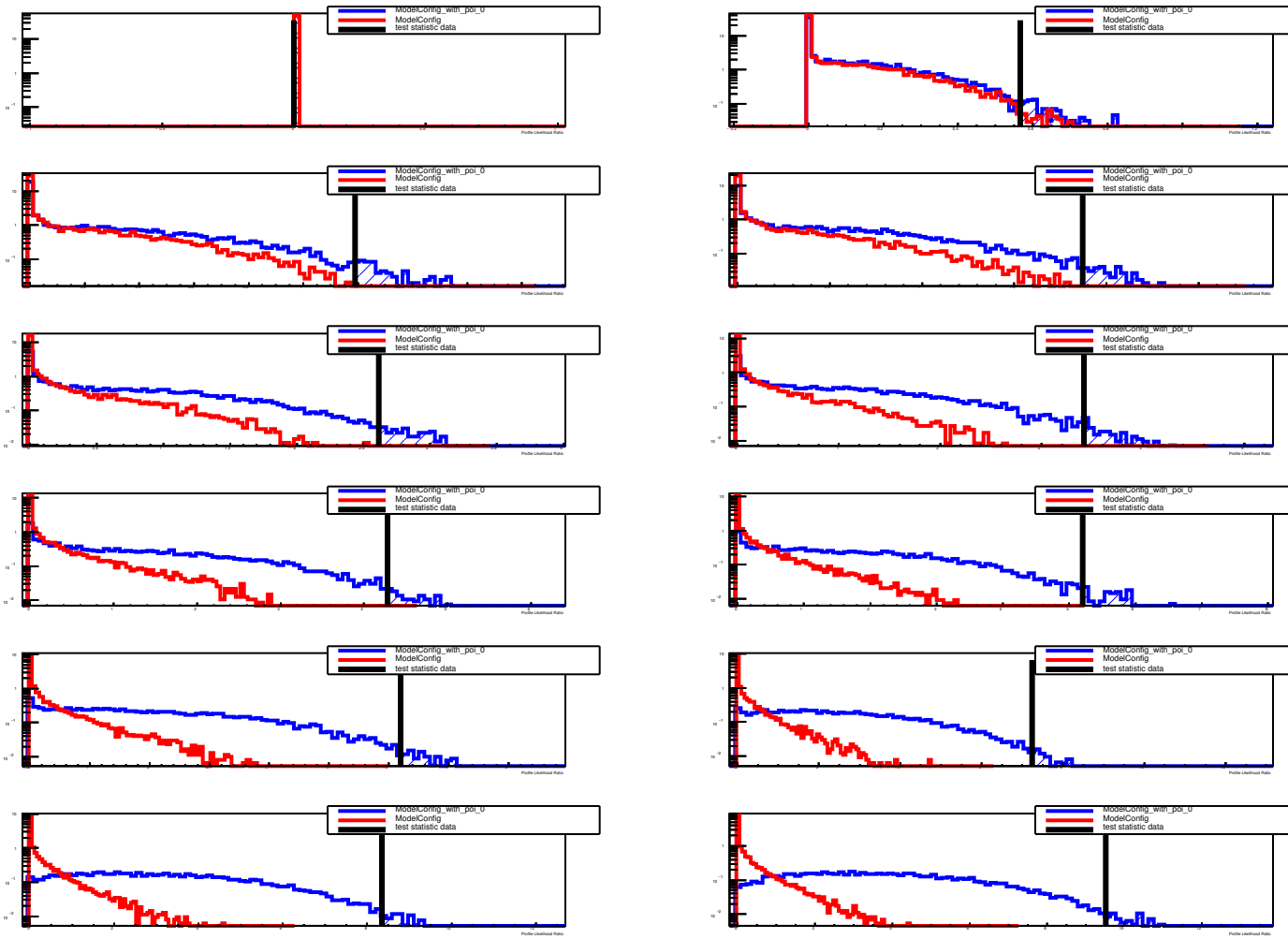

Figure 1: Test statistics

```
$ mkdir testRooStat
$ cd testRooStat
$ cp /hep300/data/khamano/statStudy/RooStat/* .
$ root -l -b -q LikeSign_ee_combine_2.C+ >& log.txt &
$ tail log.txt
```

After the job is finished, take a look at the log.txt and other *.eps files.

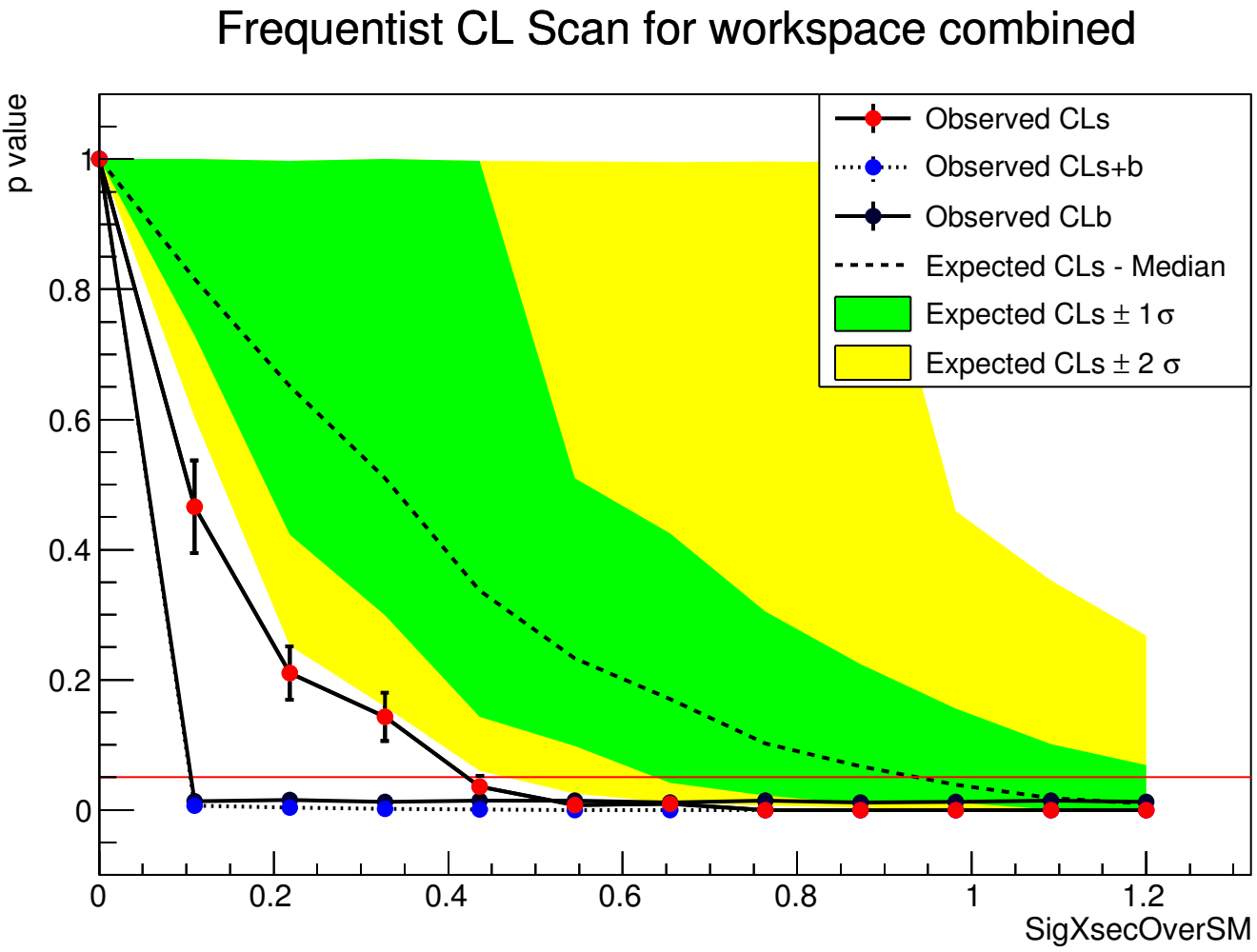

Figure 2: P-value

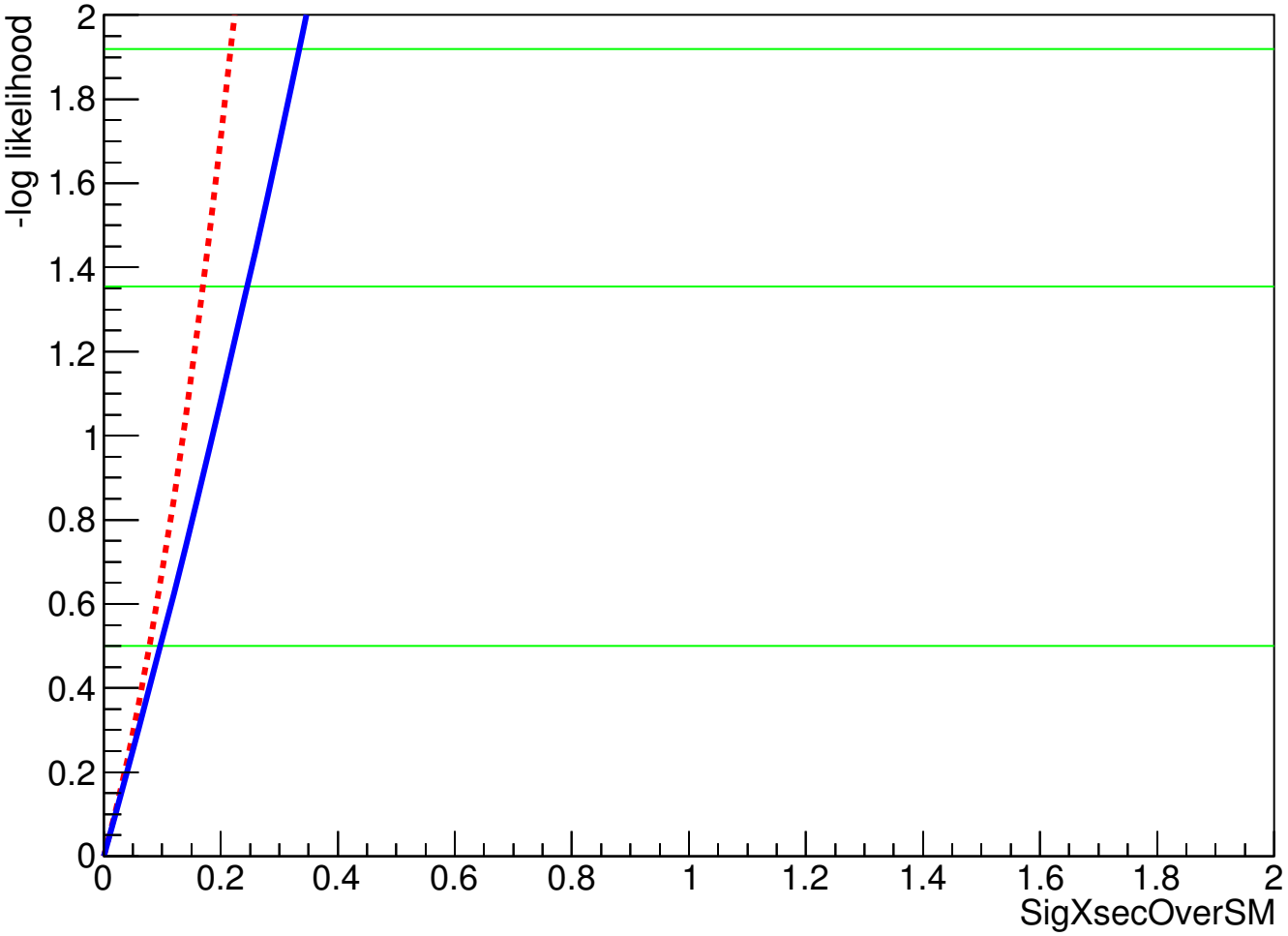

Figure 3: Profile Likelihood ratio