Statistical Methods for Particle Physics

Tutorial on multivariate methods

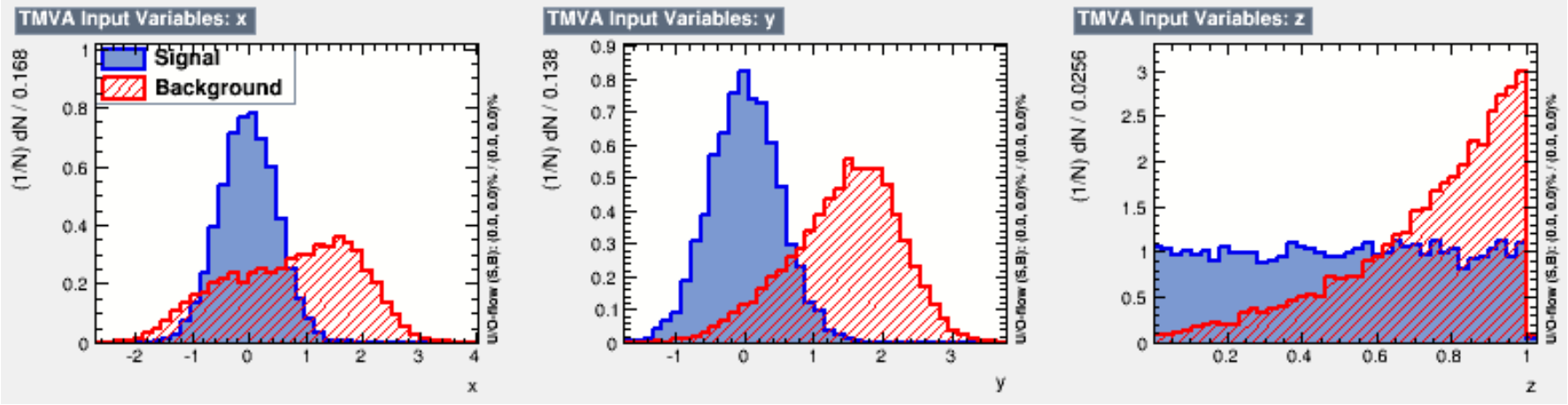
www.pp.rhul.ac.uk/~cowan/stat_trisep.html

Tutorial on Statistics
TRISEP School
27, 28 June 2016

Glen Cowan
Physics Department
Royal Holloway, University of London
g.cowan@rhul.ac.uk
www.pp.rhul.ac.uk/~cowan

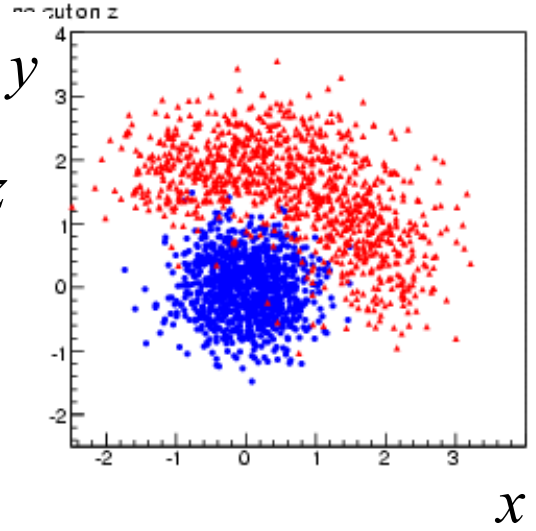
Test example with TMVA

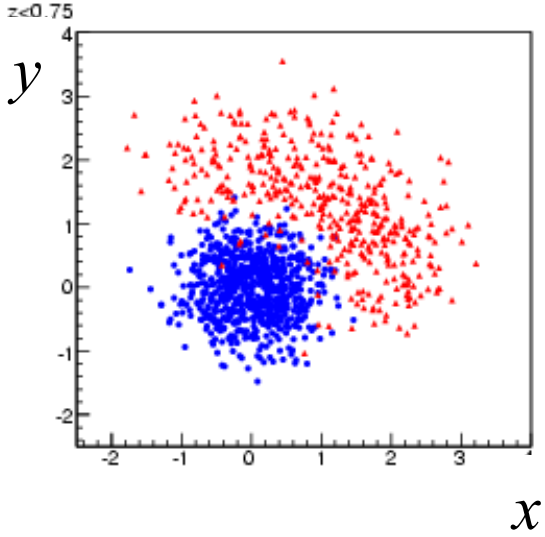
Suppose signal (blue) and background (red) events are each characterized by 3 variables, x, y, z:



Test example (x, y, z)

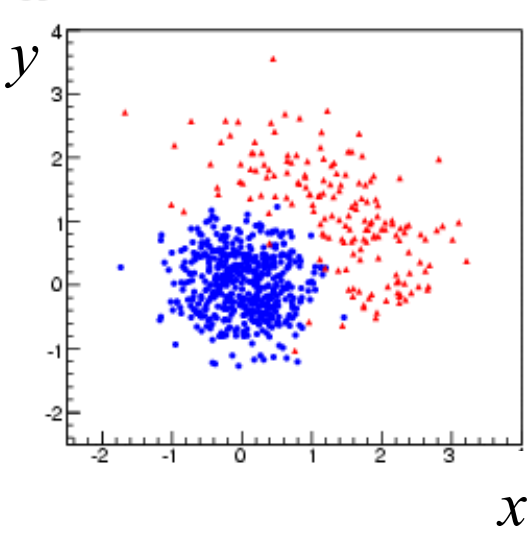
no cut on z

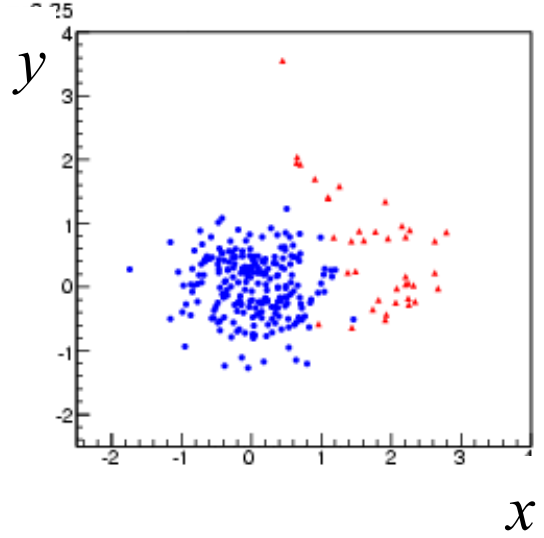




$z < 0.75$

$z < 0.5$





$z < 0.25$

Goal of test example: discovery of signal

We want to define a region of (x,y,z) space using a test statistic t and to search for evidence of the signal process by counting events in this region.

The number of events n that we observe we observe will follow a Poisson distribution with mean $s + b$, where s and b are the expected number of signal and background events. Goal is to maximize the expected significance for test of $s = 0$ hypothesis if signal is present.

We will see (tomorrow) that the expected discovery significance can be estimated using

$$Z = \sqrt{2\left((s+b)\ln\left(1+\frac{s}{b}\right)-s\right)}$$

(For $s << b$ this is approximately $s/\sqrt{b}$.)

Code for tutorial

The code for the tutorial is here:

```
http://www.pp.rhul.ac.uk/~cowan/stat/trisep/tmva
```

Copy the file `tmvaExamples.tar` to your own working directory and unpack using

```
tar -xvf tmvaExamples.tar
```

This will create some subdirectories, including

```
generate
train
test
analyze
inc
```

Generate the data

cd into the subdirectory generate, build and run the program generateData by typing

```
make
./generateData
```

This creates two data files: `trainingData.root` and `testData.root` an each of which contains a TTree (*n*-tuple) for signal and background events.

Train the classifier

cd into the subdirectory train, build and run the program generateData by typing

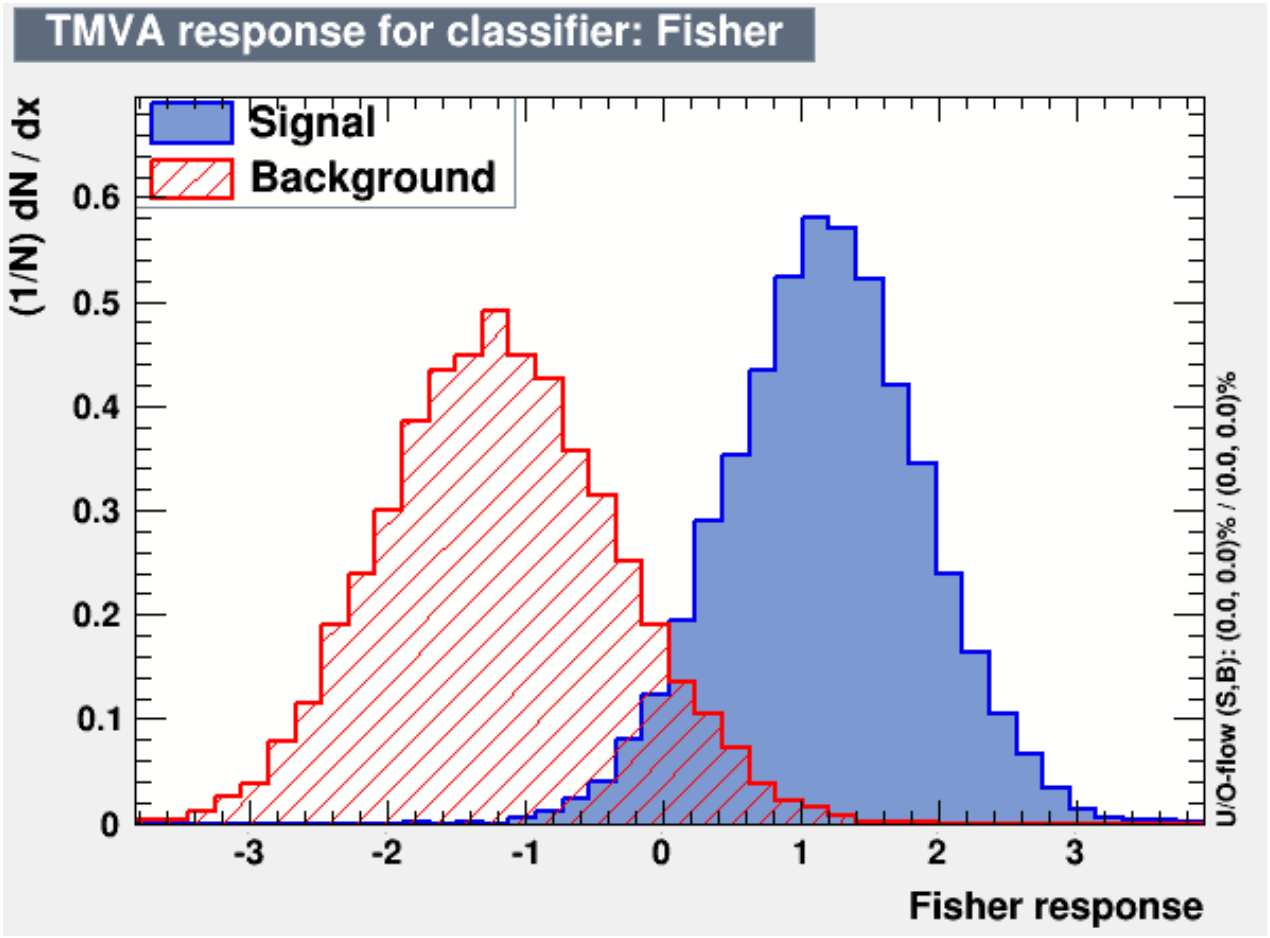
```
make
./tmvaTrain
```

This uses the data in `trainingData.root` to train a Fisher discriminant and writes the coefficients to a file `tmvaTest_Fisher.weights.xml`.

It also creates a file `TMVA.root`. Copy this file into the subdirectory test, and then you call look at various diagnostic histograms using the macros there. Try e.g.

```
.x plotall.C
```

Distribution of classifier output (Fisher)



Analyze the data

cd into the subdirectory analyze, build and run the program analyzeData by typing

```
make
./analyzeData
```

This reads in the data from `testData.root` and selects events with values of the Fisher statistic t greater than a given threshold $t_{\rm cut}$ (set initially to zero). The program counts the number of events passing the threshold and from these estimates the efficiencies

$$\varepsilon_{\rm s} = P(t \geq t_{\rm cut}|{\rm s}) \,,$$

$$\varepsilon_{\rm b} = P(t \geq t_{\rm cut}|{\rm b}) \,.$$

Discovery significance

Suppose the data sample we have corresponds to an integrated luminosity of $L = 20$ fb^{-1}, and that the cross sections of the signal and background processes are $\sigma_s = 0.2$ fb and $\sigma_b = 10$ fb.

The program then calculates the expected number of signal and background events after the cut using

$$s = \sigma_s L \varepsilon_s$$

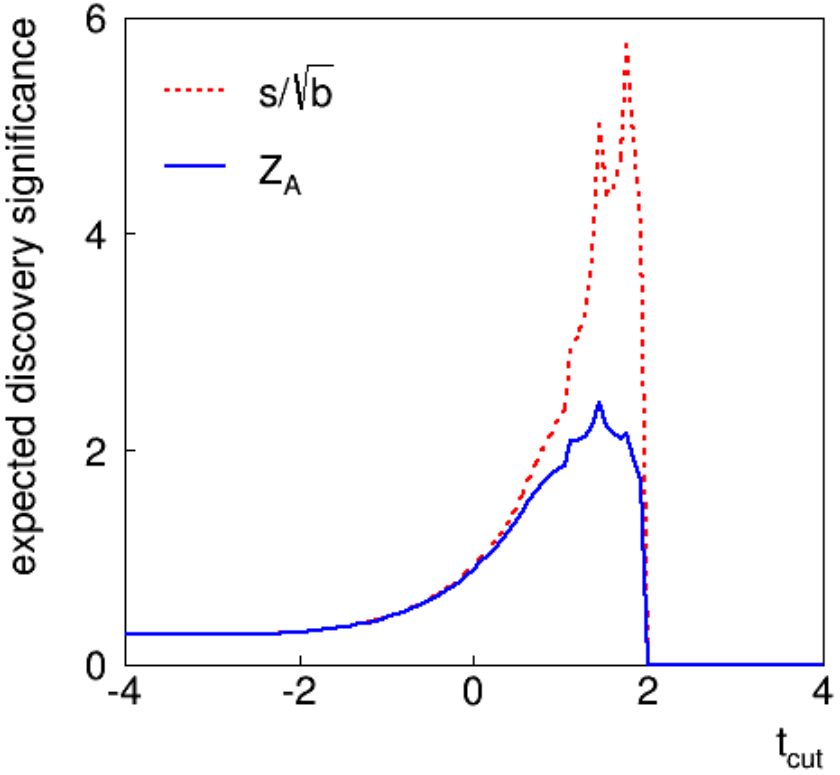
$$b = \sigma_b L \varepsilon_b$$

From these numbers we can compute the expected discovery significance using the "Asimov" formula:

$$Z = \sqrt{2\left((s+b)\ln\left(1+\frac{s}{b}\right) - s\right)}$$

Extending the example

We would like to optimize the value of t_{cut} to obtain the maximum expected discovery significance. To do this, modify the code in **`analyzeData.cc`** to loop over a range of values of t_{cut}. You should find a plot something like



Hints for making plots

To make a simple plot, you can use the root class **`TGraph`**. See

`https://root.cern.ch/root/html/TGraph.html`

Suppose you have values $x_1,\ldots, x_n$ and $y_1,\ldots, y_n$ in arrays **`x`** and **`y`** of type double and (int) length **`n`**. Then plot with

```
TGraph* g = new TGraph(n, x, y);
g->Draw();
```

This can be done either in your C++ code or in a root macro.

To make a plot with error bars use **`TGraphErrors`**.

Using different classifiers

The Fisher discriminant is a linear classifier and is not expected to give the optimal performance for this problem. You can try adding classifiers to tmvaTrain.cc by adding the lines

```
factory->BookMethod(TMVA::Types::kBDT, "BDT", "NTrees=200:BoostType=AdaBoost");
```

(adds Boosted Decision Tree with 200 boosting iterations)

```
factory->BookMethod(TMVA::Types::kMLP, "MLP", "H:!V:HiddenLayers=3");
```

(adds a Multilayer Perceptron with a single hidden layer with 3 nodes)

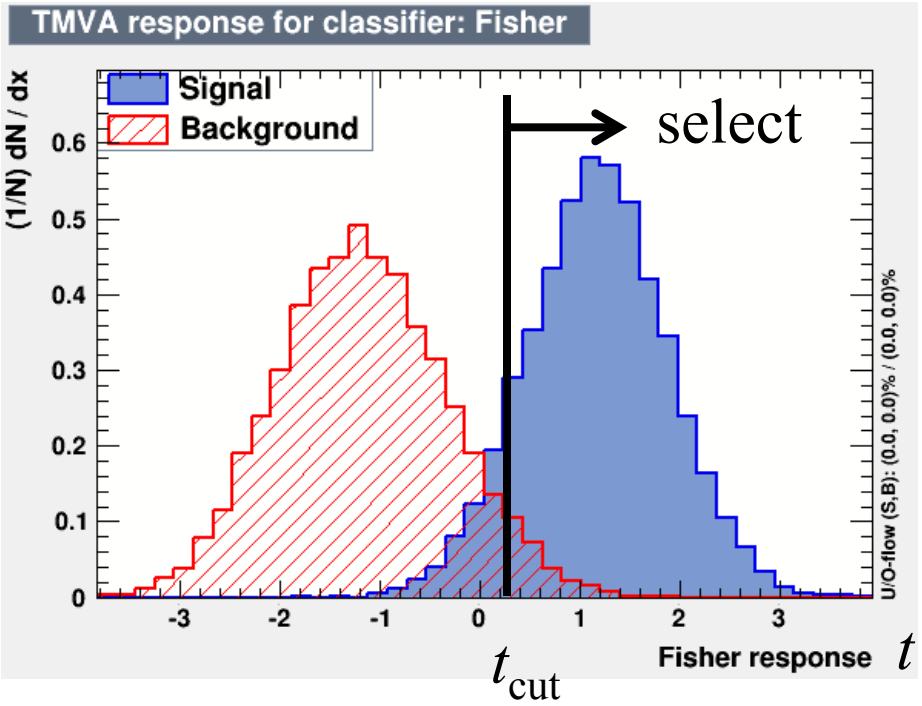
You need to make corresponding changes in **`analyzeData.cc`**

Try using these classifiers to find the maximum expected discovery significance.

You can also try modifying the architecture of the classifiers or use different classifiers (see TMVA manual at **`tmva.sourceforge.net`**).

Extension of TMVA Project

For the TMVA Project, you defined a test statistic t to separate between signal and background events.



You selected events with $t > t_{\text{cut}}$, calculated s and b, and estimated the expected discovery significance.

This is OK for a start, but does not use all of the available information from each event's value of the statistic t.

Binned analysis

Choose some number of bins (~20) for the histogram of the test statistic. In bin i, find the expected numbers of signal/background:

$$s_i = \sigma_{\rm s} LP(t \in \text{ bin } i|s) \qquad b_i = \sigma_{\rm b} LP(t \in \text{ bin } i|b)$$

Likelihood function for strength parameter μ with data n_1,..., n_N

$$L(\mu) = \prod_{i=1}^{N} \frac{(\mu s_i + b_i)^{n_i}}{n_i!} e^{-(\mu s_i + b_i)}$$

Statistic for test of $\mu = 0$:

$$q_0 = \begin{cases} -2\ln(L(0)/L(\hat{\mu})) & \hat{\mu} \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

(Asimov Paper: CCGV EPJC 71 (2011) 1554; arXiv:1007.1727)

Discovery sensitivity

First one should (if there is time) write a toy Monte Carlo program and generate data sets (n_1,..., n_N) following the $\mu = 0$ hypothesis, i.e., $n_i \sim$ Poisson(b_i) for the $i = 1,...,N$ bins of the histogram.

This can be done using the random number generator TRandom3 (see **`generateData.cc`** for an example and use **`ran->Poisson(bi)`**.)

From each data set (n_1,..., n_N), evaluate q_0 and enter into a histogram. Repeat for at least 10^7 simulated experiments.

You should see that the distribution of q_0 follows the asymptotic "half-chi-square" form.

Hints for computing q_0

You should first show that ln $L(\mu)$ can be written

$$\ln L(\mu) = \sum_{i=1}^{N} \left[n_i \ln(\mu s_i + b_i) - (\mu s_i + b_i) \right] + C$$

where C represents terms that do not depend on μ.

Therefore, to find the estimator $\hat{\mu}$, you need to solve

$$\frac{\partial \ln L}{\partial \mu} = \sum_{i=1}^{N} \left[\frac{n_i s_i}{\mu s_i + b_i} - s_i \right] = 0$$

To do this numerically, you can use the routine **`fitPar.cc`** (header file **`fitPar.h`**). Put fitPar.h in the subdirectory **`inc`** and **`fitPar.cc`** in **`analyze`**. Modify **`GNUmakefile`** to have

```
SOURCES       = analyzeData.cc fitPar.cc
INCLFILES     = Event.h fitPar.h
```

To plot the histogram and superimpose

To "book" (initialize) a histogram:

```
TH1D* h = new TH1D("h", "my histogram",
                   numBins, xMin, xMax);
```

To fill a value `x` into the histogram: `h->Fill(x);`

To display the histogram: `h->Draw();`

To superimpose ½ times the chi-square distribution curve on the histogram use:

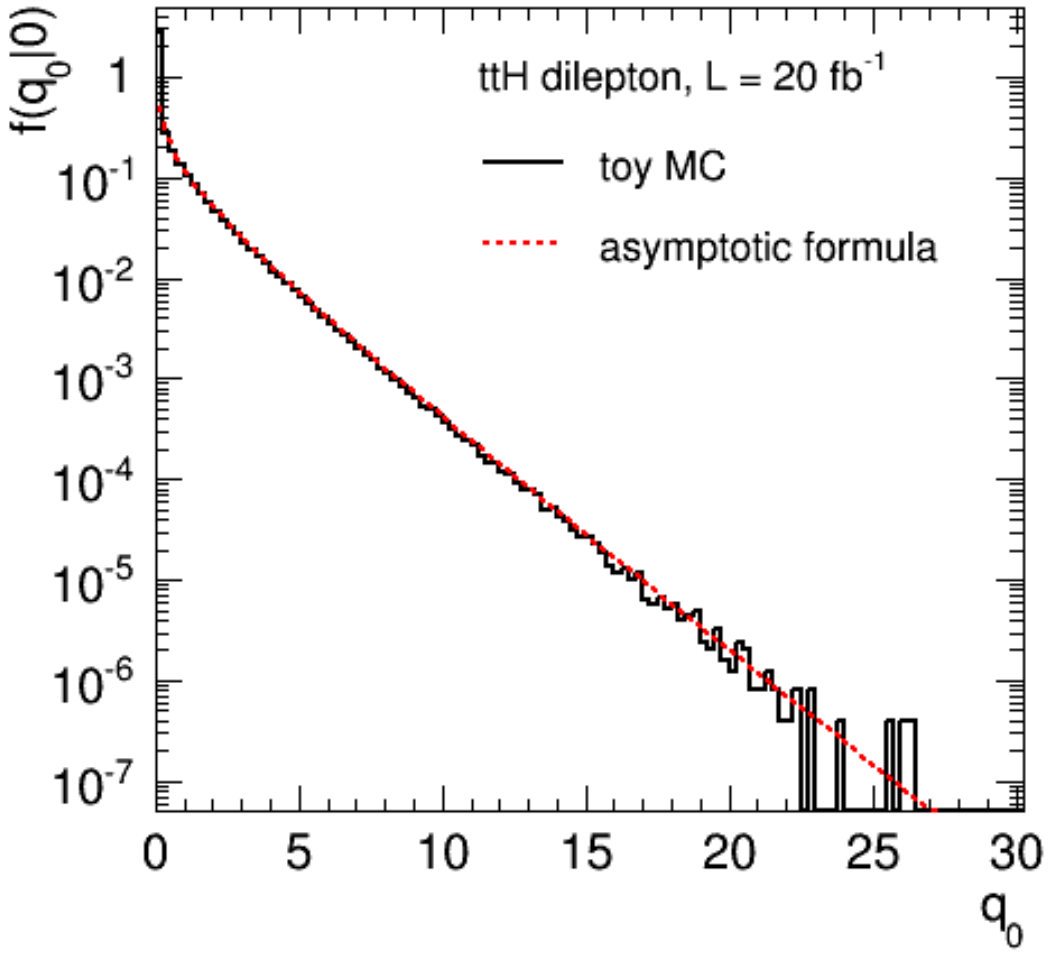
```
TF1* func = new TF1("func", ScaledChi2, 0., 50., 2);
func->SetParameter(0,1.0);       // degrees of freedom
func->SetParameter(1, 0.5);      // scale factor 0.5
func->Draw("same");
```

You can get the function `ScaledChi2.C`. from subdirectory `tools`.

Background-only distribution of q_0

For background-only ($\mu = 0$) toy MC, generate $n_i \sim \text{Poisson}(b_i)$.

Large-sample asymptotic formula is "half-chi-square".



Discovery sensitivity

Providing that the asymptotic approximation is valid, we can estimate the discovery significance (significance of test of $\mu = 0$)) from the formula

$$Z = \sqrt{q_0}$$

Median significance of test of background-only hypothesis under assumption of signal+background from "Asimov data set":

$$n_i \to s_i + b_i$$

You can use the Asimov data set to evaluate q_0 and use this with the formula $Z = \sqrt{q_0}$ to estimate the median discovery significance.

This should give a higher significance that what was obtained from the analysis based on a single cut.

The Higgs Machine Learning Challenge

An open competition (similar to our previous example) took pace from May to September 2014 using simulated data from the ATLAS experiment. Information can be found

opendata.cern.ch/collection/ATLAS-Higgs-Challenge-2014

800k simulated ATLAS events for signal ($H \to \tau\tau$) and background (ttbar and $Z \to \tau\tau$) now publicly available.

Each event characterized by 30 kinematic variables and a weight. Weights defined so that their sum gives expected number of events for 20 fb^{-1}.

Some code using TMVA is here (download and unpack as usual with tar –xvf):

www.pp.rhul.ac.uk/~cowan/higgsml/tmvaHiggsML.tar