derived from CDF/DOC/STATISTICS/PUBLIC/7904
Version 1.4
October 9, 2007

Building a More General χ^2

Tom Junk
Fermilab

Abstract

A χ^2 function is defined and discussed which is useful in comparing Poisson distributed data with a sum of models, each of which may have correlated and uncorrelated systematic uncertainties on its rate and shapes. Furthermore, each model may be estimated in each bin by a Poisson process, such as a Monte Carlo or a control sample of data, and these Poisson fluctuations are incorporated in the χ^2 function described here. A program for computing this χ^2 function is described.

1 Introduction

The results of a typical analysis on CDF consist of one or more histograms of data binned in particular variables which are chosen to separate events produced by a particular signal process from those produced by a set of background processes. Various signal hypotheses may be tested, parameterized by masses, cross sections, branching ratios, and possibly other parameters. The signals and backgrounds too may depend on a variety of parameters which are not of primary interest but which are needed for the measurement – examples include efficiencies, acceptances, integrated luminosity, and background production cross sections. The parameters which describe the signal and background processes which are not being measured or constrained by the analysis are called "nuisance parameters". Their values are needed in order to extract measurements of, or limits on, the parameters of interest, and uncertainty in

their values usually results in reduced sensitivity to the parameters of interest. The systematic errors on observables are parameterized in terms of these nuisance parameters.

The situation is that data must be compared to a model, and the predictions of the model are uncertain because of the lack of precise knowledge of the nuisance parameters. If two models are being compared, a null hypothesis and a test hypothesis, then the uncertainties in the nuisance parameters can reduce the ability of the analysis to reject one of the hypotheses – a search for a small signal in a counting experiment on top of a background with a systematic error that's of comparable size to the expected signal will not be a strong test of the signal model. Similarly, if a histogram is used and the shape of the background model's prediction is known imprecisely, the sensitivity may also suffer. In particular, if a background histogram is allowed to float in a fit in such a way as to match data in which a signal is actually present, then the sensitivity for detecting and measuring such a signal is reduced.

This situation arises very commonly when computing χ^2, even in the stages of an analysis before the limits or measurements are computed. Further complications arise from the fact that typically data are expected to consist of an incoherent sum of several different background (and signal) processes, each with its own sensitivity to nuisance parameters. Furthermore, background predictions as well as signal predictions often are estimated with Monte Carlos, scaled with luminosity, and thus have independent Poisson statistical variations in each bin[1]. Often processes are estimated using control samples in data, and these too are subject to Poisson fluctuations. Some models are not subject to Poisson fluctuations. Examples of this kind of model are a falling exponential for a lifetime distribution, or a Gaussian resolution function with some parameterization of the tails. Even these functions usually have uncertainties in their rates and shapes which may be correlated with those in the portions of the model which are estimated with Poisson-limited processes.

A more mundane issue arises in the case of small Poisson statistics. A typical χ^2 function does not handle very well cases in which the number of observed or predicted events is zero or small. Data are usually displayed in histograms with error bars of size $\sqrt{n}$ where n is the number of events observed in a bin, and these errors are often used in the computation of χ^2 (this is often called Neyman's χ^2, or χ^2_N [1]). For small n, these errors become less useful, and for bins with zero counts, an error of zero can be misleading. Typically bins with no observed counts in them are simply ignored in histogram fits using the χ^2_N function. Using the square root of the predicted value instead of the observed data is called Pearson's χ^2, or χ^2_P [1]. Neither of these χ^2 functions gives unbiased fits to histograms – Neyman's χ^2_N when applied to fitting a peak where few events are observed in the tails tends to underestimate the area of the peak, while Pearson's χ^2_P produces fits which tend to overestimate the area of the peak.

[1]Some Monte Carlos produce weighted events, where the weights of the Monte Carlo events are not all the same. Sometimes, the weights of Monte Carlo events are negative. This chisquared function does not (yet) take into account the distribution of sums of arbitrarily weighted Monte Carlo events.

A general χ^2 function has a great variety of uses. Once the number of degrees of freedom is known, one can estimate the goodness-of-fit of a model to a particular histogram using the value of χ^2. The goal of this note is to provide a χ^2 function that includes many different kinds of uncertainties that models are known to have. Section 4 below describes the issues relevant to computing the number of degrees of freedom. A χ^2 function can also be used to select one of two hypotheses, by taking a difference of chisquared values computed under the two hypotheses [2]. The difference of χ^2 values, say, between two measurements of the same quantity, may also be distributed according to the χ^2 distribution, for Gaussian-distributed measurements. This, and many other important features of pulls and their distributions, are described in [3].

2 Available Tools

An interesting proposal for an improved χ^2 function has been available for some time [1,4]. The approach is to notice that χ^2 is related to the likelihood function in the Gaussian limit (large Poisson statistics),

$$\chi^2 \approx -2\ln\left(\mathcal{L}/\mathcal{L}_0\right), \tag{1}$$

and to use this to extrapolate the meaning of χ^2 for situations where Gaussian statistics does not hold. Here $\mathcal{L}$ is just the product of the Poisson probabilities of observing the events in each bin given the summed predictions, and $\mathcal{L}_0$ is the Poisson probability of observing the same events if the prediction in each bin exactly matches the observation in that bin. For the simplest case of comparing a single data histogram against a single model with no systematic or statistical uncertainties,

$$\mathcal{L} = \prod_{i=1}^{I} \frac{t_i^{n_i} e^{-t_i}}{n_i!} \tag{2}$$

and

$$\mathcal{L}_0 = \prod_{i=1}^{I} \frac{n_i^{n_i} e^{-n_i}}{n_i!}, \tag{3}$$

where there are I bins, n_i data events are observed in bin i, and t_i events are predicted in bin i, following the notation of [4]. Using Equation 1, we may write the associated χ^2 function as

$$\chi^2 = 2\sum_{i=1}^{I} \left[(t_i - n_i) - n_i \ln(t_i/n_i)\right] \tag{4}$$

for this simple case.

Computing a difference of χ^2 values for two hypotheses H_1 and H_2 is equivalent to taking the logarithm of the likelihood ratio, a common test-statistic

$$\Delta\chi^2 = \chi_1^2 - \chi^2 = -2ln\left(\mathcal{L}_1/\mathcal{L}_2\right). \tag{5}$$

This relationship between $\Delta\chi^2$ and the likelihood ratio holds even if the hypotheses have different numbers of degrees of freedom (L_0 does not depend on the hypothesis).

In the case that the t_i are subject to systematic uncertainties, the chisquared function is modified to include their effects. In [4], the effects of relative systematic uncertainties were considered to be multiplicative for "backgrounds" and a division is performed for "signals". In this work, no distinciton is made between a signal and a background, and relative systematic uncertainties are treated multiplicatively. Using a notation close to that of [4],

$$t'_i = t_i \prod_{k=1}^{K} (1 + f_k S_k) \,, \tag{6}$$

where t'_i is a systematically varied prediction in bin i, and there are K nuisance parameters S_k. The f_k's are the relative systematic unceratainties on the normalization of the model histogram due to the nuisance parameters separately. In the program described below, the nuisance parameters S_k have gaussian constraints applied to them, unless the user asks them to be unconstrained. The scale of the systematic uncertainties are determined by the f_k, so the Gaussian constraints on the S_k are all centered on zero with unit width. The S_k are limited in fits so that the t'_i do not go negative in any bin.

In [4], the f_k's are separate for each bin i. This strategy allows the shape of the t' histogram to vary because a change in a nuiscance parameter affects each bin in a correlated but different way. The weakness in this approach is that the effects are multiplicative and symmetric. A shape error in a histogram with jet energies or a reconstructed mass, for instance, may change with the jet energy scale by shifting the scale up or down on the abscissa. A bin which may have zero or very few predicted events in it in the central value histogram (say, it is beyond a kinematic edge), may acquire a sizeable contribution when the jet energy scale is shifted up, for example. Treating the corresponding f_{ik} in this bin as a symmetric systematic uncertainty means that shifting the jet energy scale downwards may drive this bin's contents negative. The multiplicative procedure fails when the central value bin contents are predicted to be zero. Besides, it is awkward to compute shape errors as correlated multiplicative changes on bin contents. Typcally, an analyzer will make separate histograms showing the shape of the model histogram under variations of an unknown nuisance parameter, and it would be best to input these for calculation of a chisquare.

Formulas for this χ^2 are given in [4] for the case that one of the model predictions suffers from Poisson statistical fluctuations in each bin separately, where a maximization of the likelihood function is done over the scaling factor between the Poisson process used to make the prediction and the predicted value in each bin, and the true value in each bin (which is only imperfectly known due to statistical fluctuations in the estimate). The calculation simplifies due to the fact that this optimization of the Poisson means can be performed in each bin separately. For the case of one Poisson process producing the prediction, the maximization of the likelihood

can be done analytically for the Poisson means, and a numerical technique is needed for the maximization with respect to the nuisance parameters. If more than one contributing process suffers from Poisson uncertainty, then one must solve a system of coupled quadratic equations which has no analytic solution. Practical techniques exist, however [5], and the joint fitting of a sum of models to the data is accomplished by the `HBOOK` routine `HMCMLL`, which exists in `root` as the class `TFractionFitter`. Reference [4] adds in the possibility of including non-Poisson contributions too.

`TFractionFitter` does not seem to have the ability to input external constraints, let alone correlated external constraints. A common requirement of experimenters is to compute a χ^2 for data as compared to a sum of models, where each model has its own shape and systematic uncertainty on its normalization, which requires the input of constraints.

Incorporating model histogram shape uncertainty in the abovementioned tools is either awkward or even incorrect for some kinds of shape uncertainty. One may be able to include shape variation clumsily in the above by creating a shape difference histogram and fitting its fraction, but this technique lacks generality because not all histogram shape uncertainties can be parameterized in this manner. An example is the jet energy scale uncertainty on a reconstructed invariant mass distribution. Ideally one would like to transform the x-axis, or slide a distribution left and right along the axis. If the background or signal has a peak in it, or even an edge, then forming difference histograms can create unphysical negative peaks when the parameter range is scanned. If a central-value-estimate background shape histogram has a peak in it, and a systematically varied background histogram has a peak also but which is displaced, then an interpolated background histogram shouldn't have two peaks of lesser strength, but rather one peak situated midway between the two original estimates. The same applies for many kinds of signal.

3 Proposed χ^2 Function

The proposed χ^2 function follows Equations 8 and 9 of [4], with some minor modifications. The "signal" and "background" distinctions of [4] are not needed – the goal is to compare the data with a sum of models, and a test using this χ^2 can be to compare the χ^2 values of a fit including a signal and a fit omitting a signal, for example. There may even be several signal contributions present simultaneously. The "signal" and "background" distinctions of [4] are in fact used to distinguish between estimations that have Poisson discreteness in each bin and those that do not, and since we'd like to generalize this function to have arbitrarily many contributions of both kinds, the labels "signal" and "background" aren't important to the χ^2 calculator.

In the same vein, systematic uncertainties on the contributions are proposed to be handled

in a symmetric manner for all contributions – they are taken to be relative multiplicative uncertainties, and no division is done. These two prescriptions, multiplicative relative errors and division-based errors are the same to first order in the uncertainty.

Another small change with respect to [4] is that the uncertainties in the scaling factors on the Poisson contributions to the model are assumed to vary with the nuisance paramters S_k (in [4] these normalizations were their own, independent nuisance parameters). This change allows all uncertain predictions to have correlated uncertainties.

The proposed χ^2 function is

$$\begin{aligned}
\chi^2 &= 2\sum_{i=1}^{I}\Bigg[\left(\sum_{l=1}^{L} t_{li}\prod_{k=1}^{K}\left(1+f_{lk}^{t}S_k\right)+\sum_{j=1}^{J}\rho_{ji}-n_i\right) \\
&\quad -n_i\ln\left(\frac{\sum_{l=1}^{L} t_{li}\prod_{k=1}^{K}\left(1+f_{lk}^{t}S_k\right)+\sum_{j=1}^{J}\rho_{ji}}{n_i}\right) \\
&\quad +\sum_{j=1}^{J}\left(\left(\frac{\rho_{ji}}{F_j\prod_{k=1}^{K}\left(1+f_{jk}^{F}S_k\right)}-b_{ji}\right)-b_{ji}\ln\left(\frac{\rho_{ji}}{F_j\prod_{k=1}^{K}\left(1+f_{jk}^{F}S_k\right)b_{ji}}\right)\right)\Bigg] \\
&\quad +\sum_{k=1}^{K}S_k^2
\end{aligned} \tag{7}$$

The meanings of the symbols used in Equation 7 is given below. There are I bins in the histogram, and the index i runs over the bin number. The number of events observed in the data in bin i is given by n_i. There are J model prediction components which are subject to Poisson statistics in each bin, and the index j runs over these. There are L model prediction components which are not subject to Poisson statistics in each bin, and the index l runs over these. There are K independent sources of systematic uncertainty, which are parameterized by the nuisance parameters S_k, where the index k runs over the nuisance parameter index. The model prediction t_{li} is the l^{th} non-Poisson model component's prediction in bin i. For model predictions with Poisson errors in each bin, b_{ji} is the number of counts in bin i from the subsidiary Poisson measurement which determines the model contribution. These event counts need to be scaled to compute the expected contribution from this model to bin i, and the central value of this prediction is $F_j b_{ji}$ for Poisson source j. The unknown true value ρ_{ji} of the rate of Poisson component j in bin i is solved for by minimizing χ^2 with respect to all of these.

The nuisance parameters S_k are assumed to have Gaussian constraints centered on zero with unit width, which is included as part of the χ^2 function. The systematic uncertainties on t_{li} and F_j are parameterized by f_{lk}^{t} for the non-Poisson components and by f_{jk}^{F} for the

Poisson components. These f's are the fractional multiplicative uncertainties on the overall normalization of each component.

Shape uncertainties are parameterized by specifying alternative shape histograms for the t_{li} and b_{ji}, separately for each nuisance parameter. The central-value and systematically varied histogram shapes can be interpolated using a technique common at LEP [6]. Histogram extrapolation is not allowed [2], and so the nuisance parameters are constrained to vary within the range given by the shape uncertainty estimation if provided for a particular model for a particular nuisance parameter. Separate shapes need to be supplied for positive changes of the nuisance parameter and for negative changes so that the central value histogram may be interpolated in both directions. For histograms with shape errors arising from several nuisance parameters, the central value histogram is interpolated using the first varied shape, and the result of the interpolation is the starting point for an interpolation to the second varied shape, and so on. The order in which the interpolations is done in the software described below is the order in which the nuisance parameters were specified in the template histograms.

Histograms estimated from non-Poisson models are interpolated, and the normalizations of the interpolated histograms are also interpolated. One may explicitly separate shape and normalization errors by supplying shape-error histograms with the same normalization as the central-value histogram, or one may also supply shape histograms with normalizations that represent the changes in normalization that are correlated with the changes in the shapes. The user is in charge of not double-counting normalization uncertainties. The shape uncertainties are parameterized by the same set of nuisance parameters as all other uncertainties and so are correlated with all other uncertainties. An interesting feature of interpolated Poisson histograms is that if two histograms with Poisson-distributed bin contents in each bin are interpolated using [6], and the total number of entries is the same in both histograms, then the resulting interpolated histogram also has Poisson-distribted bins with the same number of entries. The underflow and overflow bins of a histogram are not considered as part of the normalization nor are they interpolated by the program supplied here. The underflow and overflow bins do not contribute to the chsquared calculation.

While the uncertainties on histogram normalizations and shapes can be correlated with each other, the nuisance parameters S_k are treated as uncorrelated. In general, any correlated errors can be broken down into sums of uncorrelated components, and indeed estimates of correlations in uncertainties are often determined by estimating the magnitudes and signs of the 100% correlated components.

[2] If histogram extrapolation is required, users of the supplied routines should do the extrpolation with the supplied routine and check the results by hand in order to see if they make sense. Often they will not.

The bin-by-bin Poisson rates ρ_{ji} may be solved for by setting

$$\frac{\partial \chi^2}{\partial \rho_{ji}} = 0, \ \forall j, i \tag{8}$$

and solving the J coupled quadratic equations which follow (they are similar to Equation 11 in [4]). These are given by

$$1 - \frac{n_i}{\sum_{l=1}^{L} \prod_{k=1}^{K} (1 + f_{lk}^t S_k) + \sum_{m=1}^{J} \rho_{mi}} + \frac{1}{F_j \prod_{k=1}^{K} \left(1 + f_{jk}^F S_k\right)} - \frac{b_{ji}}{\rho_{ji}} = 0, \ \forall j, i \tag{9}$$

These systems may be solved separately for each bin i, independently of the others. This system may be solved iteratively by setting the $\rho_{ji} = F_j b_{ji}$ on the initial step, and then solving each quadratic equation using the previous step's estimate of ρ_{ji}'s, always choosing the positive root, as it has been noted that only one physical solution exists. The iterative procedure usually settles down to a precision of one part in 10^8 after ten iterations or so. A numerical procedure using Newton's method is used in `HMCMLL` [5] for solving the same problem.

The χ^2 function is then minimized over all values of the S_k, as is recommended in [4] A requirement which arises when considering allowable ranges of systematic variation is that no bin of any prediction may go negative when the nuisance parameters are varied. Typically one truncates the Gaussians (e.g., in priors if doing a Bayesian analysis) or one chooses another function than a Gaussian. Limits are set on the `MINUIT` parameters S_k which keep the predictions non-negative. The values of these limits are computed from the f_{lk}^t and f_{jk}^F values.

4 The Number of Degrees of Freedom

`TFractionFitter` defines the number of degrees of freedom to be the number of points used in the fit minus the number of templates, which works (almost) because there are no external constraints on the normalization of each template in the fit. Adding in a count of the nuisance parameters is needed because they contribute to the χ^2 from their Gaussian constraints. A naive estimate of the number of degrees of freedom therefore is

$$n_{\mathrm{DOF}} = I - J + K \tag{10}$$

using the notation defined for Equation 7.

The chisquare distribution for a given number of degrees of freedom n_{DOF} is readily computed using the `CERNLIB` routine `prob`, which is supplied in `root` as a part of `TMath`. Unfortunately, the standard `prob` function assumes that the measurements included in the calculation of χ^2 are drawn from independent Gaussian parent distributions, where the variances of the Gaussians include both statistical and systematic fluctuations. In counting experiments, it is

often the case that nothing has a Gaussian distribution, particularly the numbers of events in bins with few expected entries. Nonetheless, it is common to use the `prob` function along with Neyman's χ^2 to estimate goodness-of-fit.

Because the likelihood construction of χ^2 behaves like Neyman's and Pearson's χ^2 for large event counts, it is just as appropriate to use `prob` for the χ^2 defined here. But there are some issues to keep in mind which arise due to the extensions introduced here and in [4].

A question comes up in which bins contribute to the number of degrees of freedom. If a histogram has more bins than is required – that is, the predictions and observations in some bins of a histogram are identically zero, then those bins ought not to contribute to the number of degrees of freedom.

For bins with a small number of expected events, observing zero events contributes a small amount to the χ^2 proposed here. Observing one event contributes a large amount to χ^2, of order the reciprocal of the prediction (the presence of nuisance parameters or Poisson predictions modifies this). This is in contrast to Neyman's χ^2 in which the uncertainty on one event observed is unity, and the contribution of this bin to Neyman's χ^2 is of order unity. Using a χ^2 based on the likelihood function will always have this feature. One consequence of this feature is that events on the tails of histograms will be given very large weights in fits to these histograms, which requires constructing fully realistic models of histogram tails. Often fits done for technical reasons (calibrations, resolution estimates) use functions which do not have complete models of the tails, and using a more "forgiving" χ^2 function can sweep issues of mismodeled tails under the rug, which is sometimes desired.

In analogy to a condensed-matter phenomenon, degrees of freedom "freeze out" when the number of events in a bin gets so small that discreteness plays an important role, with a limiting case of the distribution of the numbers of events being a delta-function at zero. In fact, the interpretation of any χ^2 function of Poisson data using the χ^2 distribution is only approximate because the possible outcomes of the experiment are discrete and countable, and some values of χ^2 are impossible to obtain because there is no number of events in any bin which gives exactly that value. The warning is that to use this χ^2 or any other as the input to `prob` requires caution and thought. A standard procedure if it is desired to have a χ^2 function which approximately follows the chi-squared distribution is to rebin the data so that no bin is empty or has few events in it.

Another issue with the number of degrees of freedom is that separate nuisance parameters may have the same effect on the normalization of a histogram, and thus the chisquared minimization procedure will adjust both of the nuisance parameters in the same way. In a simple situation where a data histogram is fit to a single non-Poisson model with two nuisance parameters with the same f_{lk}^t values, then the values of the two corresponding S_k's will always be equal to each other at the χ^2 minimum, and will thus not be independent contributions to

the total χ^2.

If the uncertainty on a model histogram's normalization resulting from a nuisance parameter is very large, and there aren't other constraints on the same nuisance parameter, then the Gaussian constraint on the nuisance parameter contributes very little to the chisquared function and shouldn't be counted as a full degree of freedom.

4.1 Proposed Number of Degrees of Freedom

As is the case in nearly all statistical computations with many contributing effects, the correct advice is nearly always to run a large sample of pseudoexperiments and see what the distribution of the outcomes is. This advice can easily be applied here – pseudoexperiments can be run simulating possible data outcomes, each pseudoexperiment chosen from a sample with a correctly fluctuated model prediction using all of the correlated model uncertainties on the rates, shapes, and Poisson ucnertainties in each bin. These pseudoexperiments are handled by a program [8] which includes this chisquared calculator as part of a limit calculator. The resulting distribution of χ^2 can then be fit to the χ^2 probability density function, with the number of degrees of freedom and the normalization left floating. For convenience, the formula is supplied here [9]:

$$f(z;n) = A\frac{z^{n/2-1}e^{-z/2}}{2^{n/2}\Gamma(n/2)}$$

where z is the random χ^2 variable, and n is the number of degrees of freedom. An example fit is

5 Available Software

The χ^2 function proposed here is calculated by a routine which runs in `root`, taking histograms as input. All of the histogram declarations in the software use the base class `TH1` so that any of the derived histogram classes can be used on input. The code is available at phystat.org.

One uses this package by creating a member of the class `csm`, by supplying a histogram of data (assumed to be Poisson distributed in each bin with an unknown mean – that is, histograms with one entry per event), and model templates with systematic error information. Then two methods, `csm::chisquare` and `csm::ndof` can compute the chisquare and number of degrees of freedom. One can also access the best fit values of the nuisance parameters, as well as a histogram which contains the model which fits the data best after minimizing over the nuisance parameters S_k and each bin's ρ values. Below are descriptions of the methods of class `csm`.

```
void csm::set_htofit(TH1 *h, char *channame)

  This method identifies a histogram of Poisson-distributed
  data.  The histogram is cloned when the method is called,
  and the clone is deleted when the csm destructor is called
  or when set_htofit is called again.
  The overflow and underflow bins are ignored in the chisquare
  calculation.  The histogram may be 1D or 2D.  Several channels,
  possibly sharing nuisance parameters, may be fit jointly, and the
  data histograms and model templates need to specify a channel
  name in order to identify them.

void csm::set_modeltofit(csm_model *model)

  Defines what to fit to the data.  The model contains predictions for each channel.
Strategy -- make a new instance of csm_model, and call csm_model::add_template()
described below to define the templates.

void csm_model::add_template(TH1 *template_hist,
                        Double_t sf,
                        Int_t nnp,
                        char* npname[],
                        Double_t *nps_low,
                        Double_t *nps_high,
                        TH1 *lowshape[],
                        Double_t *lowsigma,
                        TH1 *highshape[],
                        Double_t *highsigma,
                        Int_t pflag,
                        Int_t scaleflag,
                        char *channame)

  Adds a model component to the sum of models to be compared with the data,
  and parameterizations of the errors on this model component.
  All of the template histograms, shape histograms, and errors are copied into
  dynamically allocated storage within the csm class, and thus the originals
  do not need to persist after the add_template method is called.

template_hist   may be a Poisson or non-Poisson histogram.  If this
                histogram comes from a Poisson subsidiary process (like MC
```

```
                or a subsidiary measurement), be sure that its normalization
                corresponds to the entries made in it.  (That is, let sf
                do all of the scaling).

sf              scale factor to multiply template by to compare w/ data
                (e.g., (data_lum/MC_lum) for a MC Poisson histogram

nnp             number of nuisance parameters -- each is constrained to
                zero by a Gaussian of unit width.  Each nuisance parameter
                corresponds to one entry in the nps_low, nps_high,
                lowshape, highshape, lowsigma, highsigma arrays below.

npname          nuisance parameter names.  Correlations between systematic
                errors across templates are handled by labeling the
                separate nuisance nuisance parameters by name.

nps_low         These are the f's, fractional uncertainties on the
nps_high        normalization (sf) due to each nuisance parameter.
                Fractional uncertainties may be asymmetric -- when a
                nuisance parameter is negative, it may have a different
                effect on sf than when it is positive.  Typically
                nps_low and nps_high will have opposite signs, as
                opposite variations of a nuisance parameter will have
                opposite effects on sf -- these signs need to be
                input as opposite in this case.  But sometimes you get
                the same sign of variation, in which case nps_low and
                nps_high may have the same sign.  The relative sign
                is important across templates too.  If one template's
                normalization goes up while another goes down when
                a nuisance parameter is fluctuated (anticorrelation),
                this is reflected in the relative signs of these f's.

lowshape        Histogram corresponding to a variation of a nuisance
                parameter in the negative direction.  Used to parameterize
                shape uncertainty.  The normalization of this histogram
                is unimportant.  Set this pointer to zero if you do
                not have a shape variation for this template for this
                nuisance parameter.  If the template histogram is
                Poisson, then lowshape and highshape should have the
                same number of entries as the template histogram in order
```

to make the interpolated histogram follow Poisson
statistics too.

lowsigma — How many sigma of variation the lowshape corresponds to. (example: you may make a histogram of a variable that corresponds to changing the jet energy scale by two sigma. set this number to 2. The sign of this 2 doesn't matter). Note: histogram extrapolation is not allowed -- the nuisance parameter this shape uncertainty corresponds to is constrained to lie between -|lowsigma| and +|highsigma|.

highshape — Same as lowshape, but for positive variations of the corresponding nuisance parameter. Set it to zero if you don't have this uncertainty evaluated.

highsigma — See the description of lowsigma.

pflag — Set to 1 if the template histogram is Poisson distributed and set to 0 otherwise.

scaleflag — used by mclimit_csm to determine which templates to scale as signals in search of the right limits, and which to leave fixed (backgrounds). Does not affect chisquared calculation here.

channame — Identifies which channel this template corresponds to. Must be the same as the corresponding channame for the data histogram in set_datahist.

```
void csm_model::add_npcons(Int_t listsize, char **nplist,
   char *npname, Double_t (*f)(Double_t*))
```

Defines a function to compute one nuisance parameter in terms of a list of others. The nuisance parameter referred to by npname is computed as a functin of listsize other parameters, whose names are given in nplist. The function for computing them is f, which takes listsize Double_t arguments, and returns the value of nuisance parameter npname, given the values of the others. This feature is better documented in the mclimit_csm documentation.

```
void csm_model::print()
```

Useful printout of the details of a model.

```
void csm_model::plotwithdata(char *channame, TH1 *datahist)

  Makes a stacked plot comparing the data to the model prediction for channame.

void csm_model::candcheck(char *channame, TH1 *datahist)

  Prints out summaries of candidates and the s/b's in the bins in which they are found.

csm_model *csm_model::Clone()

  Makes an exact copy of a model.

Double_t csm::chisquared()

  The chisquared calculation described here.

Int_t csm::ndof()

  Very naive calculation of the number of degrees of freedom.  Recommended -- use
  mclimit_csm to run pseudoexperiments and histogram the resulting chisquareds, and
  fit it to a chisquared distribution.

TH1F* csm::getbestmodel()

  Returns a pointer to a histogram which is the best comparison with the data
  (minimizes chisquared).  This histogram is deleted when the class destructor
  is called, so clone it if you need to keep it.  Be sure to call the
  chisquared method to do the minimization calculation before accessing the
  best model.  The internal bestmodel histogram is a TH1F histogram with the
  same number of bins and lower edge and upper edge as the histogram passed
  in with set_htofit.  This histogram isn't used in the calculation of
  chisquared but is supplied as a diagnostic.

Calls to add_template may refer to nuisance parameters multiple times
and so an internal list is made in csm of all the parameters,
identified by name.   These methods access that list.

Int_t csm::getnparams()
```

```
  Returns the number of independent nuisance parameters given in the
  add_template calls.

Double_t csm::getparam(Int_t iparam)

  Access to value of a nuisance parameters after the minimization.  Be sure
  to call the chisquare method before calling this method.

Double_t csm::getperror(Int_t iparam)

  MINUIT's uncertainty.  Don't believe it.  Because of discrete behavior of
  interpolated Poisson histograms, a nuisance parameter can vary by a tiny
  amount and an event can flip from one bin in a model histogram to another,
  changing the chisquared by a discrete amount.  The chisquared function
  therefore has discontinuities in it and MINUIT may get a strange derivative
  if it chooses too small a finite difference.

char* csm::getpname(Int_t iparam)

  The corresponding parameter's name.

Double_t getcov(Int_t iparam, Int_t jparam)

  Get an entry out of the covariance matrix.

void csm::setminuitmaxcalls(Int_t maxminuitcalls)

  Maximum number of function calls MINUIT is allowed to do per minimization

Int_t csm::getminuitmaxcalls()

void csm::setminosflag(bool minosflag)
  true: call MINOS. Best to have printing set too if
  you're going to run MINOS.  False:  Do not call MINOS (default)

bool csm::getminosflag()

void csm::setprintflag(bool printflag)
  true:  let MINUIT print stuff out;  FALSE -- turn off MINUIT printing (default)
```

```
bool csm::getprintflag();

csm::~csm()

  Be sure to delete your instance of csm before setting up another chisquared
  calculation.  There is not facility to edit an existing list of template
  histograms -- in order to change a chisquared calculation, the csm instance
  should be deleted and the setup repeated.  One exception to this -- you can
  call set_htofit again to find the chisquared of a new data histogram without
  rebuilding the list of model templates.  This is designed for convenience
  when running pseudoexperiments.
```

6 An Example

A code snippet to run the pseudoexperiments is supplied below. The full `mclimit_csm.C` package is needed to generate the pseudoexperiments with correctly-varied correlated systematic uncertainty and to run the `csm` χ^2 calculator on each one. For the code snippet below, the models for the null and test hypotheses are the same (no distinction is needed for computation of a single χ^2, although `mclimit_csm` is built to handle the problem of testing two hypotheses against each other). The code snippet below shows how to run the pseudoexperiments, get a histogram of the χ^2 values and fit the result. For further documentation please see [8].

```
  // the null and test hypotheses have already been defined

csm* mycsm = (csm*) new csm();

mycsm->set_modeltofit(nullhyp);
        mycsm->set_htofit(histo[4][0],"tchan");  // the data histogram is in histo[4][0]

        double chisqresult;

chisqresult = mycsm->chisquared();

cout << "Chi squared: " << chisqresult << endl;

csm_model* bestmodel = mycsm->getbestmodel();
```

```
        mycanvas = (TCanvas *) new TCanvas("Canvas1","Canvas1");
        mycanvas->Divide(2,1);
mycanvas->cd(1);
        bestmodel->plotwithdata("tchan",histo[4][0]);


        TF1 *fcsd = new TF1("fcsd",
"[0]*pow(x,0.5*([1]-2))*exp(-x/2.0)/(pow(2,[1]/2.0)*TMath::Gamma([1]/2.0))",0,60);
        TH1F *nnhist = (TH1F*) new TH1F("nnhist",
"Null Pseudoexperiments -- Null Hyp Fit",100,0,60);
        TH1F *nnhist1 = (TH1F*) new TH1F("nnhist1","other Fit1",100,0,60);
        TH1F *nnhist2 = (TH1F*) new TH1F("nnhist2","other Fit2",100,0,60);
        TH1F *nnhist3 = (TH1F*) new TH1F("nnhist3","other Fit3",100,0,60);

        mclimit_csm* mymclimit = (mclimit_csm*) new mclimit_csm();
        mymclimit->set_null_hypothesis(testhyp);
        mymclimit->set_test_hypothesis(testhyp);
        mymclimit->set_null_hypothesis_pe(nullhyp_pe);
        mymclimit->set_test_hypothesis_pe(testhyp_pe);
        mymclimit->set_datahist(histo[4][0],"tchan");
        mymclimit->set_chisquarehistos(nnhist,nnhist1,nnhist2,nnhist3);
        mymclimit->set_npe(250);
        mymclimit->run_pseudoexperiments();

        mycanvas->cd(2);
        nnhist->GetXaxis()->SetTitle("#chi^{2}");
        nnhist->GetYaxis()->SetTitle("Pseudoexperiments");
        fcsd->SetParameter(0,nnhist->GetEntries());
        fcsd->SetParameter(1,nnhist->GetMean(1));

        nnhist->Fit("fcsd");
```

References

[1] S. Baker and R. D. Cousins, 'Clarification of the Use of Chi-Square and Likelihood Functions in Fits to Histograms' Nucl. Instrum. Meth. **A221** 437 (1984).

[2] L. Lyons, 'Selecting Between Two Hypotheses', OUNP-99-12 (1999).

[3] L. Demortier and L. Lyons, CDF Note 5776 "Everything you always wanted to know about pulls", (2002).

[4] T. Devlin, CDF note 3126, 'Correlations from Systematic Corrections to Poisson-Distributed data in Log-Likelihood Functions' (1999).

[5] R. Barlow and C. Beeston, 'Fitting Using Finite Monte Carlo Samples', Comput. Phys. Commun. **77** 219-228 (1993).
See also the `HBOOK` Reference Manual, CERN Long Writeup Y250 (1995).

[6] A. L. Read, 'Linear interpolation of histograms', Nucl. Instrum. Meth. **A425** 357 (1999).

[7] F. James, 'Minuit Reference Manual, Version 94.1', CERN Program Library Long Writeup D506 (1994). See also the `root` documentation for the class `TMinuit` and `http:\\root.cern.ch`

[8] T. Junk, 'Sensitivity, Exclusion and Discovery with Small Signals, Large Backgrounds, and Large Systematics', CDF note 8128.

[9] See for example, the Review of Particle Physics, Phys. Lett. B **592**, table 31.1, on p. 277 (2004).