General overview of the chi-square approach for fitting data

This document is a general overview of the chi-square approach for fitting data and hypothesis testing, including the F-test and ANOVA. Before getting into the gory details I'd like to provide an overall summary of how to fit some of the most complex data sets. The strategy of our approach is relatively simple. Suppose the observed data is given by O­­I for the ith datum. Suppose one somehow comes up with a model that seems to do a good job of accounting for the data. We will call E­I, the expected value for that datum. Ei will depend on a variety of conditions such as age, contrast of stimulus and things like whether you were standing on one or two legs while doing the experiment. The hardest part of any analysis is to come up with good models. Ei. With all that input data in place we can begin the analysis.

Step 1) Calculate the standard deviation, SD, of the data. There are a variety of ways to calculate SD. There are three broad approaches. a) The standard deviation may be well known based on the random errors in the data gathering instrument or based on knowledge of the population from which the data sample was obtained. b) In an analysis of variance (ANOVA) type of data one simply calculates a number of means then gets the variance and the SD is the square root of the variance. c) For binomial data life can be easy in that the variance is often equal to E­­I (a claim that sounds surprising at first). The process for calculating SD and E

Step 2) Use Matlab's lsqnonlin (or equivalent program in other languages) to estimate Ei the "expected" best fit to the data using a number of parameters, paramk. What lsqnonlin does is minimize the sum of squares deviation between O­i and Ei where the deviation is: devi = (Oi - Ei) / SDi. SDi is the standard deviation of either the ith datum or the ith expected value (for binomial statistics the latter is more accurate). Lsqnonlin finds Ei by minimizing chi square: chisq =sum(devi2). If the data are correlated, as often happens in neuroscience (e.g. fMRI, EEG/MEG, etc) then 1/SD­i2become the inverse of the covariance matrix, as will be discussed in the advanced topics section.

3) One challenge for Step 2 is to figure out the SDi, the standard deviation. Sometimes it is known based on the accuracy of the measuring equipment. Sometimes it is known from binomial statistics (SD=sqrt(p\*(1-p)/N)). Or more commonly it can be estimated from the data itself. For example, if there are enough repetitions for each condition then SDi = sqrt(sum((Oi - Mi)2)/(Ni -1)) where Mi and Ni are the mean and number of trials for that condition. Very often one assumes that SD is the same across all the data, so SD = sqrt(sum((Oi- Ei)2)/(Ndata -N­­params)) is estimated from the best possible model. We may need to use a very simple nonlinear transformation (like a log) to get the SD's of the various data to be about the same. If the SD had to come from the data, then rather than using a chi square test one needs to use an F test and we would write the equation in Step 2 as: chisqF =sum(devi2), inventing a new term, chisqF on left side. ChisqF is calculated the same way as chisq, but with knowledge that the SD was calculated from the sample.

4) Step 2 mentioned using lsqnonlin to determine Ei. Lsqnonlin outputs three items: a) the best fitting params (my shorthand for parameters), b) chi square, and c) the jacobian, j, from which one directly calculates the standard error, SEk, of each param and the correlations among all the params. If the params are overly correlated then it would be good to rewrite Ei in terms of a recombination of the params to reduce the correlations as much as reasonable.

Throughout this writeup I'll be calling the error bar on the data, SDi, and the error bar on the params, SEk to avoid confusion. Recall that SE is the SD of replications of the same experiment so there is some flexibility in what to call the error bar and using different letters will help separate the params and the data errors.

5) One can test out a variety of different models, E­i on different iterations with separate calls to lsqnonlin. Consider for example comparing two models with 5 and 7 parameters. As long as the 7 parameter fit uses the same 5 parameters, then one can take the difference between chisq or chisqF of the two iterations and do a chisq or F test with 2 degrees of freedom (DF). For the case of chisqF one also needs to specify the DF that was used for calculating the SD. In that way one can determine whether the 2 extra parameters were significantly different from zero.

6) An important aspect is providing information for how to plot the data in an informative manner.

The rest of this document is devoted to fleshing out the above summary. It will be applied to several simple datasets. It is very useful to play with the associated Matlab code for the Wikipedia. Please let me know of any errors or anything that is confusing. Stan (3/29/2015)

**How to fit data and introduction to chi-square (2) and F test.**

**Step 0. Acquire data.**  I'll use Oi as the variable for observed data.

DataFormat: The data will be strung out as a long row or column vector with other vectors indicating the conditions:

Oi = [6 8 4 5 3 4 8 12 9 11 6 8 13 9 11 8 7 12] %The Wikipedia data

cond1 = [1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3] %row number(subject ID?)

cond2 = [1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6] %column number(training day?)

The data, O, together with the condition 1 is precisely the Wiki F-test data. We added condition 2 to demonstrate how one can do linear regression on that condition.

Do the standard ANOVA "by hand.

**Step 1. Get the standard deviation of the WikiData by hand.**

I'll do it in a slightly different way than the Wikipedia article. The big problem with the Wikipedia method is that it isn't motivated.

1a) Start by with a three parameter fit that are the three means of the three conditions.

Means = [(6+8+4+5+3+4)/6 (8+12+9+11+6+8)/6 (13+9+11+8+7+12)] = [5 9 10]

1b) Now get the 18 deviations from the three means. For example Deviation3(1) = 6 - 5 = 1

Deviations3=[1 3 -1 0 -2 -1 -1 3 0 2 -3 -1 3 -1 1 -2 -3 2 ]

The 3 indicates that 3 means are used as the model of the data.

1c) SS3= sum(Deviations32) = 68 (one of the numbers in the Wikipedia method)

1d) DegFree3 = #data - #params = 18 - 3 = 15.

1e) SD= sqrt(SS3/DegFree3) = sqrt(68/(18-3)) = 2.129 (also in Wikipedia)

The SD is needed for calculating the chisqF. (the F in chisqF is because SD was based on the sample).

1f) Calculate chisqF3 = sum((Oi-Ei)/SD)2 =

= sum(SS3/SD2) = DegFree3 = 15 (That's why chisqF can't do goodness of fit)

2) Now let's do the same for the one parameter fit that is the null hypothesis.

2a) The grand mean = sum(data)/18 or sum([5 9 10]/3 = 24/3 = 8;

2b) Deviations1= [ 6 8 4 5 3 4 8 12 9 11 6 8 13 9 11 8 7 12] - 8 =

= [-2 0 -4 -3 -5 -4 0 4 1 3 -2 0 5 1 3 0 -1 4]

2c) SS1 = sum(Deviations12) = 152

2d) DegFree1 = #data - #params = 18 - 1 = 17

2e) chisqF1 = sum(Deviations12 /SD2 ) = 152 / 2.1292 = 33.53

3) Finally do the hypothesis test by taking the difference between the two chisquares.

3a) F = (chisqF1 - chisqF3)/(DegFree1-DegFree3) = (33.53 - 15)/(3-1) = 18.53/2 = 9.26

3b) p=F2p(F,difDegFree,DegFree3) = 0.0024 In agreement with the Wikipedia result

**Step 2. Estimate the standard deviation of the data for the general case.**

In our approach we need to know the standard deviation, SDi, of each datapoint Oi. Sometimes you are given that information. More often one needs to calculate it. There are many possible approaches (saving best for last):

1) If each datum is obtained by making say 10 repeated measurements then SDi would equal the standard error of the mean of the 10 measurements. In Matlab that is given by SE=SD(of the mean value) = std(rawdata)/sqrt(10).

2) Some types of measurements, such as use of a ruler to measure length, enable one to estimate the errors in the measuring instruments. A ruler measurement can be may have a precisions of half a tic mark (0.5 mm).

3) If the data had a binomial distribution like rolling dice (p=1/6) then SDi =sqrt(Ei(1-p)), where Ei=pi Ni is the expected value of the datum.

4) There are also many occasions when SDi=sqrt(Ei) is the correct formula if all outcomes are included in the data (and also for Poisson statistics when p is very small).

5) A common approach used in ANOVA is to assume that the SD of each datum is about the same, or the same after a nonlinear transform of the data. This approach is common in the social sciences. For the Wiki data above there are three means of 5, 9, 10. So before the analysis starts it is often easy to calculate the SD.

6) Sometimes it is difficult to do #5 because the parameters aren't simple means. Or maybe one is lazy. The "Generic" software makes it easy to use Iteration 1 to be the "optimal" iteration for automatically calculating the SD.

Whenever possible one wants to avoid the embarrassment of "naked data" (data without the associated SD).

For the Wikipedia data a common SD is calculated using Method 6. We do that by initially setting the SDi=1 and starting with a 3 parameter fit. This is the fit that Wikipedia uses. The standard deviation is calculated automatically using the standard standard deviation formula (SD=sqrt(sum((Oi - Ei)2)/df) ) where df=Ndata­-Nparams is the degrees of freedom. For this approach we have made the assumption that all the data has the same SD (SD=SDi).

**Step 3. Make a model of the data.**

There are a variety of approaches for how one can use parameters to specify a model. The model would have the same dimensions as the data (a vector of 18 elements for the Wiki data). The model should have far fewer parameters than there are data points. Typically the model doesn't come from thin air, but is based on your prior knowledge of how the various conditions would have been expected to affect the output.

An example of a typical model is the following Matlab code, WikiFtestFun, for the Wiki data. This code provides 4 different ways of fitting the Wiki data. Each parameter was chosen so that it has a meaningful interpretation. For example, in Iteration #5,

p(1) is the group mean. This is the null hypothesis

p(2) is the deviation of the first treatment from the mean. The Wikipedia site mentions this as another hypothesis.

p(3) is the difference between the 2nd and 3rd treatments. It's nice to have the difference as a parameter for a t-test.

p(4) is the slope of the 6 trials in condition #2.

At some point there will be more parameters than are really needed to adequately fit the data. We should be able to detect when the model has sufficient complexity by seeing when the chi square stops significantly decreasing as the number of parameters increases.

**cond1 = [1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3];%subject ID?**

**cond2 = [1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6];%days of training?**

**if iter==1, %For calculating SD. This is identical to iteration 4**

**expect= p(1) +p(2)\*(3\*(cond1==1) -1) +p(3)\*((cond1==3)-(cond1==2));**

**elseif iter==2, expect=p(1); %the mean of all data**

**elseif iter==3 expect= p(1)+ p(2)\*(cond1>1); %p2=difference of cond1=1 from mean.**

**elseif iter==4, %Same as iteration 1**

**expect= p(1) +p(2)\*(3\*(cond1==1) -1) +p(3)\*((cond1==3)-(cond1==2));**

**elseif iter==5, %Going beyond the Wikipedia fits to demonstrate linear regression**

**expect= p(1) +p(2)\*(3\*(cond1==1) -1) +p(3)\*((cond1==3)-(cond1==2));**

**expect=expect + p(4)\*(cond2-3.5); %orthogonal version**

**%this gives columns as: p(4)\*[5 -3 -1 1 3 5]/2 (orthog to [1 1 1 1 1 1]**

**end**

**dif=(expect-Info4FitFun.data)/Info4FitFun.SD; %dif\*dif' is chisq or chisqF**

**Note the fancy appearance of SD in denominator with no effort needed.**

The next section may be removed if it seems too complicated.

A previous time I taught this class our main focus was on the Wiki data and I had 7 iterations rather than 4 (ignoring the first iteration that was for SE). I did the two parameter fit 3 ways. For example compare the correlations between parameters 1 & 2 in iterations 2 & 3. In iteration 2 I defined parameter 2 in a simple manner. In iteration 3 I defined it to be orthogonal to parameter 1. It is worth comparing the SEs of the parameters in the two iterations. A summary of the 9 iterations is as follows (I use the abbreviation of p1 for p(1) for compactness):

For Iterations 1-5, Expected has the following dependence on condition 1 and no dependence on condition 2:

Iter=1: p1\* [1 1 1] Where the 3 numbers correspond to the 3 levels of condition 1 (note p1 means p(1)).

Iter=2: p1\* [1 1 1] + p2\* [0 1 1] Note that [1 1 1] and [0 1 1] are + correlated, so p1 and p2 are - correlated.

Iter=3: p1\* [1 0 0] + p2\* [0 1 1] Note that [1 0 0] and [0 1 1] are uncorrelated so p1 and p2 are uncorrelated.

Iter=4: p1\* [1 1 1] + p2\* [-2 1 1]/3 Note that [1 1 1] and [-2 1 1] also uncorrelated with p1 same as Iter=1.

Iter=5: same as #4 + p3\*[0 -1 1] Note that [1 1 1], [-2 1 1], [0 -1 1] are all uncorrelated

Iter=6: same as #5 + p4\*[1 2 3 4 5 6] where p4 is independent of cond1 and depends on the 6 levels of cond2.

Iter=7: same as #5 + p4\*[-5 -3 -1 1 3 5]/2 simply shifts #6 by -3.5 to make it orthogonal to [1 1 1 1 1 1]

A nice introduction to Matlab notation is to look at Model type=3 that can be written in two different ways:

expect= p(1)\*(cond1==1)+ p(2)\*(cond1>1); %p2=0 is the null hypothesis (2)

expect= p(1+(cond1>1)); %this should be identical to preceding line! (3)

Both specify p1 to be the mean of condition 1 and p2 to be the mean of conditions 2&3.

Model 5 is a 3 parameter fit similar to the Wikipedia example. Thus we hope that we get the same sum of square error (sum(dif^2)) and p-value as in Wikipedia. This would be one of the main things one checks to see if there are errors.

When one looks at the Wiki data it looks like across the 6 runs (cond2) the values are getting smaller. Model 6 tests the hypothesis that the data linearly depends on condition 2. This can be written as:

E6(i) = E5(i) + p4\*cond2(i) (4) (see above where we changed "expect" to E(i) to show the subscript (i) )

One problem with this way of writing it is that if you look at the parameter correlation matrix you will see that parameter 4 is not orthogonal to the group mean. It can be made orthogonal by removing the mean of cond2. Since we are now focusing on the 4th parameter it is useful for didactic purposes to simplify the first 3 parameters as follows:

E6(i) = p(cond1(i)) + p4\*(cond2(i) -3.5) (5)

At first sight this looks tricky since we are using the cond1 vector to specify which parameter to use for each row of the wiki data. This is exactly what Wikipedia did so now we finally get the same means that was in the Wikipedia article. I didn't do that earlier because I wanted to just add 1 parameter at a time so one could see how each separate parameter influences the result. It is worth spending some time to look at the code and look at these results to try to understand what is going on and to get some practice with Matlab.

In the above iterations we added at most one parameter at a time, from n-1 to n parameters, always maintaining the same n-1 model. As we shall see, that enables us to compare the chi squares for the two fits*. For a preview of coming attractions if the extra parameter wasn't useful in fitting the underlying behavior and it only did a better job of fitting the noise then the change in chi square is expected to be -1. For the change to be significant at the p=.025 level the chisquare would have had to decrease by >4.*

**Step 3. Calculate dev: the deviations of data from model** In order to facilitate the use of matrices for computation (a la Numerical Recipes or NR for short) the deviations are given by:

dev(i) = (O(i) - E(i))/SD(i) (6)

These are normalized deviations that measure of the difference between the observed and expected data in standard deviation units. Eq. 6 is one of the most important equations that we will encounter. This equation is THE output of the function that lsqnonlin calls. Lsqnonlin finds what is the best fitting E(i) that minimizes the sum of square deviations as will be discussed in the next section.

These normalized deviations are often called z-scores. z(i), if the observed data is known to have a Gaussian distribution and if SD(i) is known. If SD(i) is estimated from the data then it could have been called t(i) rather than z(i). For the Wiki data the SD isn't given so we will initially take SD=1 and then we will estimate the SD from the data.

**Step 4. Calculate chi-square (2).** For this section we assume that the SD or variance of the data was reliable. Chi square is simply the sum of squares of the deviations:]

2 = Chisq = I dev(i) dev(i) = I (O(i) - E(i))2/var(i) (7)

where var(i)=SD(i)2. Chi square is the most common measure of the goodness of fit. The optimal parameters of a fit are those that minimize 2. Note that Eq. 7 has a 2 distribution if the variance, var(i), was accurately known before the data was collected. If the variance is estimated based on the same data, then Eq. 7 turns out to be closely related to F. It needs to be divided by the degrees of freedom to become a true F. Step 5 will tell us what to do if the estimate of the covariance needs to come from the sampled data, O(i).

As we will discuss, there is also a connection between the SE of the parameter estimates and 2. It turns out SE(k), the standard error of the kth parameter, is the amount of shift in that parameter away from its optimal value that will increase 2 by unity, while allowing the other parameters to take on their optimal value as the shift is made (see NR or Bevington). 2SE(k) is the point where 2 increases by 4. We will come back to this later.

*Covariance and sqrtm (an aside). This detour examines what happens if there are correlations in the data:*

*If the datapoints, O(i), are correlated the variance must be replaced by covariance and Eq. 7 becomes:*

*Chisq = ij (O(i) - E(i))\* inv(covar(i, j)\*(O(j) - E(j)) (8)*

*One might worry about how to specify the deviation in Eq. 6 if there is a covariance in Eq. 8. The answer is simply to take the square root of the inverse covariance. Matlab has a special function sqrtm(M) that takes the square root of a matrix such that its square is the original matrix. Thus Eq. 6 for dev(i) becomes a matrix rather than a vector:*

*dev(i, j) = (O(i) - E(i))\*sqrtm(inv(covar(i, j)) (9)*

*and Eq. 8 becomes:*

*Chisq = ij dev(i, j) dev(j, i) (10)*

*It should be emphasized that the sqrtm function is a really wonderful special trick. It is sometimes called a 'whitening' function. To whiten data means to make its elements independent of each other. In this case the independence is over the various samples of the data. The introduction of the covariance matrix is called Generalized Least Squares (GLS).*

Goodness of fit. One of the main reasons for using the 2 statistic is that it allows one to evaluate the goodness of fit. If the model provides an excellent description of what the system is doing then the numerator and denominator of each of the deviations will be about equal with a ratio near unity and the total 2will equal the degrees of freedom, df:

df = Ndata - Nparameters (11)

Ndata is the number of terms being summed in the calculation of 2. The number of parameters must be subtracted off because in the least squares fit each parameter will be partially used to fit the noise as well as the underlying system. It turns out each parameter gobbles up a unity (1) amount from the estimated value of 2. One might ask how does one know that Eq. 11 is the unbiased estimate of the expected value of 2 for a good fitting model. There are two approaches, theoretical and by simulations. Although both approaches lead to Eq. 11, for me the simulation method is much simpler. You are encouraged to do some simulations to verify Eq. 11.

To a good approximation the 2 distribution has a Gaussian shape (especially when df>5) with a mean and SE of:

mean(2) = df (12a)

var(2) = 2 df or (SE(2) =sqrt(2df) (12b)

Suppose, for example, df=50 and chi-square=95. Thus 2expected = 50 ± 10 (since10 is sqrt(2\*50)) and the observed value of 95 is 4.5 SDs from the mean indicating a poor goodness-of-fit. There are two general reasons for a poor fit:

1) The model was inadequate and needs to be improved, hopefully without introducing too many extra parameters.

2) The model is fine, but the estimate of the SDi in the denominator of 2, was unreliable.

Reduced chisq (chisqR) and fudge factor (FF). As discussed above the expected value of 2 for a well fitting model equals the degrees of freedom (df). A normalized version of 2  is called the reduced 2, given by:

chisqR = r2 = 2 / df (13)

The expected value of r2 equals 1 for a well-fitting model. If the reduced r2 is substantially greater than 1 that would mean either the model was not to be trusted or that the presumed SDs were too small. In either case it would be good to enlarge the SDs that went into the calculation of r2. On the other hand if r2 is substantially less than 1, that could mean either the data, by accident, fit the model better than would have been expected or possibly the original SD's were overly large. To be conservative, it is reasonable to leave the values of SDi unchanged. That is, we tend to only use the following fudge factor in cases where the r2 is too large. The correction factor is given by:

FF = sqrt(r2) (14)

The corrected SD(i) is given by:

SDF(i) = SD(i)\*FF (15)

If this corrected estimate of SD is used in Eq. 6, then the new F2 = df. Here is the math for that:

devF(i) = (O(i) - E(i))/SDF(i) = (O(i) - E(i))/(SD(i) FF) = dev(i)/ sqrt(r2)

F2 = sum(devF(i)2) = sum(((dev(i)/FF)2) = 2/r2 = df (16)

This result shows that F2, after using FF to correct the SD, is fixed at df, and can no longer be used for estimating the goodness of fit. So you might ask what is the purpose for introducing FF? The answer is that the SEparam(k) of the parameter estimates depend on the SDdata(i) of the original data (see next section). If the SDs of the data are doubled by FF, so are the SEs of the parameter estimates. Also, the values of t for the t-test would be halved, and values of F for the F-test would be reduced four-fold (since F=t2).

**Step 5. Determining SEs of parameter estimates.**

We start by discussing how the parameters and their standard deviations are estimated. We will be using the Matlab fitting program lsqnonlin. The output of the program are the parameters, p, the 2 (or SSE for the case went SDi wasn't provided) and the jacobian, j that is directly connected to the linear regression design matrix. The main program has the following steps to get the parameter SE from j.

j=full(j); %something about sparse matrices

covar=inv(j'\*j); %see Numerical Recipes for meaning of covariance matrix

SEtemp=sqrt(diag(covar))';%Standard error uncorrected

If the original data had reliable SE(I) provided, then SEtemp(k) is a reliable SE for the parameter estimates. However, if SE(i) wasn't provided so it was set to 1, or if SE(i) was provided but the value of 2was unreliable then a correction (fudge). Since our focus at this point is for the Wikipedia data where SE(i) wasn't given then the the fudge factor is actually the estimate of the SD(i)=SD of the date. The code in the main program for SD is:

DegFree=18-length(params0);

SD=sqrt(sse/DegFree) %estimated like line 15 above, since the SEdata=1

SELSQ=SEtemp\*SD; %Fixes SEtemp. Like line 18 above.

These final value of SELSQ(k) are the standard errors of the parameters param(k).

For those interested in a mathematical derivation of the above, one can go to Numerical Recipes that provides a derivation for getting standard errors of the parameter estimates. [http://apps.nrbook.com/c/index.html Section 15.4 should get you close]. The derivation says the parameter covariance is the inverse of the design matrix times its transpose.

Given the parameters and their SEs, one can now do a variety of significance tests.

**Step 6. Testing null hypotheses.** We now look at four approaches for hypothesis testing, the z-test, t-test, 2 test and the F-test. Given the value of one of the four statistics for hypothesis testing: z, t,2 and F, the task is to find p, the probability that that value could have been obtained by the null hypothesis.

a) z-test. Relevant if the original SE(i) of the data was reliable, and didn't need to be estimated.

Suppose for example, the kth parameter of a model is given by param(k) = 6 ± 2. If the parameters had been estimated without the FF, the ratio of parameter to its SE is called z.

z(k) = param(k) / SE(k) (17)

For our example, z(k) = 3, so param(k) is 3 SEs away from being zero.

It is common to say that 2 SEs away from being zero means that the null hypothesis can be rejected at p=0.05. That is because the area in the tail of a Gaussian distribution above 2 SDs is 2.5% and since there are two tails, the total area on both sizes is 5%. To be totally accurate z=1.96 corresponds to a 2.5% tail of the Gaussian. For our example with z=3, the area of the two tails would be p=.0026 (a 0.26% chance of being in the tails). For our example of param(k) having z=3 we can say that just by random luck (or misfortune) there is a 0.26% chance that param(k) could have had a value of 6+2=8 or higher or 6-2=4 or lower. Alternatively one can say that the null hypothesis (param(k)=0) can be rejected with a 0.26% chance of the data being a false alarm error. A false alarm is an incorrect rejection of the null hypothesis.

b) t-test. If the FF fudge factor had been used in calculating SEi in Eq. 6, then z in Eq. 17 is replaced by t. This is an important step so some further emphasis is provided here and again in discussing the F-test. There are situations where the data is too sparse to get a reliable estimate of SE for the denominator of Eq. 6 so it is common to see 2 without the denominator. It is then called SSE, the sum of square errors, and the procedure is called ANOVA. We won't use that notation, but rather, for simplicity, consider it to be a 2 with an error in calculating the denominator, so it needs the FF correction. We do that because we can then still use all the 2 machinery for estimating parameters and their SEFFs. We will sometimes use a subscript "FF" as a reminder of when we use the FF correction to estimate the 2 denominator.

If the parameters had been estimated with the FF, the ratio of parameter to its SE is called t.

t(k) = param(k) / SEF(k) (18)

In the example above where param(k) = 6 ± 2 the ratio of Eq. 18 would be t(k) = 3. In order to determine the significance of this value (for testing the null hypothesis of param(k)=0) one needs to know, not only t, but also the number of degrees of freedom that went into estimating FF. In this case it would be the number of data points minus the number of parameters. As will be discussed in connection with the F-test the number of parameters involved with this t-test is 1. So when looking in t-test tables for the significance probability one would use DegFreedom = Ndata-1.

c) 2 test. In the Wikipedia F-test entry the goal was to test whether the dataset could be adequately fit by a single parameter (the mean of the 18 data points) or were three parameters needed, one for each.

We'll get to that Wiki example at the end of this section but first we'll start with a more generic case of testing whether the 2nd and 4th (for example) parameters of some model are significantly different from zero. Suppose for example the full model has five parameters and the restricted model has three parameters. It turns out, by the wonderful properties of 2 that the difference between the two 2 statistics is another 2 statistic whose degrees of freedom (df) equals the difference of the degrees of freedom of the two original 2s. For our example where we are interested in the null hypothesis of the 2nd and 4th parameter being not significant 2 = 2N-2 -  2N where N= number of parameters of the larger model. Since 2 has a 2distribution with 2 df, its expected value is 2 and its approximate SE is sqrt(2df) = 2. One can do approximate statistics making believe that it is a Gaussian distribution with mean and SD given by 2±2. It is often useful to have a quick estimate of the 2 reliability. However that can be a crude approximation (especially since 2 can't be negative and 2±2 makes one sweat) so it is better to use the p-value given by the 2 distribution. That distribution is called a gamma function (see Numerical Recipes for details).

d F-test. Having just gone through the discussion for the 2 test it is easy to discuss the F test. Basically one follows all the steps for the 2 test except that one makes the FF correction because of the non-trustworthy estimate of SDi in the 2  formula. The standard F-test literature doesn't call the chi square in the denominator of F a fudge factor, but rather it simply estimates SDi by the standard deviation of all the deviations from the most complete model. I prefer to calling it a "fudge factor" as a reminder that it is often the case when doing F tests that the data is binomial, so an SE is known and the 2 test could have been used, rather than setting SE=1 as is done with ANOVA.

A standard way to think of F is that it is the ratio of two reduced 2s.

F = 2r / 2r (19)

Suppose, for example, we measured 30 datapoints, and the full model had 5 parameters. Using the above example, we are interested in testing the hypothesis that 2 of the 5 parameters are superfluous. Thus the numerator of Eq. 19 is:

2r = 2/ df = 4 / 2 = 2. (20)

The denominator depends on the 2that was used for measuring the SD. There were 30 data points, fit with 5 parameters so df=25. A reasonable guess is the 2is approximately equal to the degrees of freedom, so let's assume that's the case. Thus the denominator would =1, so F=2. If the reduced 2in the denominator ware say 2r=100, that would be an indication that the initial estimate of SD was 10 times too small, and both numerator and denominator of Eq. 19 would have been 100 times too large. That factor of 100 in both numerator and denominator would have cancelled and the value of the ratio, F=4, would be unchanged. The same would hold if the initial SD were 10 times too large and both numerator and denominator would have been 0.01 times too large.

It is useful to consider how the Wikipedia F-test example works in our framework. There are 18 data points in three rows of 6 columns (see Eq.1) or as specified by cond1 and cond2 in Step 1. The two models considered by Wikipedia are: 1) one parameter that is a constant to fit all 18 data; and 2) three parameters, one for each row (cond1). The two models have 2 = 152 and 68 for 1 and 3 parameter models. The 2 for the numerator is 152 -68 = 84 giving a reduced 2r = 84/2=42. The reduced 2for the denominator is 2r = 68/15 = 4.533 Thus F = 42/4.533 = 9.265. This is identical to the value obtained in the Wikipedia F-test article. The fudge factor is FF= sqrt(4.533) = 2.129. The improved value of SDFis obtained by multiplying the original SD (SD=1) used in calculating 2 by FF, giving the improved SDF = 2.129. Finally, the p-value (see Step 7) for this F-test is:

Stat2Prob('F', F, [df1 df2],iplot) = Stat2Prob('F', 9.265, [2 15],0) < 0.0024.

The value we get for F, as well as the p-value is identical to the results found in the Wikipedia article. This overview of the Wiki F-test entry is an excellent way to learn the connections between the standard approach to ANOVA and our 2 based approach. We believe our 2 approach to be superior because of its generality to any problem, linear or nonlinear, simple or complex.

The case of df1 = 1 is special. This is the case where we are testing for the significance of a single parameter. The F-test in this case is identical to the t-test. There are two approaches for doing this test:

1) Based on the difference between the two SSEs (or 2 followed by FF correction) with the critical parameter either fixed or allowed to float.

2) Based on the single least squares fit with the parameter allowed to float. The ratio of the parameter to its SE is either z or t, depending on whether FF was used. A major advantage of this approach to the F-test is that it shifts the calculation of F to the standard least squares programs that are readily available in Matlab.

**Step 7. Going from z, t,2 and F to probabilities.**

It would be good to discuss the standalone Stats2Prob function. It is called by:

[ProbFancy,ProbSim,Prob3]=Stat2Prob(iplot,value, degfree1, degfree2)

One of the nifty things about this function is that it not only calculates the fancy hypothesis test p-value based on erf function (for z), gamma function (for chi square) and beta function (for F or t), it also calculates the p-value by doing amazingly simple simulations. It is worth looking at the simulations to see how 2 and F are simulated. Type help Stat2Prob to get the needed details on how to use it.

Several examples are:

Stat2Prob('z', 1) = 0.16 means there is a 16% chance that the value of z would be bigger than 1. and Stat2Prob('z', 2)=0.0228 and Stat2Prob('z', 1.96) = .025 for a one-tail area. For the area of z>1.96 and z<-1.96 the two-tail area would be 0.05. This corresponds to the famous p<0.05 null hypothesis test that is often used.

2 =4, df=2; Stat2Prob('chi',2, df) = 0.135 corresponding to p<0.14. Note that this isn't that far from the Gaussian approximation that would have given p<0.16 (recall our example above of chi=2±sqrt(4), so 2+2=4 is 1 SD above the mean).

t=1.96; df2=100; Stat2Prob('chi', t2, 1, df2) = 0.0528. If you increase df2 it asymptotically goes to p=0.05. In fact for general values of t it will be twice what one gets for a z-score. The doubling is because the t-test is really an F-test (F=t2). Because of the squaring the upper tail of the t or F test includes both tails of the z test. Additional F-test examples with discussion are discussed next.

The probability of getting this value of F=2 or higher by chance, even if the two parameters in question were fixed at zero is:

F=2, df1=2, df2=25, Stat2Prob('F', F, df1, df2) < 0.1564. (21)

Note that p<0.154 for the F-test is slightly larger (weaker) than it was for the 2 test where p<0.135. If in the F-test df2 had been much larger, then we would have found p<0. 135, just like for the 2 test. If there had been fewer data points the p-value would have been substantially larger, because of the greater uncertainty in estimating SDi.