Project 2: The Chi-Square Distribution
PHSX815

David Coria

March 22, 2021

1 Introduction

The chi-square distribution with k degrees of freedom, or $\chi^2(k)$, is the distribution of the sum of the squares of k independent Gaussian random variables. The chi-square distribution has a mean $= k$ and a variance $= 2k$ [1]. This distribution is a special case of the gamma distribution, and it is most notably known (to me, at least) for its use in the chi-square test for the goodness of fit between observed data and hypothetical distributions [2].

The binomial distribution is a discrete, rather than continuous, probability distribution that represents the number of successes in a sequence of n independent experiments–each with probability of success p. As seen from the probability mass function below, k successes occur with probability p^k and $n-k$ failures occur with probability $(1-p)^{n-k}$ [3]. The binomial coefficient, $\binom{n}{k}$ explains that there are that many different ways of distributing the k successes within the n trials.

Remember the parameters of the Gaussian Distribution are the mean μ, the standard deviation σ and the variance σ^2. In a standard normal distribution, as required for the chi-square distribution, mean $\mu = 0$, the standard deviation $\sigma = 1$.

Probability Distribution Function (Gaussian) [4]:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{1}$$

Probability Distribution Function (χ^2) [1]:

$$f(x) = \frac{x^{\frac{k}{2}-1} e^{-\frac{x}{2}}}{2^{\frac{k}{2}}\Gamma(\frac{k}{2})} \tag{2}$$

Probability Mass Function (Binomial Distribution) [3]:

$$f(k,n,p) = \binom{n}{k} p^k (1-p)^{n-k}$$
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

2 Significance: Chi-Squared Tests

I wanted to do something for this project using the chi-square distribution in an attempt to understand it better since it sets the foundation for the chi-squared statistic that is used extensively in "goodness of fit" tests. This statistic proved useful in my own research by helping me calculate isotopic abundances in several solar twin stars by comparing the observed spectra to Solar models with varying abundances of the same isotopologue. Here, the chi-squared statistic is defined as the weighted sum of squared deviations [2] or:

$$\chi^2 = \sum_i \frac{(O_i - M_i)^2}{\sigma_i^2} \tag{3}$$

where O_i represents observations (in my case: the flux at a certain point of the observed spectrum), M_i represents the model/calculated data (in my case: the flux at a certain point of the model spectrum), and σ_i represents the corresponding uncertainties (of the observed spectrum, in my case). This χ^2 statistic asymptotically approaches a χ^2 distribution and can then, in turn, be used to calculate a p-value by comparing the statistic to a χ^2 distribution [2]. If you calculate the χ^2 value between the observed ^{13}CO line and the each of the model lines corresponding to a different quantity of ^{13}CO, you obtain four points in χ^2 space. These four points can be fit with a parabola, and the minimum of this parabola marks the interpolated "best fit" of the data. This corresponds to a xSolar abundance. This process is pictured/explained a bit more in Figure 1.

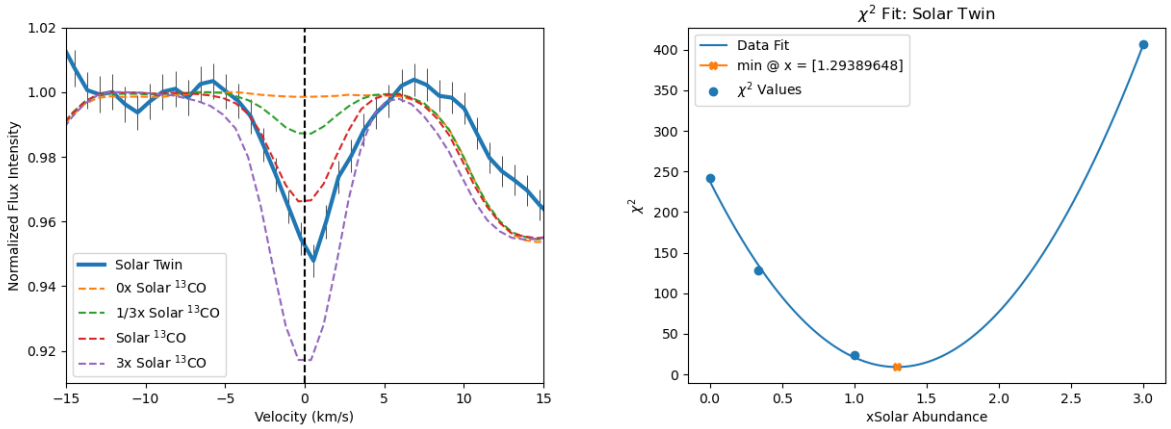

Figure 1: On the left: This is a plot of "stacked ^{13}CO absorption lines". This process (not super important for you to understand) is done for the observed spectrum of a solar twin and each of the four Solar models representing 0x, $\frac{1}{3}$x, 1x, and 3x Solar ^{13}CO abundances. This plot is used to compare the amount of a certain species in the observed star to the varying amounts in a set of models. Just taking a quick look, you might say that the observed Solar Twin has a ^{13}CO abundance between 1x and 3x Solar value. We need to perform a series of chi-squared "goodness of fit" tests to determine an actual value. On the right: This is a plot of the four χ^2 values calculated between the observed spectrum of a solar twin and each of the four Solar models representing 0x, $\frac{1}{3}$x, 1x, and 3x Solar ^{13}CO abundances. The χ^2 values are fit with a parabola which is then minimized. This minimum value by definition represents the interpolated best abundance fit of the observed line. This means that this solar twin star has a ^{13}CO abundance of ≈ 1.47 xSolar.

3 Code and Experimental Simulation

The goal for project two, as a continuation of project 1, is to use built-in numpy/scipy functions to produce a set of n independent standard Gaussian random variables, use them to construct a chi-squared distribution with a randomly determined number of degrees of freedom, repeat this process N times as specified by the user, and finally, output these randomly generated chi-square distributions to a .txt file.

PHSX815 Project2.py: This code accepts the following parameters via user input: sample size, the sample size for the randomly generated standard normal distributions; Nexp, the number of total experiments to run; n, the sample size for the binomial distribution used to generate random values for the number of degrees of freedom; prob, the probability of success in the binomial distribution, and outputfile, a filename which will be used to output the set of randomly generated chi-square distributions. The script has default values assigned, so the code will run with sample size $= 100$, Nexp $= 10$, prob $= 0.5$, and no output text file if no user input is specified.

The code begins by randomly generating (according to a discrete binomial distribution with user-specified parameters n and prob) an array of integers of length Nexp to be used as the number of degrees of freedom (DOF) i.e. the only parameter that chi-square distributions depend on. I chose to use the binomial distribution because it can generate random integers, which is what I need to assign to DOF, and has a probability parameter that we can change to make things more interesting. This file then takes the given sample size from the user and produces Nexp randomly generated standard normal (Gaussian) distributions. Then, the script takes the sum of the squares of the Nexp standard normal distributions to construct a chi-square distribution for every randomly generated value of DOF. If requested by the user, the script will produce a text file containing Nexp number of chi-square distributions, each with a randomly determined number of DOF. Finally, I print the array of randomly generated DOF, calculate the mean and variance for each randomly generated chi-square distribution, and print these arrays as well.

Updates from Project 1: This code now has the capability of repeating the experiment from Project 1 Nexp times, as specified by the user. This means that in the end, there are Nexp number of randomly generated chi-square distributions, each with a random number of DOF. Furthermore, the analysis i.e. mean and variance calculations are carried out for each chi-square distribution instead of just one at a time.

4 Analysis

The analysis for Project 1 was done for three different randomly generated $\chi^2(k)$ distributions, each with a different number of DOF. Below, we can see the histograms created for each data set by the Chi Square Generator.py script. Each plot includes a continuous χ^2 probability distribution function with a DOF value corresponding to the randomly generated data set. The script also calculated the mean and variance of each data set. The calculated values are written in the figure captions.

For Project two, experimental repetitions are done automatically, and so we can produce a greater number of chi-square distributions, each randomly generated using a standard normal distribution– using a randomly assigned number of DOF. Because we can accommodate larger experiments, I decided not to produce a plot of each chi-square distribution as that would take up too many resources. Instead, Project 2 works only with arrays. The chi-square distributions are stored in an n-dimensional array of size (Nexp, sample size). This n-dimensional array makes it much easier to calculate and display the mean and variance for each chi-square distribution.

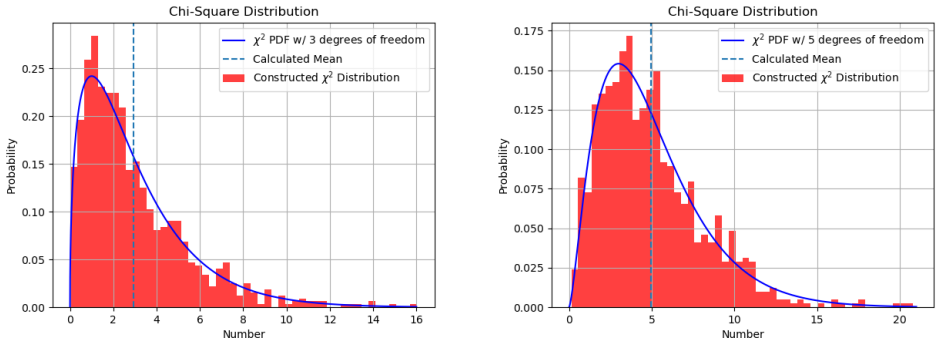

Figure 2: Left: A χ^2 distribution, in red, constructed from a set of three (3) randomly generated standard normal (Gaussian) distributions. The blue line corresponds to the continuous probability distribution function of $\chi^2(3)$. The mean for this set is 2.93. The variance is 5.83.Right: A χ^2 distribution, in red, constructed from a set of three (5) randomly generated standard normal (Gaussian) distributions. The blue line corresponds to the continuous probability distribution function of $\chi^2(5)$. The mean for this set is 4.95. The variance is 9.58.

5 Conclusion

Now that this code no longer produces histograms, we must rely less on the continuous χ^2 distributions and how well they may seem to agree with the randomly generated χ^2 distributions and instead focus on the mean and variance calculated for each distribution. The mean calculated for each data set agrees well with the theoretical mean of a χ^2 distribution–that is, the mean should be equal to DOF, the number of degrees of freedom (or the number of independent Gaussian distributions used to produce the final χ^2 distribution). Similarly, the variance calculated for each data set agrees with the theoretical variance of a χ^2 distribution which should be equal to $2k$, or double the number of degrees of freedom. Even if we did not store the DOF array, we could easily recover these random values using the array of calculated mean values for each data set. I was able to successfully reproduce N χ^2 distributions, each with a randomly generated number of degrees of freedom, using a set of independent, randomly generated standard normal (Gaussian) distributions using only built-in numpy and scipy functions.

References

[1] N/A, *Chi-square distribution*, *Wikipedia* (Feb, 2021) .

[2] N/A, *Pearson's chi-squared test*, *Wikipedia* (Jan, 2021) .

[3] N/A, *Binomial Distribution*, *Wikipedia* (March, 2021) .

[4] N/A, *Normal distribution*, *Wikipedia* (Feb, 2021) .