ECEN321 - Lab 1

Joshua Benfell - 300433229

March 21, 2020

1 Introduction

This report investigates three types of distribution; Normal, T and Chi^2. The purpose of this investigation is to firstly, further understand these distributions and secondly to see how accurately the experimental data matches with the theory.

2 Theory

2.1 Normal Distribution

For this distribution 1000 normally distributed variables were generated using the **randn()** function from the Python library, numpy. This was done around a mean of 2.5 with a variance of 16; standard deviation of 4. This generated a list of normally distributed variables.

$$\mu + randn(N) * \sigma$$

After this list was created, the mean, variance and standard devitation were obtained from the sample so that they could be compared with the input variables. For visual comparison to a normal distribution, the sample was plotted as a histogram in 30 bins. To allow comparison with the probability density function (PDF) of the normal distribution the histogram had to be scaled to a probability mass function (PMF). This was done by first scaling the bin heights by the total number of sample points, and then scaling by the width of the bins.

```
H,X = np.histogram(nums, bins = 30)
dx = X[1] - X[0]
histProbabilities = H / (N * dx)
```

$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

Using eq. (1), the theoretical and experimental PDFs were calculated and plotted along side the scaled histogram.

The other aspect of this distribution to compare is the cumulative distribution function (CDF) of the experimental data from the histogram and for both the population mean and variance and those of the sample. The theoretical CDFs where calculated with eq. (2), where erf is the error function from the scipy library and the empirical CDF of the histogram was done using the cumulative summation function (cumsum) in the numpy library.

$$\phi(x) = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right) [1] \tag{2}$$

2.2 Student t Distribution

To ivestigate the t distribution, a matrix of size $5_M \times 10000_N$ was filled with normally distributed samples with the same mean and variance as that described in section 2.1. From this sample each column of the matrix was operated on making a t-value for that column using eq. (3). This equation uses the standard deviation of the entire matrix, the true mean and the mean of the column.

$$\frac{(\mu_{column} - \mu_{true})}{\sigma_{sample}} \times \sqrt{M} \tag{3}$$

The t-values were then plotted on a histogram using the same method described in section 2.1. Plotted alongside this was theoretical PDF (tPDF) for degrees of freedom $\nu = M - 1 = 4$. On a second plot, the sorted data was plot against the rank alongside the theoretical CDF (tCDF).

2.3 Chi^2 Density

This density was investigates by first, generating a matrix in the same way as the one in section 2.2. This time, chi squared values were made by taking the sample variance for each column and dividing it by the $\frac{5-1}{\sigma^2}$. This was plotted alongside the tPDF for 4 (M-1) and 5 (M) degrees of freedom using the chi2.pdf method in the scipy library. This was repeated on a second plot but instead of the sample variance where it's the difference from the sample mean, the difference from the true mean was used.

3 Results and Discussion

3.1 Normal Distribution

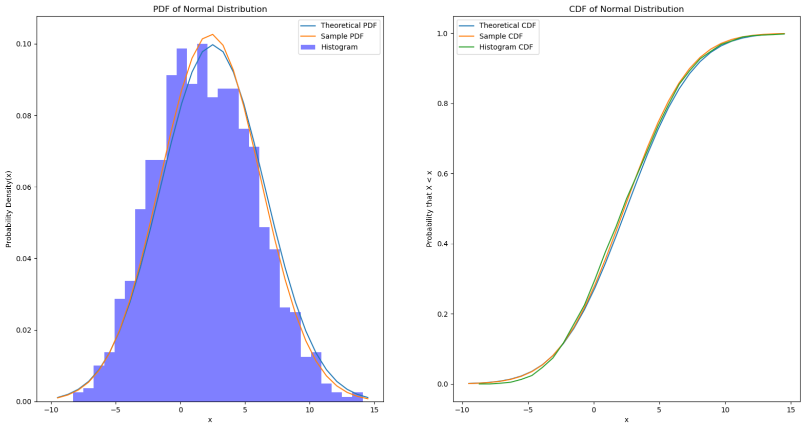

Figure 1: PDFs and CDFs of the normal distribution with true mean of 2.5

The resulting plots from the python script written for this distribution can be seen in fig. 1. The following values relate to this sample set:

- μ: 2.332364394985525
- σ^2: 15.086094766162631
- σ: 3.8840822295830235

From the above list we can see that the values of the sample have minor differences from the true values that the set was generated from. This is reflected in the plot as it can be seen that the general shape of the pmf is similar to the theoretical pdf. For this particular sample it is better reflected in the experimental pdf, both are left leaning but it is less exagerrated in the pdf.

Both the theoretical and experimental CDFs are very close to each other that any difference is minor. The CDF from the pmf has got clearer differences in it. This is reflected in the pmf as it can be seen that many more data points happen early on in this sample which causes the cdf to be above the tCDF in the second plot.

Because both of these plots have results that closely resemble the theory it can be concluded that the theory behind the normal distribution matches the experimental results.

3.2 Student t Distribution

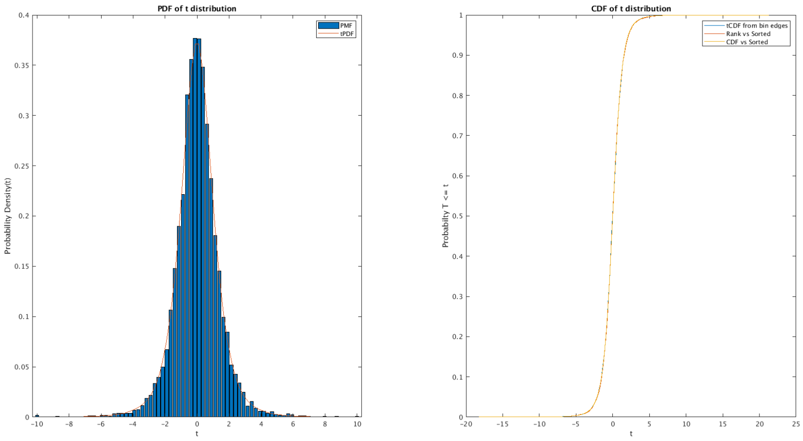

Figure 2: PDFs and CDFs of the t distribution with true mean of 2.5

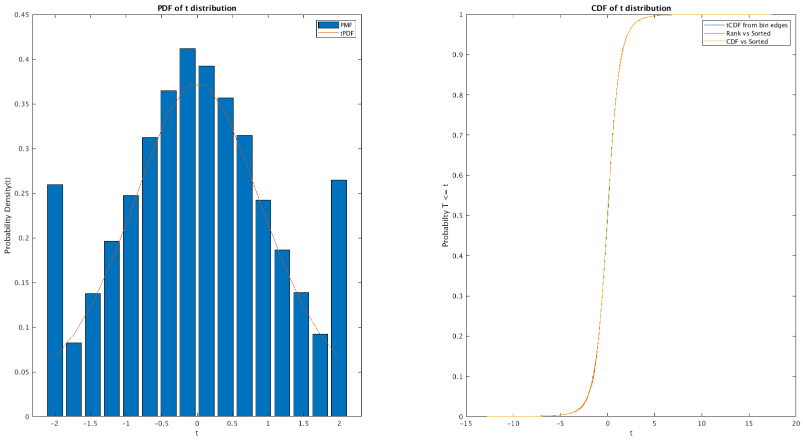

Figure 3: PDFs and CDFs of the t distribution with true mean of 2.5 over a small range

The resulting plots for this experiment can be seen in fig. 2. The experiment probability density lines up along the edges almost exactly with the theoretical pdf. This holds true for the cdf and tcdf as there is very little difference between the two lines. The tcdf was compared against two other plots, the cdf made from the pmf (histogram) and from plotting the

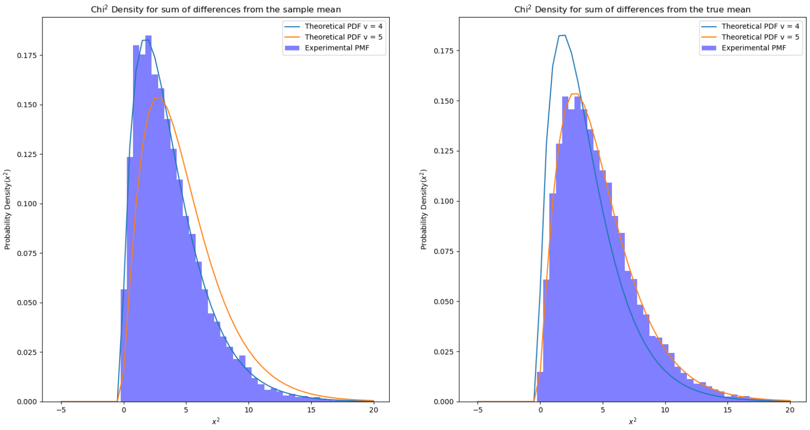

Figure 4: PDFs and CDFs of the Chi^2 distribution with true mean of 2.5

sorted data against the rank. Both ended up being very close to the theoretical line. This indicates that the t distribution holds true for normally distributed random variables.

There was some issue with generating this plot. If the range of the x axis that the histogram was generated on was too small (between ±2 in fig. 3) it would shoot up on the sides due to the long tails. This is remedied by specifying a larger range.

3.3 Chi^2 Density

The resulting plots for this experiment can be seen in fig. 4. On each plot the pdf for degrees of freedom 4 and 5 are plotted alongside the corresponding pdf. The left plot is a chi squard density from using the sample variance. From this plot we can see that it lines up very closely with the pdf where $\nu = 4$. The plot on the right is the density from using the difference between the true mean, and it can be seen that this lines up very closely with the pdf of $\nu = 5$. This indicates that the degrees of freedom of the sample pmf is influenced by how close you sample mean is to your true mean, with it being closer resulting in a higher ν value.

From both plots, it can be concluded that a chi squared density can be applied to a normally distributed random variable as the experimental results line up with the theoretical ones.

4 Conclusion

From all plots in section 3.1 it can be seen that there is a significant amount of variation in the data compared to plots in sections 3.2 and 3.3. This indicates that accuracy in statistical analysis can be improved by taking more measurements.

References

[1] Mathworks, “error function matlab erf mathworks australia,” 2020, accessed: 21/03/2020. [Online]. Available: https://au.mathworks.com/help/matlab/ref/erf.html

Appendices

A Normal Distribution Code

```python
import numpy as np
import matplotlib.pyplot as plt
# Ensure we are using tkinter for matplotlib
from scipy import special
# matplotlib.use("TkAgg")

N = 1000
mean = 2.5
variance = 16
sigma = variance ** 0.5
bins = 30

nums = sigma * np.random.randn(N) + mean

# Need Mean and Variance
sampleMean = np.mean(nums)
sampleVariance = np.var(nums)
sampleStdDev = np.std(nums)

print("Sample Mean: {}, Sample Variance: {}, Sample
   STDDEV: {}".format(sampleMean, sampleVariance,
sampleStdDev))


plt.figure(1)
plt.subplot(1,2,1)
# PMF
H,X = np.histogram(nums, bins, range=(-12+mean,12+mean)) # Make histogram values and return them and the counts for each bin.

dx = X[1] - X[0]
H = H / N
histProbabilities = H / dx
# print(histProbabilities)
# print(sum(histProbabilities))
plt.bar(X[:-1],histProbabilities,width=dx, color="blue", alpha=0.5, label="Histogram")

# PDFs

expectedPDF = ( 1/np.sqrt(2 * np.pi * variance) ) * np.exp(np.power((X-mean), 2)/(-2*variance) )  # expected one
plt.plot(X, expectedPDF, label="Theoretical PDF")

samplePDF = ( 1/np.sqrt(2 * np.pi * sampleVariance) ) * np.exp(np.power((X-sampleMean), 2)/(-2*sampleVariance) ) # uses mean and variance of sample
plt.plot(X, samplePDF, label="Sample PDF")

plt.legend(loc="upper right")
plt.xlabel("x")
plt.ylabel("Probability Density(x)")
plt.title("PDF of Normal Distribution")

# CDFs

cdf = (1/2)*(1 + special.erf((X-mean)/(sigma*np.sqrt(2))))  # Expected CDF
plt.subplot(1,2,2)
plt.plot(X,cdf, label="Theoretical CDF")

sampleCdf = (1/2)*(1 + special.erf((X-sampleMean)/(sampleStdDev*np.sqrt(2))))
plt.plot(X,sampleCdf, label="Sample CDF")

histCDF = np.cumsum(H)
plt.plot(X[1:], histCDF, label="Histogram CDF")

plt.legend(loc="upper left")
plt.xlabel("x")
plt.ylabel("Probability that X < x")
plt.title("CDF of Normal Distribution")

plt.show()
```

B t Distribution Code

```
mu = 2.5;
var = 16;
sigma = sqrt(var);
M = 5;
N = 10000;
v = M - 1;
Z = sigma * randn(M, N) + mu;
sample_sigma = std(Z);

means = sum(Z)/M;
t_vals = ((means - mu) ./ sample_sigma) .* sqrt(M);

dx = 0.25;
boundary = 2;
edges = linspace(-boundary, boundary, 2* boundary / dx);

H = hist(t_vals, edges);

hist_vals = H / (N * dx);
tpdf_vals = tpdf(edges, v);
```

```
figure(1)
subplot(1,2,1)

bar(edges, hist_vals, ”DisplayName”, ”PMF”);
hold on
plot(edges, tpdf_vals, ”DisplayName”, ”tPDF”);
hold off
xlabel('t');
ylabel('Probability Density(t)');
legend
title(”PDF of t distribution”)

subplot(1,2,2)
tcdf_vals = tcdf(edges, v);
plot(edges, tcdf_vals, ”DisplayName”, ”tCDF from bin edges”);
hold on
srt = sort(t_vals);
rnk = [1:N]/N;
cdf1 = tcdf(srt, v);
plot(srt, rnk, ”DisplayName”, ”Rank vs Sorted”);
plot(srt, cdf1, ”DisplayName”, ”CDF vs Sorted”);
legend
xlabel(”t”)
ylabel(”Probabilty T <= t”)
hold off
title(”CDF of t distribution”)
```

C Chi Squared Density Code

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import gamma
import scipy.stats as stats

def getColumn(col_num):
    return np.array([row[col_num] for row in nums])


mean = 2.5
```

```
variance = 16
sigma = variance ** 0.5
M = 5
N = 10000

v = M - 1

nums = sigma * np.random.randn(M, N) + mean

sampleStdDev = np.std(nums)
# print(nums)
chi_values = []

for i in range(N):
    col = getColumn(i)
    sample_variance = np.var(col)
    sample_mean = np.mean(col)
    col_diff2 = np.power((col-sample_mean), 2)
    chi_values.append(np.sum(col_diff2)/variance)

plt.figure(1)
plt.subplot(1,2,1)
H, X = np.histogram(chi_values, bins=50, range=(-5,20))
dx = X[1]-X[0]
plotVals = H / N
plotVals = plotVals / dx
plt.bar(X[:-1],plotVals,width=dx, color="blue", alpha=0.5, label="Experimental PMF")

chipdf = stats.chi2.pdf(X+dx/2,v)
plt.plot(X, chipdf, label="Theoretical PDF v = 4")


chipdf2 = stats.chi2.pdf(X+dx/2,M)
plt.plot(X, chipdf2, label="Theoretical PDF v = 5")
plt.legend(loc="upper right")

plt.ylabel("Probability Density($x^2$)")
plt.xlabel("$x^2$")
```

```python
plt.title("Chi$^2$ Density for sum of differences from the sample mean")

plt.subplot(1,2,2)

chi_values_2 = []

for i in range(N):
    col = getColumn(i)
    sample_variance = np.var(col)
    sample_mean = np.mean(col)
    col_diff2 = np.power((col-mean), 2)
    chi_values_2.append(np.sum(col_diff2)/variance)

H2, X2 = np.histogram(chi_values_2, bins=50, range=(-5,20))
dx2 = X2[1]-X2[0]
plotVals2 = H2 / N
plotVals2 = plotVals2 / dx2
plt.bar(X2[:-1],plotVals2,width=dx2, color="blue", alpha=0.5, label="Experimental PMF")

chipdf_2 = stats.chi2.pdf(X2+dx2/2,v)
chipdf2_2 = stats.chi2.pdf(X2+dx2/2,M)


plt.plot(X2, chipdf_2, label="Theoretical PDF v = 4")
plt.plot(X2, chipdf2_2, label="Theoretical PDF v = 5")
plt.legend(loc="upper right")
plt.ylabel("Probability Density($x^2$)")
plt.xlabel("$x^2$")
plt.title("Chi$^2$ Density for sum of differences from the true mean")
plt.show()
```