ECEN321 : Hypothesis Testing
Lab 4 Submission

Daniel Eisen : 300447549

June 8, 2020

1 Introduction

This lab cover the creation of a particular random variables (in this case Poisson), and the evaluation of them via Null Hypothesis; in this case utilising the χ^2 test.

2 Theory

Hypothesis Testing

We cover the p-value approach:

- Specify the null and alternative hypotheses. In this case the null is that the observed does not match the expected
- Using the sample data and assuming the null hypothesis is true, calculate the value of the test statistic. The χ^2 test statistic is:

  $$\sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

  Where O are the observed values from data and E are the expected theoretical.
- Using the known χ^2 distribution of the test statistic, the p-value is computed:"If the null hypothesis is true, the p-value will be higher.

3 Results

A python script was implemented to generate a Poisson distributed random variable from a random uniform disruption and then evaluated using the χ^2 test.

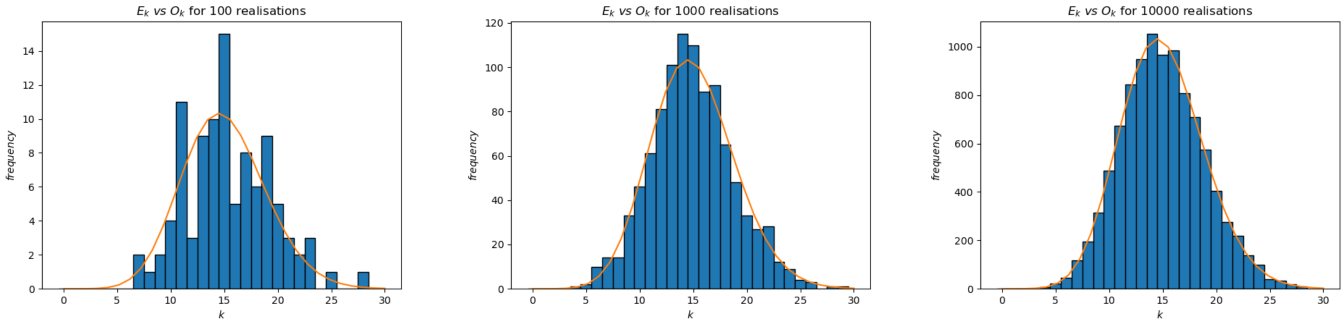

Figure 1.

Figure 1 shows three runs with differing number of realisations (as histograms), overlain with theoretical distribution. Showing the general trend of greater fit for increased N.

```
test statistic: 9.339585756506985
critical-value: 0.0002010146090515683
       p-value: 0.9997989853909485
```

```
test statistic: 43.0648485932466
critical-value: 0.955095636437383
       p-value: 0.044904363562616956
```

```
test statistic: 207.78301082362026
critical-value: 1.0
       p-value: 0.0
```

Figure 2.

Figure 2 shows the results of the χ^2 test on the same differing values of N, 100, 1000, 10000. When experimenting, I found that a N value of 2000 gave consistent results of 99% confidence with very little outliers, at 10000 it was a constant 100%.
When N become around 1000-1400 the critical value drops below 0.99 half the time.

Appendix

Part 1

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import factorial
from scipy.stats import chi2

Lambda = 3  # arrival rate
M = 100  # N of arrivals
N = 10000  # N of experiments
t = 5  # upper bounds of time interval
max_k = (2 * Lambda * t)  # upper possible k values

# Generate exp distributed inter-arrival times of events from uniform dist
inter_arrival_times = -np.log(np.random.rand(N, M)) / Lambda
# cumulatively sum to get vectors of arrival times
arrival_times = np.cumsum(inter_arrival_times, axis=1)
# Count the number of arrivals with a specified time range [0,t] <- is Poisson
observed = np.sum(arrival_times <= t, axis=1)

# vector of possible K values
K = np.linspace(0, max_k, max_k)
# theoretical arrival count distribution
expected = N * ((np.exp(-Lambda * t) * np.power(Lambda * t, K)) / factorial(K))

#  Comparision between Observed and Expected arrivals
plt.figure()
#  plot and retrieve histogram bins of observed
observed, _, _ = plt.hist(observed, bins=np.linspace(0, max_k, max_k + 1) - 0.5, edgecolor="
                                                black")
plt.plot(K, expected)
plt.title("$E_k$ $vs$ $O_k$ for {} realisations".format(N))
plt.xlabel("$k$")
plt.ylabel("$frequency$")
plt.show()

# Correct expected outer values to be >= 5, (and match transform to observed)
for k in range(expected.size):
    if expected[k] >= 5:
        expected[k] = np.sum(expected[:k + 1])
        observed[k] = np.sum(observed[:k + 1])
        expected = expected[k:]
        observed = observed[k:]
        break
for k in reversed(range(expected.size)):
    if expected[k] >= 5:
        expected[k] = np.sum(expected[k:])
        observed[k] = np.sum(observed[k:])
        expected = expected[:k + 1]
        observed = observed[:k + 1]
        break

chi2_stat = np.sum(np.power((observed - expected), 2) / expected)
critical = chi2.cdf(chi2_stat, max_k - 1)
print("test statistic: {}"
      "\ncritical-value: {}"
      "\n       p-value: {}"
      .format(chi2_stat, critical, 1 - critical))
```