Computational Statistical Experiments:

STAT 218 - 08S2 (C) Student Projects Report UCDMS 2009/5

All the projects were supervised by Raazesh Sainudiin while he coordinated the second-year course called STAT 218 - 08S2 (C) during Semester Two of 2008 (14/07/2008-12/11/2008) at the Department of Mathematics and Statistics,

University of Canterbury,

Private Bag 4800,

Christchurch, New Zealand.

Contents

Chapter 1

Thumbtack Dropping

Christopher McCracken and Tracey Wood

Introduction

A standard bronze metal thumbtack is symmetrical in shape but this does not necessarily mean that it will land with the point up or the point down (on its side) half of the time. We want to find out whether different surfaces will affect the outcome. To do this we will assume that $C_1, C_2, \ldots, C_n \overset{IID}{\sim} Bernoulli(p_c^*)$ and $W_1, W_2, \ldots, W_n \overset{IID}{\sim} Bernoulli(p_w^*)$ where $Bernoulli(p_c^*)$ is the distribution function of carpet trials and $Bernoulli(p_w^*)$ is the distribution function of the wooden floor trials.

From this we get our hypothesis:

$$H_0 : p_c^* = p_w^*$$

$$H_1 : p_c^* \neq p_w^*$$

Materials & Methods

We want to test the hypothesis $H_0 : p_c^* = p_w^*$ which says that the Bernoulli distribution functions of carpet and wooden trials are the same.

To do this we will drop a bronze metal thumbtack onto a carpeted area and a polished wood floor 400 times from a height of $50cm$. We will hold the pointed end of the thumbtack and record which way the thumbtack lands. Each time we will be careful to drop the thumbtack in the most identical way possible.

Statistical Methodology

Our sample space may be thought of as $\Omega = \{pointup, pointdown\}$. We use a $Bernoulli(p_c^*)$ RV C with $C(pointup) = 1$, $C(pointdown) = 0$ and $P(C = 1) = p_c$ to measure the outcomes of our 'Thumbtack dropping experiment' on the carpet. We have a similar $Bernoulli(p_w^*)$ RV W with $W(pointup) = 1$, $W(pointdown) = 0$ and $P(W = 1) = p_w$ to measure the outcomes of our 'Thumbtack dropping experiment' on the wooden floor. Therefore, our data for the two set of trials will be vectors of 400 0's and 1's denoted by $(c_1, c_2, \ldots, c_{400})$ and $(w_1, w_2, \ldots, w_{400})$.

Using each data vector of 400 trials we will calculate $\widehat{p_c}$ and $\widehat{p_w}$, the Maximum Likelihood Estimates (MLEs) of p_c^* and p_w^* as the sample means. Thus, $\widehat{p_c} = \frac{1}{400}\sum_{i=1}^{400} c_i$ and $\widehat{p_w} = \frac{1}{400}\sum_{i=1}^{400} w_i$. From this MLE we will calculate a 95% confidence interval.

$$\widehat{p_c} \pm z_{\frac{\alpha}{2}} \hat{se}_c$$

$$\widehat{p_w} \pm z_{\frac{\alpha}{2}} \hat{se}_w$$

The standard error, $(\hat{se}_n)$, is calculated as:

$$\hat{se}_c = \sqrt{(\widehat{p_c} \times (1 - \widehat{p_c}))/400}$$

$$\hat{se}_w = \sqrt{(\widehat{p_w} \times (1 - \widehat{p_w}))/400}$$

Permutation Test

The permutation test is designed to determine whether the observed difference between the sample means is large enough to reject the null hypothesis, $H_0 : p_c^* = p_w^*$, that the two samples are from the same Bernoulli distribution. Firstly, the difference in means between the two collected samples is calculated, $(T(obs))$. Then the observations of the carpet trials **and** wooden trials are combined to form a new array. From this new array, 400 observations are sampled at random from it without replacement. The sample mean for these observations is computed and the sample mean for the remaining 400 observations is also computed, and the difference between the resulting sample means is recorded. This process is repeated n times (e.g. $10,000$) until a reliable estimation of the distribution is reached. In this case, the purpose of the test is to try to reject the null hypothesis. The final p-value obtained will allow us to interpret the strength of the evidence against the null hypothesis.

p-value range	Evidence
(0,0.01]	Very strong evidence against H_0
(0.01,0.05]	Strong evidence against H_0
(0.05,0.1]	Weak evidence against H_0
(0.1,1)	Little or no evidence against H_0

Results

From our data we worked out two MLE's. One for the carpet trials and One for the wooden trials. *See Figure* 1

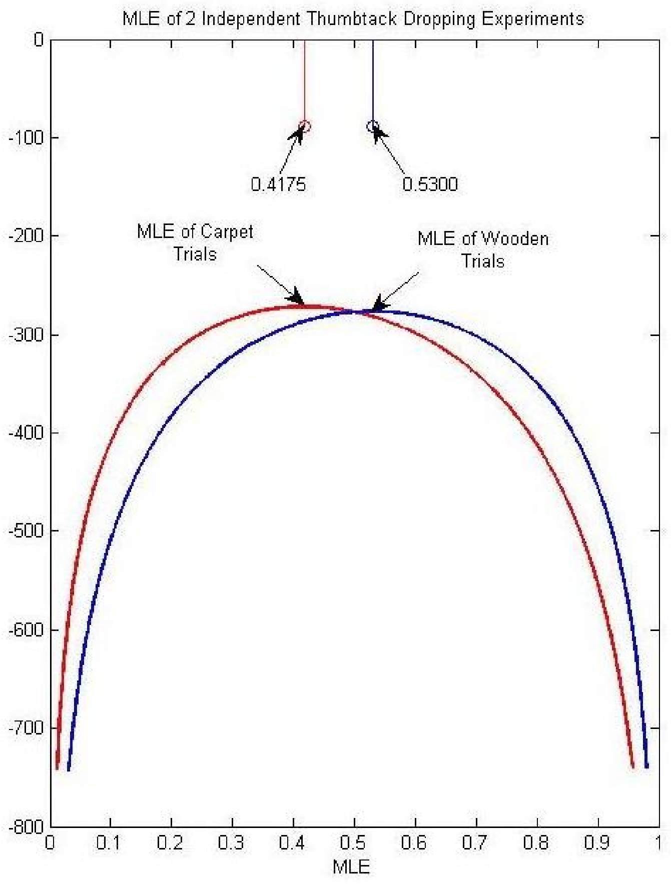


Figure 1.1: MLE's

Using the MLE's, we then created a 95% confidence interval. The intervals were:

$$\widehat{p_c} \pm z_{\frac{\alpha}{2}} \hat{se}_c$$

$$0.4175 \pm 1.96 \times 0.0247$$

$$[0.3691, 0.4659]$$

And

$$\widehat{p_w} \pm z_{\frac{\alpha}{2}} \hat{se}_w$$

$$0.5300 \pm 1.96 \times 0.0250$$

$$[0.4810, 0.5790]$$

The permutation test returned a p-value of $P = 0.0013$

Conclusion

As these confidence intervals do not overlap we can **reject** $H_0 : p_c^* = p_w^*$. If the two samples were from the same Bernoulli distribution, the confidence intervals would overlap or be the same. Further evidence to reject the null hypothesis H_0, is the p-value of 0.0013. This indicates very strong evidence against H_0 as shown in the chart in section 0.1, permutation test.

Author Contributions

Chris and Tracey were both involved in the collection and processing of data.

References

We have used the notes supplied for the Stat218 course to assist us with formulae and code for this assignment. They can be located on the course website.

Appendix A

The m-file used to calculate the p-value, to calculate and graph the MLE of the wood floor and carpet trials is:

```
% To simulate n thumbtack drops on a Wooden Floor, set theta=probability of point up and n
% Then draw n IID samples from Bernoulli(theta) RV
% theta=0.5; n=400; x=floor(rand(1,n) + theta);

x=[1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 0 ...
    1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0 0 0 0 ...
    0 0 1 0 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 1 0 1 0 1 1 0 0 ...
    0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 ...
    1 0 0 1 1 1 0 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 ...
    0 1 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 ...
    1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 1 0 ...
    1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 ...
    1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1 0 ...
    1 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 1 1]; n=length(x);
t = sum(x); % statistic t is the sum of the x_i values
% display the outcomes and their sum
display(x)
display(t)

% Analyticaly MLE is t/n
MLE=t/n
% l is the log-likelihood of data x as a function of parameter theta
l=@(theta)log(theta ^ t * (1-theta)^(n-t));
ThetaS=[0:0.001:1]; % sample some values for theta

% plot the log-likelihood function and MLE in two scales
subplot(1,2,1);
plot(ThetaS,arrayfun(l,ThetaS),'b','LineWidth',2);
hold on; stem([MLE],[-89],'b-'); % plot MLE as a stem plot
xlabel('MLE','FontSize',10); % label x-axis
title('MLE of 2 Independent Thumbtack Dropping Experiments','FontSize',10);


% Now we will find the MLE by finding the minimiser or argmin of -l
% negative log-likelihood function of parameter theta
negl=@(theta)-(log(theta ^ t * (1-theta)^(n-t)));

NumericalMLE = fminbnd(negl,0,1,optimset('Display','iter'))
% This data was collected by Chris McCracken and Tracey Wood
% Results of pin dropping experiment on carpet
Carpet=[0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 1 1 0 0 0 0 1 ...
    1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 1 ...
    0 0 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 1 ...
    0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 ...
    1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 ...
    0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 1 0 ...
    0 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 ...
    0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 0 ...
    0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 ...
    0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 1 1 1 0 0 0];
% Results of pin dropping experiment on wooden floor
```

```
Wood=[1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 0 ...
    1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0 0 0 0 ...
    0 0 1 0 1 0 1 0 0 1 0 0 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 1 0 1 0 1 1 0 0 ...
    0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 ...
    1 0 0 1 1 1 0 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 ...
    0 1 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 ...
    1 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 1 0 ...
    1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 ...
    1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1 0 ...
    1 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 1 1];
Tobs=abs(mean(Carpet)-mean(Wood));% observed test statistic
nCarpet=length(Carpet); % sample size of the Carpet data
nWood=length(Wood); % sample size of the Wood data
ntotal=nCarpet+nWood; % sample size of the pooled data
total=[Carpet Wood]; % observed data -- ordered: Carpet data followed by Wood data
B=10000; % number of bootstrap replicates
TB=zeros(1,B); % initialise a vector of zeros for the bootstrapped test statistics
ApproxPValue=0; % initialise an accumulator for approximate p-value
for b=1:B % eneter the bootsrap replication loop
    % use MATLAB's randperm function to get a random permutation of indices{1,2,...,ntotal}
    PermutedIndices=randperm(ntotal);
    % use the first nCarpet of the PermutedIndices to get the bootstrapped Carpet data
    BCarpet=total(PermutedIndices(1:nCarpet));
    % use the last nWood of the PermutedIndices to get the bootstrapped Wood data
    BWood=total(PermutedIndices(nCarpet+1:ntotal));
    TB(b) = abs(mean(BCarpet)-mean(BWood)); % compute the test statistic for the bootstrapped data
    if(TB(b)>Tobs) % increment the ApproxPValue accumulator by 1/B if bootstrapped value > Tobs
        ApproxPValue=ApproxPValue+(1/B);
    end
end
ApproxPValue % report the Approximate p-value
% To simulate n thumbtack drops on a Carpeted Floor, set theta=probability of point up and n
% Then draw n IID samples from Bernoulli(theta) RV
% theta=0.5; n=400; x=floor(rand(1,n) + theta);
x=[0 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 1 1 0 0 0 0 1 ...
    1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 1 ...
    0 0 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 1 ...
    0 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 ...
    1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 1 0 1 ...
    0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 1 0 ...
    0 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 ...
    0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 0 ...
    0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 ...
    0 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 0 1 1 1 0 0 0]; n=length(x);
t = sum(x); % statistic t is the sum of the x_i values
% display the outcomes and their sum
display(x)
display(t)

% Analyticaly MLE is t/n
MLE=t/n
```

```
% l is the log-likelihood of data x as a function of parameter theta
l=@(theta)log(theta ^ t * (1-theta)^(n-t));
ThetaS=[0:0.001:1]; % sample some values for theta

% plot the log-likelihood function and MLE in two scales
subplot(1,2,1);
plot(ThetaS,arrayfun(l,ThetaS),'r','LineWidth',2);
hold on; stem([MLE],[-89],'r-'); % plot MLE as a stem plot
xlabel('MLE','FontSize',10); % label x-axis
title('MLE of 2 Independent Thumbtack Dropping Experiments','FontSize',10);


% Now we will find the MLE by finding the minimiser or argmin of -l
% negative log-likelihood function of parameter theta
negl=@(theta)-(log(theta ^ t * (1-theta)^(n-t)));

NumericalMLE = fminbnd(negl,0,1,optimset('Display','iter'))
```

Chapter 2

Magnitude and Depth of Earthquakes in New Zealand, 2008

Fiona Morrison and Amy Smit

Abstract

This report will investigate whether there is a relationship between the magnitude (size of the quake in Richter Scale terms) of earthquakes and their focal depth (the distance in metres below sea level to the quake's epicentre) along the Australia-Pacific Plate boundary in New Zealand. The non-parametric Bootstrap method and a Confidence Interval will be the statistical methodologies used to investigate this relationship.

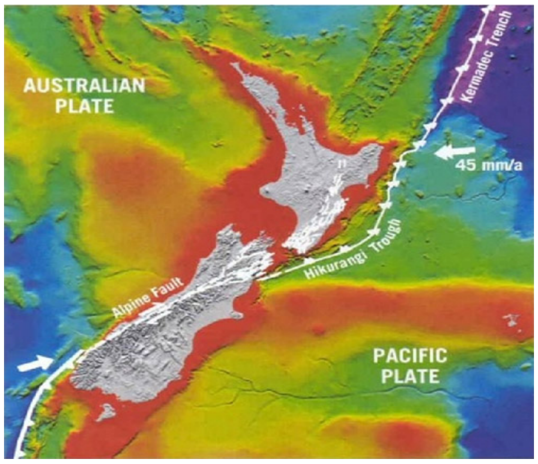


Figure 2.1: The Australia-Pacific faultline that runs through the South Island of New Zealand.

Motivation & Background

New Zealanders are literally living on the edge because the active Pacific-Australian Plate boundary passes through New Zealand and produces earthquakes. Nowhere in New Zealand is immune from the possibilities of damaging earthquakes. A major event almost anywhere in the country would affect the whole society and economy due to its small size. Not since the 1930s and early 1940s — a period in which large shallow earthquakes struck repeatedly — has New Zealand suffered major social disruption or serious economic setback due to geological hazards.

We are interested in investigating whether there is a relationship between the magnitude (size of the quake in Richter Scale terms) of earthquakes and their focal depth (the distance in metres below sea level to the quake's epicentre). The non-parametric Bootstrap method and a Confidence Interval will be the statistical methodologies used to investigate this relationship.

Method

Experiment Process

We searched the `geonet.org.nz` website and retrieved the earthquake magnitude and depth data for January 18 to August 18 of the year 2008. We then input the time, magnitude and depth data into a matrix in MATLAB and assigned a date stamp to each observation. We then simulated 1000 replications of this data using the non-parametric Bootstrap method.

Bootstrap

Bootstrap is one of the first computer-intensive statistical resampling techniques that was proposed in 1979 by statistician Bradley Efron. The bootstrap method is a statistical method for estimating confidence sets of statistics. If we assume that the samples are independently and identically distributed then we can construct the empirical DF and produce any number of samples from it. This is obtained by random sampling with replacement from the original dataset. Bootstrapping can also be used for testing a hypothesis. It is often used as an inference method based on nonparametric assumptions when parametric inference is impossible or requires very complicated formulas for the calculation of standard errors.

Data

We selected two types of data from `geonet.org.nz`: y = Magnitude — how big the earthquakes are on the Richter Scale between 1 (small) and 12 (large) — and x = Depth of the earthquakes – in metres below sea level. We selected this data from the 18th of January to 18th August in the year 2008. We assume that our pair of random variables are from a nonparametric distribution. Our paired data set is $((x_1, y_1), (x_2, y_2), \ldots, (x_{6217}, y_{6217}))$.

Statistical Methodology

$$H_0 : \theta^* = 0 \qquad H_1 : \theta^* \neq 0$$

We are testing the null hypothesis H_0 that there is no linear relationship between depth and magnitude or that $\theta^* = 0$, i.e. the correlation coefficient of our bivariate data is zero. To do this we will produce $1,000$ replications of our data set using the non-parametric Bootstrap method, and estimate the sample correlation from the samples of each replicated data set. From these sample correlation coefficients we will form a 95% confidence interval to test whether there is no relationship. If our confidence interval contains 0 then we would fail to reject the null hypothesis that there is no linear relationship between the magnitude and depth of earthquakes. If our confidence interval does not contain 0, then we would reject the null hypothesis and conclude that there is a linear relationship.

Results

$$\text{Confidence Interval} = \text{Sample Correlation} \pm 1.96\sqrt{\text{Standard Error}}$$

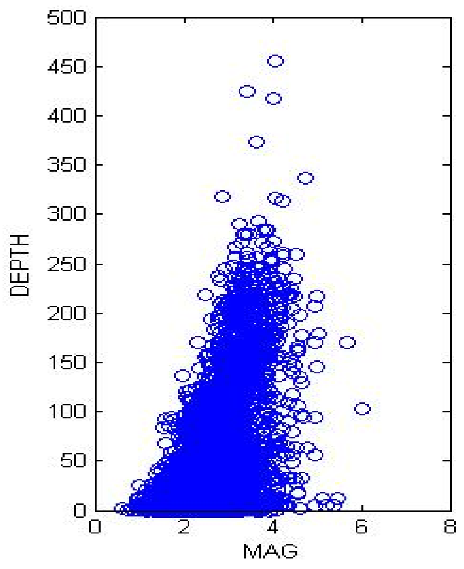
After using the Bootstrap method we constructed a 95% confidence interval of $(0.5088, 0.5471)$. Our confidence interval does not contain 0 so we reject the null hypothesis.

Conclusion

We conclude that there is very strong evidence against the lack of a linear relationship between the magnitude and depth of earthquakes recorded along the Pacific-Australian Plate boundary within New Zealand during the first eight months of 2008.

Author Contributions

Fiona - Gathered the data results, modified MATLAB code, constructed report, constructed presentation, conducted spell/grammar check.
Amy - Modified Matlab code to analyse and plot data, constructed report, and constructed presentation.



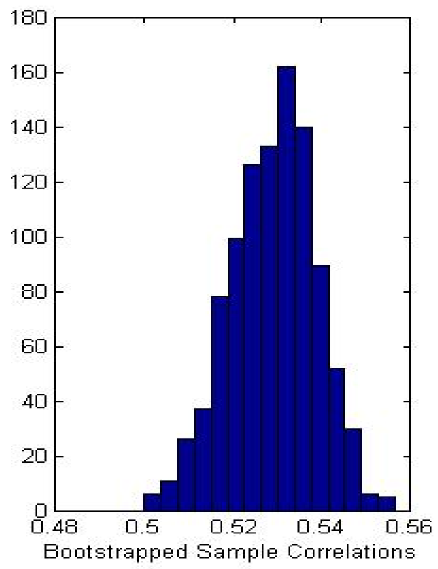


Figure 2.2: Scatter plot of the observed data. Figure 1.3: Histogram showing the sample correlations from our bootstrapped samples.

References

`http://www.geonet.org.nz`

`http://en.wikipedia.org/wiki/Bootstrapping_(statistics)`

Appendix A

Link to our earthquake data:

`http://www.math.canterbury.ac.nz/~r.sainudiin/courses/STAT218/Supplements/earthquakes.csv`

Chapter 3

Investigation of the gender specific income disparity distribution in New Zealand

Authors: Pi-Fan Yang and Timothy Weusten

Abstract

This project is designed to investigate the gender-specific income disparity for New Zealand in 2001. As money is the driving force of the economy, we tend to feel more secure and safe when we have money. We want to investigate whether there is an imbalance in the socioeconomic structure and how this is distributed. We will be using the bootstrap technique to generate a new set of data to test our null hypothesis which is that men and women have equal incomes.

Introduction

The report will cover the background of the bootstrap technique and look at the methodologies we used to research the income disparity distribution. We will then thoroughly analyze our results and state a conclusion, which we found by attempting this project.

Motivation

The motivation behind this project is to determine the average income disparity distribution between males and females. We will be investigating three areas of interest, each area containing age group sets: after University (20-24 and 25-29), career peak (35-39 and 40-44) and retirement (55-59 and 60-64). We believe that these three age groups will give us key indications of the overall gender-specific average income disparity distribution for our data set and we will be be able to attempt to reject the null hypothesis that there is no gender-specific income disparity in various age groups, including the three age groups of interest.

Visualization

We found our data from the Statistics New Zealand website, using the 2001 Census data (`http://www.stats.govt.nz/NR/rdonlyres/C8762EB8-3132-43BE-87F2-0801C6A81A7D/0/WorkTable23.xls`). We first visualise the gender-specific average income difference in each age group of interest. This visualization looked at the average income of men in a certain age group subtracted from that of women from the same age group. The difference was plotted as a stem plot for each age group (see next three Figures). Therefore, the y axis is a number stating the difference between the average income of a male and that of a female for each age group, in dollars, and the x axis gives the different income categories. The x axis contains thirteen income categories as shown below:

X Axis	Money Caterogry
X1	Loss
X2	Zero Income
X3	1 − −5,000
X4	5, 001 − −10,000
X5	10, 001 − −15,000
X6	15, 001 − −20,000
X7	20, 001 − −25,000
X8	25, 001 − −30,000
X9	30, 001 − −40,000
X10	40, 001 − −50,000
X11	50, 001 − −70,000
X12	70, 001 − −100,000
X13	100, 001 or More

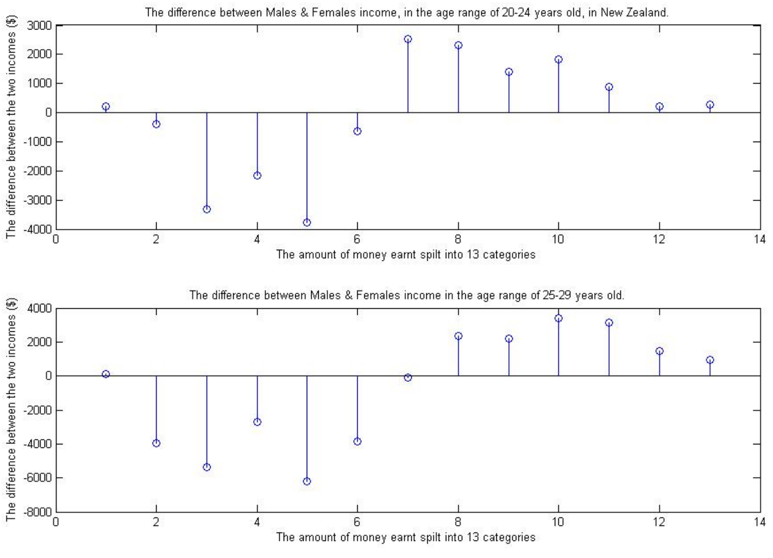


Figure 3.1: The difference in average income between Men and Women in certain income categories

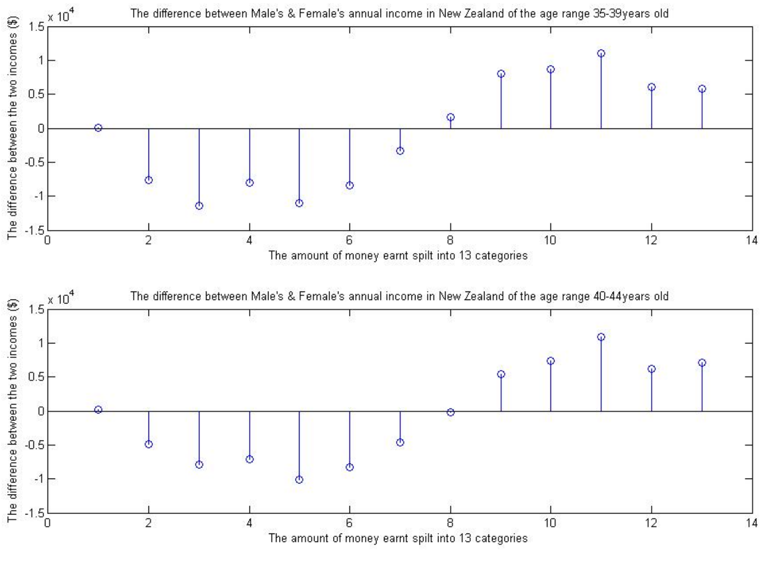


Figure 3.2: The difference in average income between Men and Women in certain income categories

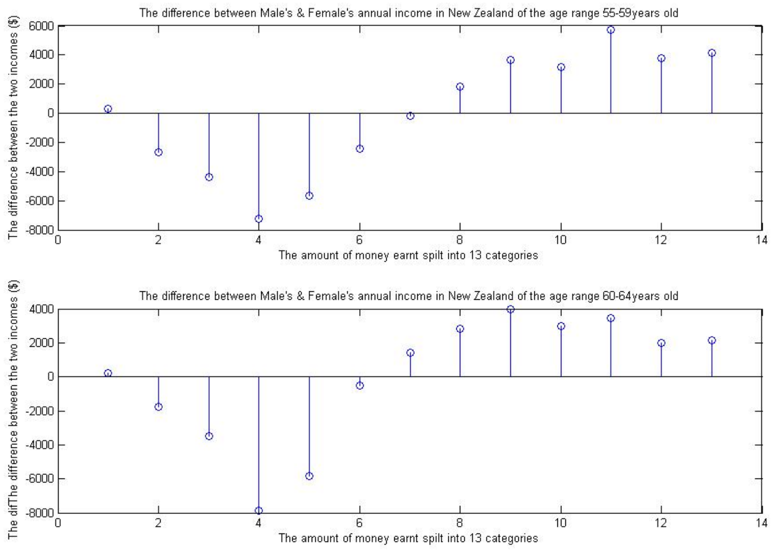


Figure 3.3: The difference in average income between Men and Women in certain income categories

Statistical Methodology

Bootstrap

The bootstrap technique is a re-sampling method that can be used to estimate the variance of an estimator, such as the sample variance. The observed data is assumed to be independent and identically distributed from non-parametric population. The bootstrap method creates a new set of B bootstrapped data sets by random sampling from our original data. We can use the bootstrapped data sets to generate the confidence interval for our statistic and the critical region of the test statistic of our hypothesis test.

Hypothesis test

$$H_0 : \mu_M = \mu_F \qquad H_1 : \mu_M \neq \mu_F$$

The null hypothesis, H_0, states that the average income for men is equal to women in the same age group. The alternative hypothesis states that the average income for men is not equal to women in the same age group.

We are using this hypothesis test for each of the age groups of 20-24, 25-29, 35-39, 40-44, 55-59 and 60-64. Then we will create confidence intervals to either reject or fail to reject H_0 via the Wald test, depending on whether our interval contains zero or not.

Let the average earning per age category be denoted by the vector D:

$$D = (0, 2500, 7500, 12500, 17500, 22500, 27500, 35000, 45000, 60000, 85000, 100000)\ .$$

Suppose we are interested in a random sample of size $n = 1,000$ males and $n = 1,000$ females from the 2001 New Zealand population and their income difference statistic T:

$$T = \sum(\theta_i^m) * D_i - \sum(\theta_i^f) * D_i$$

for our hypothesis test. The probability of sampling a male and a female from the income category i is denoted by θ_i^m and θ_i^f, respectively. We may obtain the sampling distribution of the test statistic to construct the 95% confidence interval for a Wald test from the bootstrapped statistics, as follows:

$$t^\bullet = \sum(\theta_i^{\bullet,m}) * D_i - \sum(\theta_i^{\bullet,f}) * D_i$$

The sampling distribution of T was generated from $B = 1000$ times bootstrapped data sets and put into a histogram, as shown in the Results section. We used this method to determine our 95% confidence interval for the six different age groups (20-24, 25-29, 35-39, 40-44, 55-59 and 60-64) to test the gender-specific disparity, if any, in New Zealand income for 2001. We generated two bootstrap samples with, $B = 100$ and $B = 1000$ samples, to compare the difference in the confidence intervals and to see whether the interval contains the value zero.

Results

$$95\% \text{ Confidence Interval} = \text{Mean} \pm 1.96\sqrt{\text{std error}}$$

$$\text{std error} = \text{std dev}/\sqrt{B}$$

In the Tables below, B is the number of bootstrapped samples we generated. As the sample size increased we found that our mean has decreased, theoretically getting closer to our true mean. For our 95% confidence interval we found that our standard deviation decreased which make our upper and lower bound more restricting. Therefore we can be 95% confident that our true mean is bound in our confidence interval.

We have done the bootstrapped test statistic for following age groups: 20-24, 25-29, 35-39, 40-44, 55-59 and 60-64. This was generated $B = 1000$ times and put into a histogram, as shown in the following six Figures for the specific age groups.

Age Group 20-24	Mean	Std	95% C.I	Upper bound	Lower bound
B=100	2726.1	627.7353	1.96	2849.1	2603.1
B=1000	2702.8	589.4716	1.96	2739.3	2666.3

Table 3.1: In the confidence interval, for age group 20-24, it had a range of $[2603.1, 2849.1]$, $[2666.3, 2739.3]$. We found both intervals to not contain the value zero.

Age Group 25-29	Mean	Std	95% C.I	Upper bound	Lower bound
B=100	6963.3	861.6387	1.96	7132.1	6794.4
B=1000	6927.1	818.2747	1.96	6977.9	6876.4

Table 3.2: In the confidence interval, for age group 25-29, it had a range of $[6794.4, 7132.1]$, $[6876.4, 6977.9]$. We found both intervals to not contain the value zero.

Age Group 34-39	Mean	Std	95% C.I	Upper bound	Lower bound
B=100	17218	1110.9	1.96	17436	17000
B=1000	17166	1046.3	1.96	17231	17101

Table 3.3: In the confidence interval, for age group 35-39, it had a range of $[17000, 17436]$, $[17101, 17231]$. We found both intervals to not contain the value zero.

Age Group 40-44	Mean	Std	95% C.I	Upper bound	Lower bound
B=100	16824	1142.5	1.96	17048	16600
B=1000	16767	1068.9	1.96	16834	16701

Table 3.4: In the confidence interval, for age group 40-44, it had a range of $[16600, 17048]$, $[16701, 16834]$. We found both intervals to not contain the value zero.

Age Group 55-59	Mean	Std	95% C.I	Upper bound	Lower bound
B=100	15077	1095.3	1.96	15291	14862
B=1000	15011	1040.1	1.96	15076	14947

Table 3.5: In the confidence interval, for age group 55-59, it had a range of $[14862, 15291]$, $[14947, 15076]$. We found both intervals to not contain the value zero.

Age Group 60-64	Mean	Std	95% C.I	Upper bound	Lower bound
B=100	12183	990.0177	1.96	12377	11989
B=1000	12149	917.1082	1.96	12206	12093

Table 3.6: In the confidence interval, for age group 60-64, it had a range of $[11989, 12377]$, $[12093, 12206]$. We found both intervals to not contain the value zero.

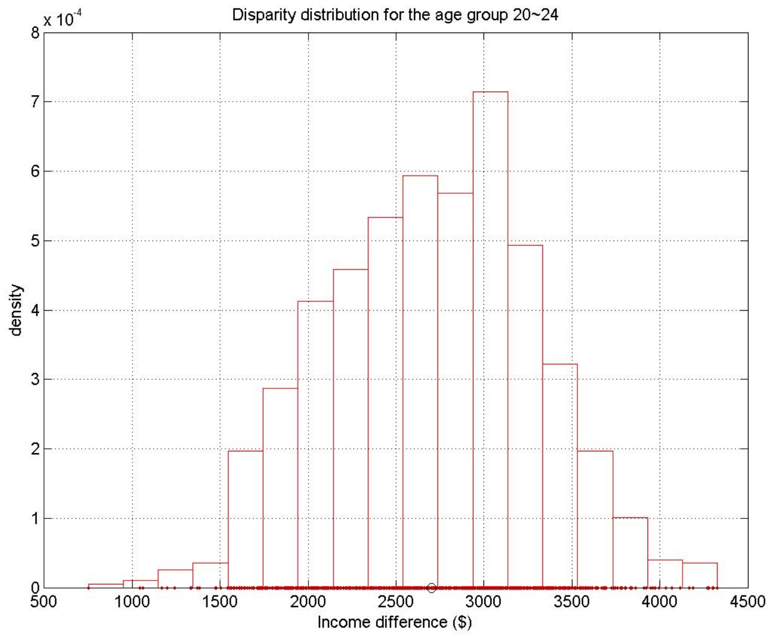


Figure 3.4: The bootstrapped histogram of test statistic T shows the difference in income. For age group 20-24, men are earning 2701.0 dollars more than their female counterparts, this is from our observed data indicated by the black circle.

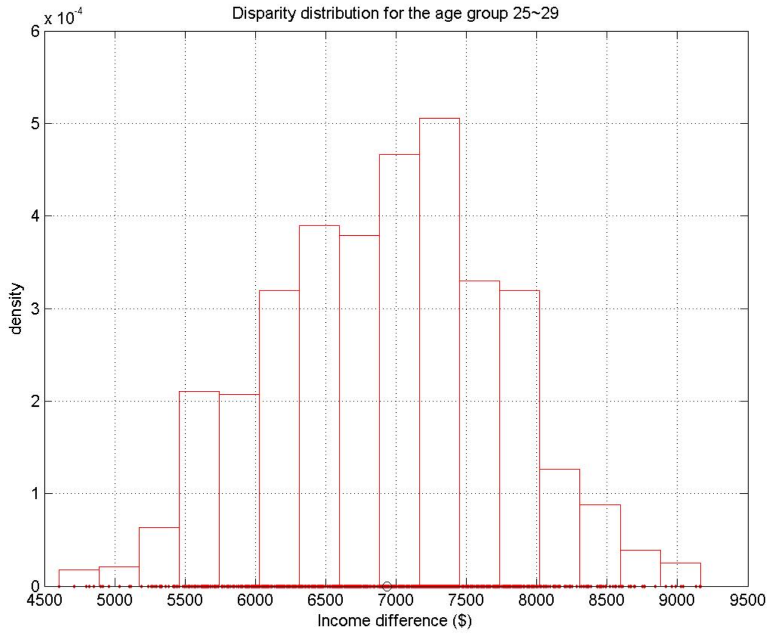


Figure 3.5: The bootstrapped histogram of test statistic T shows the difference in income. For age group 25-29, men are earning 6933.9 dollars more than their female counterparts, this is from our observed data indicated by the black circle.

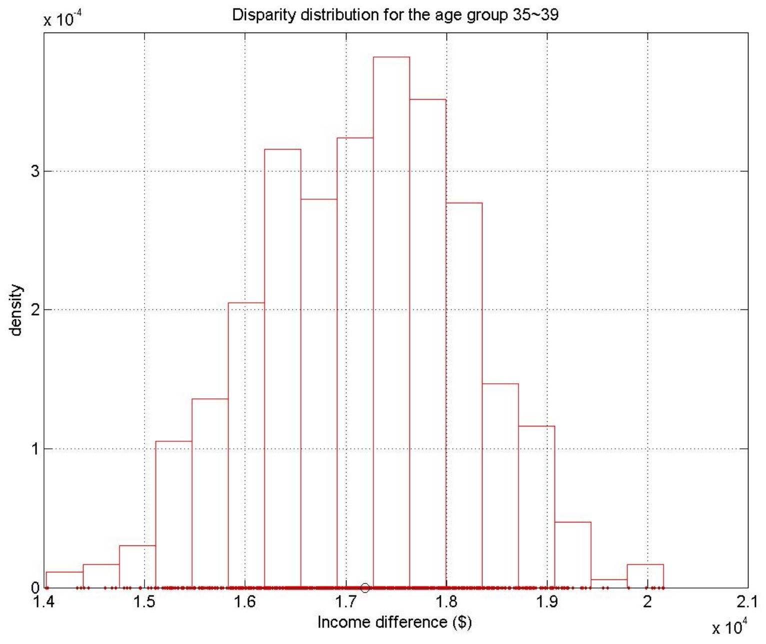


Figure 3.6: The bootstrapped histogram of test statistic T shows the difference in income. For age group 35-39, men are earning 17190 dollars more than their female counterparts, this is from our observed data indicated by the black circle.

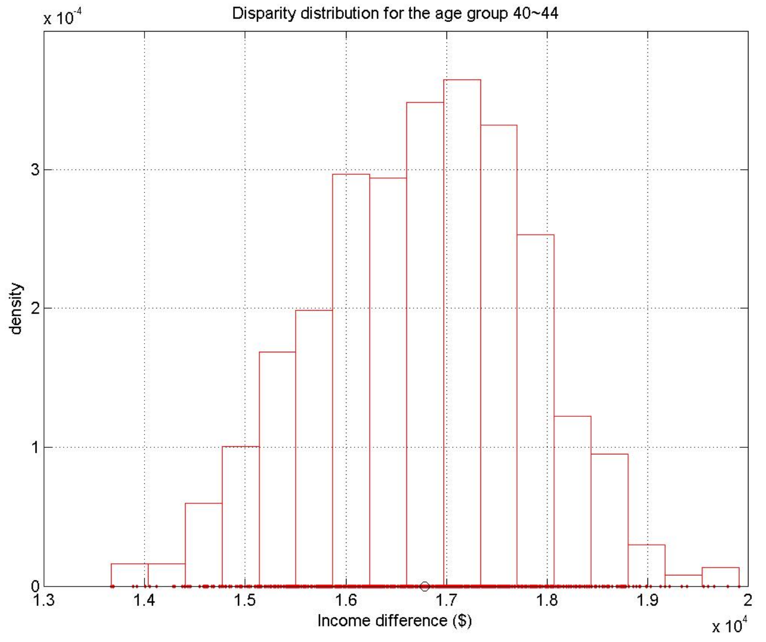


Figure 3.7: The bootstrapped histogram of test statistic T shows the difference in income. For age group 40-44, men are earning 16786 dollars more than their female counterparts, this is from our observed data indicated by the black circle.

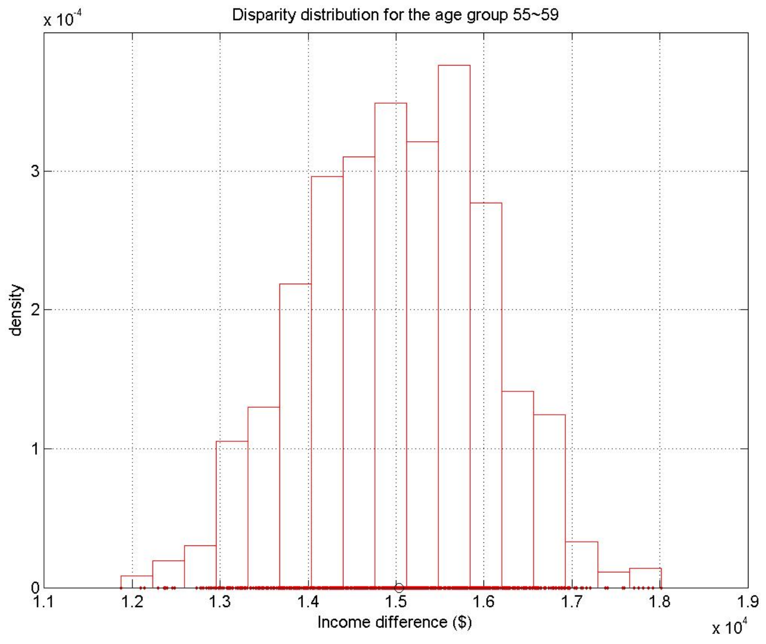


Figure 3.8: The bootstrapped histogram of test statistic T shows the difference in income. For age group 55-59, men are earning 15032 dollars more than their female counterparts, this is from our observed data indicated by the black circle.

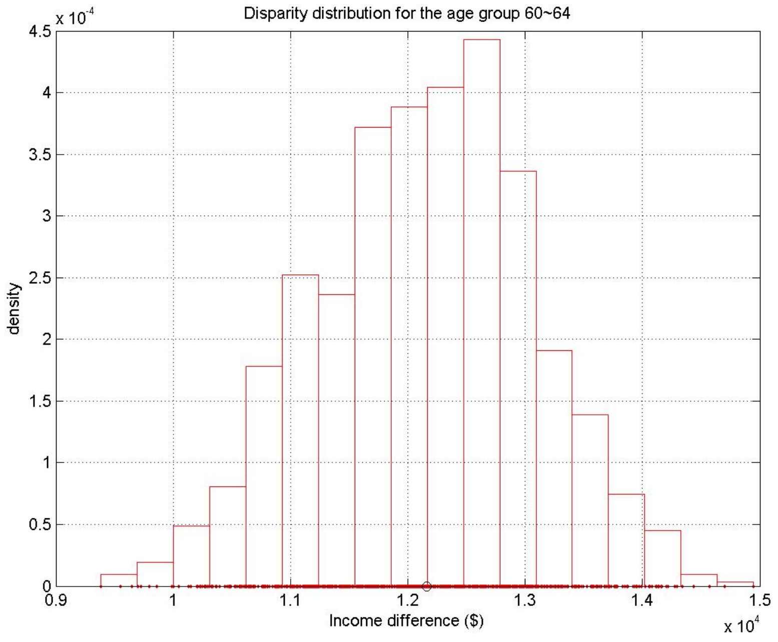


Figure 3.9: The bootstrapped histogram of test statistic T shows the difference in income. For age group 60-64, men are earning 12160 dollars more than their female counterparts, this is from our observed data indicated by the black circle.

Conclusion

We reject the null hypothesis that there is no gender-specific average income disparity with 95% confidence, because the value zero is not contained in the range of our intervals based on a Wald test using the bootstrap. Therefore we can conclude that the average income for men is different from the average income for women in the same age group. We rejected the null hypothesis for all six of the following age groups: 20-24, 25-29, 35-39, 40-44, 55-59 and 60-64.

Potential Modification

For our project, a modification we could have done is to use more recent data, for example the 2006 census data, and then comparisons could have been made between 2001 and 2006 data. We could also have analysed and compared the men with different age groups to see if age was influencing the disparity in income.

Author Contributions

Pi-Fan Yang - Found the data, constructed confidence interval tables and Matlab coding for C.I, wrote up background, results and conclusion.
Timothy Weusten - Wrote up abstract, introduction, visualizations, motivation, methodology and potential modification.
Both - Conducted spell/grammar check, constructed report and presentation.

References

`http://www.stats.govt.nz/NR/rdonlyres/C8762EB8-3132-43BE-87F2-0801C6A81A7D/0/WorkTable23.xls`

Appendix A

```
%read in data
DM=dlmread('NZINCOME.csv');


k=12; % number of categories to achieve
n=1000; % number of males or females
rand('twister',11154238); % initialise the fundamental sampler
D=[0 2500 7500 12500 17500 22500 27500 35000 45000 60000 85000 100000];
% average earning per category
```

```
Columns=[2 3 5 6 9 10];
% this looks at the specific age group 2=20-24, 3=25-29, 5=35-39, 6=40-44, 9=55-59 and 10=60-64.
for j=1:length(Columns)
    mcs=DM(2:13,Columns(j))';% counts of males in column 10 (s age group)
    mfs=mcs/sum(mcs) % frequencies -- MLE under multinomial model
    fcs=DM(15:26,Columns(j))';% counts of males in column 10 (s age group)
    ffs=fcs/sum(fcs) % frequencies -- MLE under multinomial model
    tobss=sum(mfs .* D) - sum(ffs .* D)
    % Bootstrap for average income disparity m-f....
    C=eye(12)
    B=1000; % numer of bootstrap reps
    t=zeros(1,B);
    for b=1:B % bootstrap reps
        %make data set
        bootDataM=zeros(1,12);
        bootDataF=zeros(1,12);
        for i=1:n
            bootDataM = bootDataM + C(SimdeMoivreOnce(rand(),mfs),:);
            bootDataF = bootDataF + C(SimdeMoivreOnce(rand(),ffs),:);
        end
        %compute stats
        t(b)=sum((bootDataM/n) .* D) - sum((bootDataF/n) .* D);
    end
    histogram(t,1,[min(t), max(t)],'r',2)
    hold on;
    plot(tobss,0,'ko')
end

% This only works correctly when the code  "Column=[]" has one number in it.  This number
% states which age group you are looking at.
bmean = mean(t) % find the mean for the bootstrapped data
bstd = std(t) % find the standard deviation for the bootstrapped data
bci =[bmean+1.96*bstd/(sqrt(n)) bmean-1.96*bstd/(sqrt(n))]
% finds the 95% bootstrapped confidence interval for that  age group.
```

```
IC=dlmread('NZINCOME.csv');
for i=2:3 % This shows column 2(age group 20-24) and column 3(age group 25-29)
subplot(2,1,i-1) % Creates 2 graph in one figure
```

```
stem(IC(1:13,i)-IC(14:26,i))
% this calculates the difference in the specific row and specific column Men-Female.
end
```

```
IC=dlmread('NZINCOME.csv');
for i=5:6 % This shows column 5(age group 35-39) and column 6(age group 40-44)
subplot(2,1,i-4) % Creates 2 graph in one figure
stem(IC(1:13,i)-IC(14:26,i))
% this calculates the difference in the specific row and specific column Men-Female.
end
```

```
IC=dlmread('NZINCOME.csv');
for i=9:10 % This shows column 9(age group 55-59) and column 10(age group 60-64)
subplot(2,1,i-8) % Creates 2 graph in one figure
stem(IC(1:13,i)-IC(14:26,i))
% this calculates the difference in the specific row and specific column Men-Female.
end
```

Chapter 4

Testing the fairness of the New Zealand Lotto draw

Adrian Goode, Flynn Knowles-Barley, and Danielle Rolle

Abstract

This project aims to test the randomness of the balls drawn in the weekly New Zealand Lottery. To test our hypothesis that each and every ball has an equal chance of being drawn, we will be taking the results of the past 1043 draws and running a Pearson's Chi-square test and a Kolmogorov-Smirnov test with the data. Alongside these results, we will be simulating an equal number of draws, with random data drawn from a uniform distribution. This is for comparison with our Chi-square value, and to enable us to run a two-sample K-S test between the two data sets of data.

Introduction & Motivation

Introduction

The New Zealand Lottery is a weekly draw of seven balls from a pool of possible balls numbered one through to forty. Lotto has been running every week since the 1st August 1987, and has paid out more than 3.75 billion dollars in prizes since then. The very nature of the draw implies randomness; all the balls are released into the machine at about the same time, and spun around for a few minutes before the balls are drawn out one at a time, without replacement.

Our project focuses only on the balls drawn, and does not take into account the different types of machines that may have been used over the years. Information on the machines was not readily available, and would have added a whole new level of complexity to our tests.

Motivation

Being students, money is something that always interests us. So doing a project on something that could potentially lead to a very large windfall naturally peaked our curiosity. Realistically, we didn't expect to uncover any unfairness, but it would provide an interesting chance to begin looking into the nature of randomness.

Materials & Methods

Collecting The Data

We collected our data from the New Zealand Lotto website, `http://lotto.nzpages.co.nz`, where the last 1043 draws are stored in an Excel file. To add to this data we simulated another 1043 draws using the uniform distribution.

Experiment Process

To test our hypothesis that Lotto is random and therefore the distribution of the balls is uniform we obtained results from the New Zealand Lotto site. We will use Matlab to compare these results along with an additional 1043 draws simulated from the uniform distribution.

The Chi-square portion of our testing happens in two parts. Firstly, we will simply take the first ball observed for each of the 1043 draws and use the Chi-square formula to test our hypotheses that each ball has an equal probability of being drawn. We will then use Matlab to simulate the first ball drawn for 1043 draws, analyse this data using the Chi-squared test, then compare the two sets of results.

Once this is done, and assuming it passes the test, we can then make some assumptions for the second part of our Chi-squared testing. Because the Lotto balls are drawn without replacement, after the first ball is drawn the probability of each subsequent ball being drawn increases slightly. We can get around this complication by assuming that because each of the initial balls drawn is uniform, each subsequent ball will also be uniform. So now we can simply sum the number of observations of each ball number over the entirety of the 1043 draws, and compare them against each other and the expected value.

The Kolmogorov-Smirnov testing also happens in two parts. First we will perform a simple one-sample K-S test to compare our sample with our estimated probability distribution. This will test the difference between the empirical distribution function of the sample and the cumulative distribution function of our estimated uniform distribution. Then with the help of Matlab we will simulate another 1043 draws of six balls, drawn from a uniform distribution. We will use this new data to perform a two-sample K-S test to see whether or not the samples appear to be drawn from the same distribution.

Statistical Methodology

Chi-squared Test

Firstly we will define the Chi-squared distribution. The Chi-square distribution has one parameter k, a positive integer that specifies the number of degrees of freedom, which in our case is 40, the number of balls we are using, minus 1. The Chi-square distribution is a special case of the gamma distribution. To test our Chi-squared test statistic we compare this to the critical value, this is obtained by taking the alpha value and looking down the Chi-squared table following the degrees of freedom. We calculated the Chi-squared test statistic using the formula:

$$\chi^2 = \sum_{i=0}^{n} \frac{(O_i - E_i)^2}{E_i}$$

We then calculated the Chi-squared test statistic for the simulated data and compared this to our observed data.

Kolmogorov-Smirnov Test

We decided then to do the Kolmogorov-Smirnov test to compare our data to the uniform distribution and the simulated data. The Kolmogorov-Smirnov test is a form of minimum distance estimation and is used as a nonparametric test of equality of one dimensional probability distributions used to compare a sample with a reference probability distribution which in our case is the uniform distribution. This test can also be used to compare two samples. It quantifies a distance between the empirical distribution function f the sample and the cumulative distribution function of the reference distribution, or between the empirical distributions of the two samples. The k-s test gives out 2 test statistics, kn+ which calculates the maximum positive distance of our sample minus the cdf and kn- calculates the maximum positive distance of the cdf minus our sample. We used the single sample test to test our data with the uniform cdf using the formula:

$$K_n^+ = \sqrt{n} \max_{1 \leqq j \leqq n} (\frac{j}{n} - F(X_j)$$
$$K_n^- = \sqrt{n} \max_{1 \leqq j \leqq n} (F(X_j) - \frac{(j-1)}{n})$$

We then used the 2 sample test to compare our data to the simulated data using the formula:

$$D_{n,n'} = \max_x |F_n(X) - F_{n'}(X)|$$

Results

Chi-squared Critical Region The critical region for our Chi-squared tests, obtained from the Chi-squared distribution tables, with degrees of freedom being 39 and an alpha value of 0.05 is 54.57. We will use this value to determine whether or not we can reject the null hypothesis by comparing it with our calculated Chi-squared value obtained from our data.

Chi-squared results for the first balls:

	Data Values	Observed Count	Expected Count	$(O-E)^2/E$
	1	29	26.075	0.328116
	2	25	26.075	0.044319
	3	30	26.075	0.590819
	...	...	...	...
	38	28	26.0755	0.142114
	39	22	26.075	0.636840
	40	35	26.075	3.054865
Total		1043	1043	
Chi Test Result				25.494726

Chi-squared results for the simulation of the first balls over 1043 draws:

	Data Values	Observed Count	Expected Count	$(O-E)^2/E$
	1	25	26.075	0.044319
	2	25	26.075	0.044319
	3	28	26.075	0.142114
	...	...	...	...
	38	21	26.0755	0.987752
	39	25	26.075	0.044319
	40	24	26.075	0.165125
Total		1043	1043	
Chi Test Result				33.54842

Approximate p-value:

0.9523

All six balls over 1043 draws Chi-squared results:

	Data Values	Observed Count	Expected Count	$(O-E)^2/E$
	1	179	156.45	3.250255
	2	153	156.45	0.076078
	3	159	156.45	0.041562
	...	...	...	...
	38	165	156.45	0.467257
	39	172	156.45	1.545557
	40	141	156.45	1.525743
Total		6258	6258	
Chi Test Result				22.89485

One sided Kolmogorov-Smirnov Test Result:

0.0554

Two sided Kolmogovor-Smirnov Test Result:

0.0629

Conclusion

We found using the Chi-squared test that the calculated test statistic for our data was well below the critical value at $\alpha = 0.05$, degrees of freedom = 39, which gives us no evidence to reject the null hypothesis.

The Kolmogorov-Smirnov test fell below the critical value in both the one sided and two sided tests, showing that it is significantly close to zero when $\alpha = 0.05$ and with degrees of freedom = 40, which gives evidence to show that the distribution is extremely close to that of the uniform distribution.

In relation to our statistical question, this means that there is insufficient evidence to suggest that our data is not from $deMoivre(\frac{1}{40}, \ldots, \frac{1}{40})$, and so we can conclude that the Lotto draws are fair, with every ball having an equal chance of being drawn.

Author Contributions

Adrian Goode Conducted Matlab statistical tests, analysis of data, and contributed to written report

Flynn Knowles-Barley and Danielle Rolle Wrote and compiled written report and analysis of data

Appendices

I

k.m

```
function x = k(numbers, sim);
%
% This function calulates the kn+ and kn- test statistics for the input
% data. This function can take one or two paramaters. If there are 2
% parameters the function calculates the k-s test statistics using the one
% sample test and then calculates the two sample test using both inputs.
%
% File Dates : Created  12/10/08  Modified  16/10/08
% Author(s)  : Adrian Goode
%
% Call Syntax:  x = k(numbers, 0);
%               x = k(numbers, sim);
%               k(numbers, 0);
%               k(numbers, sim);
%
% Input      : numbers = array of the disribution of lotto numbers
%              sim = an array of simulated unifromly distributed lotto
%              numbers used to compare to observed distribution. This may
%              also be set to zero
%
% Output     : x = the k values for the k-s test statistic first line is
%              the k values for the numbers distribution, the second row is
%              for two sample k-s test statistic.
%
%initialise variables
max1 = 0; %used to find kn+ to compare with uniform distribution
max2 = 0; %used to find kn- to compare with uniform distribution
max3 = 0; %used to find kn+ to compare with another distribution e.g simulated uniform
max4 = 0; %used to find kn- to compare with another distribution e.g simulated uniform

%calls to the empiricalDist to create the empirical distribution for the
%array of data
z = empiricalDist(numbers);
if sim
    y = empiricalDist(sim);
end

%for loop to find the different max variables
for i = 1:1:40
    if sim ~= 0 && (y(i) - z(i)) > max3
        max3 = (y(i) - z(i));
    end
    if sim ~= 0 && (z(i) - y(i)) > max4
        max4 = (z(i) - y(i));
    end
    if (i/40 - z(i)) > max1
        max1 = (i/40 - z(i));
    end
    if (z(i) - i/40) > max2
        max2 = (z(i) - i/40);
    end
end

%calculates the kn+ and kn- values
x(1, 1) = (40)^(1/2)* max1;
x(1, 2) = (40)^(1/2)* max2;
if sim ~= 0
    x(2, 1) = (40)^(1/2)* max3;
    x(2, 2) = (40)^(1/2)* max4;
end

x =x;
```

II

deMoivreEqui.m

```
function x = deMoivreEqui(u,k);
%
% return samples from deMoivre(1/k,1/k,...,1/k) RV X
%
% File Dates : Created  08/06/07  Modified  08/06/07
% Author(s)  : Raaz
%
% Call Syntax:  x = deMoivreEqui(u,k);
%               deMoivreEqui(u,k);
%
% Input      : u = array of uniform random numbers eg. rand
%              k = number of equi-probabble outcomes of X
% Output     : x = samples from X
%
x = ceil(k * u) ; % ceil(y) is the smallest integer larger than y
% floor is useful when the outcomes are {0,1,...,k-1}
%x = floor(k * u);
```

III

k.m

```
function x = k(numbers, sim);
%
% This function calulates the kn+ and kn- test statistics for the input
% data. This function can take one or two paramaters. If there are 2
% parameters the function calculates the k-s test statistics using the one
% sample test and then calculates the two sample test using both inputs.
%
% File Dates : Created  12/10/08  Modified  16/10/08
% Author(s)  : Adrian Goode
```

```
%
% Call Syntax:  x = k(numbers, 0);
%               x = k(numbers, sim);
%               k(numbers, 0);
%               k(numbers, sim);
%
% Input      : numbers = array of the disribution of lotto numbers
%              sim = an array of simulated unifromly distributed lotto
%              numbers used to compare to observed distribution. This may
%              also be set to zero
%
% Output     : x = the k values for the k-s test statistic first line is
%              the k values for the numbers distribution, the second row is
%              for two sample k-s test statistic.
%
%initialise variables
max1 = 0; %used to find kn+ to compare with uniform distribution
max2 = 0; %used to find kn- to compare with uniform distribution
max3 = 0; %used to find kn+ to compare with another distribution e.g simulated uniform
max4 = 0; %used to find kn- to compare with another distribution e.g simulated uniform

%calls to the empiricalDist to create the empirical distribution for the
%array of data
z = empiricalDist(numbers);
if sim
    y = empiricalDist(sim);
end

%for loop to find the different max variables
for i = 1:1:40
    if sim ~= 0 && (y(i) - z(i)) > max3
        max3 = (y(i) - z(i));
    end
    if sim ~= 0 && (z(i) - y(i)) > max4
        max4 = (z(i) - y(i));
    end
    if (i/40 - z(i)) > max1
        max1 = (i/40 - z(i));
    end
    if (z(i) - i/40) > max2
        max2 = (z(i) - i/40);
    end
end

%calculates the kn+ and kn- values
x(1, 1) = (40)^(1/2)* max1;
x(1, 2) = (40)^(1/2)* max2;
if sim ~= 0
    x(2, 1) = (40)^(1/2)* max3;
    x(2, 2) = (40)^(1/2)* max4;
end
```

```
x =x;
```

IV

empiricalDist.m

```
function x = empiricalDist(numbers);
%
% return the empirical distribution of the array and the plot of this
%
% File Dates : Created  12/10/08  Modified  16/10/08
% Author(s)  : Adrian Goode
%
% Call Syntax:  x = empiricalDist(numbers);
%               empiricalDist(numbers);
%
% Input      : numbers = array of the disribution of lotto numbers
%
% Output     : x = empirical distribution for the data
%
%intialise varibles
x = [40, 1];
cumulative = 0;
total = 6258;

%calculates the empirical distribution of the array of our data points
for i = 1:1:40
    cumulative = numbers(i) + cumulative;
    x(i) = cumulative/total;
end

%plots x the empirical distribution
plot(x);

x=x;
```

V

OneballSim.m

```
function x = OneballSim(y);
%
% return the Approximate P-Value of the data using the Monte Carlo
% approximation.
%
% File Dates : Created  16/10/08  Modified  17/10/08
% Author(s)  : Adrian Goode
%
% Call Syntax:  x = OneballSim(y);
%               OneballSim(y);
%
```

```
% Input      : y = array of numbers from the original draws
%
% Output     : x = number of each ball drawn from the 1043
%

Expt = 1043/40 * ones(40, 1); % expected values
Tobs = findchi(y, Expt);% observed test statistic
B = 1000000; % number of bootstrap replicates
TB = zeros(1,B); % initialise a vector of zeros for the bootstrapped test statistics
ApproxPValue = 0; % initialise an accumulator for approximate p-value
for b = 1:B % enter the bootsrap replication loop
    x = ceil(rand(1043, 1)*40); % generates uniformly random first dran balls for lotto
    BC = 1:40;
    [z, Bins] = hist(x, BC); % calculates the frequency of each ball
    TB(b) = findchi(z', Expt); %calculates the chi squared test statistic for the generated data
    if(TB(b)>Tobs) % increment the ApproxPValue accumulator by 1/B if bootstrapped value > Tobs
        ApproxPValue = ApproxPValue + (1/B);
    end
end
ApproxPValue % report the Approximate p-value
```

VI

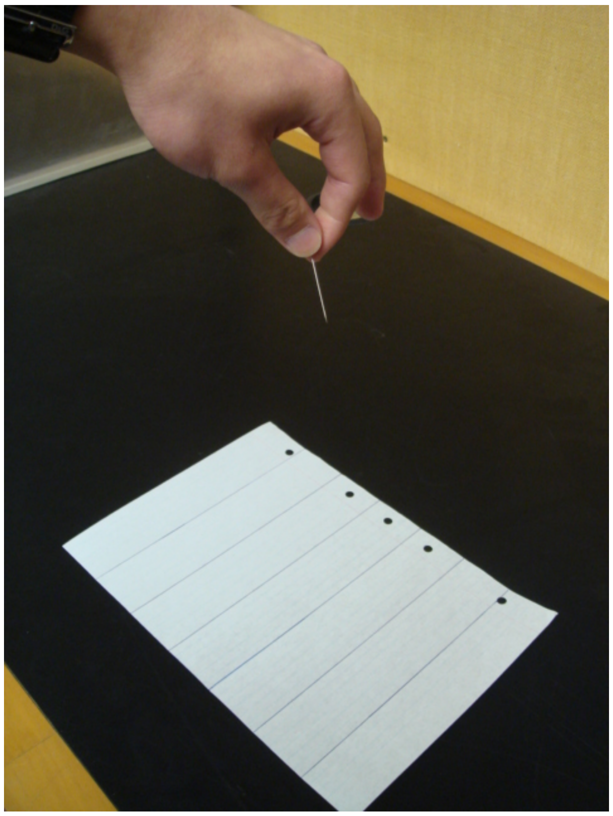
findChi.m

```
function chi = findChi(Observed, E);
%
% returns the chi squared value for the data passed in
%
% File Dates : Created  12/10/08  Modified  17/10/08
% Author(s)  : Adrian Goode
%
% Call Syntax:  x = findchi(Observed, E);
%               findchi(Observed, E);
%
% Input      : Observed = the array of observed values
%              E = the expected values
%
% Output     : x = chi squared test statistic
%
chi = sum(((Observed - E) .^2)./E);
```

Chapter 5

Testing the Approximation of π by Buffon's Needle Test

Amanda Hughes
and Sung-Lim Suh



Abstract

This project is designed to investigate Buffon's Needle experiment. We replicated the concept of Buffon's Needle and tested the null hypothesis that the outcomes from tossing a needle are directly related to π. The Delta Method and non-parametric methods have been employed in this project.

Introduction & Motivation

The report will firstly cover the background and motivation of this project. Next we will explain the geometric background to why this experiment approximates π and we will look at the statistical methodologies used. Following this we will discuss our results and conclusion. Finally we will discuss potential modifications.

Background - Comte de Buffon

Comte de Buffon was born September 7 1707 in Montbard, France. He studied law in Dijon and medicine in Angers. After his studies, he had a chance to tour around France, Italy, and England to explore his knowledge in science. When he returned to France, Buffon published translations of one of Isaac Newton's works and his interest in science was now clear.

Background - Buffon's Needle

Buffon's Needle Problem was first stated by Comte de Buffon in 1733, the solution was published later in 1777. The problem involves finding the probability that a needle of length l will land on a line, given a floor with equally spaced parallel lines (or floorboards) a distance d apart.

Motivation

The motivation behind this project is to reconstruct Buffon's Needle Experiment. We wanted to see if an approximation of π was found by this somewhat simple experiment over 200 years ago.

Materials & Methods

Materials

We first constructed a board which had several parallel lines. We did this by ruling lines on a piece of A4 paper. The width between the lines was either the same length as the needle or smaller than the length of the needle or larger than the length of the needle. The needle we used was a sewing needle of length 37mm. Instead of changing the needle to a smaller or larger needle for the three trials we ruled up three different sheets of A4 paper, one for each of the situations. The sheet that had the lines the same distance apart as the length of the needle had lines spaced 37mm apart. The sheet that had the lines closer together than the length of the needle had lines spaced 35mm apart. The sheet that had the lines further apart than the length of the needle had lines spaced 42mm apart. We recorded the data by entering it into Excel as the experiment was taking place. Sung-Lim tossed the needle and Amanda recorded the data.

Method

We tossed a needle 200 times for each of the three different distances apart of the lines. Sung-Lim tossed the needle for all trials so that the tossing could be done as identically as possible. Sung-Lim held the needle at the same height and dropped it in exactly the same way for each trial. Sung-Lim called out each toss as either "Yes" for the needle landing on a line or "No" for the needle not landing on a line. Any decisions that had to be made over whether the needle crossed the line or not were made by Sung-Lim. If the needle rolled or bounced off the page we disregarded that trial. A break was taken every 100 trials so that all trials could be done as identically as possible.

Geometric Aspect

We need to look at how this experiment approximates π. Let us begin by looking at the general case, where the needle is the same length as the distance between the floorboards.

Imagine a floor on which there are an infinite number of parallel lines (floorboards) spaced two units apart. You toss a needle that is also two units in length onto the floor. We want to find the probability that the needle crosses a line in the floorboards. l=the length of the needle and d=the distance between the lines.

Take one board to examine. Let x be the shortest distance from the mid point of the needle to the nearest line. Let θ be the angle between x and the needle. Imagine that there is a vertical line down from the end of the needle closest to the line. $\cos(\theta)$ gives the distance from the midpoint of the line to this imaginary line.

As you can imagine, the needle can fall in an infinite amount of ways and positions relative to the parallel lines. Therefore, θ also has an infinite amount of possibilities. We will limit these possibilities in two ways. Firstly only consider the line that is closest to the midpoint of the needle. Secondly we will divide the board into two by drawing an imaginary line down the board halfway between two lines and only consider needle positions on the right side of our halfway line. We will consider the needles whose middle falls to the left of this imaginary line to be a rotation of the right side of the imaginary line. So we now only have two

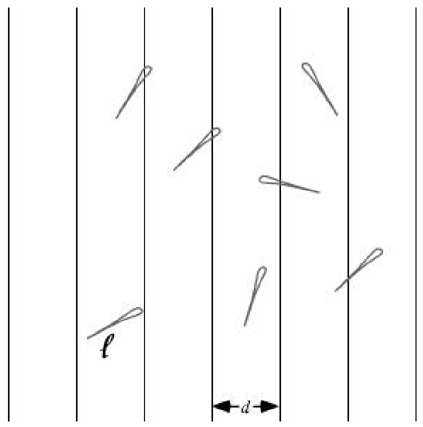


Figure 5.1: Example of Needle Tosses

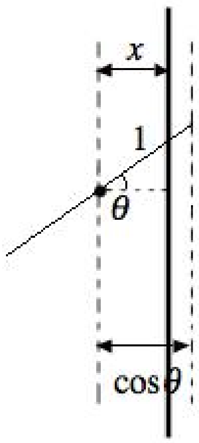


Figure 5.2: Explaining the outcomes of the needle toss

general situations to look at. One of these situations is shown in the above picture, where θ is less than $\pi/2$ radians. The other situation is where θ is larger than $\pi/2$ radians. In this case we will consider the angle θ to actually be the angle below of x and we will think of this angle as negative. Therefore, the support of x is 0 to 1 and the support of θ is $-\pi/2$ to $\pi/2$. This is shown in figure 5.3.

The shaded area in figure 5.3, represents when the needle crosses a line. This happens when $x < \cos(\theta)$. The total outcome space is π and the area under $\cos(\theta)$ from $-\pi/2$ to $\pi/2$ is 2 (integrating $\cos(\theta)$ from $-\pi/2$ to $\pi/2$). Therefore, the chance of a needle falling on a line is $2/\pi$.

For the case where the needle is shorter in length than the distance between the floorboards we need to

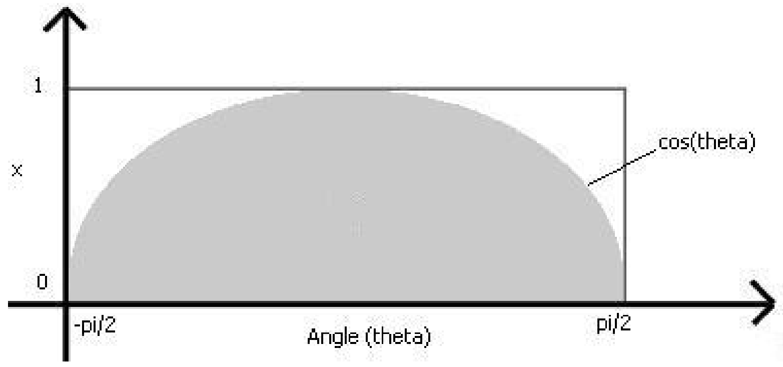


Figure 5.3: Outcome Space of Buffon's Needle Experiment for General Case

modify the above slightly. We need to account for the exact length difference between the needle and the floorboards. This is added to the above explanation by multiplying $2/\pi$ by y, which is the ratio of length of needle and distance between the lines, ie. $y = l/d$. Therefore the chance of a shorter needle landing on a line is $2y/\pi$.

$$\Pr(\text{needle crosses line}) = \int_0^{2\pi} (\frac{l|\cos\theta|}{d}) \frac{d\theta}{2\pi}$$

$$\Pr(\text{needle crosses line}) = \frac{2l}{d\pi} \int_0^{\frac{\pi}{2}} \cos\theta \, d\theta$$

$$\Pr(\text{needle crosses line}) = \frac{2l}{d\pi}$$

$$\Pr(\text{needle crosses line}) = \frac{2y}{\pi}$$

For the case where the needle is longer in length than the distance between the floorboards we get a more complex outcome. We again need to account for the exact length difference between the needle and the floorboards. Therefore, the chance of a longer needle landing on a line is:

$$\Pr(\text{needle crosses line}) = \tfrac{2}{\pi}(y - \sqrt{y^2 - 1} + \sec^{-1} y)$$

Statistical Methodology

For this experiment we looked at the three different distances apart of the lines. For each of the three circumstances we used the same hypotheses.

H0(null hypothesis): $\phi^* = \frac{2}{\pi}$, the outcomes of tossing a needle are directly related to π

H1(alternative hypothesis): $\phi^* \neq \frac{2}{\pi}$, the outcomes of tossing a needle are not directly related to π

For the general case:

$$(\Theta, X) \overset{IID}{\sim} Uniform([\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times [0,1])$$

as shown in figure 1.3. We know that for our simplest situation, where the length of the needle is the same as the distance between the lines:

$$\phi^* = \Pr((\Theta, X) \in \text{Shaded Region of figure 5.3}) = \tfrac{2}{\pi}$$

$$N_1, N_2, ..., N_{200} \overset{IID}{\sim} Bernoulli(\phi^*)$$

This is the probability of a needle landing on a line. The trials for each of the three different distances apart of the lines are 200 independent and identically distributed Bernoulli trials.

Our maximum likelihood estimator of ϕ^* is:

$$\hat{\phi}_{200} = \tfrac{n_1+n_2+...+n_{200}}{200}$$

This is the sample mean. But what we really want is a function of ϕ^*, namely, $\Psi(\Phi) := 2/\Phi$. We now need the Delta Method. The Delta Method gives us the needed correction to transform an estimate and its confidence interval. By the Delta Method:

$$\pi \approx \psi^* = g(\phi^*) = \frac{2}{\phi^*}$$

and the maximum likelihood estimate of ψ^* is:

$$\hat{\psi}_{200} = g(\hat{\phi}_{200}) = \frac{2}{\hat{\phi}_{200}}$$

Next we can calculate the standard error of our estimator of ψ^*, namely, $\hat{\Psi}_n = g(\hat{\Phi}_n)$, and subsequently confidence intervals:

$$se(\hat{\Psi}_n) = |g'(\phi)|se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |\tfrac{d}{d\phi}g(\phi)|se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |\tfrac{d}{d\phi}(2\phi^{-1})|se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |-2\phi^{-2}|se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = (2\phi^{-2})se(\hat{\Phi}_n)$$

where the estimated standard error is:

$$se(\hat{\psi}_{200}) = (2\phi_{200}^{-2})se(\hat{\phi}_{200}) = (2\phi_{200}^{-2})\tfrac{\hat{\phi}_{200}(1-\hat{\phi}_{200})}{200}$$

Now we can calculate the 95% confidence interval for $\psi^* \approx \pi$:

$$\hat{\psi}_{200} \pm 1.96se(\hat{\psi}_{200})$$

Now for the needle shorter in length than the distance between the lines, where $\phi^* = \frac{2y}{\pi}$ and therefore by the Delta Method: $\pi \approx \frac{2y}{\phi^*}$, $\hat{\phi}_{200}$ is the sample mean of this set of data.

$$\psi^* = g(\phi^*) = \tfrac{2y}{\phi^*}$$

$$\hat{\Psi}_{200} = g(\hat{\phi}_{200}) = \tfrac{2y}{\hat{\phi}_{200}}$$

From this we can calculate the standard error of the estimator of ψ^*, namely, $\hat{\Psi}_n$, and subsequently confidence intervals as follows:

$$se(\hat{\Psi}_n) = |g'(\phi)| * se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |\tfrac{d}{d\phi} g(\phi)| * se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |\tfrac{d}{d\phi}(2y\phi^{-1}| * se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |-2y\phi^{-2}| * se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = (2y\phi^{-2}) * se(\hat{\Phi}_n)$$

where the estimated standard error is

$$se(\hat{\psi}_n) = (2y\hat{\phi}_{200}^{-2}) * se(\hat{\Phi}_n) \quad \text{and} \quad se(\hat{\Phi}_n) = \tfrac{\hat{\phi}_{200} * (1-\hat{\phi}_{200})}{200}$$

Now we can calculate the 95% confidence interval for $\psi^* \approx \pi$:

$$\hat{\psi}_{200} \pm 1.96 se(\hat{\psi}_{200})$$

Now for the needle longer in length than the distance between the lines, where $\phi^* = \frac{2}{\pi}(y - \sqrt{y^2-1} + sec^{-1}y)$ and therefore by the Delta Method: $\pi \approx \frac{2}{\phi^*}(y - \sqrt{y^2-1} + sec^{-1}y)$, $\hat{\phi}_{200}$ is the sample mean of this set of data.

$$\psi^* = g(\phi^*) = \tfrac{2}{\phi^*}(y - \sqrt{y^2-1} + sec^{-1}y)$$

$$\hat{\psi}_{200} = g(\hat{\phi}_{200}) = \tfrac{2}{\hat{\phi}_{200}}(y - \sqrt{y^2-1} + sec^{-1}y)$$

From this we can calculate the standard error of $\hat{\Psi}_{200}$, the estimator of ψ^*, and subsequently its confidence interval:

$$se(\hat{\Psi}_n) = |g'(\phi)|se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |\frac{d}{d\phi}g(\phi)|se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |\frac{d}{d\phi}\frac{2}{\phi}(y - \sqrt{y^2-1} + sec^{-1}y)|se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = |-2\phi^{-2}|(y - \sqrt{y^2-1} + sec^{-1}y)se(\hat{\Phi}_n)$$

$$se(\hat{\Psi}_n) = (2\phi^{-2})(y - \sqrt{y^2-1} + sec^{-1}y)se(\hat{\Phi}_n)$$

where the estimated standard error is:

$$se(\hat{\Psi}_n) = (2\hat{\phi}_{200}^{-2})(y - \sqrt{y^2-1} + sec^{-1}y)se(\hat{\Phi}_n) \quad \text{and} \quad se(\hat{\Phi}_n) = \frac{\hat{\phi}_{200}(1-\hat{\phi}_{200})}{200}$$

Now we can calculate the 95% confidence interval for $\psi^* \approx \pi$:

$$\hat{\psi}_{200} \pm 1.96 se(\hat{\psi}_{200})$$

Results

For the needle same in length as the distance between the lines, we got an approximation for π of 3.0769. The 95% Confidence Interval (2.7640, 3.3898), contains π.

For the needle shorter in length than the distance between the lines, we got an approximation for π of 2.9122. The 95% Confidence Interval (2.5861, 3.2384), contains π.

For the needle longer in length than the distance between the lines, we got an approximation for π of 2.8042. However, the 95% Confidence Interval (2.5769, 3.0316), does not contain π.

Conclusion

For the first two cases the 95% Confidence intervals both contain π so we cannot reject the null hypothesis that the outcomes from tossing a needle are directly related to π. For the final case the 95% Confidence interval does not contain π so we reject the null hypothesis and conclude that for this case the outcomes from tossing a needle are not directly related to π. The first two outcomes agree with Buffon's Needle while the third one may in fact be false, because our samples were quite small in hindsight.

Potential Modification

There are several ways we could improve this experiment. Firstly we could increase sample size and see if we can get better approximations of π. Secondly we could investigate the chance of a needle landing on a line on a square tiled floor, this is the Buffon-Laplace Needle. We could also extend the idea to not only cover needles and lines but also shapes and more complicated arrangements of tiles or lines.

Author Contributions

Amanda Hughes

Original concept; research; recording of the data collection; MATLAB analysis; writing and construction of the report; LateX writing and compiling; some of the slides for presentation and some of the script for the presentation (mainly sections on the geometric aspect of the problem).

Sung-Lim Suh

Research; collected data; majority of slides for presentation; majority of script for presentation, some of which was used in the report (Background on Comte de Buffon and Background of Buffon's Needle).

Many thanks to our lecturer Dr. Raazesh Sainudiin who spent much of his time discussing the geometric aspect of this report with us. We really appreciated being able to stop by his office almost whenever and have a discussion.

References

Texts:

Stigler,Stephen M.,*Statistics on the Table: The History of Statistical Concepts and Methods*, Harvard University Press, USA, 2002

Electronic Texts:

Hyna, Irene; Oetiker, Tobias; Partl, Hubert; Schlegl, Elisabeth.,*The Not so Short Introduction to LATEX 2e or LATEX 2e in 90 Minutes*, 2000

Websites:

`www.angelfire.com/wa/hurben/buff.html`

`www.mathworld.wolfram.com/BuffonsNeedleProblem.html` www.mste.uiuc.edu/reese/buffon/buffon.html

www.nndb.com/peopl/755/000091482/

www.ucmp.berkeley.edu/history/buffon2.html

www.youtube.com/watch?v=Vws1jvMbs64

Appendix

Data and MATLAB Code:

```
data% this is the data for the experiment where the needle is the same length as the distance between the lines
datalines=[1 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 0 1 0 0 1 0 1 1 1 ...
    1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 ...
    0 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 0 1 ...
    1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
    1 0 0 1 0 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 1 ...
    1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 0 1]
MLE_theta_star=mean(datalines)% find the maximum likelihood estimator, the sample mean
piapprox=2*(1/MLE_theta_star)% estimate of pi
StdErr=sqrt((MLE_theta_star*(1-MLE_theta_star))/200)% standard error
CI=[piapprox-(1.96*StdErr*2/(MLE_theta_star)^2),piapprox+(1.96*StdErr*2/(MLE_theta_star)^2)]% 95% Conficence Interval ...
for our approximate of pi

short% this is the data for the experiment where the needle is shorter in length than the distance between the lines
datalines=[1 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 0 0 ...
    1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0 ...
    1 1 1 0 0 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1 0 1 ...
    1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 0 1 1 0 1 0 1 1 1 1 1 ...
    0 1 1 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 0 1 0 0 0 1 1 1 1 1 0 0 1 0 0 ...
    0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1]
MLE_theta_star=mean(datalines)% find the maximum likelihood estimator, the sample mean
x=37/42%x=length/distance
piapprox=(2*x)*(1/MLE_theta_star)% estimate of pi
StdErr=sqrt((MLE_theta_star*(1-MLE_theta_star))/200)% standard error
CI=[piapprox-(1.96*StdErr*(2*x)/(MLE_theta_star)^2),piapprox+(1.96*StdErr*(2*x)/(MLE_theta_star)^2)]% 95% Conficence ...
Interval for our approximate of pi

long% this is the data for the experiment where the needle is longer in length than the distance between the lines
datalines=[0 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 1 1 ...
    0 0 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 ...
    1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 ...
    1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 1 0 0 1 ...
    1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 ...
    0 1 1 0 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1]
MLE_theta_star=mean(datalines)% find the maximum likelihood estimator, the sample mean
x=37/35%x=length/distance
piapprox=(2*(1/MLE_theta_star))*(x-sqrt((x^2)-1)+asec(x))% estimate of pi
```

```
StdErr=sqrt((MLE_theta_star*(1-MLE_theta_star))/200)% standard error
CI=[piapprox-(1.96*StdErr*2/(MLE_theta_star)^2)*(x-sqrt((x^2)-1)+asec(x)),piapprox+(1.96*StdErr*2/(MLE_theta_star)^2)* ...
(x-sqrt((x^2)-1)+asec(x))]% 95% Conficence Interval for our approximate of pi
```

Chapter 6

Relationship Between Magnitude of Earthquakes and Distance From The Fault Line

Tonya den Baars and Sarah Ramsey

Abstract

This project is designed to compare the epicentres of earthquakes in New Zealand to the fault line between 18-Jan-2008 02:23:44 to 18-Aug-2008 19:29:29. We are testing the null hypothesis that there is no relationship between the magnitude of an earthquake and its distance from the fault line to the epicentre.

Introduction & Motivation

Firstly, this report covers the background and motivation of this project. This is then followed by the methodologies used, which are explained before discussing the results and conclusion of this experiment.

Background- Earthquakes

The earth's crust is made up of tectonic plates that lock together. Stress builds up where the plates meet together and release seismic waves. The waves shake the ground, causing what we know to be an earthquake. The fault line is where two tectonic plates meet. New Zealand is situated half on the Pacific Plate and half on the Australasian Plate. In the South Island, the fault line runs right down the middle of the Southern

Alps, and is more commonly known as the Alpine Fault line. We used the Alpine Fault line to approximate the remainder of the fault line of New Zealand. We chose to look at where the earthquakes are actually occurring in relation to the fault line because it is expected that the earthquakes should occur reasonably close to the fault line as that is where the seismic waves are released from.

Background- Richter (magnitude) Scale

Wikipedia (2008) quotes, "The Richter (magnitude) scale assigns a single number to quantify the amount of seismic energy released by an earthquake. It is a base-10 logarithmic scale obtained by calculating the logarithm of the combined horizontal amplitude of the largest displacement from zero on a Wood-Anderson torsion seismometer output".

Background- Longitude and Latitude

The earth is divided up by a series of invisible lines used to reference points in the earth. The horizontal lines are called latitude and run parallel with the equator, (which is 0 degrees). The vertical lines are called longitude and do not run parallel to each other, instead they are half circles which converge at the north and south poles.

Motivation

The motivation behind this project is to see whether there is a relationship between the distance of the earthquakes epicentre from the fault line and the magnitude. We have a mild Geography background and thought it would be interesting because we have studied the earthquake process at secondary school level. Earthquakes occur frequently and it is always good to have a bit of common knowledge about them and the fault line, which helps us to understand where they may occur.

Materials & Methods

Collecting the Data

We accessed the data from www.geonet.org.nz. From here, we could select the variables (date, longitude, latitude, magnitude), that we wanted to use. We chose to look at all earthquakes occurring between $18-Jan-200802:23:44$ to $18-Aug-200819:29:29$. Our lecturer, Raazesh Sainudiin, cleaned up the data (deleted incomplete data), and converted it from an .xls file to a .csv file which was easier for us to use when carrying out the Matlab code.

Seismometers are instruments that measure and record motions of the ground, including those of seismic waves generated by earthquakes. Records of seismic waves allow seismologists to map the interior of the

Earth, and locate and measure the size of these earthquakes (Wikipedia, 2008).

Figure 6.1: This is a photo of a seisommeter used to measure the magnitude and location of earthquakes.

Statistical Methodology

First we estimated the fault line and then used the data to look at the distribution of magnitude and distance from the fault line, where we could then carry out the non parametric Bootstrap method.

1. Use an estimate of the end points of the fault line to calculate the gradient of the fault line. This was done by using the equation $m = y2 - y1/x2 - x1$. We found estimates for the end points of the fault line at Picton and a point off coast of the south west corner of New Zealand.

2. Calculate distances from the epicentres of earthquakes to the fault line.

3. Plot the bivariate data of magnitude and distance from the fault line.

4. Calculate the correlation coefficient.

5. Use the Bootstrap method to calculate a non parametric confidence interval.

We chose to do a non parametric experiment because there is an infinite number of locations where earthquakes can occur. For this experiment, we are testing whether there is a correlation between magnitude and distance from the fault line. Commands were included in our Matlab code which calculated the correlation coefficient of the sample. The null hypothesis of this project is $H_0 : C = 0$ and the alternative hypothesis is $H_1 : C \neq 0$, where C is the correlation coefficient. To test this hypothesis, we used non-parametric bootstrap method to obtain a 95% confidence interval for this value. These values were formed by sampling the data uniformly. The observed data was obtained from `geonet.org.nz`, which allowed us to choose variables, such as, date, longitude, latitude and magnitude, that we wanted to use for our project. We used 6128 samples from different earthquakes throughout New Zealand. This whole process was then repeated 1000 times through the Bootstrap method and from this we obtained a 95% bootstrapped confidence interval.

Results

We calculated the gradient of our approximated fault line first by finding the longitude and latitude from degrees, minutes, seconds to degrees decimals as they were easier to work with. We then used the equation $m = y2 - y1/x2 - x1$, using the end points of Picton and Milford Sound to calculate the gradient. 'x' was then set to zero to find the 'y' intercept. This then gave the equation of $y = 0.5457x - 136.25$ which we then used in our Matlab code to help with our normal projections to find the distances from the fault line.

We produced three graphs which were scatterplots of longitude against latitude, magnitude against distance from the fault line and a histogram of the bootstrap sample correlations.

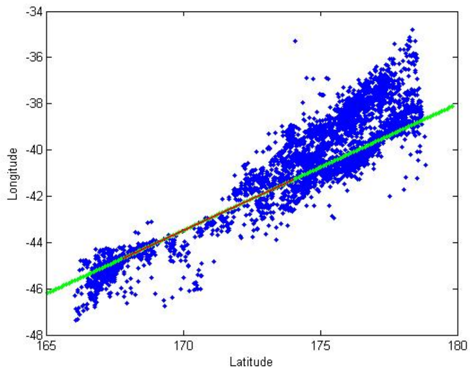


Figure 6.2: This graph shows the longitude and latitude of each of the earthquakes. This includes all of New Zealand's earthquakes (North Island and South Island). The line through the middle is our approximated fault line through New Zealand that was used in the analysis. The red part of the fault line is our approximation of the Alpine Fault Line. From this graph, we can see that the earthquakes' distribution follows the fault line.

The sample correlation is 0.1236. This means that there is a very weak positive correlation between magnitude and distance from the fault line. If there was a strong correlation then the number would be closer to -1 or 1. No correlation is represented by zero.

The Bootstrapped Confidence Interval is $(0.0972, 0.1523)$. This confidence interval does not include zero, which follows the sample correlation confirming that there is a very weak positive correlation between magnitude and distance from the fault line.

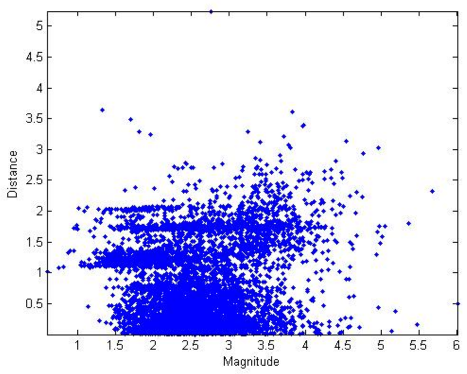


Figure 6.3: This graph shows a scatterplot of the magnitude against distance from the fault line. There is no clear visual relationship between these two variables. However, the correlation coefficient calculated shows a value of 0.1236 which suggests a very weak correlation between magnitude and distance from the fault line.

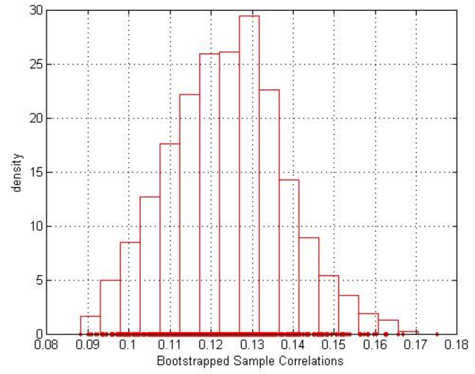


Figure 6.4: This graph shows a histogram of the bootstrapped correlations. This graph shows the distribution of bootstrapped correlations over 1000 repetitions. The sample correlations are reasonably normally distributed, however it could be argued that they are slightly skewed to the right.

Conclusion

We can now conclude that we reject the null hypothesis $H_0 : C = 0$ as there is a sample correlation of 0.1236 and a bootstrapped confidence interval of $(0.0972, 0.1523)$ which does not include zero. This gives significant evidence that there is a relationship between magnitude of earthquakes and their distance from the fault line.

Potential Modifications

We chose to keep all of the data from New Zealand when carrying out this project. However, we think the North Island's data has affected the correlation of the South Island's earthquake data. From looking at the graphs we can see that the North Island's earthquake data is further away from the fault line than the South Island's. By taking out the North Island data, we think there would be a stronger correlation between magnitude and distance from the fault line.

The fault line that we approximated is a straight line that only represented Picton to Milford Sound and was used as a linear estimate of the fault line for the whole of New Zealand. By looking at pictures of the fault line in New Zealand it is clear that it is not a straight line and curves in places. A more sophisticated analysis of the distance to fault line is possible using similar statistical methodologies.

Author Contributions

Raazesh Sainudiin created the MatLab code from which we could then carry out our project. All of the report write up and power point slides for the oral presentation was done equally between Sarah Ramsey and Tonya den Baars.

References

Wikipedia. (2008). Richter Magnitude Scale. Retrieved 6/10/08 from `http://en.wikipedia.org/wiki/Richter_magnitude_scale`. Wikipedia. (2008). Seismograph. Retrieved 13/10/08 from `http://en.wikipedia.org/wiki/Seismograph`.

Appendix A

NZEarthQuakesBootMagDist.m

```
%% Load the data from the comma delimited text file 'earthquakes.csv' with the following column IDs
% CUSP_ID,LAT,LONG,NZMGE,NZMGN,ORI_YEAR,ORI_MONTH,ORI_DAY,ORI_HOUR,ORI_MINUTE,ORI_SECOND,MAG,DEPTH
EQ=dlmread('earthquakes.csv',','); % the matrix EQ has the data
```

```
size(EQ) % report the dimensions or size of the matrix EQ
clf
MaxD=max(datenum(EQ(:,6:11)));
MinD=min(datenum(EQ(:,6:11)));
datestr(MaxD)
datestr(MinD)

LonData=EQ(:,3);
LatData=EQ(:,2);
MagData=EQ(:,12);

%subplot(1,3,1)
figure
plot(LonData,LatData,'.') % plot the 3rd column against the 2nd column -- LONGitude Vs. LATtitude
xlabel('Latitude');ylabel('Longitude');
hold on;
%% Faultline as a line between Faultx1x2 and Faulty1y2
FaultLat12=[167+47/60 174.014];% Milford Sound to the northern Port city of S.Island
FaultLon12=[-(44+41/60) -41.283];
SlopeFaultLine=(FaultLon12(2)-FaultLon12(1))/(FaultLat12(2)-FaultLat12(1))
IntcpFaultLine=-136.25
%function Lon = FaultLineFunc(Lat)=SlopeFaultLine*Lat+IntcpFaultLine

line(FaultLat12,FaultLon12,'color','r','LineWidth',2) % fault line

LATS=min(LonData)-1:.1:max(LonData)+1;
plot(LATS,(SlopeFaultLine .* LATS)+IntcpFaultLine,'g.','LineWidth',3)
line(FaultLat12,FaultLon12,'color','r','LineWidth',2) % fault line

Distance=(abs(LatData-((SlopeFaultLine .* LonData)+IntcpFaultLine)))/sqrt(SlopeFaultLine^2+1);
%subplot(1,3,2)
figure
plot(MagData,Distance,'.')
xlabel('Magnitude');ylabel('Distance');
axis([min(MagData) max(MagData) min(Distance) max(Distance)]);
%hist(Distance)
%% correlation between magnitude and distance
CC=corrcoef(MagData,Distance); % use built-in function to compute sample correlation coefficient matrix
SampleCorrelation=CC(1,2) % plug-in estimate of the correlation coefficient

%% Bootstrap
B = 1000; % Number of Bootstrap replications
BootstrappedCCs=zeros(1,B); % initialise a vector of zeros
N = length(MagData); % sample size
rand('twister',778787671); % initialise the fundamental sampler
for b=1:B
    Indices=ceil(N*rand(N,1));% uniformly sample random indices from 1 to N with replacement
    BootstrappedMag = MagData([Indices]); % bootstrapped Magnitude data
    BootstrappedDis = Distance([Indices]); % bootstrapped Distance data
    CCB=corrcoef(BootstrappedMag,BootstrappedDis);
    BootstrappedCCs(b)=CCB(1,2); % sample correlation of bootstrapped data
```

```
end
%plot the histogram ofBootstrapped Sample Correlations with 15 bins
%subplot(1,3,3);
figure;
%histogram(BootstrappedCCs,15);

histogram(BootstrappedCCs,1,[min(BootstrappedCCs),max(BootstrappedCCs)],'r',2)
xlabel('Bootstrapped Sample Correlations')
% 95% Normal based Confidence Interval
SehatBoot = std(BootstrappedCCs); % std of BootstrappedMedians
% 95% C.I. for median from Normal approximation
ConfInt95BootNormal = [SampleCorrelation-1.96*SehatBoot, SampleCorrelation+1.96*SehatBoot]
% 95% Percentile based Confidence Interval
ConfInt95BootPercentile = ...
    [qthSampleQuantile(0.025,sort(BootstrappedCCs)),...
    qthSampleQuantile(0.975,sort(BootstrappedCCs))]
builtin('cd','P:\Stat218');
```

project.txt

```
NZEarthQuakesBootMagDist


ans =


        6128          13




ans =


18-Aug-2008 19:29:29




ans =


18-Jan-2008 02:23:44




SlopeFaultLine =


    0.5457




IntcpFaultLine =


 -136.2500




SampleCorrelation =


    0.1236




ConfInt95BootNormal =
    0.0958    0.1513


ConfInt95BootPercentile =


    0.0972    0.1523
```

Chapter 7

Testing the Average Time of an Olympic 100m Sprinter

Daniel Laws

16 October 2008

Abstract

One of the most watched events at the Olympics is the Men's 100m sprint final. Everyone wants to know, who is the fastest man on Earth. This year's Olympic final at the Beijing games more than lived up to expectations. This was mainly due to the fact that the World record time was reduced to 9.69 seconds. Six of the athletes also produced times underneath the 10 second time barrier, which is more than any other race to date.

Introduction

The Beijing 2008 Olympics proved that the very top athletes are getting faster. However, is this true for the other Olympic athletes? I hypothesize that as time goes by the average Olympic sprinter is increasing in speed. The hypothesis will be tested by using distributions of the 100m sprint races from the Athens 2004 Olympics and the Beijing 2008 Olympics. In order to prove this assumption I will need to falsify the following null hypothesis.
H_0: There is no difference between the distributions from Athens 2004 and Beijing 2008.
H_a: There is a difference between the distributions from Athens 2004 and Beijing 2008.

Materials & Methods

The data for the Athens Olympic times and the Beijing Olympic times were received from the IAAF website. All the data was taken from any athlete who competed in a 100m sprint regardless of whether it was a heat, quarter-final, semi-final or final. In total 141 athletes' times were received from Athens 2004 and 142 athletes' times were received from Beijing 2008. Once the data was received a permutation test was performed on the data to prove if the the two distributions were of the same distribution. The two Empirical CDF's were plotted to give more evidence for the decision.

Statistical Methodology

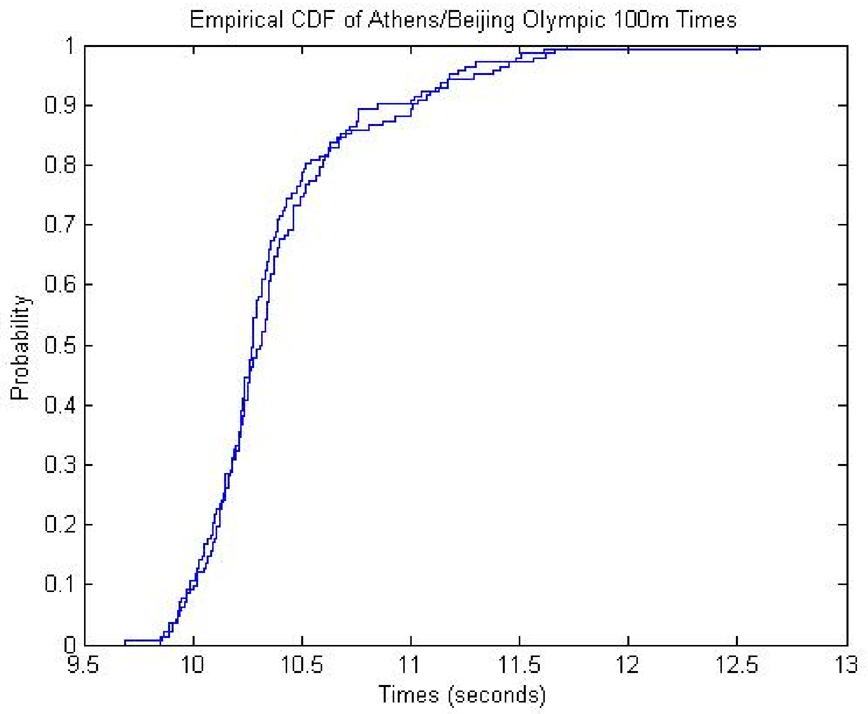
In order to test the null hypothesis that there is no difference in the two distributions, the non-parametric permutation test was used. The parametric test was used because it is an accurate test that tries to reject the null hypothesis.

Firstly the difference in means of the observed data is calculated. Next the data is randomly permuted and into two new arrays and the difference in means of these arrays is calculated. The two mean differences are compared with each other. The data is permuted and its mean difference is compared with the observed mean difference 10,000 times. The amount of times the permuted mean difference is larger than the observed mean difference is summed up and divided by 10,000 to give the p-value.

The reason the distributions can be permuted is because the test is trying to reject the null hypothesis. When the distributions are permuted the data becomes more even between the two arrays due to the randomness of the permutation. Therefore the mean difference of the arrays will give a value that would occur when the distributions are similar. Which means a small p-value would occur when the observed mean difference is large and the distributions are significantly different.

Results

P-Value = 0.5535



Conclusion

In conclusion the p-value is large which suggests there is significant evidence not to reject the null hypothesis. The plot of the Empirical CDFs also shows that there is significant evidence not to reject the null hypothesis. This is because the two distributions are very similar and there is hardly any variation. This means that there is no significant difference in means between Athens 100m times and Beijing 100m times. Therefore the average Olympic athlete is not increasing in speed as time goes by, even though the World record is falling.

Potential Modification

This test was performed by comparing the distributions from Athens 2004 and Beijing 2008. There could have also been tests including the distributions from Sydney 2000 and possibly Atlanta 1996. This would be able to show if the null hypothesis still holds. However it would not be wise to compare distributions too far in the past due to technological differences in timing.

References

http://www.iaaf.org/index.html

Appendix A

MATLAB Code

```
% Olympic Athletes Times to complete a 100m sprint at their respective Olympic Games
Athens=[10.12 10.22 10.27 10.35 10.45 10.54 10.75 11.00 10.13 10.19 10.28 10.36 10.67 10.76 11.62...
    10.07 10.15 10.21 10.23 10.29 10.32 10.62 11.72 10.02 10.08 10.15 10.33 10.50 11.02 11.30 11.66...
    10.09 10.18 10.26 10.34 10.37 10.52 10.62 11.22 10.33 10.35 10.39 10.41 10.67 10.68 10.85 11.13...
    10.06 10.15 10.24 10.28 10.32 10.72 11.25 11.56 10.18 10.21 10.23 10.28 10.50 10.70 11.05 11.17...
    11.18 10.39 10.40 10.43 10.48 10.49 10.51 10.76 11.17 10.11 10.20 10.22 10.27 10.27 10.58 10.76...
    9.93 10.12 10.12 10.15 10.19 10.21 10.24 10.34 9.89 10.12 10.26 10.26 10.26 10.28 10.42 10.43...
    9.96 10.15 10.15 10.24 10.36 10.38 10.48 10.02 10.05 10.11 10.17 10.22 10.24 10.29 10.39 9.93...
    9.99 10.18 10.23 10.24 10.29 10.30 10.32 10.07 10.09 10.11 10.22 10.28 10.29 10.32 10.35 9.95...
    9.97 9.97 10.02 10.12 10.22 10.28 10.28 9.85 9.86 9.87 9.89 9.94 10.00 10.10];
Beijing=[10.20 10.24 10.26 10.28 10.49 10.61 10.81 11.09 10.16 10.17 10.21 10.35 10.44 10.51 11.17...
    10.24 10.26 10.32 10.34 10.35 10.46 11.00 11.29 10.15 10.22 10.22 10.25 10.58 10.63 11.01 11.45...
    10.22 10.29 10.29 10.34 10.52 10.62 11.00 12.60 10.13 10.25 10.26 10.39 10.53 10.66 11.11 11.51...
    10.25 10.34 10.35 10.37 10.57 10.73 11.38 11.61 10.28 10.34 10.37 10.46 10.52 10.60 10.63 11.07...
    10.40 10.46 10.46 10.49 10.58 10.70 11.03 11.41 10.35 10.39 10.42 10.46 10.60 10.87 11.14 11.48...
    9.99 10.09 10.23 10.32 10.33 10.36 10.46 10.93 9.99 10.09 10.16 10.21 10.25 10.32 10.33 10.37...
    10.05 10.07 10.10 10.11 10.21 10.24 10.36 10.37 9.92 10.04 10.09 10.14 10.18 10.27 10.33 10.35...
    10.02 10.08 10.14 10.23 10.24 10.31 10.40 9.85 9.94 9.97 10.01 10.05 10.13 10.18 10.20 9.91 9.93...
    9.94 10.03 10.05 10.10 10.16 10.18 9.69 9.89 9.91 9.93 9.95 9.97 10.01 10.03];
```

```
% Non-parametric Permutation Test determining whether the two Olympic
% distribution are of the same distribution.
Tobs=abs(mean(Athens)-mean(Beijing));   % Observed Mean Difference Test statistic
nAthens=length(Athens);                 % Sample size of Athens Distribution
nBeijing=length(Beijing);               % Sample size of Beijing Distribution
ntotal=nAthens+nBeijing;                % Sample size of pooled data
total=[Athens Beijing];                 % Observed data in one array
B=10000;                                % Number of bootstrap replicates
TB=zeros(1,B);                          % Initialise a vector of zeros for the bootstrapped test statistics
ApproxPValue=0;                         % Initialise an accumulator for approximate p-value
```

```
for b=1:B   % Enter the bootsrap replication loop
    % Use MATLAB’s randperm function to get a random permutation of
    % indices{1,2,...,ntotal}
PermutedIndices=randperm(ntotal);
    % Use the first nleft of the PermutedIndices to get the bootstrapped
    % left-side data
BAthens=total(PermutedIndices(1:nAthens));
    % Use the last nright of the PermutedIndices to get the bootstrapped
    % right-side data
BBeijing=total(PermutedIndices(nBeijing+1:ntotal));
    % Compute the test statistic for the bootstrapped data
TB(b) = abs(mean(BAthens)-mean(BBeijing));
if(TB(b)>Tobs)
    % increment the ApproxPValue accumulator by 1/B if bootstrapped value >
    % Tobs
ApproxPValue=ApproxPValue+(1/B);
end
end
ApproxPValue                            % Report the Approximate p-value
```

```
% ECDF plots of the Athens 2004 ditribution and Beijing 2008 distribution
ecdf(Athens)        % Athens 2004 100m times ECDF distribution
hold on
ecdf(Beijing)       % Beijing 2008 100m times ECDF distribution
title('Empirical CDF of Athens/Beijing Olympic 100m Times')
xlabel('Times (seconds)')
ylabel('Probability')
```

Appendix B

Raw Data of Olympic Athletes Times

Athens	Athens	Athens	Athens	Athens		Beijing	Beijing	Beijing	Beijing	Beijing
10.12	11.30	10.28	9.89	10.30		10.20	10.24	10.26	10.28	10.49
10.22	11.66	10.50	10.12	10.32		10.61	10.81	11.09	10.16	10.17
10.27	10.09	10.70	10.26	10.07		10.21	10.35	10.44	10.51	11.17
10.35	10.18	11.05	10.26	10.09		10.24	10.26	10.32	10.34	10.35
10.45	10.26	11.17	10.26	10.11		10.46	11.00	11.29	10.15	10.22
10.54	10.34	11.18	10.28	10.22		10.22	10.25	10.58	10.63	11.01
10.75	10.37	10.39	10.42	10.28		11.45	10.22	10.29	10.29	10.34
11.00	10.52	10.40	10.43	10.29		10.52	10.62	11.00	12.60	10.13
10.13	10.62	10.43	9.96	10.32		10.25	10.26	10.39	10.53	10.66
10.19	11.22	10.48	10.15	10.35		11.11	11.51	10.25	10.34	10.35
10.28	10.33	10.49	10.15	9.95		10.37	10.57	10.73	11.38	11.61
10.36	10.35	10.51	10.24	9.97		10.28	10.34	10.37	10.46	10.52
10.67	10.39	10.76	10.36	9.97		10.60	10.63	11.07	10.40	10.46
10.76	10.41	11.17	10.38	10.02		10.46	10.49	10.58	10.70	11.03
11.62	10.67	10.11	10.48	10.12		11.41	10.35	10.39	10.42	10.46
10.07	10.68	10.20	10.02	10.22		10.60	10.87	11.14	11.48	9.99
10.15	10.85	10.22	10.05	10.28		10.09	10.23	10.32	10.33	10.36
10.21	11.13	10.27	10.11	10.28		10.46	10.93	9.99	10.09	10.16
10.23	10.06	10.27	10.17	9.85		10.21	10.25	10.32	10.33	10.37
10.29	10.15	10.58	10.22	9.86		10.05	10.07	10.10	10.11	10.21
10.32	10.24	10.76	10.24	9.87		10.24	10.36	10.37	9.92	10.04
10.62	10.28	9.93	10.29	9.89		10.09	10.14	10.18	10.27	10.33
11.72	10.32	10.12	10.39	9.94		10.35	10.02	10.08	10.14	10.23
10.02	10.72	10.12	9.93	10.00		10.24	10.31	10.40	9.85	9.94
10.08	11.25	10.15	9.99	10.10		9.97	10.01	10.05	10.13	10.18
10.15	11.56	10.19	10.18	10.23		10.20	9.91	9.93	9.94	10.03
10.33	10.18	10.21	10.23	10.34		10.05	10.10	10.16	10.18	9.69
10.50	10.21	10.24	10.24	10.29		9.89	9.91	9.93	9.95	9.97
11.02						10.01	10.03			

Chapter 8

GUIs for Some Statistical Concepts

Jennifer Harlow

Abstract

The visual display of data is an important aspect of statistics. Most of the work in this area has concerned data analysis but visual tools are also relevant to understanding statistical methods and concepts. This paper discusses the development of graphical user interfaces (GUIs) to illustrate two topics in the STAT218 Computational Statistics course: sampling from a multinomial distribution, and the von Neumann Rejection Sampler.

Introduction & Motivation

The visual or graphical display of data as part of statistical analysis is commonly dated to the late 18th century when Priestly and Playfair used graphics to illustrate their works. Since Tukey's influential *Exploratory Data Analysis* (Tukey, 1977) "made statistical graphs respectable" (Tufte, 1983, p.53), visual displays have become increasingly important. Developments in computing power and graphics hardware and software have been exploited to be able to handle and visualise datasets of a size and complexity that would probably stun Tukey, let alone Playfair (Card, Mackinlay & Shneiderman, 1999). The principles for good visual displays, such as those set out in Tufte's own classic *The Visual Display of Quantitative Information* (Tufte, 1983) and the deployment of computing power for the graphical display of statistical data have been focused on rapid data analysis and interpretation.

For students, graphics may need to fulfil a different role. When data is displayed, often it is the illustration of concepts through data rather than analysis of the data itself that is important: the emphasis changes from information density to information clarity. Computer packages offer powerful graphics capabilities for data analysis but learning to use the package may distract from the central point of understanding the concepts (Nolan & Temple, 2003). Sedig & Liang (2008) use the term visual cognitive tools (VCTs) for the "external mental aids that maintain and display visual representations of information", where 'information' here means "structures, objects, concepts, ideas and problems" (Sedig & Liang, 2008p. 147), not just the data in an investigation. Graphical displays which are ideal for data analysis have to be adapted for students in order to become an effective VCT to illustrate the actual processes undertaken by the statistician or elucidate an underlying principle.

This project uses two examples from the STAT218 Computational Statistics course (2008) to explore the potential for the use of graphical user interfaces (GUIs) as VCTs for this course. The examples chosen are the multinomial distribution and the von Neumann Rejection Sampler.

The multinomial distribution is a discrete probability distribution which can be thought of as the outcome from n independent trials where each trial results in exactly one of some fixed finite number k of possible outcomes and the probability $f(i)$ of outcome $i \in \{1, 2...k\}$ in any one trial is θ_i, such that $\sum_{i=1}^{k} \theta_i = 1$ and $f(i) = 0$ for $i \notin \{1, 2...k\}$. A multinomial random variable parameterised by (n, θ) is a random vector $X = (X_1, X_2, ..., X_k)$ where each X_i is the number of times that outcome $i \in \{1, 2, ..., k\}$ is observed in the n trials and $\theta = (\theta_1, \theta_2, ..., \theta_k)$. The binomial is the $k = 2$ case of the multinomial. For simplicity, in this project the multinomials used have $\theta_i = \frac{1}{k}$, i.e., each of the $i \in \{1, 2, ..., k\}$ outcomes of a single trial is equally likely.

Rejection sampling, as developed by John von Neumann, is also known as the acceptance-rejection technique. It is a method to draw independent samples from a target random variable X with probability density function $f(x)$ in situations where it is difficult to directly generate samples from X. This is typically because the desired $f(x)$ is empirical or formed from a mixture model rather than having a form which can be conveniently integrated and an inverse inverse sampling function formed (Robert & Casella, 1999).

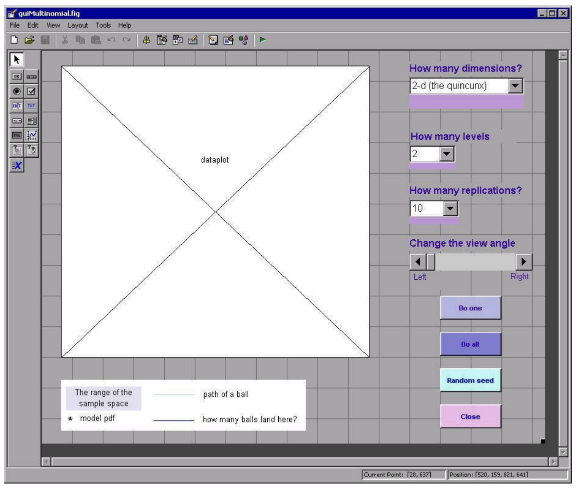
Both of these topics can be difficult for students. In the case of the multinomial, visualisation and interaction can help to relate the behaviour and characteristics of a multinomial random variable to the (relatively student-friendly) binomial and to explore how the multinomial probability mass function relates to the number of trials n. In the case of rejection sampling, the sampling algorithm can be demonstrated by creating visual displays of the process broken down into steps, and the student's belief in the process can be affirmed by showing how the shape of the distribution of the accepted samples becomes more like the target density function as the sample size increases.

Materials & Methods

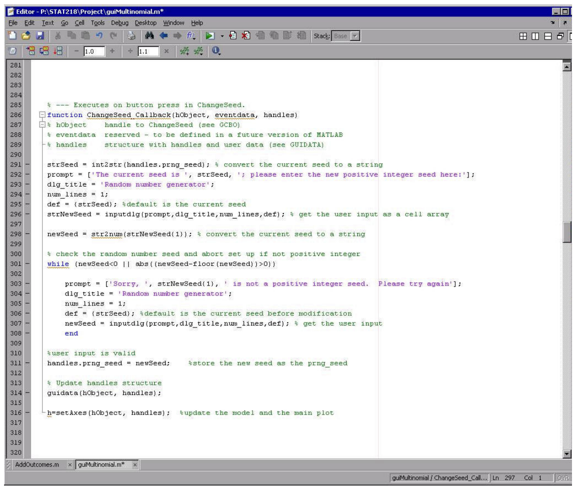
An interface, in computer terms, is the way in which a human user communicates with computer processes or applications (Chen, 2003). A graphical user interface (GUI) is an interface which allows the user to interact with the computer through graphical objects (objects rendered on a screen). Most GUIs are built on the so-called WIMP model (windows, icons, menus, and push-buttons/point-and-click/pull-and-drag) (Broneck, 2003). In computing environments many users are now so familiar with graphical user interfaces that they are largely unaware of alternatives such as command-line instructions.

The two GUIs for this project were developed in MATLAB using MATLAB GUIDE graphical user interface development environment. A MATLAB GUI consists of a base figure or graphical object and a 'm-file' containing the functions which drive the behaviour of the GUI, written in MATLAB's programming language. Other components of the GUI are added to the base figure. These other components can include objects such as graphs, push buttons, displays of text, pop-up lists, areas for user input, etc. Customised menus and toolbars can also be created and added. The responses of the GUI to user actions (such as pressing a button) are programmed in the m-file.

GUI programming is 'event-driven', i.e., functions are called or activated by events. For any component of a GUI, including the main figure, different responses can be programmed for different events concerning that component. The function associated with a particular event is the 'callback' of that event for that component. For example, a button to close the GUI will have a callback function which contains the code to close the figure and programming for that button will specify that this is the callback activated on the button's click event. Most but not all events are initiated by user actions. The callbacks can themselves initiate other GUIs, such as messages to the user (message boxes), or requests for user input. Figure 8.1 below shows the Multinomial GUI in its design state together with the code for the callback function for the Close button



(a) The Multinomial GUI in design state


Editor - P:\STAT218\Project\guiMultinomial.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
AddOutcomes.m
guiMultinomial.m*
guiMultinomial / ChangeSeed_Call... Ln 297 Col 1 OVR

```
% --- Executes on button press in ChangeSeed.
function ChangeSeed_Callback(hObject, eventdata, handles)
% hObject    handle to ChangeSeed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

strSeed = int2str(handles.prng_seed); % convert the current seed to a string
prompt = ['The current seed is ', strSeed, '; please enter the new positive integer seed here:'];
dlg_title = 'Random number generator';
num_lines = 1;
def = {strSeed}; %default is the current seed
strNewSeed = inputdlg(prompt,dlg_title,num_lines,def); % get the user input as a cell array

newSeed = str2num(strNewSeed{1}); % convert the current seed to a string

% check the random number seed and abort set up if not positive integer
while (newSeed<0 || abs((newSeed-floor(newSeed))>0))

    prompt = ['Sorry, ', strNewSeed{1}, ' is not a positive integer seed.  Please try again'];
    dlg_title = 'Random number generator';
    num_lines = 1;
    def = {strSeed}; %default is the current seed before modification
    newSeed = inputdlg(prompt,dlg_title,num_lines,def); % get the user input
    end

%user input is valid
handles.prng_seed = newSeed;    %store the new seed as the prng_seed

% Update handles structure
guidata(hObject, handles);

h=setAxes(hObject, handles);  %update the model and the main plot
```

(b) Callback function for the Close button

Figure 8.1: Designing and programming a GUI using MATLAB and GUIDE

Using the event-driven programming framework, the GUI can be programmed to respond to user actions by showing information, changing displays, performing calculations, initiating further events or calling other functions, amongst many other responses.

The Multinomial GUI

The programming code used in the Multinomial GUI is closely based on code developed by Bry Ashman and Raazesh Sainudiin and made available to the author by Raazesh Sainudiin. The author gratefully acknowledges the use of this code in the GUI. This code simulates samples from a multinomial distribution using a fundamental sampler of pseudo-random numbers. This GUI also used code developed by Raazesh Sainudiin to calculate the value of the probability mass function $f(x : n, \theta)$ for given realisation x of a multinomial random vector X parameterised by n and $\theta = (\frac{1}{k}, \frac{1}{k}, ..., \frac{1}{k})$.

To this basis the author has added code to generate the sample space of a multinomial (n, θ) experiment (θ is a k-dimensional vector $(\frac{1}{k}, \frac{1}{k}, ..., \frac{1}{k})$ as above). This code is shown in Appendix A. The author also simplified and standardised original formulation of Ashman and Sainudiin, which had slightly different treatments for the multinomial with $k = 2$ and $k = 3$, so as to avoid duplicated coding for essentially similar routines.

The remaining programming of the GUI, relating to the functionality of components, was carried out by the author.

The Rejection Sampler GUI

The programming code related to rejection-sampling from a Laplace ($\lambda = 1.0$) distribution to simulate samples from a Normal(0,1) distribution was written by the author based on the algorithms and functions used during the course (Sainudiin & Lee, 2008). Code for rejection sampling from another distribution in the GUI was written by the author. The procedure for calculating the bin widths and heights of each bin for a histogram of the sampled data was provided by Raazesh Sainudiin. The author gratefully acknowledges the use of this procedure in this project.

As with the Multinomial GUI, the programming of the GUI relating to the functionality of the components was carried out by the author.

Methodology

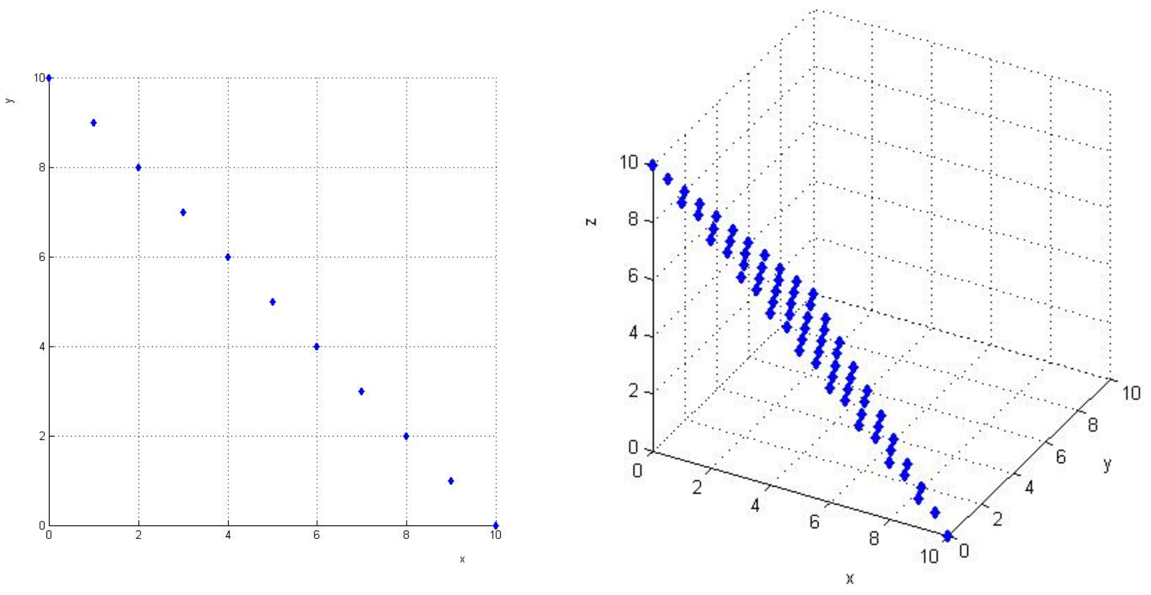
The Multinomial GUI

The Multinomial GUI requires a way to represent the multinomial distribution visually. In this project, a directional travel analogy is used. The sequence of trials is represented as a journey in which the outcome of any of the n trials is interpreted as the direction taken at that point (out of k possible directions) and the final outcome can be interpreted in terms of an end point of the journey in relation to the starting point. A multinomial random vector $X = (X_1, X_2, ...X_k)$ is the summative effect of n separate independent and identically distributed directional movements. This builds on the substantial work already done in this course using a directional travel analogy, including the re-creation of Galton's Quincunx by Ashman and Lawrence (2007). In keeping with the Quincunx approach, the GUI depicts a sample, when in motion (i.e., accumulating its trials, or the equivalent of dropping from peg to peg for the Quincunx), as a ball.

For this project, the direction analogy is interpreted as travel in orthogonal directions. A multinomial random vector $X = (X_1, X_2, ...X_k)$ is considered as the sum of n independent and identically distributed trials in which the result of any one trial is the i^{th} eigen vector (the k-dimensional vector where all elements except the i^{th} are 0 and the i^{th} element is 1). The $k = 2$ case or binomial can be visually represented on a Cartesian (x, y) axis. Using this analogy each trial can result in travel either in the direction (1,0) or the direction (0,1) (the model used in the project assumes that $\theta_1 = \theta_2 = \frac{1}{k} = \frac{1}{2}$). Thus any outcome in the sample space of all possible outcomes can be represented visually as points on a scatter plot drawn on an (x, y) Cartesian axis.

The analogy can be expanded by graphically represented the 'path' taken by a particular sample. For any trial (or stage of the journey) m ($1 \leq m \leq n$), a sample x can be thought of as being at the point represented by $\sum_{j=1}^{m} e_j$ where e_j is the eigen vector which is the outcome of the j^{th} trial (i.e., either (1,0) or (0,1)). This point can be plotted on the Cartesian (x, y) axis for each $m = 1, 2, ..., n$ and thus the 'path' of the sample plotted. This demonstrate that different paths can lead to the same outcome in the sample space and provides a visual interpretation of the binomial coefficient $\binom{n}{x}$.

The same approach can be taken for a $k = 3$ multinomial, plotting outcomes on an (x, y, z) Cartesian axes. This works reasonably well and provides a way to visualise the sample space of a multinomial(n, θ) $(\theta = (\theta_1, \theta_2, \theta_3))$ random variable as $(n + 1) \times (n + 2)$ points on a 2-simplex (triangle) on a sloping plane (for the project GUI, $\theta = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, i.e., in any trial, travel in the x, y or z directions is equally likely). The representations on Cartesian axes of the sample space for a multinomial random vector with number of trials $n = 10$ are shown below in Figure 8.2 for $k = 2$ and $k = 3$. The 3-d version of the Quincunx has been termed the Septcunx (Sainudiin & Lee, 2008).



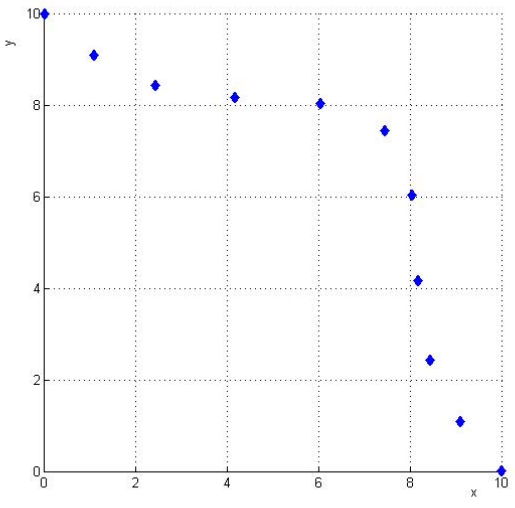
(a) n=10 and k=2 (b) n=10, k=3

Figure 8.2: Multinomial sample spaces represented on Cartesian axes

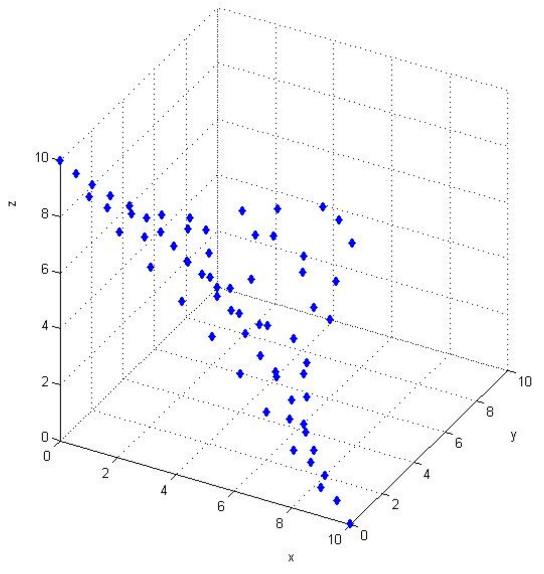
The major disadvantage of the Cartesian coordinate representation is that it is not good for representing $k > 3$ multinomials. Directions could be interpreted as turns of less than 90 degrees, but the resulting representation is not as intuitively clear to the viewer. An alternative representation which retains some aspects of the travel/direction analogy but which can be adapted for higher dimensions is the use of orbit plots or tours. An orbit plot uses multiple axes and projects a high-dimensional representation into a number of low-order representations on the different axes (Young, Valero-Mora, & Friendly, 2006). This would be interesting to explore but is beyond the scope of this project.

The other issue for the Multinomial GUI is how to convey information about the probability mass function

(PMF) of the multinomial - how to plot an outcome in the sample space in such a way that it also gives some sense of the probability of that outcome. Ashman and Sainudiin's original code only plotted the sample space for the binomial case and used an approach which projects the point out from the plane containing the sample space. The projection is orthogonal to the plane and the length of the projection is proportional to the PMF for that outcome and to the number of trials n. This seems to succeed well in giving an impression of the PMF but does means that the point actually plotted to represent an outcome is a transformation of the Cartesian representation of the outcome vector itself. Similarly the proportion of realisations having a particular outcome can be represented by a version of the stem plot, or actually drawing the projected line from the representation of the outcome on the sample space plane and giving it a length proportional to the proportion of realisations having that outcome. The author made some minor adjustments to the scaling of the projection from the sample plane and used the same approach to plot representations of outcomes in the sample space of $k = 3$ multinomials, adjusting the scaling to allow for the fact that a multinomial with n trials has $(n+1) \times (n+2)$ possible outcomes (a quadratic in n) whereas a binomial with n trials has $(n+1)$ possible outcomes. The scaling factor has to be increased for the $k = 3$ depiction so that the rise and fall of the PMF is still visible, but this does mean that comparisons with a depiction for the $k = 2$ case are even less meaningful. The representations on Cartesian axes of the sample space with PDF-scaling for a multinomial random vector with number of trials $n = 10$ are shown below in Figure 8.3 for $k = 2$ and $k = 3$.



(a) n=10 and k=2



(b) n=10, k=3

Figure 8.3: Multinomial sample spaces represented on Cartesian axes with scaling for PMF

There is some danger that viewers may be confused by the transformation of the sample space or the addition of stem plot lines. An alternative which could be used for the PMF is to colour each outcome in the sample space according to its PMF, using a colour range that viewers will associate with height or a sense of extremes. Possibilities include blue $\rightarrow$ green $\rightarrow$ brown $\rightarrow$ whitey-blue (associating to maps where relief is denoted with these colours, blue for water, whitey-blue for snow-capped mountains), or blue $\rightarrow$ red (associating with temperature, blue for cold and low, red for hot and high). Figure 8.4 shows the sample space of a multinomial random vector ($n = 10$ and $k = 3$) where the markers representing outcomes in the sample space are coloured according to the PMF of those outcomes, using blues for outcomes with the lowest PMF and reds for highest (the very pretty effect achieved may be somewhat lost on readers in black and white). Similarly marker sizes proportional to the PMF could be used. However, although both of these methods have some potential for representing and contrasting the sample spaces of different multinomials ($k = 2$ or $k = 3$) for different values of the parameter n, trying to overlay information about the proportion of realisations of each possible outcome in sample space so that it can be distinguished from information about the PMF, it is difficult to achieve the desired visual effect without again resorting to drawing some sort of projection.

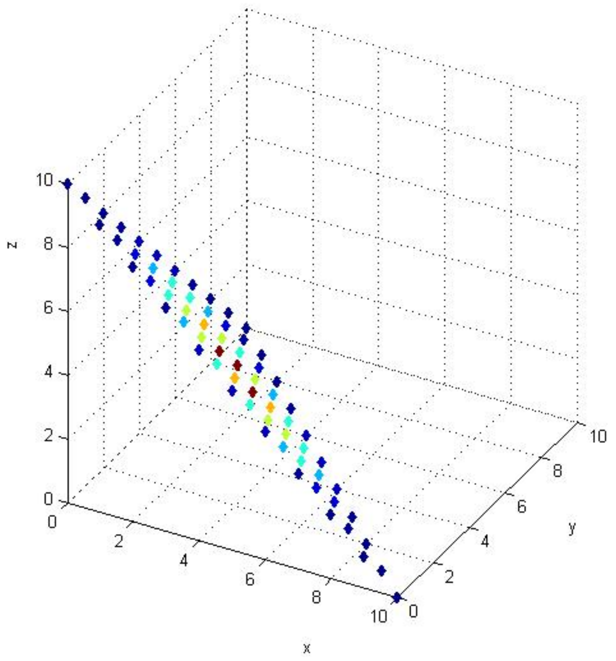


Figure 8.4: Sample space of multinomial n=10, k=3 with PMF indicated by marker colouring from blue (low) to red (high).

It should also be emphasised that the whole travel/direction analogy is just one way in which the multinomial could be represented visually. Many other representations would be possible. For example, we could have used an analogy like a fruit basket. After n independent additions of a random fruit from k possible types of fruit to a basket, the contents of a basket can be considered as a multinomial (n, θ) random variable. This might be fun graphically (although fruit are not offered as part of the standard componentary in GUIDE), but presents difficulties in representing the many overflowing baskets of fruit that would result from a large n in a small space on the GUI. MATLAB does have functions for creating sounds of different frequency and volume. It would be possible to represent the realisation of a multinomial random vector aurally, but a group, such as a number of samples or all the outcomes in the sample space, would be harder to deal with coherently. The author did some limited experimentation with sounds but found that a communal computer laboratory is not the best venue for this.

The Rejection Sampler GUI

Rejection sampling is a method to draw independent samples from a target random variable X with probability density function $f(x)$ in situations where it is difficult to directly generate samples from X. Rejection sampling involves sampling from a more tractable function g but adding a step to randomly accept or reject this sample. The decision to accept or reject is independent of the sample itself but is such that the density of the samples accepted, over the support of X, mirrors that of the targeted density function $f(x)$. This can be done by pairing each sample x_p proposed from g with an independent sample u drawn from the Uniform$[0, 1]$ distribution and accepting the proposed sample x_p only if $u \leq \frac{f(x_p)}{g(x_p)}$. The proposal function g must be such that $f(x) \leq g(x)$ for any x in the support of X (and the support of g must contain the support of f). This may mean that a parametric probability density function is scaled to give g (von Neumann's letter to Stanilaw Ulam in 1947 in which he first described his method used two uniform distributions, one for the proposal function and one for the acceptance-rejection decisions (Eckhardt, 1989)). Von Neumann and Ulam were discussing the trials and tribulations of simulating neuron diffusion at the Los Alamos National Laboratory; our use in the Rejection Sampler GUI is somewhat less exciting. The main example incorporated into the GUI is simulating samples from X as a Normal(0,1) random variable using rejection sampling with proposals from a Laplace (λ=1.0) distribution.

In this situation we need to find the constant a such that $f(x) \leq a \times g(x)$ for all x in the support of X. It can be shown that for the Normal(0,1) and Laplace(1.0) distributions, a = $\sqrt{\frac{2}{\pi}} exp(\frac{1}{2}) \approx 1.3155$ (Sainudiin & Lee, 2008). The proposal function used in the GUI is therefore the Laplace scaled by 1.3155. The rejection sampling method used to get one sample from the target distribution follows the steps outlined above, where sampling from the Laplace is performed using an inversion sampler.

The target and proposal functions are plotted on the same (x, y) axis in the GUI. A lower and upper bound for the x-axis for each available target function is specified in the code. The support of a Normal(0,1) random variable is $(-\infty, +\infty)$ but in the GUI the plot is shown over the interval $-5 \leq X \geq 5$. The plots of the continuous functions f and g are created by generating a set of discrete points on the x-axis 0.01 apart, calculating $f(x)$ and $g(x)$ (hence $ag(x)$) for each x-value and plotting a line through all of the pairs $(x, f(x))$ and $(x, ag(x))$.

The Rejection Sampler GUI is programmed to break the rejection sampling process up so that the user can 'walk through' the algorithm a step at a time. The GUI also allows the user to simulate samples in batches of 50 or 100, or to complete a full sample of specified size from a list of 10 (unequally spaced) sample sizes between 10 and 50,000.

When the step-by-step mode is used the proposal x_p from g is shown mapped onto ag and f as the points $(x_p, ag(x_p))$ and $(x_p, f(x_p))$. The 'bound' for rejection sampling $f(x_p)/ag(x_p)$ is then calculated. The bound is illustrated as a red line across a second (x, y) axis adjacent to the plot of the target and proposal functions. The y-axis in this second plot is scaled to to have the same height as $ag(x_p)$ and is labeled with 0 (the minumum, at the level of $y = 0$ on the target & proposal plot) and 1 (the maximum, level with the height of the proposal mapped onto ag $(a(g(x)))$ in the target and proposal plot). A sample u from a Uniform(0,1) random variable is then generated and plotted on the second (x, y) axis so that it will be on or below the red line representing the bound if $u \leq f(x_p)/ag(x_p)$ and above if $u > f(x_p)/ag(x_p)$. Thus the second plot shows whether this proposal x_p is to be accepted or rejected. If it is accepted the total number of samples shown in the GUI is incremented by one, otherwise the total number of rejections shown is incremented.

A vector of the accepted samples (the Sample) is maintained in the GUI code and when the number of samples is greater than one a histogram of the Sample is depicted after a proposal is accepted (if in step-by-step mode) or after a batch of samples has been added to the Sample. The bin-width chosen for a histogram has "an enormous effect on the appearance of the resulting histogram" (Wand, 1997, p. 59), and can reveal or hide important features of the data. The Rejection Sampler GUI needs to have a robust but flexible method of calculating bin widths to accommodated different sizes of samples from different distributions rather than one fixed bin width for all data. The method developed by Wand (1997) used in the GUI attempts to estimate, from the sample to be histogrammed, a mean integrated squared error optimal (MISE-optimal) bin width for the underling probability density function f from which the data is drawn. In general, for small sample sizes the bin width will be larger but the method used is complex and takes into account more features of the data than just the sample size. As stated above, the code implementing the Wand method was supplied to the author by Raazesh Sainudiin. Interested readers are referred to Wand (1997) for the full treatment of the method. At present no attempt is made in the GUI to explain or comment on the histogram or the reason for changing bin sizes. Strictly, the GUI should also not attempt to draw the histogram at all for small sample sizes. The rationale for doing so is it that, by showing it for very small sizes, the user is able to get a sense of how meaningless such a graphic actually is in these cases.

A different example is also included in the Rejection Sampler GUI where the target function is a quadratic of the form $f_c(x) = e^{(-2+cos(c_1x+c_2x^2)+sin(c_3x+c_4x^2))}$ over $x \in [-10, 10]$ ($f_c(x)$ is the unnormalised form, i.e., [-10,10] $\int_{-10}^{10} f(x)\,dx \neq 1$). In this case the proposal distribution is the Uniform(-10,10) and the constant a needed to ensure that the proposal function envelopes the target is $a = 20$. This function is included as a better example than the Normal of the kind of target density f that rejection sampling may be a good choice of sampling method for.

The fundamental sampler

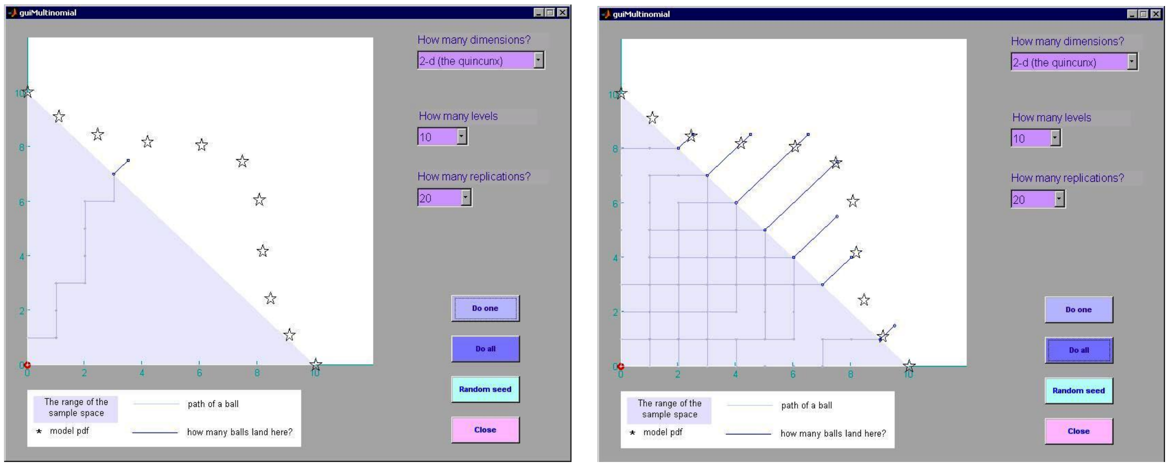
In both GUIs, sampling from a uniform (0,1) distribution is carried out using MATLAB's rand() function. This generates pseudo-random numbers using the Mersenne Twister algorithm by Nishimura and Matsumoto ("Mersenne Twister home page", n.d.) by default. Both GUI's specify a default seed in the code and include a push-button component to allow the user to examine and change the seed.

Results

This section of the project report uses screen shots of the Multinomial GUI and Rejection Sampler GUI in action illustrate the results of the project.

The Multinomial GUI

Figure 8.5 shows the Multnomial GUI in Quincunx ($k = 2$) mode, simulating first one and then 20 replications from a multinomial random vector with $n = 10$, $\theta = (\frac{1}{2}, \frac{1}{2})$. The range of the sample space is indicated with a light blue simplex (which is a 1-simplex or line for 2-dimensional coordinates (x, y)), and the area under the simplex is also filled in light blue. The outcomes in the sample space, scaled as described above, are represented with a pentagram-star and the proportion of the replications resulting in each outcome in the sample space is indicated with the stem plot. The 'history path' of a replication from the origin (0,0) to its final representation is shown below the simplex.


guiMultinomial

How many dimensions? 2-d (the quincunx)
How many levels 10
How many replications? 20

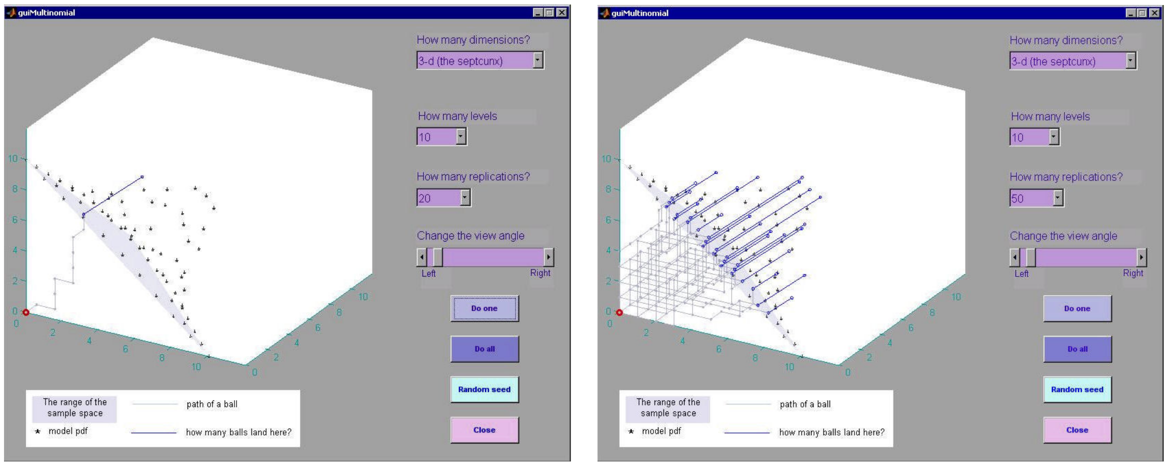
guiMultinomial

How many dimensions? 2-d (the quincunx)
How many levels 10
How many replications? 20

(a) n=10 and k=2, one replication completed (b) n=10, k=2, 20 replications completed

Figure 8.5: The Multinomial GUI in Quincunx form

Equivalent screenshots for the Multnomial GUI in Septcunx ($k = 3$) mode are shown below in Figure 8.6, simulating first one and then 50 replications from a Multinomial random vector with $n = 10$, $\theta = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The simplex is now a 2-simplex or triangle.


guiMultinomial

How many dimensions? 3-d (the septcunx)
How many levels 10
How many replications? 20

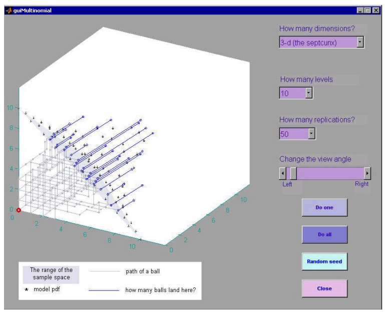
guiMultinomial

How many dimensions? 3-d (the septcunx)
How many levels 10
How many replications? 50

(a) n=10 and k=3, one replication completed (b) n=10, k=3, 50 replications completed

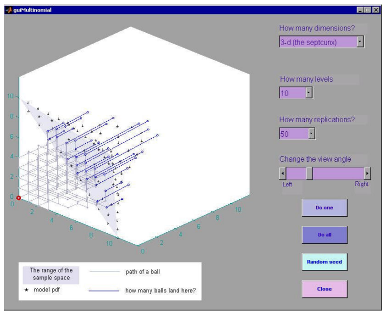
Figure 8.6: The Multinomial GUI in Septcunx form

When the GUI is put into Septcunx mode a default viewing position is provided which angles the view to try to maximise the 3-d effect that the GUI aims for. The user is also provided with an additional control in the form of a slider bar which allows them to interactively rotate the Septcunx plot through about 60 degrees about the vertical axis. Figure 8.7 below shows the effect of rotating the view of the Septcunx using the slider.



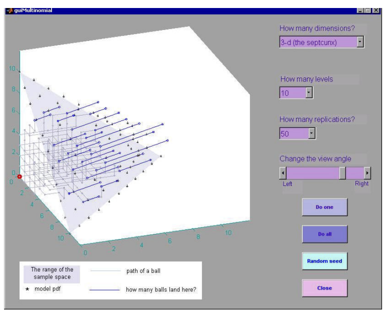
How many dimensions? 3-d (the septcunx)
How many levels 10
How many replications? 50
Change the view angle

(a) Starting point of rotation



How many dimensions? 3-d (the septcunx)
How many levels 10
How many replications? 50
Change the view angle

(b) Part-way through rotation



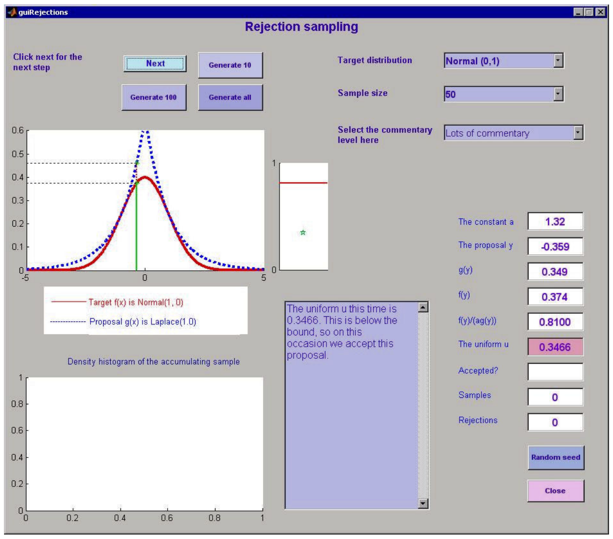
How many dimensions? 3-d (the septcunx)
How many levels 10
How many replications? 50
Change the view angle

(c) Towards the end of a rotation

Figure 8.7: The Multinomial GUI in Septcunx form, n=10 and k=3, 50 replications completed

The Rejection Sampler GUI

A screen shot of the Rejection Sampler GUI mid-way through the process of simulating one sample step-by-step is shown below in Figure 8.8. At this stage the bound $\frac{f(x)}{ag(x})$ for a proposal x is being compared to a Uniform(0,1) random variable u. u in this case is below the bound and so the proposal will be accepted.

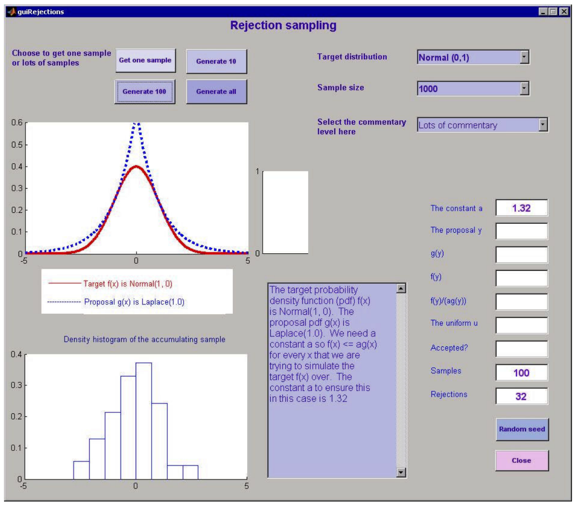

guiRejections
Rejection sampling
Click next for the next step
Next
Generate 10
Generate 100
Generate all
Target f(x) is Normal(1, 0)
Proposal g(x) is Laplace(1.0)
Density histogram of the accumulating sample
Random seed
Close
Target distribution Normal (0,1)
Sample size 50
Select the commentary level here Lots of commentary

The uniform u this time is 0.3466. This is below the bound, so on this occasion we accept this proposal.

The constant a	1.32
The proposal y	-0.359
g(y)	0.349
f(y)	0.374
f(y)/(ag(y))	0.8100
The uniform u	0.3466
Accepted?	
Samples	0
Rejections	0

Figure 8.8: Part-way through simulating one sample: comparing u with $f(x)/ag(x)$

The accumulated sample at any point is shown in a histogram. Figure 8.9 shows the Rejection Sampler after simulating 100 and then 10,000 samples from the target Normal(0,1) distribution, illustrating how the sample will look more and more like the target density function as the sample size increases. Figure 8.9 also shows the change in bin width used for the histogram.


guiRejections
Rejection sampling
Get one sample
Generate 10
Generate 100
Generate all
Target f(x) is Normal(1, 0)
Proposal g(x) is Laplace(1.0)
Density histogram of the accumulating sample
Random seed
Close

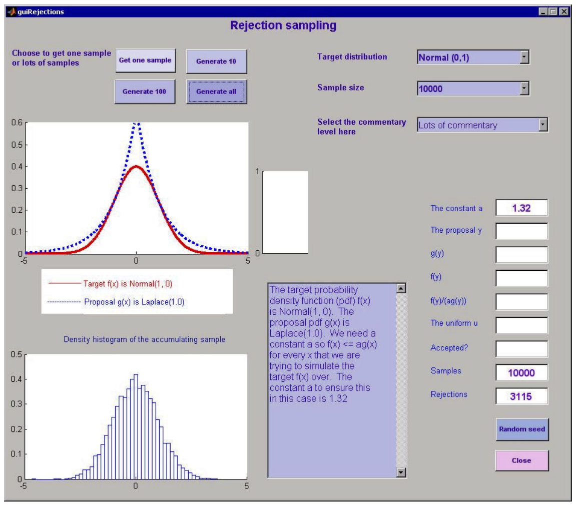
Choose to get one sample or lots of samples

Target distribution: Normal (0,1)
Sample size: 1000
Select the commentary level here: Lots of commentary

The target probability density function (pdf) f(x) is Normal(1, 0). The proposal pdf g(x) is Laplace(1.0). We need a constant a so f(x) <= ag(x) for every x that we are trying to simulate the target f(x) over. The constant a to ensure this in this case is 1.32

The constant a	1.32
The proposal y	
g(y)	
f(y)	
f(y)/(ag(y))	
The uniform u	
Accepted?	
Samples	100
Rejections	32

(a) Sample size 100


guiRejections
Rejection sampling
Get one sample
Generate 10
Generate 100
Generate all
Target f(x) is Normal(1, 0)
Proposal g(x) is Laplace(1.0)
Density histogram of the accumulating sample
Random seed
Close

Choose to get one sample or lots of samples

Target distribution: Normal (0,1)
Sample size: 10000
Select the commentary level here: Lots of commentary

The target probability density function (pdf) f(x) is Normal(1, 0). The proposal pdf g(x) is Laplace(1.0). We need a constant a so f(x) <= ag(x) for every x that we are trying to simulate the target f(x) over. The constant a to ensure this in this case is 1.32

The constant a	1.32
The proposal y	
g(y)	
f(y)	
f(y)/(ag(y))	
The uniform u	
Accepted?	
Samples	10000
Rejections	3115

(b) Sample size 10,000

Figure 8.9: The Rejection Sampler GUI with sample histograms

Conclusion

This project has explored how GUIs based on two topics covered in STAT218 can be used as visual cognitive tools (VCTs) to enhance student learning. The project GUIs do this by adapting graphical displays already used in the course in the following ways:

- Break processes into steps with illustrations of each stage, highlighting important points;
- Allow the user to compare the effects of using different parameters;
- Allow the user controlled interaction with the data and displays;
- Provide the graphical data displays for the user so that the user is not distracted by the mechanics of creation.

Potential Modification

GUI design and human interaction technologies are specialised fields in their own right. This project is intended as a demonstration of the potential for GUIs as VCTs, and the actual GUIs created probably violate most if not all of the rules for good GUI design. Considerable improvements could be made on the design of these GUIs as human/technology interfaces. In addition the programming could undoubtedly be improved and made cleaner and more efficient.

Both GUIs could also be expanded, or different versions created, to illustrate different aspects their topics. For example, with the Multinomial GUI, Galton's original purpose of the Quincunx could be emulated and the GUI used to demonstrate the Central Limit Theorem.

More ways in which the user can move and rotate the plot of multinomial outcomes could be added to the Multinomial GUI. More emphasis could be given to the meaning of the multinomial coefficient $\binom{n}{x_1,x_2,\ldots x_k} = \frac{n!}{x_1!x_2!\ldots x_k!}$, possibly by tracking how many different paths in the Cartesian representation of the multinomial can lead to each outcome.

It would also be interesting to depict the multinomial in ways other than the Cartesian representation used in this project and possibly to include multinomials with $k > 3$. Representation through sound or orbit plots could be explored.

Other target and sampling distributions could be added to the rejection sampler. The efficiency of the sampler (the total number of proposals required to obtain a sample of a given size) could be quantified and the number of rejections related to the difference between the area under the proposal function $g(x)$ and the area under the target $f(x)$ over the support of X. The user could try different proposal functions to explore their efficiency. This could lead to an exploration of the density function of the number of rejections (which is geometric, with parameter related to the efficiency of the sampler).

Finally, other GUIs could be constructed. At a simpler level than the rejection sampler, a similar GUI could illustrate inversion sampling for various random variables. Or, a GUI might be a good way to demonstrate the Central Limit Theorum in action or to illustrate bootstrapping or parametric hypothesis testing.

References

Ashman, B., & Lawrence, R. (2007). Estimating the binomial probability p for Galton's Quincunx. In *Computational statistical experiments: STAT218-07S2(C) Student Projects Report UCDMS 2008/5 (pp. 47-51).* Retrieved October 4, 2008, from `http://www.math.canterbury.ac.nz/~r.sainudiin/courses/STAT218/projects/Stat218StudentProjects2007.pdf`

Broneck, K. (2003). Graphical user interface. In *Encyclopedia of new media* (pp. 207-209). Thousand Oaks, CA: Sage.

Card, S.K., Mackinley, J.D., & Shneiderman, B. (Eds.). (1999). *Readings in information visualization: Using vision to think.* San Francisco, CA: Morgan Kaufmann.

Chen, S.S. (2003). Interface. In *Encyclopedia of new media* (pp. 244-245). Thousand Oaks, CA: Sage.

Eckhardt, R. (1989). Stan Ulam, John von Neumann, and the Monte Carlo method. In N.G Cooper (Ed.), *From cardinals to chaos: Reflections on the life of Stanislaw Ulam.* (pp.131-137). Cambridge, NY: Cambridge University Press.

Mersenne Twister home page (n.d.). Retrieved October 16, 2008, from `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html`

Nolan, D., & Temple Lang, D. (2003). *Case Studies and computing: Broadening the scope of statistical education.* Retrieved October 14, 2008, from `http://www.stat.berkley.edu/~nolan/Papers/isi03.pdf`

Robert, C.P, & Casella, G. (1999). *Monte Carlo statistical methods.* New York, NY: Springer.

Sedig, K., & Liang, H. (2008). Learner-information interaction: A macro-level framework characterizing visual cognitive tools. *Journal of Interactive Learning Research*, 19(1), 147-173.

Sainudiin, R., & Lee, D. (2008). *Computational statistical eExperiments I version 0.2SL.* Retrieved October 16, 2008, from `http://www.math.canterbury.ac.nz/~r.sainudiin/courses/STAT218/Lectures.pdf`

Tufte, E.R. (1983). *The visual display of quantitative information.* Cheshire, CN: Graphics Press.

Tukey, J.W. (1977). *Exploratory data analysis.* Reading, MA: Addison-Wesley.

Wand, M.P. (1997). Data-based choice of histogram bin width. *The American Statistician, 51*(1), 59-64.

Young, F., W., Valero-More, P.M., & Friendly, M. (2006). *Visual Statistics: Seeing data with dynamic interactive graphics.* Hoboken, NJ: John Wiley & Sons.

Appendix A

matlab m-files used to generate the sample space of a Multinomial experiment

```
function x=MultinomialOutcomes(n, k)
% create a matrix of the sample space assuming equiprobable thetas
% inputs :  n the number of trials in the multinomial
%           and k which for an equiprobable multinomial defines theta
%            to that theta(i) = 1/k for each dimension i=1,2 ... k
%
A = zeros(1,k); % starting point
y = AddOutcomes(A, 1, n, k); % call function to do the hard work
%cleans up extra rows of zeros which will come back in the
% A returned by AddOutcomes;
while sum(y(1,:))==0% sum of first row of y
    T=size(y,1); % rows of y
    y=y(2:T,1:k); % take off the top row since it's zeros
end;
x=y; % return the cleaned up y



function x=AddOutcomes(A, j, n, k)
% called as a recursive function from MultOutcomes()
% fills in the sample space of the Multinomial by calling itself
% inputs :  a 1xk vector A
%           and the column we are now filling j
%           and n the number of trials in the multinomial
%           and k which for an equiprobable multinomial defines theta
%            to that theta(i) = 1/k for each dimension i=1,2 ... k
%
x=zeros(1,k);
e=zeros(1,k);
e(1,j)=1;
tot=sum(A,2); % the sum of the components of A so far
space = n-tot; % can be zero
if j==k % we are at the end
```

```
    A = A+space*e; % how much we can add (can be zero)
    x=A;
end;
if j<k % not at the end
    for t=0:space
        A(1,j)=t;
        %A = A+(t*e);   % add t of the current component vector
        %x = AddOutcomes(A,j+1,n,k); % and pass it along to fill the next column
        x=vertcat(x,AddOutcomes(A,j+1,n,k));
        while sum(x(1,:))==0% sum of first row of x
            T=size(x,1); % rows of x
            x=x(2:T,1:k);
        end;
    end;
end;
```