Management Science and Information Systems Studies

Final Year Project Report


Ferdia de Paor March 2015
THE STATISTICS DISCIPLINE

A Friendly Web Interface to R Using R Shiny for Some Frequently Used Statistical Analysis Methods

20th March 2015

Prepared By: Ferdia de Paor         Supervisor: Arthur White
I declare that the work described in this dissertation has been carried out in full compliance with the ethical research requirements of the School of Computer Science and Statistics.

Signed: _________________________

Ferdia de Paor

20/03/2015
ABSTRACT

The aim of this project was to develop the first part in an interactive web platform collecting, analysing, processing and visualising many case studies. The web platform was built using the R Shiny framework. The first part of the platform covers four introductory statistical topics and is also useful for seeing the capabilities of Shiny and the potential for this project to progress further.
PREFACE

The Statistics Discipline, hereafter referred to as “the client”, is providing courses in Statistics in Trinity College at both undergraduate and postgraduate levels. The teaching is mainly conveyed in class via lectures, tutorials and computer labs. The main client contact throughout the project was Dr Jason Wyse.

To complement this teaching the client wanted the first part of an interactive web platform to be developed. This platform would help students to illustrate and understand the various course contents delivered by the staff in the Statistics Discipline.

The developed system meets the needs of the client as agreed in the project terms of reference. There is definite scope to expand on the web platform, R shiny has proven to be a very accessible web framework with some powerful functionality.

The main difficulties that arose over the course of the project were issues relating to running the Shiny system off of GitHub on some of the college computers. This was due to issues navigating the college’s proxy server. The development process was challenging but also very rewarding learning how to use a new web framework.

I would like to thank my client contact Jason Wyse for his cooperation and enthusiasm over the course of the project. His feedback was invaluable to the project and he was a pleasure to work with. Acknowledgements must also be given to my supervisor Arthur White, whose advice and statistical expertise ensured the successful completion of this project. I would also like to thank my parents for their constant support throughout my university education, without them I would not be where I am today.
THE STATISTICS DISCIPLINE
A Friendly Web Interface to R Using R Shiny for Some Frequently Used Statistical Analysis Methods

20 March 2015

TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>NO</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>INTRODUCTION AND SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.1 The Client</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.2 The Project Background</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.3 Terms of Reference</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1.4 Project Outcomes</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1.5 Summary</td>
<td>3</td>
</tr>
<tr>
<td>2.</td>
<td>SYSTEM OVERVIEW</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.1 System Objectives</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2.2 System Structure</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2.3 User Interface Walkthrough</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2.4 Technical Environment</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.5 System Overview Diagram</td>
<td>8</td>
</tr>
<tr>
<td>3.</td>
<td>DESCRIPTION OF WORK DONE</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3.1 Requirements Gathering</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3.2 Research</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>3.3 System Design</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>3.4 System Development</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>3.5 Testing and Implementation</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>3.6 Challenges Encountered</td>
<td>17</td>
</tr>
<tr>
<td>4.</td>
<td>CONCLUSIONS AND RECOMMENDATIONS</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>4.1 Conclusions</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>4.2 Recommendations</td>
<td>19</td>
</tr>
</tbody>
</table>
## APPENDICES

<table>
<thead>
<tr>
<th>NO</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Original Project Outline</td>
<td>A.1</td>
</tr>
<tr>
<td>B.</td>
<td>Interim Report</td>
<td>B.1</td>
</tr>
<tr>
<td>C.</td>
<td>User Manual</td>
<td>C.1</td>
</tr>
<tr>
<td>D.</td>
<td>Design Documentation</td>
<td>D.1</td>
</tr>
<tr>
<td>E.</td>
<td>Source Code</td>
<td>E.1</td>
</tr>
</tbody>
</table>

REFERENCES
1. INTRODUCTION AND SUMMARY

This chapter introduces the client, provides the project background and the terms of reference. Also included is a summary of the remaining chapters.

1.1 The Client

Founded in 1966, the Statistics discipline is providing courses in Statistics in Trinity College at both undergraduate and postgraduate levels. The teaching is mainly conveyed in class via lectures, tutorials and computer labs. The client contact for this project was Dr Jason Wyse, from the Statistics Discipline. Dr Wyse currently teaches Stochastic models in space and time.

1.2 The Project Background.

The Statistics Discipline to support teaching for each course, makes use of case studies. These case studies need to be collected, understood and processed with a dedicated software environment to illustrate the statistical methodology applied to different domains. Such a database of case studies needs to be continuously updated to follow the latest science and to keep the students interested with innovations.

The aim of this project is to develop the first part in an interactive web platform collecting, analysing, processing and visualising many case studies. These will then be used by staff and students to illustrate and understand the various course contents delivered by the staff in the Statistics Discipline. The web platform would be built using the R Shiny framework. This web platform would be able to be easily maintained and, if necessary, hosted online so that both students and staff could access it. The first part of the platform would only cover four introductory statistical topics but would be useful to see the capabilities of Shiny and the potential for this project to progress further.

The free environment R (r-project.org) is widely used for statistical computing in research and now also in industry. Most of the staff in the discipline already uses it (with alternatives) and would be familiar with this language. Shiny is a web application framework for R to turn analysis into interactive web applications, giving users the power to process their own datasets easily or run examples provided with user friendly GUIs through an internet browser or through R Studio.
1.3 Terms of Reference

Primary

- Develop an interactive web interface that covers four relevant ubiquitous statistical topics, including
  1. T-tests
  2. Regression
  3. ANOVA
  4. Chi-square tests

- This web interface will be built using the R shiny web application framework
- The Interface will have an emphasis on being user friendly, attractive and approachable in terms of design.

Secondary

- The capacity to upload one’s own dataset to the web interface will be investigated. With an emphasis on data control and the capacity to give a warning message or caveat if the wrong data format is uploaded.

1.4 Project Outcomes

The following section provides a summary of the main outcomes reached in this project.

- An interactive web interface has been developed using R Shiny that covers the four statistical topics agreed with the client and laid out in the terms of reference.
- The web interface makes use of numerous datasets some built in to R and some taken from staff notes.
- The web interface is user friendly with an attractive and approachable user interface.
- The web interface has a section where a user can upload their own dataset and view it. The user will receive a warning message if the dataset is in the wrong format.
- The web interface was not hosted online but is available to run from the source files included on an attached CD as well as available online from a GitHub repository.
1.5 Summary of Report

- **Chapter 2** provides an overview of the developed system and the technical environment.

- **Chapter 3** describes the work done over the course of this project. This includes requirement gathering, research, system design and development, system testing and system implementation.

- **Chapter 4** presents conclusions for the project and recommendations for the further development of the project.
2. SYSTEM OVERVIEW

This chapter provides an overview of the objectives of the developed system. Explains the system’s structure, explains how the system operates, presents the systems overview diagram and also includes a brief summary of the systems technical environment.

2.1 System Objectives

The system objectives were to meet the requirements of the client, as proposed in the terms of reference. Central to this was to develop the first part of an interactive web platform collecting analysing, processing and visualising many case studies. This web platform was to be used by staff and students to illustrate and understand various course contents delivered by the staff in the Statistics discipline.

Currently the client must find individual datasets for teaching purposes and write the R code during lectures or labs to demonstrate statistical concepts. This process can be time consuming as datasets often get lost when staff pass over the teaching of a class to a colleague. The client then has to find new suitable datasets and write the code appropriately. The web platform is designed to help eliminate this problem by storing all the datasets needed online with the analysis already coded. This would make the sharing of teaching materials easier and enable the client to prepare teaching materials in advance.

The system is to be built using the R Shiny web application framework. Shiny is a framework developed by R Studio that allows users to turn their R analysis into interactive web applications. It gives the user the power to process their own datasets easily or run examples provided with user friendly GUIs through an internet browser with little web development knowledge needed. Shiny also gives the user the power to make interactive applications which automatically update when a user changes a parameter - similar to an Excel spreadsheet.

R Shiny was selected by the client as R is a widely used statistical analysis tool by staff and students in the Statistics discipline. The logic of this is that if this web interface was a success R Shiny would be easy for the client to learn as they already have a wealth of experience using R.

The system had a number of specific requirements which were set out in the terms of reference with the client and which were changed slightly during its development. There was also a requirement to cover four ubiquitous statistical topics, including t-tests, regression, ANOVA and Chi-square tests.

A large emphasis was put on the system’s user interface being friendly, attractive and approachable in terms of design. A lot of work was put in to this aspect of the system with considerable time being spent learning Latex so that equations and formulas could be displayed clearly and concisely. A lot of work was also done learning how to implement CSS themes with Shiny to give the web interface a very approachable and clear appearance. Cascading Style Sheets (CSS) is a style language which gives HTML documents a sophisticated look. Since the Shiny app user-interface is an HTML document, you can use CSS to control how you want your Shiny app to look.
In the original terms of reference for the system there was a requirement for the system to have the capacity to allow the user to upload one’s own dataset for analysis. The user would also receive caveats or warning messages if the data appeared to be incompatible with the analysis type chosen. This requirement was later changed after discussion with the client. The client’s primary objective was for the web application to be built as a teaching tool rather than a statistical dashboard capable of analysis. This feature was still partially implemented and a section with an upload feature was included in the finished application that allows the user to upload and view and dataset.

An objective that was suggested during the systems interim presentation was that the project be hosted on a GitHub repository for version control and ease of access for future users of the application. This was implemented and a GitHub profile was created for the client. The use of GitHub also allowed for easier implementation of the system due to Shiny’s capability to be run straight from GitHub by a function run in R Studio.

2.2 System Structure

As already mentioned the Web Interface was implemented as a Shiny application. A Shiny application structurally consists of two main components. These are:

- a user interface R script (ui.R)
- a server R script (server.R)

An R script is simply a text file containing lines of R code. Every Shiny application requires these two scripts saved in the same directory to run. This directory can be saved locally on your own machine or can be stored online in a GitHub repository. The directory name is the name of the app. Other optional files are also kept in this directory saved in a folder labelled ‘www’. The ‘www’ folder is where any files needed by the app are stored such as large datasets, images or CSS files.
User Interface Script (ui.R)

The user interface script controls the layout and appearance of the Shiny app. Shiny has numerous pre-built layouts that make constructing a good looking user interface very straightforward. The user interface doesn't have any R analysis code in it; its sole purpose is the aesthetics of the Shiny app. It tells the application where to put R analysis by calling on the server script functions when they are needed.

Server Script (server.R)

The server script is where all the R analysis goes on and all the calculations and plots are made. The Server script is full of functions - each storing analysis. The user interface script tells the app where everything should be positioned and then calls on the relevant server function to tell it where to put it.

Running a Shiny App

The Shiny app can then be run by using a function in R Studio to compile these scripts into a Shiny app. There are separate functions to run a locally stored app or an app stored online in a GitHub repository. More information on these functions can be found in the user manual in the appendices.

![Diagram of how Shiny runs in R Studio](image)

**FIGURE 2.2.1 – Diagram of how Shiny runs in R Studio**

2.3 User Interface Walkthrough

This section provides a brief walkthrough of the user interface for the application and its structure. The web interface is split up in to six sections and the user can access each of these sections via a navigation bar that runs across the top of the web interface at all times. There are many different ways of setting up the User interface but the Navbar set up is the best for an application with multiple pages.
The navigation bar is accessible at all points while within the app and makes the application easily navigable for the user. The six sections are as follows ‘Home’, ‘T-Tests’, ‘Chi-Square’, ‘ANOVA’, ‘Regression’ and ‘File Upload’. A screenshot of the navigation bar can be seen below.

![FIGURE 2.3.1 – Screenshot of navigation bar](image)

**Home**
This tab is home page of the web interface, which acts as an introductory page to the web interface.

**T-Test**
The first page in this section introduces hypothesis testing and the t-distribution. In the next tab the paired T-test is introduced using the sleep dataset as an example, which is a dataset built in to R.

**Chi-Square**
The Chi-square section starts with an introductory section that introduces the Chi-square and its formula. The next tab introduces a famous dataset of student admissions at Berkeley University in 1975 which is a good example of Simpson’s paradox and also a suitable dataset to demonstrate a Chi-square test.

**ANOVA**
The ANOVA section has an introduction which introduces the main concepts behind ANOVA. The next tab introduces a dataset taken from Eamon’s notes and displays it graphically. The third section explains how you would decompose the variance mathematically by hand and the final section shows how you would do it using the ANOVA table function in R.

**Regression**
The regression section has an introduction that explains regression. The next tab introduces a dataset of spot welds taken again from Eamon’s notes. The final tab displays the data graphically with regression line, it also shows the prediction interval and the confidence interval.

**File Upload**
The file upload section is only one page and demonstrates Shiny’s file upload functionality. It allows the user to upload a CSV file to the app which is then printed for the user. The Application prints a warning message if the wrong file type is uploaded.
2.4 Technical Environment

The web application was developed entirely using the R studio integrated development environment (IDE) implementing the R Shiny package. R-studio is a free and open source integrated development environment for R, a programming language used for statistical computing and Graphics.

The R Shiny package is made by the same developers behind R studio. R Shiny makes it easy to build interactive web applications using R code without any prior web development experience. Each Shiny application is made up of two R scripts - one for the user interface and one for all the R analysis.

R shiny makes use of many prebuilt widgets and wrapper functions that implement other software libraries for Shiny. In making my web interface I made use of a number of these wrapper functions. The MathJax function was used to display mathematical formulas. MathJax converts mathematical notation like Latex into HTML so that it can be displayed in a web browser. Shiny also allows the implementation of Cascading Style Sheets (CSS) to change the theme of the web interface. I was able to make use of free CSS themes to change the appearance of my Shiny app.

A version of the web application was also kept in an online repository on GitHub. GitHub is a popular website for storing code in online repositories. Shiny files kept on Github can be run straight from R Studio using built in functions included in the Shiny package.

2.5 System Overview Diagram

![System Overview Diagram](image)

Figure 2.5.1 – System Overview Diagram

The system overview diagram for a shiny application is rather straight forward but still illustrates clearly how the different components interact with each other. R Studio calls on the Shiny files either from a local directory or from a GitHub repository and compiles them in to a shiny application. R Studio then sends the application on to the web browser.
3. DESCRIPTION OF WORK DONE.

This chapter outlines the work that was completed over the course of the project. The project was achieved in a number of stages over a number of months.

The four stages were:
- an initial requirements gathering stage
- a research stage
- a system design and development stage
- a testing and implementation stage.

The web interface was developed using an incremental development model.

The Incremental Development Model is a software development model where you advance your software in individual steps with each version adding new functionality. The model was appropriate for this project as each version could be presented to the client and a steady stream of feedback could be obtained. I could then incorporate this feedback while working on the next phase of the project. This development model proved advantageous over alternatives like the waterfall model where the client would have only seen the system when it was completed. This way offered the possibility of building on the client’s steady stream of feedback to produce an end product close to their required specifications.

![Incremental Development Model Diagram](image)

**FIGURE 3.0.1 – Incremental Development Model Diagram**

3.1 Requirements Gathering

Prior to research and development, the needs of the client had to be clearly defined. The client’s main requirement was the development of an interactive web interface to R using R Shiny that could be used for teaching some basic statistical concepts. Further discussions with my client contact Dr Jason Wyse helped to establish a clearer idea of what exactly the client was looking for. This led to the formation of the terms of reference outlined in Section 1.3. The original project outline which can be seen in Appendix B put a lot of emphasis on the web application being able to upload a user’s own dataset for analysis. After meeting with the client and discussing this aspect of the project, Dr Wyse clarified that he was more concerned with the capabilities of Shiny as a teaching
resource to be investigated rather than that the web interface should be used as an analysis tool for uploaded datasets. He wanted the four topics mentioned in the project outline to be covered; he left how these would be covered up to me. Part of the challenge of the project was to see what could and couldn’t be achieved with Shiny. The issue of how the finished web application would be implemented was also discussed in these early client meetings. As Shiny is a web framework it is of course designed to be hosted online. It was suggested that the finished web interface could be hosted on the Statistics Discipline’s local MacNeill server. The client in the end decided against this option as this was only the first part of the web interface. The client was happy to run the application locally or from the GitHub repository for the time being. If the application was a success then possibly later versions would be hosted online.

3.2 Research

With the requirements clearly defined, the research phase of the project could get underway. There were two main branches of research that had to be completed for this project. These were software development research and statistical content research for each of the four topics to be included.

Software Research

Prior to starting this project I was a relatively experienced R user but I had never used R Shiny and had no idea how it worked or any of its functionality. I spent a lot of time learning how to use R Shiny, investigating different features to see if they would be useful for my own system, and seeing what was already out there which might provide inspiration for my own application’s design and content.

The entirety of my research in to how to use Shiny was done online. The Shiny website was my most valuable research resource. The website has a number of tutorials on Shiny to get you started and help you make your first applications. These tutorials were very good for a basic overview and introduced a lot of Shiny’s features. Also on the Shiny website is a gallery section. This section was very useful as it has examples of Shiny applications for the majority of Shiny’s features as well as the code used. This was very helpful during development of my application. If, at any point, I was struggling to get a certain feature or widget to work properly I could check the gallery to see an example.

The Shiny website also has a section which has articles on lots of topics such as user interface layout, widgets and reactive programming. This was also a valuable resource during development. Other useful resources during the software research phase of my project was Stack Overflow, which is a website where programmers who are having trouble with their code post their problems and members of the Stack Overflow community answer them. There is a dedicated R Shiny section on Stack Overflow which was particularly useful as sometimes people would have experienced similar problems to me.
Content Research

A lot of research had to be done for the application’s statistical contents. The main purpose of the web interface was to cover the four statistical concepts laid out on in the terms of reference and agreed with the client. The four statistical topics I had to cover were T-tests, Regression, ANOVA and Chi-square tests. Each topic would need to have a brief introduction, a dataset to demonstrate the analysis on, as well as countless plots and R outputs explained along the way. This took some time as it was sometimes difficult to tell if something was effective until it was tested. My supervisor, Arthur White, was very helpful during this process as he could advise me from his own teaching experience in the statistics discipline.

R has a lot of built-in datasets already included with the software and I used two of these datasets for the T-test and Chi-square sections of my web interface. The two built-in datasets I used were a dataset called ‘Sleep’ and one called ‘UCBadmissions’. The Sleep dataset is a dataset which showed the effects of two soporific drugs (increase in hours of sleep compared to control) on ten patients. I used this dataset to demonstrate a paired t-test. This dataset worked well as it was small and it was easy to display graphically so the user could easily see where all the calculations were coming from. The ‘UCBadmissions’ dataset is a dataset which displays information on the applicants to graduate school at Berkeley for the six largest departments in 1973 classified by admission and sex. This dataset was a fun and interesting dataset which I used to explain a Chi-Square test; the data could also be displayed very attractively using mosaic plots.

One of my main resources for content for the web interface was notes supplied to me by my project supervisor. These were notes from a postgraduate course in introductory statistics that the Statistics Discipline was teaching. The notes dealt with all of the topics I was covering and in a lot of detail as well as offering many different datasets and examples. I based the ANOVA and Regression sections of my application on these notes. For the ANOVA section of my application I used an example dataset drawn from these notes. This dataset was a set of five Boron measurements from four different labs and the aim of the study was to test if there was any systematic variance between the Boron measurement techniques in each lab. This dataset worked well for the ANOVA section of my application as the data could be plotted attractively. I then explained the method for decomposing the variance by hand for the data, as well as how you would do it in R using the ANOVA table function and explained its output. For the regression section of the application I used another dataset from Eamon’s notes, this was a dataset of spot welds and it concerned establishing a quantitative relationship between the shear strength of spot welds and their diameter. From this relationship a basic linear regression model could be built.
3.3 System Design

After the requirements were clearly established and research had been done on Shiny and the statistical content for the application it was possible to begin the system design phase of the project. During this phase the appearance and structure of the web interface were decided.

The system was designed as previously stated using an incremental development framework. This meant that there were many versions of the application which changed with each incremental development step.

Structural Design

An initial structural design decision was whether to make one application that covered all the topics needed or to make a separate application for each individual topic. There were arguments for both. However, the single application had the advantage of having everything together in one application, meaning there was less chance of files being lost and also that the user wouldn’t have to run multiple applications which would take from the user experience.

One concern from having only one application for all the topics was that once the application got past a few hundred lines of code it would become difficult to maintain the code in R studio. In particular, when someone was trying to add to the code in a later version, they could find it difficult to do so as they first had to traverse hundreds of lines of somebody else’s code. Another concern was that once the application got quite large it wouldn’t run as well or efficiently as smaller individual applications. After some research this concern was disregarded as the application would have to be substantially larger than my system to notice an effect to run time. It was decided after discussion with the client to opt for the single application system design. The project wasn’t so large that it would be unmanageable if some strict coding standards were stuck to. Also a single application would be a lot more user-friendly compared to multiple single applications.

Application Interface design

A lot of work went into the Application Interface. One of the main requirements of the project as laid out in the terms of reference was that the Interface would have an emphasis on being user friendly, attractive and approachable in terms of design.

The first step in designing the Application Interface was deciding which application layout to use. R shiny has many easy to implement application layouts. One great feature of the Shiny user interfaces is they have a feature called “fluid page” which means that the user interface automatically scales for the size of the browser it is being displayed on to fill all available width.

After experimenting with a few styles, the navigation bar page application layout was chosen. This layout places a navigation bar to the top of the web interface that is
accessible at all times and allows the user to navigate the application with ease. No matter where the user is in the app, they can always change to a different section easily by just clicking on the appropriate heading on the navigation bar. The content headings were Home, T-Test, Chi-Square, ANOVA, Regression and File Upload. This application layout was also the most suitable layout for an application with numerous sections. A screenshot of the navigation bar can be seen below.

![Screenshot of Navigation Bar](image)

**FIGURE 3.3.1 – Screenshot of Navigation Bar**

### Subsection's Interface

Each subsection’s interface varied from topic to topic. The Home and File Upload sections of the web interface were only one page so they had very basic layouts. Each of the four statistical topics subsections made use of the tab panel layout. This layout split content up by putting different tabs across the top of the interface so the user could navigate between each page of the topic. A screenshot of the tab panel layout for the t-test section can be seen below. You can see that for the t-test section there are two layers of tab panels. The first layer lets the user toggle between the introduction and paired t-test sections and the second layer allows the user to navigate through the paired t-test section itself. This layout worked very well and proved very effective at organising the different sections. More information on the user interface for each section can be found in the user interface walkthrough section in chapter 2.3.

![Screenshot of t-test section](image)

**FIGURE 3.3.2 – Screen shot of t-test section**

### 3.4 System Development

After the system had been designed, the development of each section took place. As previously stated, the web interface was developed using an incremental development framework. What this meant in practise was that as each section was developed, I would present it to my supervisor and client, receive feedback and update the system. I would then begin work on the next section and repeat this process.

Different increments took different periods of time as different problems were encountered at each stage of development. At the beginning of the development process I was also a less experienced Shiny user so it took me some time to get to grips with the basics compared to later on in the development process where I could develop Shiny applications a lot quicker.
I shall now run through the development process for each section of the web interface.

Home

This tab's main purpose was to act as an introductory page to the web interface. Where any information about the application could be displayed, I put the application's title and a picture of the college on this page. I thought this added to the aesthetic value of the user interface instead of jumping straight into a topic as soon as the application was loaded.

T-Test

This was the first topic that I developed. It was also probably the topic I spent the most time on. The first page in this section is an introductory section and introduces hypothesis testing and states the four steps of any hypothesis test. Below this there is an introduction to the T-distribution and how theory tells us that this is normally distributed. Below this is an interactive plot that plots the T-distribution and the normal distribution on top of each other. There is a slider to the right of this plot which lets the user control the degree of freedom of the T-distribution and re-plots it reactively. This effectively shows that as the degrees of freedom are raised the T-distribution converges on the normal distribution.

The next page of the T-test section focuses on the paired T-test. At the top of the page there is a brief introduction to the paired T-test and when it is used. Below this there is a side and main panel. The side panel is the dataset which is used for this example printed in table form. The dataset used was a built in dataset to R called Sleep which shows the effect of two soporific drugs (increase in hours of sleep compared to control) on ten patients.

The main panel is split into four sub-panels. Each deals with a different aspect of the paired T-test. The first panel just shows the data graphically using boxplots, firstly by two separate boxplots for each drug and then a single boxplot of the difference between each group.

The second panel introduces the T-statistic and shows how this is calculated using the Sleep dataset as an example. All of the mathematical notation in this tab was written using the MathJax wrapper function in Shiny. MathJax is a html function that allows Shiny developers to display mathematical notation attractively in a web browser.

The third panel shows how the P value is calculated using R and calculations previously obtained. It shows the code needed and explains what the P value is. Also covered in this tab is how to calculate the critical value by hand using a statistical table's book and by using R functions. The final section of this tab shows the conclusions we can draw from this example using the values we have calculated. How we can reject the null hypothesis and there is also a graphical representation of the data showing the Critical value and the T statistic on the same plot.
The fourth panel in the paired T-test section of the application shows how you could obtain all the values of the third panel using the T-test function in R. This panel's main purpose is to show the power of R functions while also reinforcing some of the concepts covered in earlier sections.

Chi-Square

The Chi-square section was the second increment of the software development phase. The Chi-square user interface also uses a tab panel layout with three different sections. These sections were ‘Introduction’, ‘Student Admissions at UC Berkeley’ and ‘A Closer Look’. I will briefly run through each section’s contents.

The Introduction panel introduces Chi-Square tests and explains what they are commonly used for. The Chi-Square test formula is printed in the introduction panel as well. This was displayed again using the MathJax wrapper function.

The second panel in the Chi-square section of the application introduces the dataset that is used. The UCB admissions dataset is another dataset that is already built in to R. This proved useful as it already had help files available for it. The dataset shows data on the applicants to graduate school at Berkeley for the six largest departments in 1973 classified by admission, sex and department. The dataset is quite famous as it is a good example of Simpson’s Paradox. If you examine the general admission statistics there seems to be a clear gender bias in favour of male students but when you breakdown the data and look at admissions by department it becomes clear that there isn’t a bias at all and that more female students apply for the tougher departments. This panel first shows the initial apparent bias by comparing the general acceptance rate based on gender. Next it runs a number of Chi-Square tests and shows there is significance for some and not for others.

The third panel in the Chi-Square section takes a closer look at the data and explains why this is happening. It shows that female students are actually applying to the departments that have higher rejection rates and that there isn’t a gender bias in any of the departments except one which is slightly in favour of female students. Finally the last thing on this panel is an interactive mosaic plot which shows acceptance based on gender for each department. The user can toggle between each department by using the drop down menu to the left of the plot.

ANOVA

The ANOVA section of the application was added in the third increment of the software development process. The ANOVA section of the application again uses a tab panel style of user interface to organise its contents into four panels. They are ‘Introduction’, ‘Data’, ‘Decomposing the Variance’ and ‘ANOVA table.

The ‘Introduction’ tab introduces the basic concepts of ANOVA, what it’s commonly used for and then explains how ANOVA are calculated.
The 'Data' tab introduces the dataset used for the ANOVA section of the application. This dataset is taken from the notes used to teach module ST125 Introduction to statistics 2. The dataset has Boron measurements from four different labs and the study aims to test if there were any systematic variance between the Boron measurement techniques in each lab or were the variances just due to chance. The dataset is then graphically represented by two plots, the first as a dot plot and the second using boxplots.

The ‘Decomposing the Variance’ tab shows the mathematics behind the calculations involved in an ANOVA. The formulas for calculating the sum of squares total and the degrees of freedom total are printed clearly in this section.

The ‘ANOVA table’ tab shows how ANOVA can be calculated using R functions. The output from the ANOVA table function is displayed for the Boron dataset along with an explanation of the output.

Regression

The regression section of the application was part of the fourth incremental step used in the development process. The regression section used a tab panel layout to split up its content. The first panel was an introductory section that explains the basic concepts behind regression. The general formula for a simple linear regression was also printed here. The second tab introduced the dataset used for the regression section. This was a dataset of spot welds taken again from the notes used to teach the module ST125 Introduction to Statistics 2. The final tab displays the data graphically with an interactive plot. That lets the user add a regression line; the prediction interval and the confidence interval by checking different check boxes.

File Upload

The file upload section was added in the final development increment and is only one page. This section demonstrates Shiny’s file upload widget; it allows the user to upload a CSV file to the app which is then printed for the user. The Application prints a warning message if the wrong file type is uploaded.

3.5 Testing and Implementation

The system was developed using an incremental development structure, so the system testing was done with every increment of the development process. The application during development was only run from local files so there was never any issue getting it to run on my own machine. Once the system had been completed testing began on various machines though out the college and on numerous networks.
As discussed earlier the client specified that they didn’t need the system to be implemented online as it was only to be the first part of the web platform and being able to run it off R Studio was acceptable for their current needs. So the project was implemented in two ways. The files were submitted on a cd which is attached with this project. All the user needs to run the Shiny application is to install the Shiny package on to their R studio and then they need to copy the files from the attached cd to their working directory. The application can then be loaded up from there.

The second method for implementing the shiny app is it can be loaded from the GitHub repository which I made especially for this project. Full instructions of how this can be done are located in the user manuals that can be found in the appendices located at the end of this report.

3.6 Challenges Encountered

There were many challenges encountered during the development of the web interface some more severe than others.

One challenge in developing the web interface was the sometime lack of resources on Shiny when an issue was encountered. Shiny is a relatively new web framework for R and though its website does have a lot of content for learning how to use it, it is not an exhaustive source of information. Other resources such as the online coding forum Stack Overflow were useful. Apart from these though there wasn’t the same degree of learning resources as are available for more common programming languages. This could be an issue for bugs in the code, where a lot of trial and error debugging would have to take place. This could be very time consuming and tedious and was one of the main challenges of implementing a new web framework.

Another big challenge was storing my code in a GitHub repository online. This process was a lot more difficult than at first anticipated. I had some coding experience but had never used GitHub before. I eventually worked out how to get my files onto the repository by using the GitHub desktop application but I still would be a less than confident user of the service.

The File upload section of the Shiny application took some time to get working. There wasn’t much content online on data control for the upload widget. The example application on the Shiny website let you upload any file you wanted. So I had to work this out intuitively. This took several attempts to get working correctly.

Another challenge that was encountered was revising the basic statistical concepts covered in the application. I hadn’t studied some of the statistical topics that were included in the web interface in a few years. The result was that I had to do a good bit of revision to make sure that the information conveyed was correct. It was also quite difficult to explain some of the concepts in this domain as I didn’t want to make the application uninviting by putting lines and lines of text into it. All this made me appreciate the difficulty faced by lecturers of preparing interesting and informative lecture material.
There was an additional challenge encountered when trying to run the web interface from GitHub on some of the college machines and networks. During the testing phase of the system implementation, I tried running the web interface using the GitHub function in R Studio from a number of different machines in the college. Some of the computers in the labs, such as the MSISS labs couldn’t even load the Shiny package because the version of R studio was too old. The web interface could be run off students own machines if they were connected to the TCD Wi-Fi connection but there seemed to be some issues running it off some of the other college Wi-Fi connections. For example the one used by my supervisor Arthur White led to some errors with the proxy when he attempted to run it from his own machine.

4. Conclusions and Recommendations

4.1 Conclusions

The finished system fully meets the requirements of the client as outlined in the terms of reference (see section 1.3). The first part of an interactive web platform has been developed using the R Shiny web application framework. The web platform makes use of numerous case studies to illustrate and explain four ubiquitous statistical subjects. Below is the outline of how each requirement (as specified in the Terms of Reference) was met:

- An interactive web interface has been developed using R Shiny that covers the four statistical topics agreed with the client and laid out in the terms of reference.
- The web interface makes use of numerous datasets some built in to R and some taken from notes for courses from the statistics discipline.
- The web interface is user friendly with an attractive and approachable user interface which implements an attractive CSS theme.
- The web interface has a section where a user can upload their own CSV dataset and view it. The user will receive a warning message if the dataset is in the wrong file format.

R Shiny is an easy to use and powerful framework with an attractive user interface and many useful features that are easily customisable for the developer’s needs. It has the potential to be a valuable teaching and interactive analysis tool for the statistics discipline. The interactive features proved to be especially effective at demonstrating sometimes complex concepts. I thought that Shiny’s reactive outputs and plots that change as the user alters a parameter were an especially effective teaching feature.
Shiny’s strengths lie in its attractive and friendly user interface and ease of implementation by any experienced R user. Ultimately I am confident that the developed web platform can assist the client and will be the first of many implementations of R Shiny by the Statistics Discipline.

In summary, the project has proved to be a challenging but rewarding task. The experienced gained learning how to use R Shiny has been invaluable. I also learnt some useful skills implementing other languages and applications as part of my development such as using Latex for writing maths notation and learning how to use GitHub.
4.2 Recommendations

I describe below a few recommendations for the continuation of this project and the implementation of Shiny applications in the future by the Statistics Discipline.

Further Shiny development

A number of recommendations are proposed for further development of the Shiny web interface. Included in the appendices of this report is a user manual with detailed instructions on how this web interface can be edited and added too. I would suggest that the client follow these instructions if they wish to add to this application. The user manual gives instructions on the basic workings of Shiny and instructions on how to add this.

If the client wanted to learn how to implement some of Shiny’s more complex features such as reactive features or more advanced layouts, I would highly recommend completing the online tutorials on Shiny’s website. It is a very useful development resource with an abundance of information on all Shiny features and is updated regularly.

I also recommend that further developments be stored on GitHub as this is a great resource for building on older versions of software. As well as this the Shiny package includes a handy function to run Shiny apps straight from GitHub repositories.

Implementing the Shiny Interface

I have some further recommendations regarding the implementation of the web interface. I suggest that if this project is taken further it should eventually be hosted online on a website so that is easily accessible for both students and staff. Currently the web interface can be run in R studio from local files and from the GitHub repository set up for the client.

I recommend updating R Studio to the latest version in all the computer labs. Some of the college computers are currently running older versions of R studio that aren’t able to install the Shiny package. These machines will need to be updated if the web interface is to be used in labs or lectures.

I also think that some research should be done on how to correctly navigate the college proxy as it seems to block some of Shiny’s functionality thereby causing errors. For example if you use the “run from GitHub” function for Shiny on some of the college networks an error message is displayed and it is unable to run. This is not a problem on the main TCD Wi-Fi for students but seems to be an issue on some of the staff private networks. An investigation into this would be worthwhile as it is probably easily fixed and being able to run the interface directly from GitHub is a great advantage.
THE STATISTICS DISCIPLINE

A Friendly Web Interface to R Using R Shiny for Some Frequently Used Statistical Analysis Methods

Appendices

Prepared By: Ferdia de Paor            Supervisor: Arthur White
## APPENDICES

<table>
<thead>
<tr>
<th>NO</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Original Project Outline</td>
<td>A.1</td>
</tr>
<tr>
<td>B</td>
<td>Interim Report</td>
<td>B.1</td>
</tr>
<tr>
<td>C</td>
<td>User Manual</td>
<td>C.1</td>
</tr>
<tr>
<td>D</td>
<td>Design Documentation</td>
<td>D.1</td>
</tr>
<tr>
<td>E</td>
<td>Source Code</td>
<td>E.1</td>
</tr>
</tbody>
</table>

REFERENCE
A. Original Project Outline

PROJECT 27

Client: Statistics Discipline
Project: A friendly web interface to R using R shiny for some frequently used statistical analysis methods
Location: Lloyd Building
Client Contact: Jason Wyse
Dept. Contact: Jason Wyse

Client Background

The Statistics Discipline is providing courses in Statistics in Trinity College at both undergraduate and postgraduate levels. The teaching is mainly conveyed in class via lectures, tutorial and labs.

Project Background

To support this teaching, for each course, many case studies need to be collected, understood and processed with a dedicated software environment to illustrate the statistical methodology applied to different domains. Such a database of case studies needs to be continuously updated to follow the latest developments in science and keep the students interested with the latest innovations.

Client Requirement

This project aims at developing a first part of an interactive web platform collecting, analysing, processing and visualising many case studies to be used by staff and students to illustrate and understand the various course contents delivered by the staff in the Statistics discipline.

The free environment R (r-project.org) is widely used for statistical computing in both research and now also in industry. Most of the staff in the discipline already use it (with other alternatives) and would be familiar with this language. A web base R-Studio platform is already in use for research purposes in the statistics discipline, accessible from both inside and outside campus to run processes on a cluster via a simple web browser. Shiny (see http://shiny.rstudio.com/gallery/) is a web application framework for R to turn analysis into interactive web applications, giving users the power to process their own datasets easily or run examples provided with user friendly GUIs through an internet browser.

What is involved for the student?

Develop an interactive web interface with R shiny which covers four relevant ubiquitous statistical topics, including
1. t-tests
2. regression
3. ANOVA
4. chisquare tests

The interface will be user friendly, and attractive and approachable in terms of design, with capacity to upload ones own dataset and capacity to give caveats or warning messages if the data appears incompatible with the analysis type chosen.
B. **Interim Report**

**Project – A friendly web interface to R using R shiny for some frequently used statistical analysis methods**

Client – Statistics Discipline  
Supervisor – Arthur White  
Client Contact – Jason Wyse

**Terms of Reference**

- Develop an interactive web platform that will assist the school of Computer Science and Statistics in teaching basic statistical methods through visualisation of classic statistical datasets.  
- This will be implemented using the R shiny web application framework  
- Interactive web applications will be made to demonstrate t-tests, regression, ANOVA and Chi-square tests.

**Work to date**

I have met with my client Jason Wyse to discuss what he hopes to gain from the completion of this project. I have begun teaching myself how to use the shiny framework by completing online tutorials and completing some practise projects. I have made a prototype shiny web application to demonstrate a t-test and demonstrated it to my supervisor and client to get feedback.

**Interim Conclusions**

I have gained some skills in R shiny and have made a prototype web app for the paired t-test example using the sleep dataset. I have a good idea of how I will approach the web apps for the other statistical methods and have started to work on them. I have been trying to make the web applications clear and accessible so that they are easy to understand even to a novice statistician.

**Further work to be planned**

Further work to be done will include making interactive web applications to demonstrate Regression, ANOVA and Chi-square tests. I will maintain frequent correspondence with my client to ensure that the web applications meet their needs and produce many prototypes along the way to demonstrate my progress and receive feedback. I will also add the capacity for a user to upload their own dataset to the web applications and to give caveats or warning messages if the data is incompatible. Accompanying user manuals will also have to be produced to explain how the software works and how it could be adjusted for individual needs if necessary.
C. **User Manual**

Please see the user manual attached as a separate document at the end of the report. The manual includes instructions for installing the Shiny package in R studio, how to run a Shiny application and a brief explanation of how the system could be adapted or added to.
D. **Design Documentation**
This appendix describes the processes employed in the design and development of the finished system.

**Software Methodology – incremental Development Model**
Software methodologies provide structure during the development process. The model used in the construction of this web interface was the incremental method. This model involves the following phases:

- Requirements Analysis
- Design
- Development
- Implementation
- Testing

As the system progressed, modifications were made, repeating the design and development stages in the next iteration.
E. Source Code

This appendix contains a sample of the source code for the developed system. The source code is comprised of two R scripts. The user interface script (ui.R) and the server script (server.r) where all the R analysis takes places. This source code is also available on the CD which is submitted as part of this project as well as hosted online on a GitHub repository. Details of how to access this is explained in the attached user manual.

**Ui.R**

```r
#SHINY UI SCRIPT
#Where the layout of the shiny app is decided
shinyUI(
  navbarPage("R SHINY",
    inverse= TRUE,
    #Home tab panel
    tabPanel("Home",
      img(src="trinity.jpg"),
      h3("Frequently used statistical
        analysis methods using R-Shiny"),
      h5("A Fourth year Project for the Department Of Statistics")
    ),
  #T-test tab panel
  tabPanel("T-Test",
    tabsetPanel(
      tabPanel("Introduction",
        h4("Introduction"),
        hr(),
        p("The t-test looks at the t-statistic, t-
          distribution and degrees of
          freedom to determine a p value (probability) tha
          whether the population means differ. The t-
          test is one of a number of hypothesis tests."),
        hr(),
        p("For any Hypothesis test we follow these steps:"),
        p("Step 1: State the null and alternative hypothes
is."),
        p("Step 2: Determine the level of significance."),
        p("Step 3: Compute the test statistic."),
        p("Step 4: Make a decision and draw conclusions"),
        hr(),
        p("In the case of t-tests we are interested in
```
tells us that this is normally distributed. We also factor in uncertainty about the sample deviation. For this reason we use a t-distribution),

```
sidebarPanel(sliderInput("dfslide", "Degrees of Freedom", min=0, max=30, value=3)
```

We also factor in uncertainty about the sample deviation. For this reason we use a t-distribution, which is normally distributed. While theory tells us that this is normally distributed, we also factor in uncertainty about the sample deviation. For this reason we use a t-distribution.

```
hr()
```

```
mainPanel(plotOutput("df"))
```

```
hr()
```

```
tabPanel("Paired T-Test",
    h4("Paired T-Test"),
    p("The Paired T-Test is used when you have two groups which are correlated in some way in this example the dataset shows the effect of two soporific drugs (increase in hours of sleep compared to control) on 10 patients"),
    hr(),
    sidebarPanel(
      "The Sleep dataset shows the effect of two soporific drugs (increase in hours of sleep compared to control)
      
      on 10 patients.",
      tableOutput("sleep")
    ),
    wellPanel(
      "The Dataset we will use in this example is the sleep dataset which comes built in to R and is visible on the left.",
      p("The Sleep dataset shows the effect of two soporific drugs (increase in hours of sleep compared to control)"
      
      on 10 patients.")
    )
    
    mainPanel(
      tabsetPanel(
        tabPanel("The Dataset",
          wellPanel(
            "The Dataset we will use in this example is the sleep dataset which comes built in to R and is visible on the left.",
            p("The Sleep dataset shows the effect of two soporific drugs (increase in hours of sleep compared to control)
            
            on 10 patients.")
          )
        )
      )
    )
```
code("?sleep"), "into your R console or click "
', a("here", href = "https://stat.ethz.ch/
R-manual/R-patched/library/datasets/html/sleep.html")
),
selectInput("tbox", label = "Plots of the data", choices = c("Boxplots of both groups","Boxplot of difference of two groups"), selected = "Boxplots of both groups"),
plotOutput("Tboxplot")
),
tabPanel("The T statistic",

p("The T statistic is calculated using the following formula"), withMathJax(),
h2('$$t = \frac{\bar{d}}{\frac{s}{\sqrt{n}}}$$'),
wellPanel(),
h5("$$\begin{align} t &= \text{the t-statistic} \\
\bar{d} &= \text{The mean difference of the 2 groups} \\
s &= \text{The Standard deviation of the difference of the 2 groups} \\
n &= \text{The population size} \\
\end{align}$$")
),
p("so we get"),
h3('$$ t = \frac{-1.58}{\sqrt{10}}$$'),
h3('$$ t = -4.06$$')
),
tabPanel("The P value and Critical value",

h3("P Value"), p("Formally the P value is the probability of observing an event more extreme than the data. We can calculate this using the T-statistic and the degrees of freedom and the pt() function in R")
),
p("The T-statistic we calculated was -4.06 and there were 9 degrees of freedom"), p("so we can calculate the P value by entering the following code in to R"),
code("pt(4.06, 9, lower.tail = FALSE")
), p("but our test is two tailed so we should multiply it by 2 so we enter"),

p("but our test is two tailed so we should multiply it by 2 so we enter")
)
This gives us a p value of
verbatimTextOutput("pvalue").

Using R by entering the following line of code ",
code("qt(0.025, 9, lower.tail= FALSE)"),
hr(),
verbatimTextOutput("crit"),
hr("Conclusions"),
code("t.test(extra ~ group, data=sleep, paired=TRUE)"),
hr(),
verbatimTextOutput("paired")

We can find the critical value by using our tables books or
value by using our tables books or

We can also calculate this using R by entering the following line of code ",
code("qt(0.025, 9, lower.tail= FALSE)"),
hr(),
verbatimTextOutput("crit"),
hr("Conclusions"),
code("t.test(extra ~ group, data=sleep, paired=TRUE)"),
hr(),
verbatimTextOutput("paired")

This gives us a p value of
verbatimTextOutput("pvalue").

Using R by entering the following line of code ",
code("qt(0.025, 9, lower.tail= FALSE)"),
hr(),
verbatimTextOutput("crit"),
hr("Conclusions"),
code("t.test(extra ~ group, data=sleep, paired=TRUE)"),
hr(),
verbatimTextOutput("paired")

We can find the critical value by using our tables books or

We can also calculate this using R by entering the following line of code ",
code("qt(0.025, 9, lower.tail= FALSE)"),
hr(),
verbatimTextOutput("crit"),
hr("Conclusions"),
code("t.test(extra ~ group, data=sleep, paired=TRUE)"),
hr(),
verbatimTextOutput("paired")

In this case we have degrees of freedom = 9 and an alpha value of .05 for a 95% confidence interval.

Our T-statistic is 4.06 which is well outside the critical value of 2.26 so we can reject the null hypothesis.

This is represented graphically below as you can see the critical value represented in red and the t statistic in green.

Here is a copy of the R code used to obtain the output below.

t.test function which we can use to get the same results

R has a handy T-test function which we can use to get the same results

The function in R is called t.test()

"to get more information on t.test() enter ?t.test() in to your R console or click"

Here is a copy of the R code used to obtain the output below.

t.test function which we can use to get the same results

R has a handy T-test function which we can use to get the same results

The function in R is called t.test()

"to get more information on t.test() enter ?t.test() in to your R console or click"

Here is a copy of the R code used to obtain the output below.

t.test function which we can use to get the same results

R has a handy T-test function which we can use to get the same results

The function in R is called t.test()

"to get more information on t.test() enter ?t.test() in to your R console or click"

Here is a copy of the R code used to obtain the output below.

Here is a copy of the R code used to obtain the output below.

Here is a copy of the R code used to obtain the output below.
A Chi square test is a statistical test commonly used to compare observed data with data we would expect to obtain according to a specific hypothesis.

$\chi^2 = \sum_{i=1}^I \sum_{j=1}^J \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$

Where

$$\begin{align*}
\chi^2 &= \text{Test Statistic} \\
I &= \text{# number of rows in contingency table} \\
J &= \text{# number of columns} \\
end{align*}$$

Chi square tests are commonly used to test if deviations in expected values are due to chance or by other contributing factors.

This dataset of student admissions at Berkeley University in 1973 is a famous example of simpsons paradox and can also be used to demonstrate a Chi Square test.
At first glance it appears that the university has a clear sex bias in favour of males as there seems to be a much higher acceptance rate of males compared to females. Males have an acceptance rate of 45% compared to females who have an acceptance rate of 30%.

The Chi square test for Gender vs Admission is also highly significant suggesting a bias based on Gender.

However if we look at Gender vs Department for each individual department you can see that there is only significance for department a.

However, if we test Admissions against Department, results are again highly significant.
Similarly for Department against Gender

But when you take a closer look at the data you can see that actually more females apply for the departments with higher rejection rates and in general the acceptance rates for males and females are very similar and sometimes even favours females.

A nice way to graphical display the data is by using mosaic plots. Mosaic plots are suitable in this instance as they clearly display the proportions of admissions so are easily comparable.
You can see male students in blue and female in red.

#Anova tab panel

**Data:**

For this example we will look at a dataset of Boron measurements for the same adhesive taken from four different labs. The aim of the study was to test if there was any systematic variance between the Boron measurement techniques in each lab.

The statistical test for between-group differences is based on the ratio of the two mean squares: $MS(\text{Between-group})/MS(\text{Within-group})$. If this is large, it suggests systematic between-group differences.

The basic idea underlying the simplest form of ANOVA is that the variation between individual observations can be viewed as having two components: within-group and between-group.

Within-group variation, on the other hand, is seen as (at least potentially) systematic: different groups are different, or are treated differently, in ways that may lead to higher or lower average responses.

Between-group variation, on the other hand, is seen as (at least potentially) systematic: different groups are different, or are treated differently, in ways that may lead to higher or lower average responses.

The statistical test for between-group differences is based on the ratio of the two mean squares: $MS(\text{Between-group})/MS(\text{Within-group})$. If this is large, it suggests systematic between-group differences. The basic idea underlying the simplest form of ANOVA is that the variation between individual observations can be viewed as having two components: within-group and between-group.

The system being studied.

No reasons why any two observations should be different, apart from the influence of the chance variation that affects the system being studied.

The within-group component is seen as pure chance variation - within-group.

Conditions are held constant and there are no reasons why any two observations should be different, apart from the influence of the chance variation that affects the system being studied.

The basic idea underlying the simplest form of ANOVA is that the variation between individual observations can be viewed as having two components: within-group and between-group.
different days, so the variation between them reflects medium term chance of measurement error within the laboratories. The Boron measurements are taken at the parts per million (ppm) level.

The Boron measurements are taken at the parts per million (ppm) level.

"without any formal statistical analysis it is obvious from looking at the data graphically that the average results in laboratories 3 and 4 are different; it is not clear, however, whether laboratories 1 and 2 differ by more than the chance analytical day-to-day variation which is clearly present in all laboratories.")

The data displayed graphically with a boxplot"

"We will now run through the technical side of decomposing the variance "

$$\text{SS Total} = \text{SS Within} + \text{SS Between}$$

$$IJ - 1 = I(J - 1) + I - 1$$
tabPanel("ANOVA Table",
        h4("The ANOVA TABLE:")
        ,
        p("A Look at the ANOVA table function in R and its output")
        ,
        hr()
        ,
        verbatimTextOutput("Anova_table")
    )

#Regression tab panel
    tabPanel("Regression",
        tabsetPanel(
            tabPanel("Introduction",
                h4("Introduction:"),
                p("Regression analysis is, perhaps, the most important data modelling technique in statistics.")
            ,
                hr()
            ,
                p("Regression is concerned with fitting equations to data. In the simplest case this means that we have a variable X which we believe is related to a second variable Y and which we wish to use to predict Y; the relationship between the two variables is believed to be linear. We do not assume that X is a perfect predictor of Y: in the systems we study there is always chance variation, which prevents us from making perfect predictions.")
            ,
                wellPanel(
                    h4("The Basic Regression Formula")
                    ,
                    h3("$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$")
                    )
            )
        ,
        tabPanel("Data",
            h4("Data")
        ,
            hr()
        ,
            p("The dataset used in this section is a dataset of spot welds with their Strength and Diameter")
        ,
            hr()
        ,
            tableOutput("welds")
        )
    ,
    tabPanel("Regression",
        plotOutput("Regression")
        ,
        hr()
        ,
        checkboxInput("reg", label = "Regression Line", value = FALSE)
        ,
        p("The Regression line can be thought of as an "average line" through the data at any given diameter, the strength of replicate welds will vary about this line; this is the line that (approximately) joins up the means of the strength values for different weld diameters.")
        ,
        hr()
        ,
        checkboxInput("pre", label = "Prediction Interval", value = FALSE)
        ,
        p("The prediction interval represents where we predict with 95% certainty the strength for a single spot weld diameter would fall")
    )
hr(),

checkboxInput("con", label = "Confidence Interval", value = FALSE),

p("The confidence interval represents where we are 95% confident that the interval covers the true long-run mean at this strength")

# File Upload tab panel

tabPanel("File Upload",
  h4("File Upload"),
  sidebarLayout(
    sidebarPanel(
      fileInput('file1', 'Choose CSV file to upload', accept = c('text/csv',
        'text/comma-separated-values',
        'text/tab-separated-values',
        'text/plain',
        '.csv',
        '.tsv'))
    ),
    mainPanel(
      tableOutput('upload'),
      h1(textOutput("warning1"))
    )
  )
)

# load css theme

includeCSS("background.css")
Server.R

1. #SHINY SERVER SCRIPT
2. #Where all the R code is placed
3. #Load all needed packages here
4. require(ggplot2)
5. library(tools)
6. #Load in any datasets here or declare global variables that are used by a few functions
7. #Lab data for ANOVA examples
8. Lab1= c(4.9,5.7,5.1,5.3,5.4,5.5)
9. Lab2= c(5.4,5.5,4.8,4.9,5.2,5.4)
10. Lab3= c(5.8,6.0,6.0,5.5,5.9,5.8)
11. Lab4= c(4.5,4.9,4.7,4.7,4.4,4.8)
12. #Setting up boron data for use with GGPlot
13. Boron_data <- data.frame(Lab=as.factor(rep(1:4, each =6 )), Boron=c(Lab1,Lab2,Lab3,Lab4))
14. #read in data from notes chapter 6 pg 2
15. Strength = c(3327,3240,3435,4362,4490,5556,5870,5796,6630,6410)
16. Diameter = c(5.0,5.0,5.5,5.5,5.5,6.0,6.0,6.5,6.5,6.5,7.0,7.0)
17. shinyServer(function(input, output) {
18.  ##########################
19.  ## CHI SQUARE FUNCTIONS ##
20.  ##########################
21.  #table of gender and admission
22.  output$data <- renderTable({
23.    apply(UCBAdmissions,c(1,2),sum)
24.  })
25.  #gender and department
26.  output$data1 <- renderTable({
27.    apply(UCBAdmissions,c(2,3),sum)
28.  })
29.  output$data2 <- renderTable({
30.    apply(UCBAdmissions,c(1,3),sum)
31.  })
32.  output$proptable <- renderTable({
33.    proptable<-prop.table(apply(UCBAdmissions,c(1,2),sum),2)
34.    round(proptable,2)
35.  })
36.  output$mosaicadmit <- renderPlot({
37.  })
mosaicplot(apply(UCBAdmissions,c(1,2),sum),
    xlab = "Admit", ylab = "Sex",
    main = "Admittance",color=TRUE)
}

# print all 6 departments
output$mosaic <- renderPlot{
  opar <- par(mfrow = c(2, 3), oma = c(0, 0, 2, 0))
  for(i in 1:6)
    mosaicplot(UCBAdmissions[,,i],
      xlab = "Admit", ylab = "Sex",
      main = paste("Department", LETTERS[i]),col = hcl(c(10, 120)))
    mtext(expression(bold("Student admissions at UC Berkeley Mosaic Plots")),
      outer = TRUE, cex = 1.5)
  par(opar)
}

#switch statement to switch departments plot
output$mosaicA <- renderPlot{
  if (is.null(input$mo))
    return()
  switch(input$mo,
    "A"=mosaicplot(UCBAdmissions[,,1],
      xlab = "Admit", ylab = "Sex",
      main = paste("Department", input$mo),col = hcl(c(240,0)))))

    "B"=mosaicplot(UCBAdmissions[,,2],
      xlab = "Admit", ylab = "Sex",
      main = paste("Department", input$mo),col = hcl(c(240,0)))))

    "C"=mosaicplot(UCBAdmissions[,,3],
      xlab = "Admit", ylab = "Sex",
      main = paste("Department", input$mo),col = hcl(c(240,0)))))

    "D"=mosaicplot(UCBAdmissions[,,4],
      xlab = "Admit", ylab = "Sex",
      main = paste("Department", input$mo),col = hcl(c(240,0)))))

    "E"=mosaicplot(UCBAdmissions[,,5],
      xlab = "Admit", ylab = "Sex",
      main = paste("Department", input$mo),col = hcl(c(240,0)))))

    "F"=mosaicplot(UCBAdmissions[,,6],
      xlab = "Admit", ylab = "Sex",
      main = paste("Department", input$mo),col = hcl(c(240,0)))))

# chi square test sex and admit
output$chi_sa <- renderPrint({
  chisq.test(apply(UCBAdmissions,c(1,2),sum))
})

# chi square test admit and department
output$chi_ad <- renderPrint({
  chisq.test(apply(UCBAdmissions,c(1,3),sum))
})
#chi square test sex and department

```r
output$chi_sd <- renderPrint({
  chisq.test(apply(UCBAdmissions,c(2,3),sum))
})
```

#switch statement to switch Chi square test for each department

```r
output$chi_dept <- renderPrint({
  switch(input$pick,
         "A" = chisq.test(UCBAdmissions[,1]),
         "B" = chisq.test(UCBAdmissions[,2]),
         "C" = chisq.test(UCBAdmissions[,3]),
         "D" = chisq.test(UCBAdmissions[,4]),
         "E" = chisq.test(UCBAdmissions[,5]),
         "F" = chisq.test(UCBAdmissions[,6]),
   )
})
```

#plot of critical value and t statistic

```r
output$tstat <- renderPlot({
  x1 <- seq(-5, 5, len = 100)
  y1 <- dt(x1, 5)
  plot(x1, y1, type="l")
  lines(x1, rep(0, 100))
  x2 <- seq(5, 2.26, len = 100)
  y2 <- rep(0, 100)
  x3 <- seq(2.26, 5, len = 100)
  y3 <- dt(x3, 5)
  x4 <- seq(5, 4.06, len=100)
  x5 <- seq(4.06,5, len=100)
  y4 <- dt(x5,5)
  polygon( c( x2, x3 ), c( y2, y3 ), density = 20, col = 2 )
  polygon( -c( x2, x3 ), c( y2, y3 ), density = 20, col = 2 )
  polygon( c( x4, x5 ), c( y2, y4 ), density = 20, col = 3 )
  polygon( -c( x4, x5 ), c( y2, y4 ), density = 20, col = 3 )
})
```

#plot of degrees of freedom

```r
output$df <- renderPlot({
  x <- seq(-5, 5, length=100)
  hx <- dnorm(x)
  plot(x, hx, type="l", lty=1, lwd=2.5, xlab="x value",
       ylab="Density", main="Comparison of t Distributions")
  lines(x, dt(x,input$dfslide), lwd=1.5, col="red")
})
```

#Maths equation
output$math <- renderUI{
  withMathJax(helpText('Dynamic output 1: $\alpha^2$'))
}

#plot of sleep data
output$boxplot <- renderPlot{
  ggplot(sleep,aes(x=group,y=extra,fill=group))+ geom_boxplot()
}

#boxplots of two groups and difference
output$Tboxplot <- renderPlot{
  sleep1 <- (sleep[1:10, 1] - sleep[11:20, 1])
  sleep1 <- data.frame(diff=sleep1)

  switch(input$tbox,
    "Boxplots of both groups" =
      ggplot(sleep,aes(x=group,y=extra,fill=group))+ geom_boxplot(),
    "Boxplot of difference of two groups" =
      ggplot(sleep1, aes(y=diff, x=factor(1)))+ geom_boxplot(fill="lightblue"),
  )
}

#output of paired t-test
output$paired <- renderPrint{
  t.test(extra ~ group, data = sleep, paired=TRUE)
}

#output for calculating critical value
output$crit <- renderPrint{
  qt(0.025, 9, lower.tail= FALSE)
}

#output for calculating P value
output$pvalue <- renderPrint{
  2*pt(4.06, 9, lower.tail= FALSE)
}

#sleep dataset table
output$sleep <- renderTable{
  sleep
}

#Boxplot of Boron data for ANOVA example

boron_table = data.frame(Lab1,Lab2,Lab3,Lab4)
oboron_table
output$Anova_box <- renderPlot({
  ggplot(Boron_data, aes(x=Lab, y=Boron,fill=Lab)) + geom_boxplot()
})

# Dotplot of Boron data for Anova example
output$Anova_dot <- renderPlot({
  ggplot(Boron_data, aes(x=Lab, y=Boron,colour=Lab,size=5)) + geom_point()
})

# Anova Table Function
output$Anova_table <- renderPrint({
  x <- c(Lab1, Lab2, Lab3, Lab4)
  Lab <- rep(c("Lab1", "Lab2", "Lab3", "Lab4"), each = 6)
  anova(lm(x~Lab))
})

####################
### Regression #####
####################

output$Regression <- renderPlot({
  # construct dataset
  spot_welds <- data.frame(Strength=c(Strength), Diameter=c(Diameter))
  # linear model
  model <- lm(Strength~Diameter)
  # add prediction interval
  predict.int <- predict(model, interval = "prediction")
  # add confidence interval
  conf.int <- predict(model, interval = "confidence")
  # plot data
  plot(Strength~Diameter)
  # add regression line
  if (input$reg == TRUE) {
    abline(model)
  }
  # add prediction interval
  if (input$pre == TRUE) {
    lines(Diameter, predict.int[, 2], lty = 2, col = 2)
    lines(Diameter, predict.int[, 3], lty = 2, col = 2)
  }
  # add confidence interval
  if (input$con == TRUE) {
    lines(Diameter, conf.int[, 2], lty = 3, col = 3)
    lines(Diameter, conf.int[, 3], lty = 3, col = 3)
  }
})

output$welds <- renderTable({
  spot_welds <- data.frame(Strength=c(Strength), Diameter=c(Diameter))
})
```r
spot_welds

})

###File Upload #######

#Checks uploaded file to see if it is csv if not warning message displayed
output$upload <- renderTable{
  #assign uploaded file to a variable
  File <- input$file1
  #catches null exception
  if (is.null(File))
    return(NULL)
  #validate statement to check file uploaded is format we want
  validate(
    need(file_ext(File$name) %in% c(
      'text/csv',
      'text/comma-separated-values',
      'text/tab-separated-values',
      'text/plain',
      'csv',
      'tsv'
    ), "Wrong File Format please try again!")
  read.csv(File$datapath)
})
```

REFERENCES


Websites:

www.shiny.rstudio.com
www.r-tutor.com
http://stackoverflow.com/questions/tagged/shiny
http://www.cookbook-r.com/
THE STATISTICS DISCIPLINE
A Friendly Web Interface to R Using R Shiny for Some Frequently Used Statistical Analysis Methods

20 March 2015

TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>NO</th>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>INTRODUCTION AND OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>INSTALLING SHINY IN R STUDIO</td>
<td>1</td>
</tr>
<tr>
<td>3.</td>
<td>RUNNING A SHINY APPLICATION</td>
<td>2</td>
</tr>
<tr>
<td>3.1</td>
<td>From Local Directory</td>
<td>2</td>
</tr>
<tr>
<td>3.2</td>
<td>From GitHub Repository</td>
<td>3</td>
</tr>
<tr>
<td>3.3</td>
<td>Running Shiny on your Browser</td>
<td>4</td>
</tr>
<tr>
<td>4.</td>
<td>NAVIGATING THE WEB INTERFACE</td>
<td>5</td>
</tr>
<tr>
<td>5.</td>
<td>UPLOADING A DATASET TO THE WEB INTERFACE</td>
<td>6</td>
</tr>
<tr>
<td>6.</td>
<td>EDITING AND ADDING TO THE SHINY APPLICATION</td>
<td>8</td>
</tr>
</tbody>
</table>
1. INTRODUCTION AND SUMMARY.

This user manual has been prepared to help current and prospective users of the interactive web interface understand how the developed system works. The manual runs through first how to install the Shiny package in R Studio, then briefly runs through how the web interface is navigated through and to use the file upload section of the application. Finally the user manual gives a brief explanation of how the system could be adapted or added to.

2. INSTALLING SHINY IN R STUDIO

Installing the Shiny package in R Studio is a very straightforward process and is no different from installing any other package in R Studio. First of all make sure you are running an up to date version of R Studio as older versions are unable to run the Shiny package.

Once you have loaded up your most recent version of R Studio all you have to do to download and install the shiny package is enter the following line of code

```
> install.packages("shiny")
```

This will download and install the shiny package to your R Studio but you still won’t be able to use the shiny package until you load it up. To do this enter the following line of code in to your console.

```
> library(shiny)
```

Once you have completed this the Shiny package will be installed to your machine and ready to use.
3. **RUNNING A SHINY APPLICATION**

This section will explain the process of running a shiny app both from a local directory and from a GitHub repository.

3.1 **From a local directory**

To run a Shiny app from a local directory there is a number of steps involved. The first thing you must do is make the directory where the Shiny application is stored your working directory. To do this, follow these steps.

Enter the following code to change your working directory, except place where you have saved your Shiny app in between the inverted commas. In this example you can see I have set my working directory to a folder on my desktop called Shiny Web Interface

```r
> setwd("C:/Users/Ferdia/Desktop/Shiny Web Interface")
```

Once you have set your working directory to the correct location there is now two ways that you can run the Shiny app. The simplest method is to just enter the following line of code:

```r
runApp("app_name")
```

Replacing where it says “app_name” with the name of the directory you have saved your shiny app. This will load the shiny app from your working directory. A screen shot of this process can be seen below. You can see for this example that the app I loaded is called “SHINY_APP”.

```r
> runApp("SHINY_APP")
```
The second way to load up a shiny application from a local directory is to open either the ui.R or server.R file for the application that you want to run. Once either of these scripts is open in R studio, you can run the script by clicking the run app button in top right hand corner of the panel. You can see the run app button in the below screenshot in the top right corner of the script panel. Once you click this button the app will load up.

![Screenshot of R Studio with Run App button highlighted](image.png)

### 3.2 From a GitHub repository

To run a shiny application from a GitHub repository you have to use the `runGitHub()` function that comes as part of the Shiny package. This function works by downloading the Shiny applications files from the specified GitHub repository and then running them in R studio. To do this you must specify the username of the GitHub user and the name of the repository that the Shiny files are stored in. To do this enter the following lines of code:

```r
runGitHub("repository_name","user_name")
```

Replacing where it says “repository_name” with the name of the repository and where it says “user_name” with the GitHub username. A screen shot of the code needed to run this projects web interface can be seen below. You can see that the repository is called “Shiny_Stats_Interface” and the user name is “Stats-TCD”.

![Screenshot of R Console running runGitHub function](image.png)

This function has been tested on a number of machines and connected to a number of different networks. It works fine on the TCDwifi network but sometimes encounters errors on some of the other Trinity networks. This is mentioned in the challenges encountered section of the report.
3.3 Running shiny on your browser

A lot of the wrapper functions included in R Shiny only operate properly when loaded in a browser. For example the web interface made for this project has a lot of LateX code for displaying maths equations. This code implements the Math Jax wrapper function that's part of the Shiny framework. This code will only appear as maths equations while running in a web browser.

The two screenshots below demonstrate this, the first screenshot is of the introduction section of the Chi Square section loaded in R Studio. The second screenshot is of the same section except this time loaded in a web browser.

You can clearly see the difference it makes. To load shiny in your browser all you have to do is click the “open in browser” button that is located at the top of the screen when you load it in R Studio. You can see this button in the first screenshot just above the Home button on the navigation bar.

Shiny application running in R Studio
4. Navigating the web interface

The shiny web interface is very simple to navigate once loaded; all the user has to do is click on whatever section they want to open. The web interface has many layers of tabs that are used to navigate around the application. They are very self-explanatory and even a novice computer user should be able to navigate it with ease. Below is a screen shot of the t-test section of the web interface where you can see the multi layers of the interface clearly.

The same Shiny application running in a browser

\[ \chi^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \]

Where

\( \chi^2 \) = Test Statistic

\( I \) = \# number of rows in contingency table

\( J \) = \# number of columns

Chi square tests are commonly used to test if deviations in expected values are due to chance or by other contributing factors.
5. Uploading a dataset to the web interface

Users can upload their own CSV format datasets to the web interface. To do this they just have to follow these simple steps.

From the home screen navigate to the ‘File upload’ section of the web application seen here in the screenshot below circled in red.

From the file upload screen, click the choose file button. A window will pop up and will ask you to locate the file you wish to upload. When you have found the file you are looking for select it and click the open button.
Your chosen dataset should appear in the application. An example of a successful file upload can be seen here in the screenshot below.

If you try and upload a dataset in the incorrect file format you will receive a warning message.
6. **Editing and adding a to the shiny application**

In this section I will run through how the Shiny interface can be adapted or added to. This section is quite brief and will only run though some of the basics of Shiny development. For a more comprehensive guide see [www.Shiny.com](http://www.Shiny.com).

**Basic Shiny layouts**

One of the strengths of Shiny is that it is very easy to create attractive layouts in apps with relative ease. In Shiny you declare what kind of layout you want to use and then place a set of brackets after the layout function. Everything within these brackets will be included in your chosen layout and you can do this for as many layers you like.

**Editing a text field in Shiny**

All of the text in the Shiny interface is entered in the `ui.R` script. You first must tell the application what size and style of text you want to enter. There are numerous different text styles you can use a list of the few main ones can be seen here below.

```
p
<h1>
h2
<h2>
h3
<h3>
h4
<h4>
h5
<h5>
h6
<h6>
```

A paragraph of text
A first level header
A second level header
A third level header
A fourth level header
A fifth level header
A sixth level header

You then put whatever you want to write inside brackets and inverted commas. So if you wanted to print hello world in paragraph style text you would enter the following line of code.

```
p("HELLO WORLD")
```

**Adding R analysis to your Shin App**

All R analysis is placed in functions in the `server.r` section of your code. Every separate part of analysis needs its own function for example. I will run through a very basic example now. Let's say you wanted to print the sleep dataset in table form in your app. This is how you would go about it.

In the `server.R` section of your application you would declare an output variable for this example it is called `sleep`. You now have to choose a render function, a render function tells the user interface what it's about to render in this case we are rendering a table. There are numerous different render functions for different types of output. A table of different render functions can be seen here.
After you have declared your output variable and chosen your render function you now have to put your R analysis in to the function. For this example the R code is incredibly basic and all you have to put in is the word sleep, which will print the sleep dataset.

A screenshot of the code for the server.R file can be seen here below.

Now you have finished the code needed for the server section of the application but you must tell the user interface side where you want this table to be printed.

To do this you must place an output function in your user interface function. Output functions tell the user interface where to put the output and what type of output it is. A list output functions can be seen here below. For this example we want to print a table so we the tableOutput() function.

A screen shot of the code needed to print the sleep dataset as a table can be seen here below. You can see the output function has been placed within a side bar panel layout and along with a little bit of text explaining the dataset.
The output function works by calling on the variable name given to the output in the server section.

You can see here a screen shot of what appears in the application. The side bar panel here on the left

This was a very basic example of how you add R analysis to your application. The process stays the same though for more complex outputs. The basic steps are:

1. In the server.R file declare your output variable and choose the correct render function
2. Enter your R analysis within the render function
3. In ui.R file choose the correct output function and place it where you want your analysis to appear
4. Enter the name of your output variable in to the output function