Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Design and development of an app for statistical data analysis learning

MEMÒRIA

Autor:	Eduard Serrahima de Cambra
Director/s:	Lluís Marco Almagro
Convocatòria:	Juny 2015

ETSEIB

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

UPC

Abstract

As statistics grows in importance in our society due to the need to analyze the increasing amounts of data available, it is necessary to reinforce the role of statistics in education. However, the non-existence of free open-source statistical tools designed specifically for educational purposes make the task more difficult. Concern about this fact led to this project.

An app designed for statistical data analysis learning has been developed from scratch, and fully programmed using R. Diverse studies and profound research have been carried on to design the application, named StatClip. Different aspects have been taken into account, following a logical sequential structure for implementing the application: review of similar existing programs, characterization of target users, definition of user needs and translation into program characteristics. The result is a fully functional program with several important and useful functionalities implemented. More importantly, though, the whole architecture and structure of the software has been designed to make it easier to continue developing and adding functionalities to StatClip.

An emotional design study (more specifically, a study following the kansei engineering model) has been as well performed, to take into consideration sensations and feelings that StatClip's user interface conveys on users. The results led to rethink some features of the program's user interface (UI), although in general the initial UI's design was found adequate and correct.

A lot of working effort in this project has been devoted to program code for the application. As a result, the app's skeleton and structure, as well as many important functionalities, are already programmed. From now on, StatClip can continue growing and expanding, with the addition of new functionality, so that it becomes a complete and useful statistical educational tool.

Index

ETSEIB

List of figures

Introduction

"Probability and Statistics are very important components in our society and in a lot of scientific disciplines. They should, therefore, constitute an important part of the basic knowledge of every citizen" [1], wrote Miguel de Guzmán (translated from the original in Spanish). *"There are many countries that include these subjects in their secondary education programs, but in few of them this education has the desired efficiency".*

Statistics is becoming increasingly important in education, and this increase is due to the necessity (claimed by UNESCO, amongst other institutions) to give students "statistical literacy" [2]. In our current world, this literacy is necessary in common activities such as reading the newspapers or understanding election results, surveys and even medical diagnosis.

This "statistical literacy" includes having a critical point of view with statistical information, as well as understanding the basic statistical ideas and language. Burrill and Biehler [3], after a detailed study of various theoretical educational models and the statistics presence in educational curriculums in several countries, propose a list of fundamental statistical ideas:

- **Data**: Statistics has been defined as the science of data, and Moore [4] pointed out that the main point in statistics is reasoning from empirical data. While in other mathematics branches' education data and its context are normally imaginary and the main interest is in concepts, in statistics the data context is essential.
- **Graphics**: Due to its abundant presence in the media and Internet, the understanding of statistical graphics is important. Knowing how to plot data in an easy and understandable manner is, as well, really useful.
- **Random variability**: The study of variability is typical in statistics. In statistical education, it is important that students learn to perceive variability and to manage models to control and predict it.
- **Distribution**: Statistics teaching has to develop in students the ability to read, analyze and make inferences from data distributions.
- **Association and correlation**: The importance of the association concept in decision taking in uncertainty environments is high, and studies show that sometimes no mathematical rules are used in association, leading to incorrect conclusions.
- **Probability**: One of the main characteristics of Statistics is that it uses random models. It is important, then, that students are able to understand and work with these models, as well as to understand the probability concept.
- **Sampling and inference**: Relating the characteristics of the sample data with the

population they represent is the main goal in statistics, and it allows us to decide which data to collect and obtain conclusions with a certain degree of confidence.

A methodology for introducing these statistical ideas or concepts in students is starting, from early primary education, to work on projects involving the understanding of these ideas. An example can be the one explained in *El sentido estadístico y su desarrollo* (original in Spanish) [2], in which students were asked “Will women outpace men in the 200-meter race in the Olympics? In which year?” and had to analyze and study the available data to try to come to an answer to the question.

The difficulty with this approach is that there is the need to have statistical tools that allow to perform these analyses. These tools, moreover, need to be easily and freely accessible, and they have to be thought and directed towards statistical education. Furthermore, statistics nowadays makes no sense without the usage of computational techniques, as even in the simplest projects with small data sets doing calculations and graphics by hand is really slow and difficult. In addition, a great emphasis in computational aspects in statistics is counterproductive, as students focus on the calculation details, losing interest in the important concepts supporting the analysis and in the interpretation of results. In consequence, education in statistics needs computational tools that allow students to learn the statistical ideas and concepts.

And with the idea to cover this need, my project started.

After some discussions with Lluis Marco, director of this project, the idea of developing a tool, an app, to cover the just referred need started to grow. Although ambitious, it was highly motivating and opened in front of me a vast way of possibilities to explore.

Even though it can seem unnecessary, finding a name for the still not even started to design app was relatively important. Relatively because we did not need a name to start working; important because it gives the project a new shape, a new strength. Lluis Marco told me about a name that Pere Grima (Professor at the Department of Statistics) had thought of when the idea of this project first came to their minds, years ago. A name for a small and simple program, yet powerful, to help learning statistics. A name which made you directly think about statistics when heard, and that also reminded of a light, simple, fast app.

And that name, the name of the app for statistical data analysis learning, was **StatClip**.

Objectives of the StatClip project

The project tries to face the difficulties in the design and implementation of an app thought to learn statistics and data analysis. The main objectives of the project are the following:

1. To perform an exhaustive research of already existing statistical software (including both commercial and open-source programs, as well as programming languages). This is very important to understand what already exists, and to try to find what is needed. It also helps to get inspired towards the design of StatClip.
2. To decide to whom the app is addressed (users) and exhaustively analyzing their needs.
3. To define the list of characteristics of the program.
4. To design theoretically all the features and functionalities of StatClip, as well as its User Interface.
5. To select a computational tool to build the app.
6. To build a prototype of StatClip, using the selected computational tool. More specifically, building the inner architecture of the app, and implementing the functionalities, always keeping in mind to make StatClip easily expandable.
7. To test and validate the design of the app, using emotional design techniques.

The seven objectives just referred conform the main idea behind this project. The StatClip project combines theoretical research with a strong practical work: programming a statistical app. Moreover, several statistical analysis techniques are used in the emotional design study.

Scope of the project

Being StatClip such an ambitious project, it is important to define its scope.

On the existing programs and pieces of software research, three different parts will be considered. In the first place, presenting programming languages that can be appropriate for the design of StatClip, focusing in languages widely used for data analysis and statistical programming. Secondly, a review of commercial statistical software, for they are widely used in the industrial world and thus can serve as inspiration and give ideas when designing StatClip. And in the third place, an extensive review of the most used or important freely available statistical programs; to see which approaches they have, collect ideas, etc.

In the users' and program characteristics' definition, both main users and operator users will be explained and considered. Main users are those for whom the app is designed for, and thus they are the most important. However, operator users (which are the ones working and managing the program) are important too and their needs must be considered in the design.

The theoretical design will cover all StatClip's features. From the design of its functionalities (which to introduce and how) to the graphical user interface. User inputs and StatClip's outputs will be defined and explained. Functionalities that were firstly programmed will be explained in more detail.

The different tools or languages considered will be described, and each option’s pros and cons will be detailed and considered for the final decision. Exhaustive and detailed introduction to the selected language will be then provided.

Referring now to the actual building of the app, it is important to note that the main objective is to construct the inner architecture of StatClip. That is, building the structure so that StatClip is easily expandable and new functionalities can be easily added. In this project we set the base for StatClip to keep growing. StatClip, at the end of this project, will be a fully functional app with several functionalities, prepared to be continued and expanded to achieve the complete app for statistical data analysis learning.

The emotional design study will be designed to see which visual parameters affect users’ perception of StatClip. More specifically, a kansei engineering study will be carried on. It will include the preparation of a questionnaire and volunteers filling it in, and the analysis of the obtained results. The analysis and synthesis of the data extracted from the study will be used to confirm the initial design or to propose changes in StatClip’s UI.

1. Software Resources for education in statistics

1.1. Study of the available programs

After having introduced the main idea and scope of the project, and as it has been explained in the Introduction, it is of great importance to carry on a profound investigation and research on which programs and pieces of software are already freely available; this is, determining the state of the art.

This research will lead to understanding and knowing the different pieces of software available online, and to learning what is and what is not already done, as well as to get ideas and inspiration.

There is a large number of free statistical programs; however, the scope of the project covers not only “pure formal” statistics, but also data plotting and visualization. In consequence, not only mathematical software has to be looked for, but also programs focused on presenting and plotting datasets. This widens the already large amount of options to get immersed in.

As it has already been stated, the main idea of the project is to develop a program mainly addressed to people learning statistics, and specially thought for secondary education students.

1.1.1. Programming languages

The idea is to be able to use the designed software with no need to have any programming knowledge, but it is necessary to introduce some of the programming languages that are used the most in the statistical world, as may be used for making the app.

1.1.1.1. R

The most used programming language for statistical purpose all around the world is R. R is an implementation of the S language, created at the University of Auckland, New Zealand [5].

It is a powerful language, specially designed for statistical data analysis. Learning R, however, is not easy. And for students with no programming background, the inner difficulty of the language makes even more difficult the task of learning the principles of data analytics.

A lot of the different software analyzed for this first part of the project is based on R.

1.1.1.2. Python

Nowadays really used due to its versatility and flexibility, Python is a high-level programming language [6]. Its design philosophy focuses on syntax simplicity and code readability.

There are a lot of online free libraries on data analysis for Python (such as, for example, Pandas), and it is possible to work and create graphs and plots using some of those libraries [7]. Due to its flexibility and easiness, Python is, as R, one of the most used programming languages for data analysis.

Some of the programs studied are based as well on Python.

1.1.1.3. GNU Octave

Octave has some similarities with Matlab, and that is why it is often considered as its free alternative. It allows the user to perform complex numerical computations, and solve linear and nonlinear problems.

GNU Octave is not designed particularly for statistical and data analytics calculations. However, as we are talking about a powerful tool for numerical analysis, it can be used with no major problem with this purpose. It provides, as well, capabilities for data visualization and manipulation.

However, Octave has no GUI (Graphical User Interface). The only way to perform the calculations is using the command line. This, again, requires from the user previous programming knowledge.

1.1.2. Commercial Statistical Software

Even though, as it has been already stated, the aim of the project is to create a free program, it is important to revise the already existent commercial software.

The most used programs for statistical analysis at a professional level are not freely available. Studying their characteristics and most important features can provide ideas and a view of what is considered primordial in a data analysis tool.

In this chapter, three pieces of software will be briefly commented.

1.1.2.1. Minitab Statistical Software

Minitab is a statistics package which was first developed by researchers Barbara F. Ryan, Thomas A. Ryan, Jr., and Brian L. Joiner in 1972 [8]. Currently on its version 17, Minitab is a powerful program capable of performing a wide variety of statistical calculations and simulations.

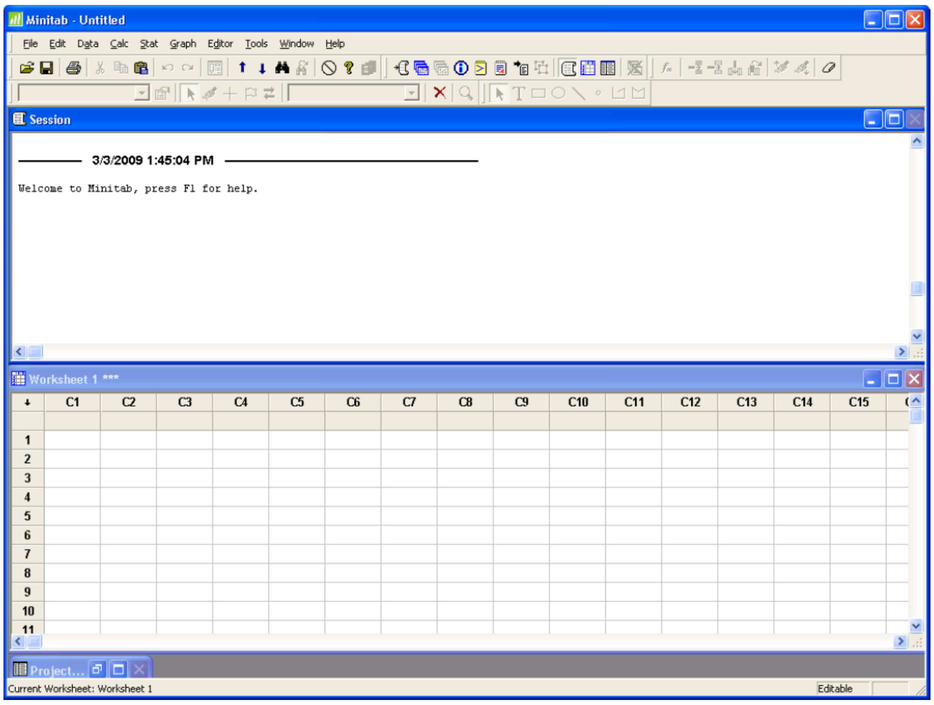
Minitab is nowadays often used together with the implementation of Six Sigma and other process improvement statistics-based methodologies.

Widely used in the industrial world, it is also common in universities, although its relatively high license price makes it unavailable for schools or high schools.

Its user interface is defined by the following characteristics:

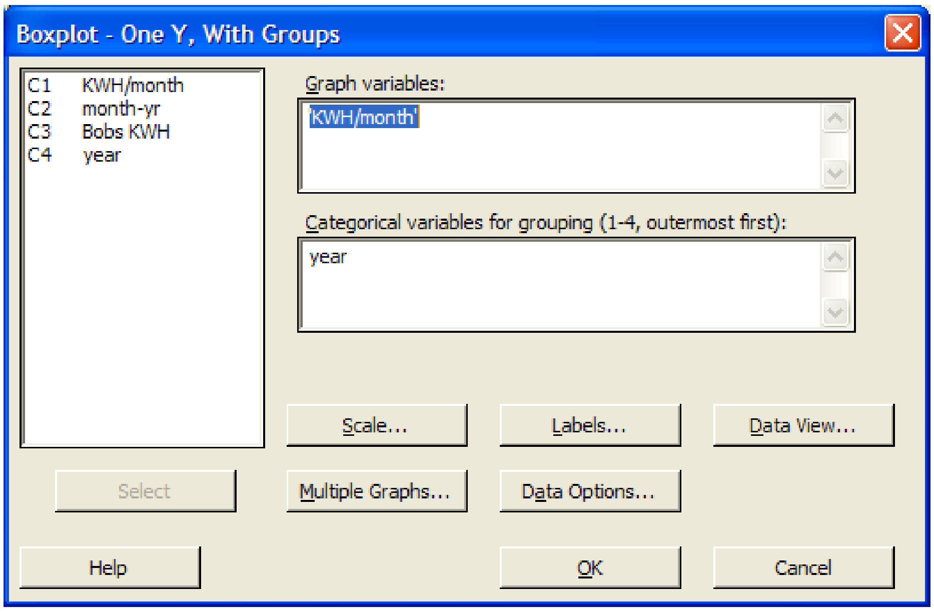
- Main screen divided in two independent windows. One (Data Window) contains a grid in which the user can introduce the data manually, directly import it from another software or generate it by performing a simulation. The other screen (Session Window) shows the numerical results of the operations performed on the data in the first screen.
- A toolbar with all the drop-down menus at the top part of the window. Those menus contain all the operations, calculations, tests, etc. Minitab is able to perform.
- Each action taken opens a new window. By this it is meant that, for example, if “Histogram” is selected, a dialog box window will appear asking the user to introduce the necessary conditions for the operation to be plausible.
- Plots will appear, as well, in pop-out windows. Numerical results however, and as it has already been stated, will be displayed on the Session Window.

The properties just explained can be seen on the images below.


Minitab - Untitled
File Edit Data Calc Stat Graph Editor Tools Window Help
Session
Worksheet 1 ***
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15
Project...
Current Worksheet: Worksheet 1
Editable

```
———— 3/3/2009 1:45:04 PM ————————————

Welcome to Minitab, press F1 for help.
```

Figure 1.1. Minitab Session and Data windows



C1	KWH/month
C2	month-yr
C3	Bobs KWH
C4	year

Graph variables: 'KWH/month'

Categorical variables for grouping (1-4, outermost first): year

Figure 1.2. Dialog box for Boxplots

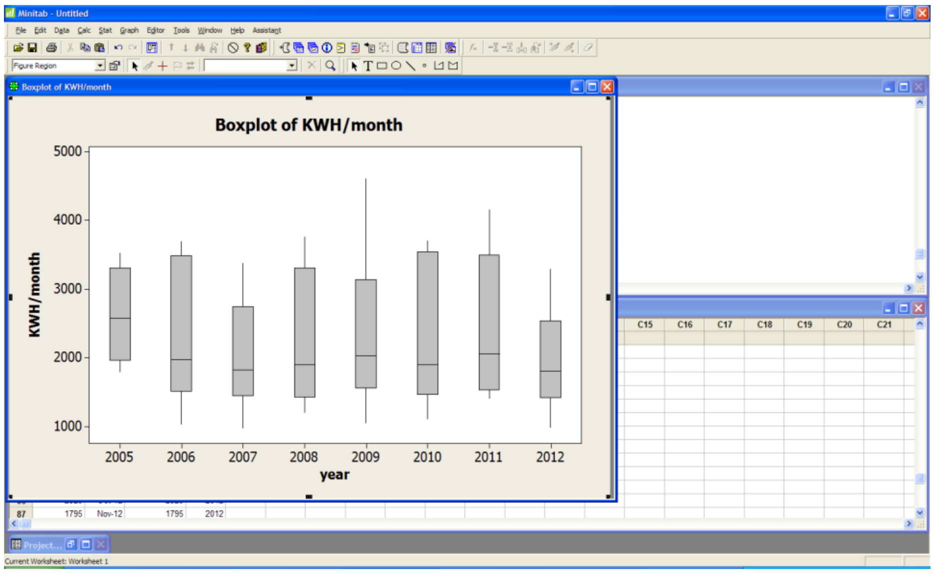


Figure 1.3. View of the resulting Boxplot (pop-out window)

1.1.2.2. JMP

Developed by SAS Institute Inc., it is fully programmed using C (about 160.000 lines). It was conceived and initially designed by John Sall [9].

JMP originally stood for “John’s Macintosh Program”, as the idea was incorporating the usability and beauty of Macintosh operating systems in a statistical program.

JMP focuses on interactivity using the mouse, giving users the possibility to see reactive changes on the results. This makes a difference with other programs, such as Minitab; in which users must introduce the operation conditions and then get a fixed result. JMP allows users to interact with the outcome of the operations performed.

Another difference with Minitab Statistical Software is that JMP is more data-driven, instead of method-driven.

JMP, as Minitab, has a Home Window and boxes that open when choosing the operation or computation to perform. In JMP’s case, though, a data window is not opened by default as in Minitab. In Figure 1.4 we can see JMP’s Home Window.

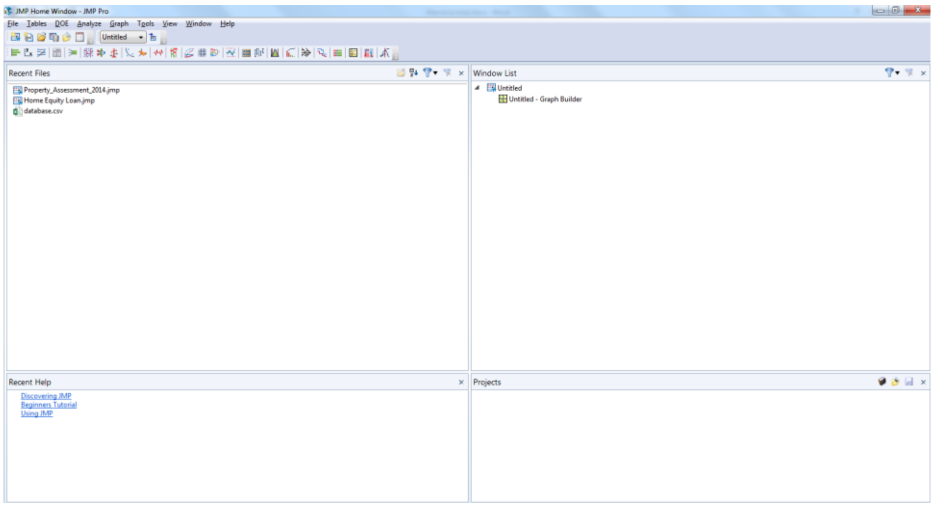
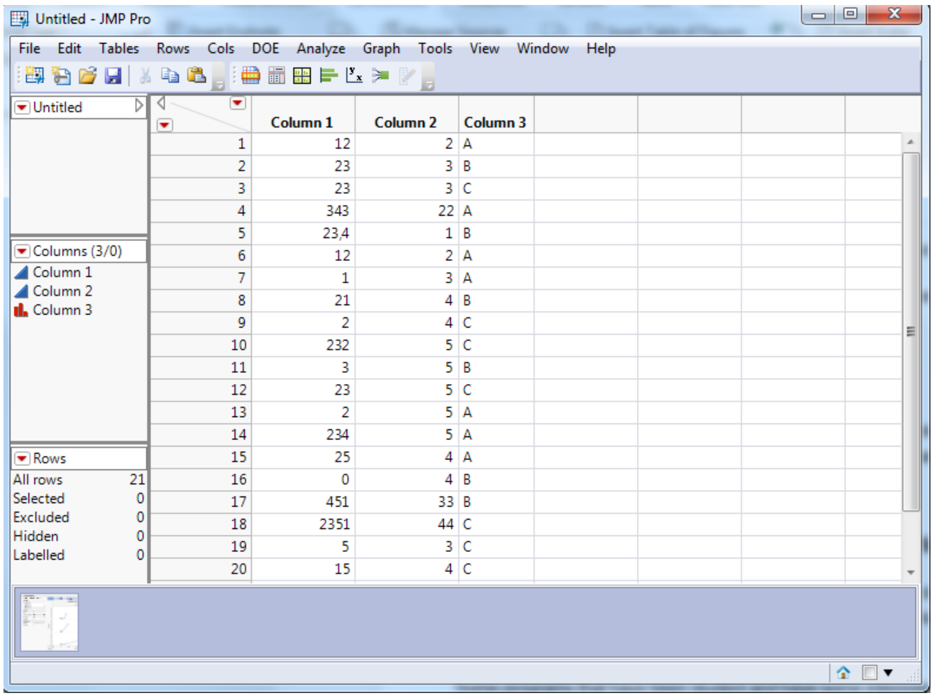


Figure 1.4. JMP. Home Window

When we open or create a Data table, the following window opens up.


Untitled - JMP Pro
File Edit Tables Rows Cols DOE Analyze Graph Tools View Window Help
Untitled
Columns (3/0)
Column 1
Column 2
Column 3

	Column 1	Column 2	Column 3
1	12	2	A
2	23	3	B
3	23	3	C
4	343	22	A
5	23,4	1	B
6	12	2	A
7	1	3	A
8	21	4	B
9	2	4	C
10	232	5	C
11	3	5	B
12	23	5	C
13	2	5	A
14	234	5	A
15	25	4	A
16	0	4	B
17	451	33	B
18	2351	44	C
19	5	3	C
20	15	4	C

Rows	
All rows	21
Selected	0
Excluded	0
Hidden	0
Labelled	0

Figure 1.5. Data Table Window

Numbers and characters can be introduced in the columns, and operations can be performed selecting from the drop-down menus at the top of the window. As it has been said, JMP looks for its operations to be highly interactive and user friendly. A good example is what we see in Figure 1.6 and Figure 1.7: the Graph Builder. Users have to drag the variables into the different drop zones. This gives users the possibility of "playing" with the different possibilities in an easy and fast way. Changing from one graph to the other is just a click away, and only by dragging the variables to a different position the graph changes.

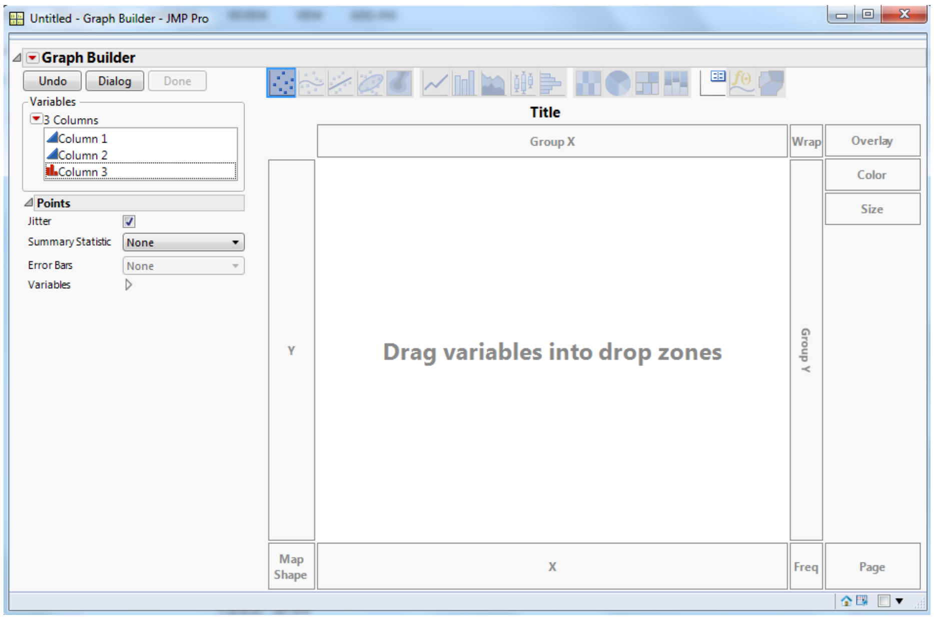


Figure 1.6. JMP Graph Builder

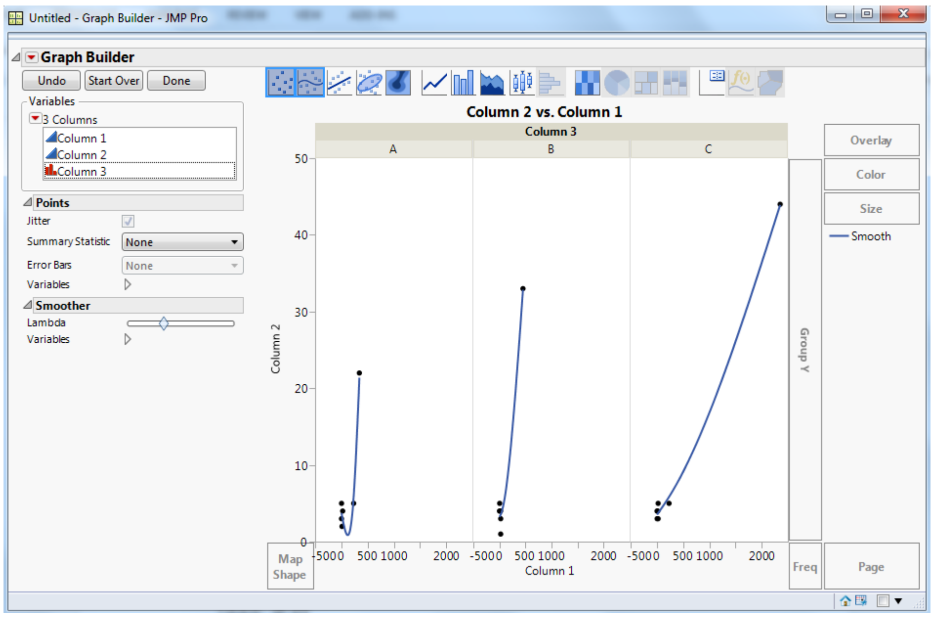


Figure 1.7. JMP Graph Builder (2)

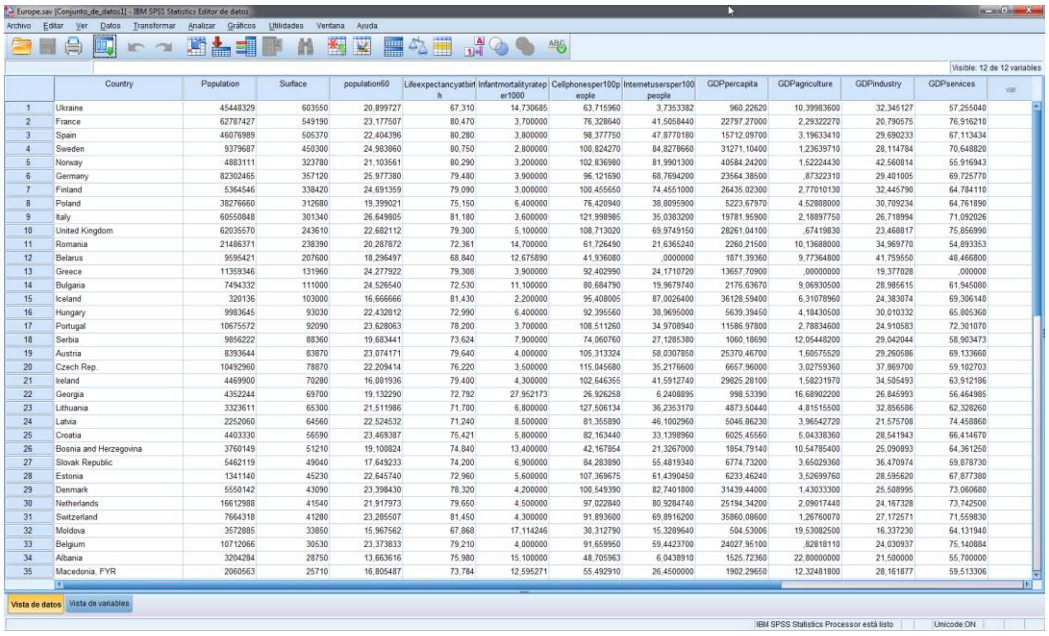
JMP is used mainly in the industrial and academic worlds, similarly to Minitab. The mentioned interactivity in JMP makes this software very appropriate for educational purposes.

1.1.2.3. SPSS

Originally standing for *Statistical Package for the Social Sciences*, SPSS is a statistical application for data analysis. First released in 1968, SPSS became part of IBM in 2009. Its graphical user interface was developed using Java.

SPSS includes many methods specially used in psychology, sociology and social sciences in general. This software, moreover, has R and Python integration packages, with which users can combine SPSS' power with the flexibility of Python or R programming.

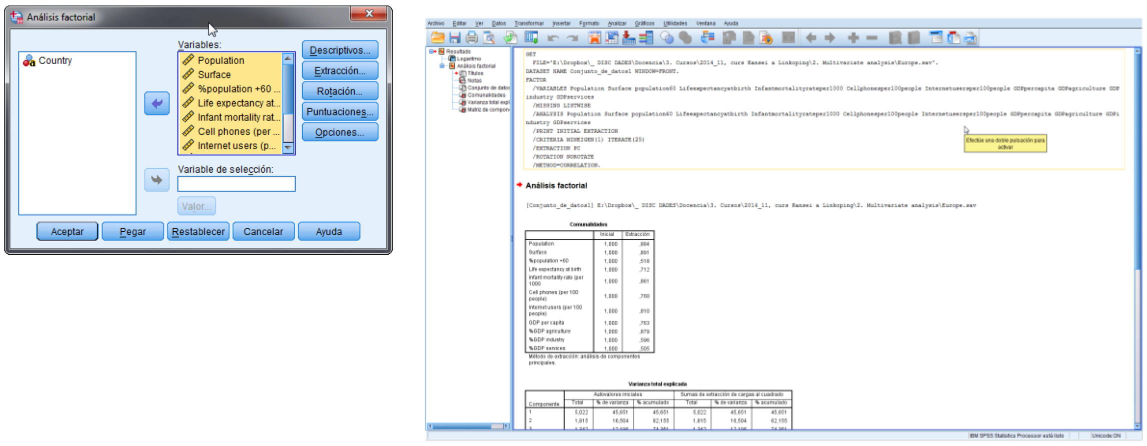
Figure 1.8 shows SPSS' data editor. Operations are performed selecting them from the dropdown menus.


Europe.sav [Conjunto_de_datos1] - IBM SPSS Statistics Editor de datos
Archivo
Editar
Ver
Datos
Transformar
Analizar
Gráficos
Utilidades
Ventana
Ayuda
Visible: 12 de 12 variables
Vista de datos
Vista de variables
IBM SPSS Statistics Processor está listo
Unicode:ON

	Country	Population	Surface	population60	Lifeexpectancyatbirth	Infantmortalityrateper1000	Cellphonesper100people	Internetusersper100people	GDPpercapita	GDPagriculture	GDPindustry	GDPservices	var
1	Ukraine	45448329	603550	20,899727	67,310	14,730685	63,715960	3,7353382	960,22620	10,39983600	32,345127	57,255040	
2	France	62787427	549190	23,177507	80,470	3,700000	76,328640	41,5058440	22797,27000	2,29322270	20,790575	76,916210	
3	Spain	46076989	505370	22,404396	80,280	3,800000	98,377750	47,8770180	15712,09700	3,19633410	29,690233	67,113434	
4	Sweden	9379687	450300	24,983860	80,750	2,800000	100,824270	84,8278660	31271,10400	1,23639710	28,114784	70,648820	
5	Norway	4883111	323780	21,103561	80,290	3,200000	102,836980	81,9901300	40584,24200	1,52224430	42,560814	55,916943	
6	Germany	82302465	357120	25,977380	79,480	3,900000	96,121690	68,7694200	23564,38500	,87322310	29,401005	69,725770	
7	Finland	5364546	338420	24,691359	79,090	3,000000	100,455650	74,4551000	26435,02300	2,77010130	32,445790	64,784110	
8	Poland	38276660	312680	19,399021	75,150	6,400000	76,420940	38,8095900	5223,67970	4,52888000	30,709234	64,761890	
9	Italy	60550848	301340	26,649805	81,180	3,600000	121,998985	35,0383200	19781,95900	2,18897750	26,718994	71,092026	
10	United Kingdom	62035570	243610	22,682112	79,300	5,100000	108,713020	69,9749150	28261,04100	,67419830	23,468817	75,856990	
11	Romania	21486371	238390	20,287872	72,361	14,700000	61,726490	21,6365240	2260,21500	10,13688000	34,969770	54,893353	
12	Belarus	9595421	207600	18,296497	68,840	12,675890	41,936080	,0000000	1871,39360	9,77364800	41,759550	48,466800	
13	Greece	11359346	131960	24,277922	79,308	3,900000	92,402990	24,1710720	13657,70900	,00000000	19,377028	,000000	
14	Bulgaria	7494332	111000	24,526540	72,530	11,100000	80,684790	19,9679740	2176,63670	9,06930500	28,985615	61,945080	
15	Iceland	320136	103000	16,666666	81,430	2,200000	95,408005	87,0026400	36128,59400	6,31078960	24,383074	69,306140	
16	Hungary	9983645	93030	22,432812	72,990	6,400000	92,395560	38,9695000	5639,39450	4,18430500	30,010332	65,805360	
17	Portugal	10675572	92090	23,628063	78,200	3,700000	108,511260	34,9708940	11586,97800	2,78834600	24,910583	72,301070	
18	Serbia	9856222	88360	19,683441	73,624	7,900000	74,060760	27,1285380	1060,18690	12,05448200	29,042044	58,903473	
19	Austria	8393644	83870	23,074171	79,640	4,000000	105,313324	58,0307850	25370,46700	1,60575520	29,260586	69,133660	
20	Czech Rep.	10492960	78870	22,209414	76,220	3,500000	115,045680	35,2176600	6657,96000	3,02759360	37,869700	59,102703	
21	Ireland	4469900	70280	16,081936	79,400	4,300000	102,646355	41,5912740	29825,28100	1,58231970	34,505493	63,912186	
22	Georgia	4352244	69700	19,132290	72,792	27,952173	26,926258	6,2408895	998,53390	16,68902200	26,845993	56,464985	
23	Lithuania	3323611	65300	21,511986	71,700	6,800000	127,506134	36,2353170	4873,50440	4,81515500	32,856586	62,328260	
24	Latvia	2252060	64560	22,524532	71,240	8,500000	81,355890	46,1002960	5046,86230	3,96542720	21,575708	74,458860	
25	Croatia	4403330	56590	23,469387	75,421	5,800000	82,163440	33,1398960	6025,45560	5,04338360	28,541943	66,414670	
26	Bosnia and Herzegovina	3760149	51210	19,100824	74,840	13,400000	42,167854	21,3267000	1854,79140	10,54785400	25,090893	64,361250	
27	Slovak Republic	5462119	49040	17,649233	74,200	6,900000	84,283890	55,4819340	6774,73200	3,65029360	36,470974	59,878730	
28	Estonia	1341140	45230	22,645740	72,960	5,600000	107,369675	61,4390450	6233,46240	3,52699760	28,595620	67,877380	
29	Denmark	5550142	43090	23,398430	78,320	4,200000	100,549390	82,7401800	31439,44000	1,43033300	25,508995	73,060680	
30	Netherlands	16612988	41540	21,917973	79,650	4,500000	97,022840	80,9284740	25194,34200	2,09017440	24,167328	73,742500	
31	Switzerland	7664318	41280	23,285507	81,450	4,300000	91,893600	69,8916200	35860,08600	1,26760070	27,172571	71,559830	
32	Moldova	3572885	33850	15,967562	67,868	17,114246	30,312790	15,3289640	504,53006	19,53082500	16,337230	64,131940	
33	Belgium	10712066	30530	23,373833	79,210	4,000000	91,659950	59,4423700	24027,95100	,82818110	24,030937	75,140884	
34	Albania	3204284	28750	13,663616	75,980	15,100000	48,705963	6,0438910	1525,72360	22,80000000	21,500000	55,700000	
35	Macedonia, FYR	2060563	25710	16,805487	73,784	12,595271	55,492910	26,4500000	1902,29650	12,32481800	28,161877	59,513306	

Figure 1.8. SPSS Data editor

The method for performing statistical analysis in SPSS is the usual one in all point-and-click statistical software packages. Once the method is selected from the menu, a dialog box appears (Figure 1.9, left). Results are shown in a new window, combining text results with graphs, when necessary (Figure 1.9, right).



```
GET
  FILE='E:\Dropbox\_ DISC DADES\Docencia\3. Cursos\2014_11, curs Kansei a Linkoping\2. Multivariate analysis\Europe.sav'.
DATASET NAME Conjunto_de_datos1 WINDOW=FRONT.
FACTOR
  /VARIABLES Population Surface population60 Lifeexpectancyatbirth Infantmortalityrateper1000 Cellphonesper100people Internetusersper100people GDPpercapita GDPagriculture GDP
industry GDPservices
  /MISSING LISTWISE
  /ANALYSIS Population Surface population60 Lifeexpectancyatbirth Infantmortalityrateper1000 Cellphonesper100people Internetusersper100people GDPpercapita GDPagriculture GDPi
ndustry GDPservices
  /PRINT INITIAL EXTRACTION
  /CRITERIA MINEIGEN(1) ITERATE(25)
  /EXTRACTION PC
  /ROTATION NOROTATE
  /METHOD=CORRELATION.
```

Análisis factorial

```
[Conjunto_de_datos1] E:\Dropbox\_ DISC DADES\Docencia\3. Cursos\2014_11, curs Kansei a Linkoping\2. Multivariate analysis\Europe.sav
```

Comunalidades

	Inicial	Extracción
Population	1,000	,884
Surface	1,000	,891
%population +60	1,000	,518
Life expectancy at birth	1,000	,712
Infant mortality rate (per 1000	1,000	,861
Cell phones (per 100 people)	1,000	,760
Internet users (per 100 people)	1,000	,810
GDP per capita	1,000	,763
%GDP agriculture	1,000	,879
%GDP industry	1,000	,596
%GDP services	1,000	,505

Método de extracción: análisis de componentes principales.

Varianza total explicada

Componente	Autovalores iniciales			Sumas de extracción de cargas al cuadrado		
	Total	% de varianza	% acumulado	Total	% de varianza	% acumulado
1	5,022	45,651	45,651	5,022	45,651	45,651
2	1,815	16,504	62,155	1,815	16,504	62,155

Figure 1.9. Dialog box and results windows in SPSS 22

1.1.3. Freely available Statistical and Data Analysis Software

There are lots of different programs thought to perform numerical and statistical analysis. The aim of this section is to refer those that have a set of characteristics that can be similar to the ones desired for our application.

Some programs that have been studied and have some interesting characteristics will be more extensively defined, while others will be just mentioned by its important or different conditions. R and Python packages or libraries have not been included amongst the following programs, as they are considered part of the programming language.

1.1.3.1. Statistical Lab

Program based on R, designed by CeDiS (Center Für Digitale Systeme), of the Berlin University, and thought specifically to support education in statistics [10].

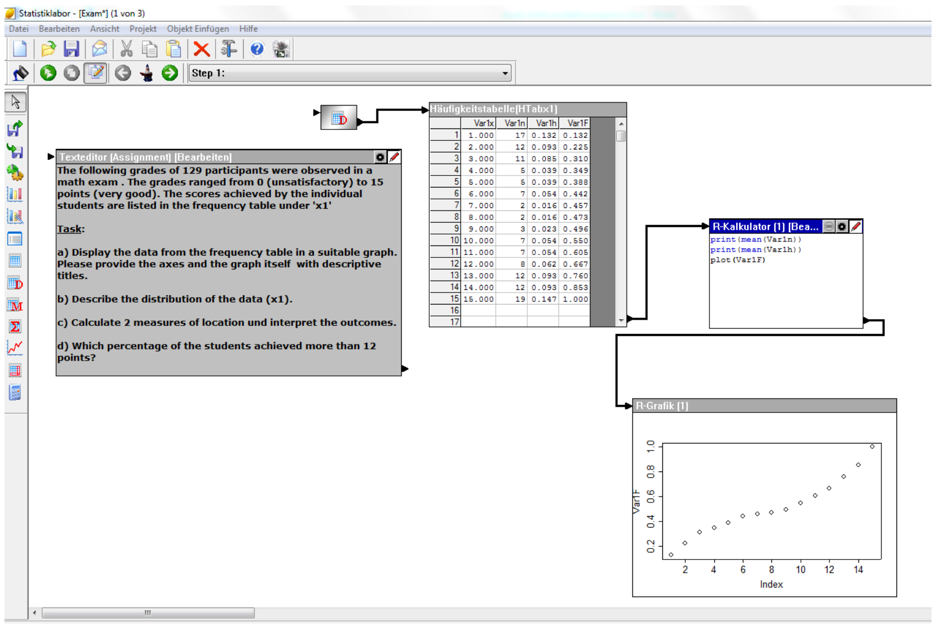
Available only for Windows operating system, it has a user interface specially designed to simplify interaction and make complex statistical calculations easier.

Its approach is interesting; when initially opening it, we face an empty window with only the menus. We can create tables and graphs, connecting them with arrows (linking them), as we can see in Figure 1.10, where we connect the data table to the R Calculator, and this last to a graph displayer.

Even though the GUI of this software is not easy to understand at first, it is original and makes more sense the more you use it. However, the fact that every menu is only in German makes the task really complicated.

Moreover, the main actions of the program are carried on by the R Calculator, which is nothing more than an R command line. In consequence, it is necessary to learn how to program in R to be able to fully use this software.

Taking this into consideration, then, Statistical Lab can be appropriate for learning statistics at a higher level, and for people with previous programming knowledge.


Statistiklabor - [Exam*] (1 von 3)
Datei Bearbeiten Ansicht Projekt Objekt Einfügen Hilfe
Step 1:
Texteditor (Assignment) [Bearbeiten]
Häufigkeitstabelle(HTabx1)
R-Kalkulator (1) [Bea...
R-Grafik (1)
Var1F
Index

The following grades of 129 participants were observed in a math exam . The grades ranged from 0 (unsatisfactory) to 15 points (very good). The scores achieved by the individual students are listed in the frequency table under 'x1'

Task:

a) Display the data from the frequency table in a suitable graph. Please provide the axes and the graph itself with descriptive titles.

b) Describe the distribution of the data (x1).

c) Calculate 2 measures of location und interpret the outcomes.

d) Which percentage of the students achieved more than 12 points?

	Var1x	Var1n	Var1h	Var1F
1	1.000	17	0.132	0.132
2	2.000	12	0.093	0.225
3	3.000	11	0.085	0.310
4	4.000	5	0.039	0.349
5	5.000	5	0.039	0.388
6	6.000	7	0.054	0.442
7	7.000	2	0.016	0.457
8	8.000	2	0.016	0.473
9	9.000	3	0.023	0.496
10	10.000	7	0.054	0.550
11	11.000	7	0.054	0.605
12	12.000	8	0.062	0.667
13	13.000	12	0.093	0.760
14	14.000	12	0.093	0.853
15	15.000	19	0.147	1.000
16				
17				

```
print(mean(Var1n))
print(mean(Var1h))
plot(Var1F)
```

Figure 1.10. Statistical Lab

1.1.3.2. Statcato

Statcato is a program under the GNU General Public License that was programmed fully using Java [11]. It can perform elementary statistical computations, including:

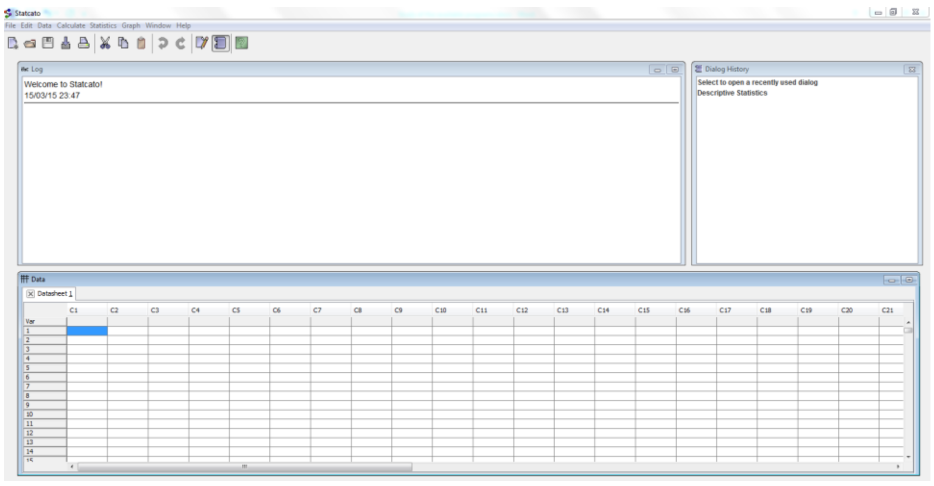
- Generate patterned data and random samples.
- Sort and standardize data.
- Probability calculations.

The main point in Statcato is that it is an open source program really similar to Minitab, although less powerful. Its graphical interface, as can be seen in Figure 1.11, is practically identical to Minitab’s.

This can be a really useful feature, as it allows the user with previous Minitab experience to a fast adaptation to Statcato. Moreover, it has the advantages of an easy and user-friendly environment, even though it has not been specifically designed for educational purposes.

The idea of having two separate screens (Data and Log, in this case) is well-spread amongst statistical software, as it gives the user the chance to see both the raw data and the results of the calculations.

As Statcato is free software with a General Public License (GPL), its source code can be downloaded (directly from its website), distributed or modified with no problem.



Welcome to Statcato!
15/03/15 23:47

Select to open a recently used dialog
Descriptive Statistics

Figure 1.11. Statcato

1.1.3.3. SalStat Statistics

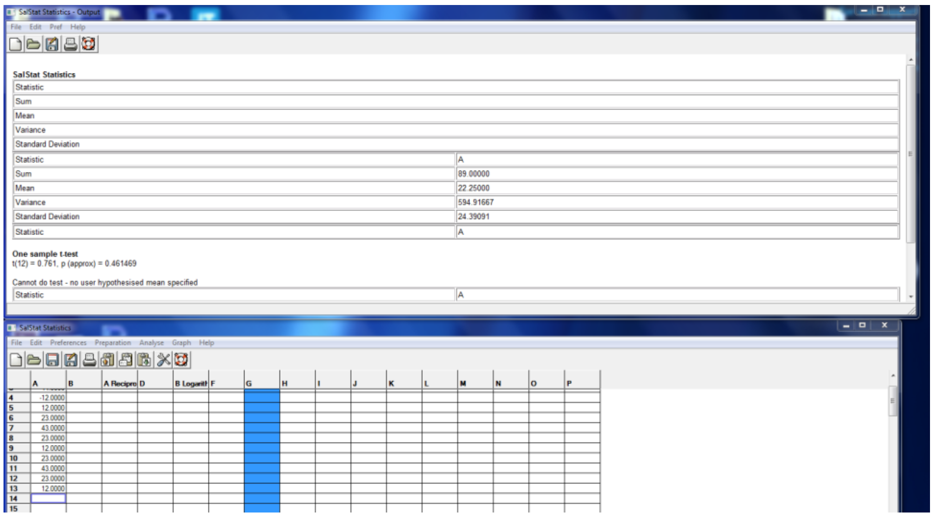
This program, written in Python, can run on Windows, Linux and Macintosh OS X. SalStat is an open source project fostered by *Thought into Design Ltd* [12].

Its graphical interface has some similarities with Minitab Statistical Software: a window with the data (table in which the user can modify the values) and an output window, although it is not as easy to use. Understanding its interface is not immediate, even having experience with Minitab, and may not be the most appropriate software for learning, as includes no explanations nor information on how to perform calculations and its meaning.

It can perform various statistical calculations, including descriptive statistics, parametric and non-parametric comparisons, correlations, linear regression and ANOVA. It also allows users with programming abilities to create new functions and calculation routines, using Python.

As it is an Open Source project, as well, anyone can download, see, modify and distribute the code that conforms SalStat.

To conclude, SalStat is an option to perform basic statistics from which parts of the code may be useful in the case that Python is the language chosen to program the app.


SalStat Statistics - Output
File Edit Pref Help
SalStat Statistics
File Edit Preferences Preparation Analyse Graph Help

```
SalStat Statistics
Statistic
Sum
Mean
Variance
Standard Deviation
Statistic            A
Sum                  89.00000
Mean                 22.25000
Variance             594.91667
Standard Deviation   24.39091
Statistic            A

One sample t-test
t(12) = 0.761, p (approx) = 0.461469

Cannot do test - no user hypothesised mean specified
Statistic            A
```

	A	B	A Recipro	D	B Logarit	F	G	H	I	J	K	L	M	N	O	P
4	-12.0000															
5	12.0000															
6	23.0000															
7	43.0000															
8	23.0000															
9	12.0000															
10	23.0000															
11	43.0000															
12	23.0000															
13	12.0000															
14																
15																

Figure 1.12. SalStat graphical user interface

1.1.3.4. SOFA Statistics

Released with open source AGPL3 (Affero General Public License) license and developed by Paton-Simpson & Associates Ltd, SOFA (Statistics Open for All) Statistics is a program designed with the intentionality of being an easy-to-use and user-friendly program [13].

It can be used in Linux, Mac and Windows and is programmed using Python, being its source code freely available.

What happens to be the most interesting fact about SOFA is its GUI. Completely different to most of the free available software, it focuses on usage easiness and visually attractive reporting.

Basically, the structure of SOFA usage is the following:

- The user is asked to import or manually introduce (two separate buttons) the data.
- The user can create report tables and frequency tables from the imported data.
- The fourth button on the initial screen guides the user to the graphics elaboration screen.
- The fifth opens a new window in which the user can select which statistical analysis to apply on the imported data.

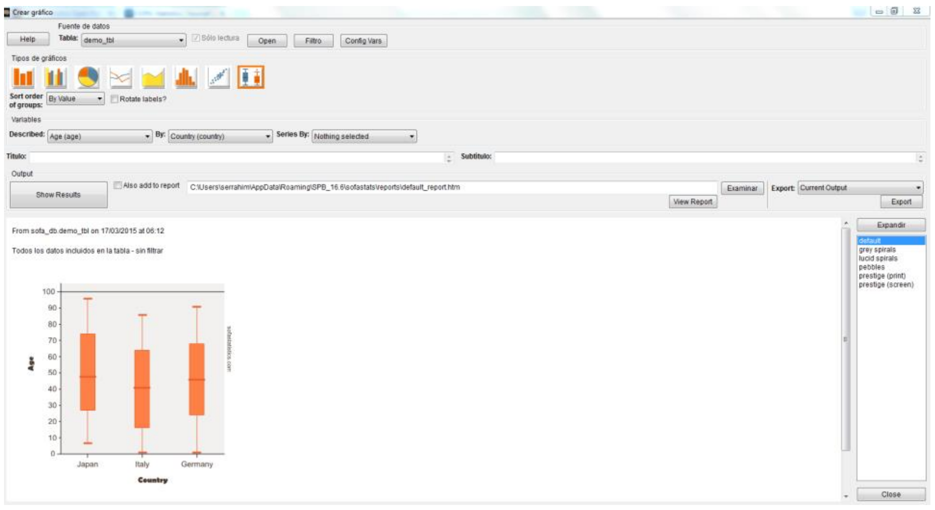


Tabla: demo_tbl
Sólo lectura
Sort order of groups: By Value
Rotate labels?
Described: Age (age)
By: Country (country)
Series By: Nothing selected
Titulo:
Subtitulo:
Also add to report C:\Users\serrahim\AppData\Roaming\SPB_16.6\sofastats\reports\default_report.htm
Export: Current Output

From sofa_db.demo_tbl on 17/03/2015 at 06:12

Todos los datos incluidos en la tabla - sin filtrar

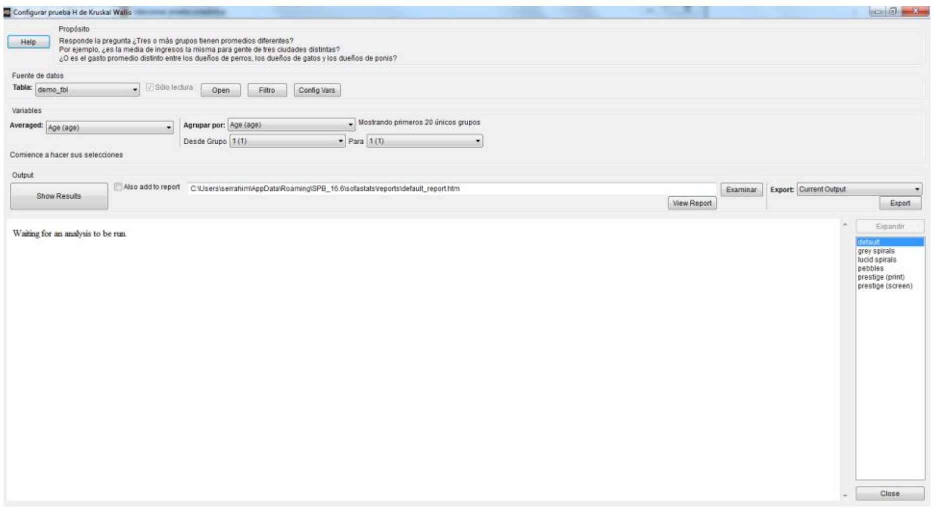
default
grey spirals
lucid spirals
pebbles
prestige (print)
prestige (screen)

Figure 1.13. SOFA Statistics: Graphics window

The software, as can be seen in Figure 1.13, allows the user to create different sorts of graph and plots, selecting the variables of a previously opened or created dataset.

The same happens with the statistical calculations page. Firstly, operation to be carried on has to be selected (options available: ANOVA, Chi Square, Correlation – Pearson’s, Correlation – Spearman’s, Kruskal-Wallis H, Mann-Wihitney U, t-test – independent, t-test – paired and Wilcoxon signed ranks). Then, the test has to be configured (Figure 1.14).

What turns to be one of the most interesting features of the program, however, is the helping and learning tips that appear in the different windows. They indicate the meaning of the calculations and give examples of correct utilization.


Configurar prueba H de Kruskal Wallis

Propósito
Responde la pregunta ¿Tres o más grupos tienen promedios diferentes?
Por ejemplo, ¿es la media de ingresos la misma para gente de tres ciudades distintas?
¿O es el gasto promedio distinto entre los dueños de perros, los dueños de gatos y los dueños de ponis?

Fuente de datos
Tabla: demo_tbl ☑ Sólo lectura

Variables
Averaged: Age (age)
Agrupar por: Age (age) Mostrando primeros 20 únicos grupos
Desde Grupo 1 (1) Para 1 (1)
Comience a hacer sus selecciones

Output
☐ Also add to report C:\Users\serrahim\AppData\Roaming\SPB_16.6\sofastats\reports\default_report.htm
Export: Current Output

Waiting for an analysis to be run.

Figure 1.14. Statistical Calculation Configuration. SOFA Statistics

1.1.3.5. Others

Research into the available programs has shown a wide variety of proposals. Apart from more complete ones, such as the ones just mentioned, there are lots which have been designed for just specific purposes and functionalities. In addition, some programs are launched with commercial purpose, and only reduced versions are freely available.

Some of the most interesting programs found under these conditions are reported here.

CSPro

Census and Survey Processing System [14], developed by the US Census Bureau and ICF International. It allows the user to enter, edit, map and tabulate census and surveys data.

Tool designed with a really specific purpose, and thought for professionals and organizations in need to perform demographic and socioeconomic analysis, as well as population mapping and analytics.

With training courses with prices up to 2.500 US$, it is clear that it is not a program designed for educational purpose. Moreover, it is not an open source software and, consequently, its source code is not available online.

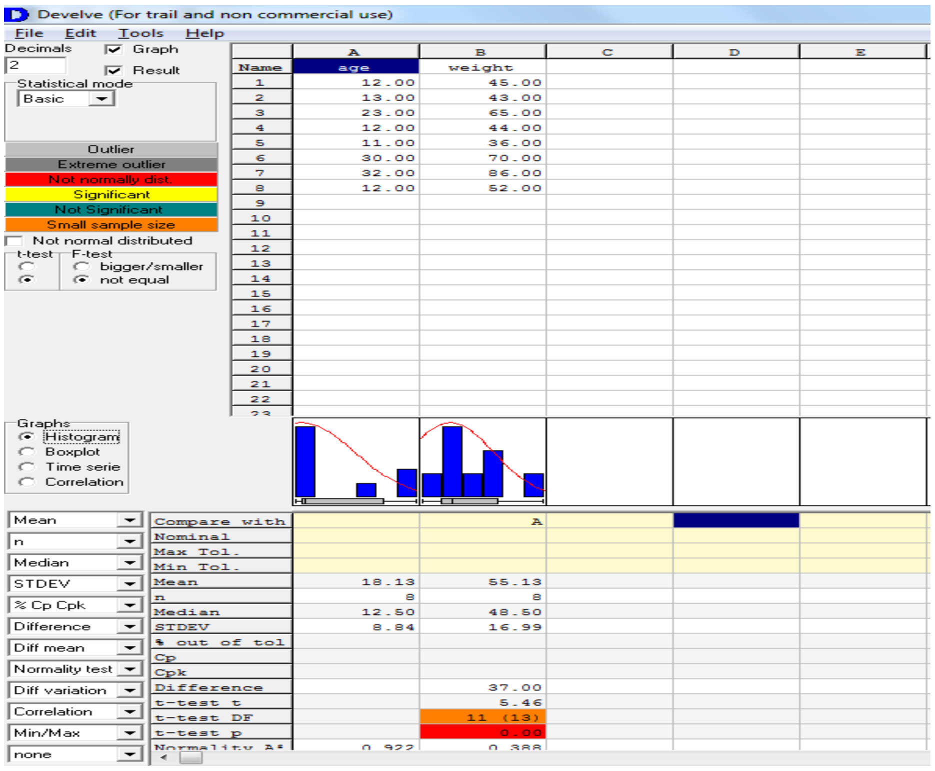
Develve [15]

Although there is only a free version available (the complete software costs 75$), it has some characteristics worth mentioning.

Again similarly to Minitab, its screen is divided in a table in which the user introduces the data and a table and graph region in the lower part in which the results are displayed.

One of its most interesting points is that, right under each data column, there is a small space in which a graph is displayed. It can be seen in Figure 1.15, where a histogram of the age and weight columns is found under each of them. In the results table, each field to be displayed can be chosen, and a color code indicates the degree of significance in test measures.

As Develve is a commercial software, its source code is not freely available. However, it is a program with a really interesting GUI that can give some ideas towards the design of our app.



Decimals: 2 [✓] Graph [✓] Result

Statistical mode: Basic

[ ] Not normal distributed

t-test / F-test: ( ) bigger/smaller [✓ not equal]

Graphs: [✓ Histogram] ( ) Boxplot ( ) Time serie ( ) Correlation

	A	B	C	D	E
Name	age	weight			
1	12.00	45.00			
2	13.00	43.00			
3	23.00	65.00			
4	12.00	44.00			
5	11.00	36.00			
6	30.00	70.00			
7	32.00	86.00			
8	12.00	52.00			
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					

Mean / n / Median / STDEV / % Cp Cpk / Difference / Diff mean / Normality test / Diff variation / Correlation / Min/Max / none

Compare with		A			
Nominal					
Max Tol.					
Min Tol.					
Mean	18.13	55.13			
n	8	8			
Median	12.50	48.50			
STDEV	8.84	16.99			
% out of tol					
Cp					
Cpk					
Difference		37.00			
t-test t		5.46			
t-test DF		11 (13)			
t-test p		0.00			
Normality A²	0.922	0.388			

Figure 1.15. Develve graphical interface

2. Users and program characteristics definition

After analyzing the already existing pieces of software, when finally deciding whether the piece of software to be designed has to have certain features, it is of high importance to perfectly define the users to whom the program is aimed. Knowing exactly which kind of user will be working with StatClip will allow to determine and decide how the program is designed.

To achieve this, a profound study of the most important users of the software is necessary.

2.1. Users

Users are, by definition, those people or organizations that can be affected by the development of the project [16].

It is clear, then, that it is necessary to take into account all the users that interact with the program at any stage of its development, testing or already consolidated periods. In consequence, then, developers, maintainers, final users, etc. will be considered in this chapter.

There are two types of user that are particularly important when developing an engineering project: main users and operator users.

2.1.1. Main Users

The main users of StatClip are those for whom the program is designed and thought.

StatClip is, and has been from the very beginning, a piece of software thought specially to enforce statistical data analysis learning. That is, StatClip is meant to be a free data analysis software carefully designed to help learning the statistical concepts while analyzing.

StatClip's principal users are, generally speaking, students.

Under this students group we can find:

- Pre-university students, facing introductory units in statistics, probability and data analytics.
- University students in introductory statistical analysis courses.
- Students (or professionals) in need to quickly perform simple analysis or plots.
- In general, anyone interested in learning how to apply statistical knowledge to data analytics at a basic level.

2.1.2. Operator Users

The needs of this kind of users are of great importance in the design of StatClip. While it is true that the software is not specifically designed for them, they will be constantly using it and in charge of its correct performance.

In this second group, the following ought to be included:

- Statistics and mathematics teachers and professors, as they are in need to learn and use the software in order to apply it for education.
- Developers and maintainers, as they will be in charge of improving StatClip, as well as solving any issues or bugs the code may have. It will be under their responsibility, as well, to take care of the user interface and access and distribution of the software.

2.2. Needs to be covered by StatClip

After defining the most important users, it is necessary to find the basic needs of each of them. Those needs will define the project, and lead the program to be as useful as possible.

In order to facilitate the understanding and prioritization of the needs, they will be classified again by the category of the users.

2.2.1. Main Users' needs

Needs follow an importance order, and in consequence there are some which are more necessary to be covered. In the case this project is facing, the following are basic features that have to be taken into account.

- Simplicity: It is of high importance that StatClip is as simple as possible. The concept "simplicity" in this project is referred to finding easy ways to analyze and present the results of the different calculations. Being StatClip a piece of software thought for data analytics learning, the way menus, inputs and outputs are organized is of great importance.
  Simplicity means keeping the user interface (UI) as clean as possible. Only the necessary information has to be displayed on the screen, and it has to be presented in a logical and perfect order.
  It is of great importance that only the necessary features are included, and any non-indispensable items should be avoided.
  The program needs to avoid any complexities and difficulties in its usage, as it has to

lead to understanding the statistical concepts beyond the user interface.

- Accessibility and Versatility: StatClip needs to be easy to access and use. It must be developed in a way that allows users to perform the analytics with no difficulties.
  To be open to every student, it is a must that users can work on StatClip on any Operating System, in any computer. The software has to be programmed to work in different environments without a problem, in order to be usable in each of the different situations students may be in.
  StatClip needs to be thought to allow quick and easy calculations and plotting. In consequence, access to StatClip must be comfortable and should not imply tough and long installation processes, nor require the user to install extra pieces of software to be able to make it work.
- Visual impact: Although it is a must that StatClip remains always as simple and "neat" as possible, the software has to allow the user to create visually beautiful and powerful graphs and plots. While the most important is to focus on the statistical concept, it is necessary to turn StatClip into an easy option to plot data in an attractive way. Furthermore, StatClip's user interface must also look up-to-date and modern, and convey the perception of being easy to use.
- Low or no Cost: StatClip, as it has been said previously, is meant to be a free option for basic data analytics. There are, in the market, several commercial options which are really powerful for data analytics and industrials statistics. Their license prices, though, and as it has been explained, make it difficult for students, schools or individuals interested in statistics to access them.
  Being free is part of StatClip's philosophy, in order to become a helpful and widespread tool.

These three just mentioned characteristics constitute the basic philosophy on which StatClip stands. The tool needs to be simple, free, accessible and versatile, because those are some of the needs that potential StatClip users have.

2.2.2. Operator Users' needs

While the software is principally designed to cover in the best possible way the needs of the main users, it is of great importance to take into account the operator users' requirements.

- Code readability and understandability: Whichever the programming language chosen to develop the software, it is a must that the code is written in the clearest and neatest way possible, as well as with all the necessary comments and explanations to make sure it is readable and easy to understand. Functions need to be well-written and the logic behind the algorithms must be perfectly explained.
  This measure will help both developers and maintainers, as the code will be neat and readable, and professors and teachers, as for them the limitations and possibilities of

the software will be clearer.

- Templates and pre-designed code files: To facilitate the task of developing new features for StatClip, templates must be designed. The existence of these templates (specially thought for items such as graphs, which may have similar structures) can make it easier to keep the same style and structure throughout all the construction and programming phase. Moreover, it will allow the programmer to focus on the mathematical and logical process instead of the UI (User Interface).
- Open-Source: StatClip is not designed to make profit, but to cover an existing need. The open-source community gives the software the possibility of improving constantly, as programmers and developers can read the code, debug it, find its problems and look for better ways to perform the calculations and processes.
  To achieve this sharing and collaborative environment, the usage of the appropriated tools is necessary. In this case, “GitHub” has been used. GitHub is now the world’s largest code host, and is designed to simplify sharing code.
  GitHub allows users to fork other users’ repositories to their own account, to be able to edit the code completely separated from the original repository. However, if this second user wants to recommend the changes to the original programmer, a Pull Request can be used. If accepted, the original repository will merge with the new modifications.

2.3. Program characteristics

Once the needs of the different considered users have been defined, it is important to list the basic features StatClip must have.

- Simple, almost minimalistic.
- Easily accessible.
- Versatile and usable.
- Focused on statistical concepts.
- Able to plot data in a visually attractive way.
- Free.
- Understandable and well-explained code.
- Based on defined templates, to make programming and maintenance easier.
- Easy to expand.
- Open-source and collaborative.

3. Design of the solution

After defining the needs of each kind of user and, from this, summarizing the most important characteristics StatClip should have, the process to design the software begins.

The first step is the theoretical design; that is, developing the philosophy and first sketches of what StatClip has to be when finally programmed. To help decide the first steps of the conceptual design, a meeting (which included a brainstorming exercise) was held. The participants in that meting were, together with me:

- Pere Grima: Professor at the Statistics and Operations Research department at UPC BarcelonaTech.
- Lourdes Rodero: Collaborating Lecturer at the Statistics and Operations Research department at UPC BarcelonaTech.
- Lluís Marco: Collaborating Lecturer at the Statistics and Operations Research department at UPC BarcelonaTech, and director of this project.

3.1. Theoretical design: StatClip's philosophy

The session included ideas from diverse aspects of the program, from its Graphical User Interface to more conceptual situations.

- Website: As StatClip is thought to be easily accessible and usable, a process including downloads and installation is always slow and non-comfortable. If the app is hosted on an Internet server, however, the situation is perfectly solved. Usable from different Operating Systems (as the only need is to have a web browser), StatClip will allow users to perform their calculations and plots online, and only save the final results or graphs.
  This measure saves time and makes it easier for the regular user to work on StatClip. However, this will only allow to work on StatClip while connected to the Internet. An added advantage is that users will always have the most updated version of the app.

Referring now to the philosophy of the GUI.

- Window organization: Three vertical regions.
  - Left region: Menu. Always visible, and containing all the different tools StatClip has.
  - Central region: Configuration Options. This second column contains all the parameters and variables the user can edit or introduce to configure the desired operation.

  - Right region: Results. The third region displays the results of the performed operation. Those results can be both numerical and text outputs, as well as graphs. In principle, there are no editable contents in this third region; all the conditions and variations must be introduced in the second region.

  This order is chosen as it is thought to be the most logical and user-friendly, resembling the usual left to right reading approach in western languages, and providing the user interface of consistent appearance and order in all tools.
  Only the particular case of the data related tabs (loading, simulation…) will not follow the just explained structure. In this case, the Menu would be still in the same position, but the rest of the screen would be divided in two horizontal spaces. The top one would contain the Configuration Options, and the second one would contain a Data Table. This configuration will be adopted to allow more wide space to the Data table.
- User Interface regions: Their sizes must be constant and consistent at all tabs. This means region widths must be equal in all tabs, in order to not impact the visual appearance of StatClip.
  If it was not possible to maintain this complete equality, it is indispensable that same category tabs (Graphs, Statistics, etc.) do keep the same measures between them.

Once decided some of the concepts related to the user interface, some more about the general idea of StatClip.

- When a user starts configuring a calculation or plot, the output keeps updating automatically with the different changes the user introduces. That is, as soon as it is possible, the results are shown and updated at every change introduced. This measure is taken to reinforce the interactivity between users and the software. Moreover, it has an educational reason. The possibility of seeing “real-time” effects on every change and modification introduced helps gain knowledge of the importance of each factor and variable, and also find which combinations are illogical or incorrect.
  The other possibility was adding an “OK” button that executed the code once every condition had been introduced. This situation (which would be in line with programs such as Minitab or SPSS) would turn StatClip into a much more static program, losing what is considered one of its most important features.

Regarding the usage of data, the following was decided:

- The menu needs to have a tab (or more) that allows the user to introduce data to StatClip. Some features (graphs, statistical calculations, etc.) cannot work if no data has been uploaded (or created).

- StatClip needs to allow users to:
    - Upload their own data. To make the process cleaner and even simpler, the only option to upload the data will be “copying and pasting” directly from the user’s file.
    - Create a simulated data set (with data randomly generated following a probability distribution).
    - Use a predefined data set, stored in the software.
- The uploaded data can be seen in a table. This table may be editable, if possible, although it is not necessary, as re-uploading is meant to be really simple and fast.
- Users can filter and sort the table.
- It was thought to add a tab to define of which kind was each variable (numeric or factor). It has been neglected, as the software will be able to choose the options depending on the operation to perform. It will be displayed in the most intuitive way possible.

3.1.1. Menu Options

During the meeting, high importance was given to decide what should and should not be included in StatClip. Keeping in mind that StatClip is a piece of software designed for introductory statistics, but still powerful, and taking into account all the ideas previously commented, the options of the menu have been decided.

The menu has been divided in four basic groups, each containing the operations the program can perform: “Data”, “Graphs”, “Computations” and “Statistics”. These for sub-menus group all the processes StatClip does.

Options in each group are explained below. Some of them are more extensively explained, the ones which have already been implemented or are about to be. Some others, though, have just been designed theoretically and, in consequence, its explanation is less detailed.

3.1.1.1. Data

- **Load Data Set**: Users can directly paste the data copied from an Excel, or similar, file. It is possible, as well, to load one of the predefined data sets.
- **Create Simulated Data**: Users can create a data table containing simulated data. The user introduces the number or variables (columns) and rows. There are several distributions (Normal, Binomial, Poisson, Exponential and Uniform) to choose from. Once the distribution is chosen, the configuration options are updated depending on the selection. For each option users must introduce:
    - Normal:
        - Mean
        - Standard Deviation

- Binomial:
  - Number of trials
  - Probability of success
- Poisson:
  - Lambda parameter
- Exponential:
  - Rate (Rate = 1 / mean)
- Uniform:
  - Distribution Lower Limit
  - Distribution Upper Limit

StatClip, following its highly reactive philosophy, has to use the data that has last been updated.

3.1.1.2. Graphics

Graphs and plots are a basic part of StatClip. Providing the possibility of plotting data in an attractive and statistically relevant way has to be one of the main strengths of the designed software. Graphics can reveal statistical information in a visual way, although, *"just as statistical calculation, they are only as good as what goes into them"* [17][14].

Each graph has a set of options that users must select to get the desired plot. This options are divided in two groups, depending on their importance in the final result. This two groups are: Main Options, and Appearance Options.

The following are the types of graph that will be included in StatClip (a list that may be updated in future versions).

- **Histogram**: A histogram is a plot that shows the distribution of the numbers along some scale [18]. It can be plotted in a frequency (or count) scale, or in a density scale. This fact becomes especially important when working with histograms that have different bin widths (bar width).
  For histograms, the available choices for users are:
  - Main Options:
    - Variable to plot
    - Number of intervals
    - Stratification variable
  - Appearance Options:
    - Select if user wants Density on Y axis

- Color for the exterior lines of the bars
- Color for the bars' fill
- Opacity degree
- Plot title
- Plot title font size
- X and Y axis' titles
- Axis' titles font size

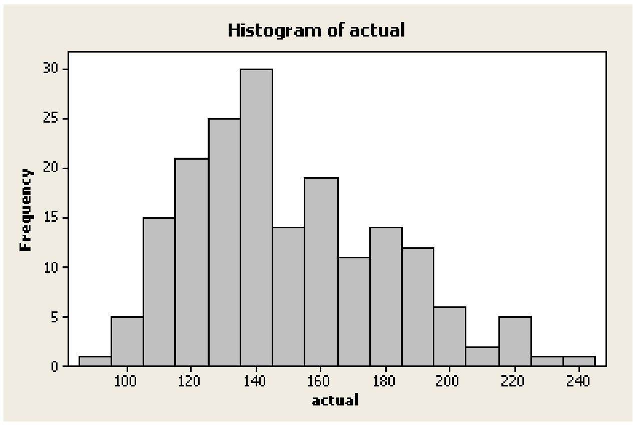


Figure 3.1. Histogram

- **Time Series Plot**: After selecting the time variable (X), user has to choose the Y variable to plot. If wanted, more than one variable can be plotted in the same graphic. Time Series Plots are used to study the evolution over time of a certain variable, and evaluate patterns and behavior of the data.
  Details such as the color, line thickness, graphic title, etc. will be editable by the user.

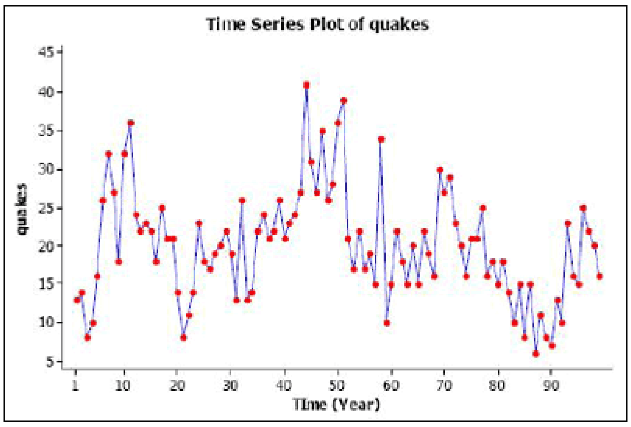


Figure 3.2. Time Series Plot

- **Dotplot**: In the same way than in the histogram, users are allowed to choose the variable to plot, the number of intervals and, optionally, a grouping variable**.**

A dotplot is a kind of graph that can be used to assess the distribution of continuous data. Especially useful when working with relatively small datasets, dotplots are constructed dividing the X axis in bins and drawing the values in its corresponding bin [19].

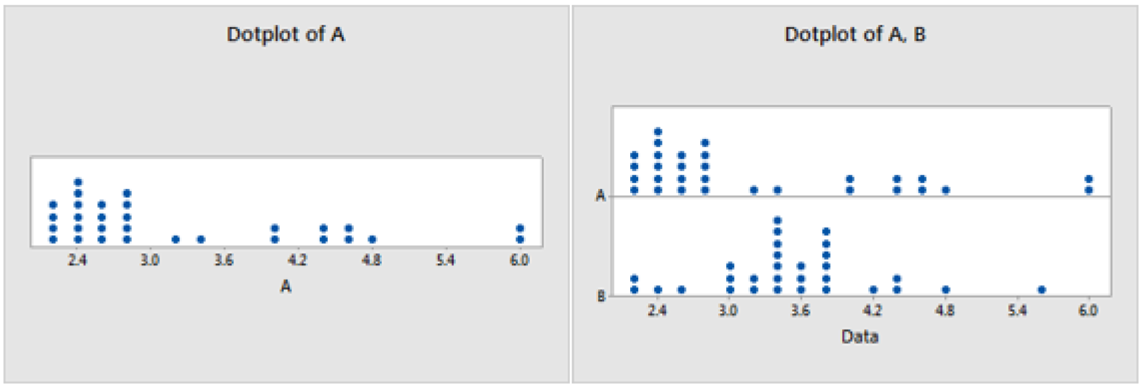


Figure 3.3. Simple dotplot (left). Dotplot with groups (right)

- **Pie Chart**: A Pie Chart is a circular statistical graphic that is divided into slices to illustrate numerical proportion. In a pie chart, the area of the slice (and thus the arc length and central angle) is proportional to the quantity it represents. Users will have to select the variable to plot. It will be possible to select, as well, a grouping variable.
- **Bar Chart**: A bar chart is a graphic that represents grouped data with rectangular bars that have length proportional to the quantity they represent. They are widely used to show comparison amongst categories. One axis of the graph shows the categories being compared, and the other axis represents a discrete value.
  For example, we could be comparing the amount of rain in two consecutive years. The discrete value axis would be representing the months. The other axis would represent the amount of rain. And in each month we would have two bars, one representing each year**.**
  Users will have to introduce the variables to analyze and, if necessary, a grouping variable.
- **Scatterplot**: Scatterplots are used to visualize the relationship between two variables. Building a scatterplot consists in plotting one variable against each other, resulting in a cloud of points that can be used to see if there is a relationship between those two variables.
  StatClip users can select:
  - Main Options:
    - X variable

    - Y variable
    - Stratification variable (optional)
  - Appearance Options:
    - Jitter: If this option is selected, the points are slightly moved from its real position so that the user is able to see the amount of points that are in a region of the plot. Especially useful when we obtain superposed points.
    - Points' size
    - Opacity degree of the points
    - Points' color
    - Plot title
    - Plot title font size
    - X and Y axis' titles
    - Axis' titles font size

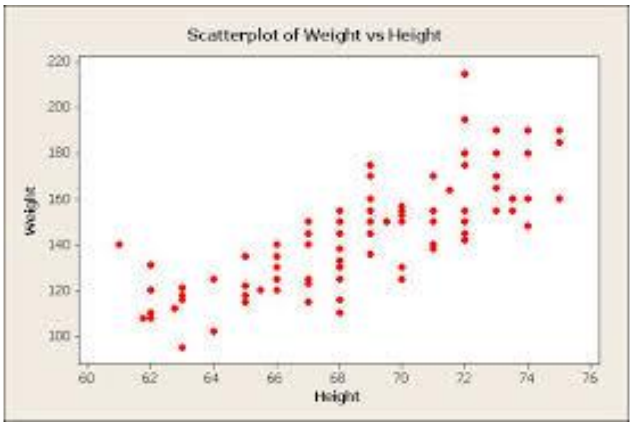


Figure 3.4. Scatterplot

- **Matrix Plot:** A Matrix Plot is nothing more than the comparison of several variables of a data set. This variables, disposed in the way of a matrix (variable 1 is first row and first column) the comparison is carried on with a Scatterplot graphic. This allows to easily compare each variable with all the others at a time, making it easier to find, for example, the necessary variable transformations in a regression.

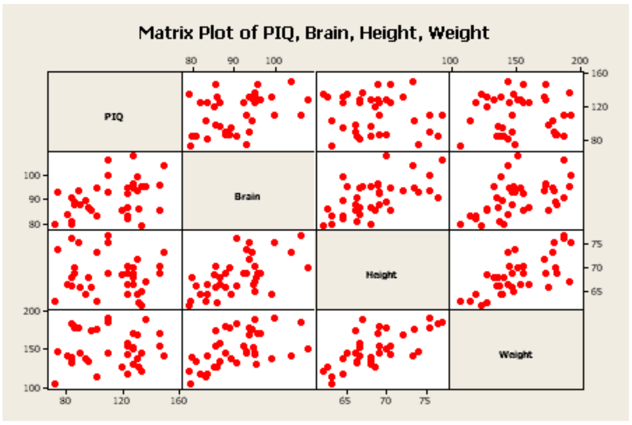


Figure 3.5. Matrix Plot

- **Boxplot**: Boxplots are quite complex plots that summarize the distribution of a given data set. They consist of three different parts:
  - A box, covering 50% of the data. The limits of the box are the first and third quartiles. A line is drawn at the median value.
  - Whiskers that extend from the limit of the box to indicate how far the data goes on each side of the box. Whiskers, however, must never be longer than 1.5 times the Inter Quartile Range (IQR). Points further than 1.5 times the IQR are considered outliers and drawn individually.

  In StatClip, users will choose the variable (or variables) for which to plot the Boxplot, and also an optional stratification variable.

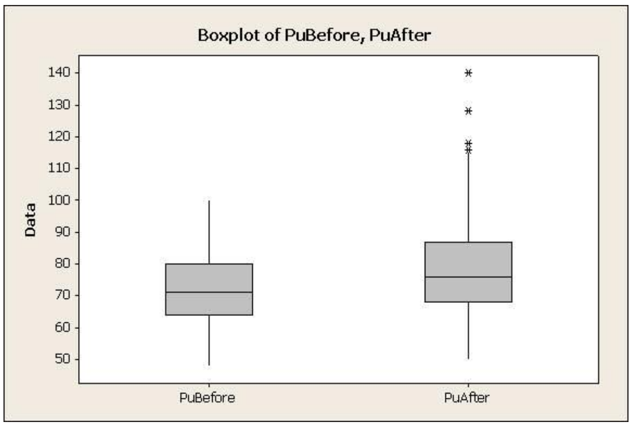


Figure 3.6.Boxplot

- **Bubble Plot**: Being very similar to scatterplots, users can select mainly the same options. The difference is the selection of the size variable. This variable defines the size of the bubbles, which are plotted in Cartesian axis, with the chosen X and Y variables.

Bubble plots are an adaptation of Scatterplots that give extra information about another continuous variable of the data set.

- Main Options:
    - X variable
    - Y variable
    - Size variable
    - Stratification variable (optional)
- Appearance Options:
    - Jitter: If this option is selected, the points are slightly moved from its real position so that the user is able to see the amount of points that are in a region of the plot. Especially useful when we obtain superposed points.
    - Bubbles size range (size of the smallest and size of the biggest one)
    - Opacity degree of the points
    - Points' color
    - Plot title
    - Plot title font size
    - X and Y axis' titles
    - Axis' titles font size

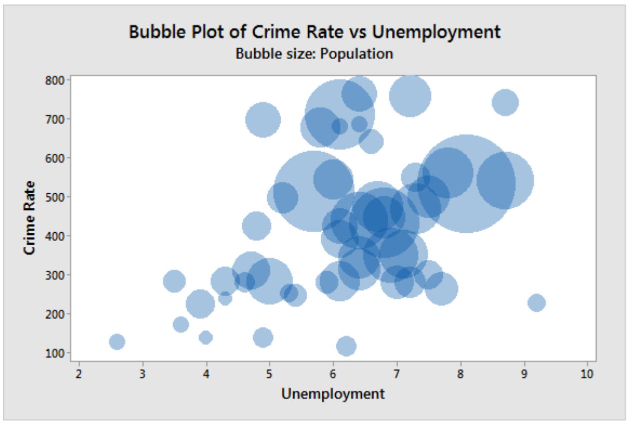


Figure 3.7. Bubble Plot

- **Multi-vari Chart:** A multi-vari chart is a graphical representation of the relationship between factors and a response. They can help carrying out an investigation and study patterns of variation from many possible causes on a single chart. They allow to display positional or cyclical variations in processes. They can also be used to study variations within a subgroup, between subgroups, etc.
- **Maps**: This last graphic can be more complex than the other suggested above. User will have to provide data in a specific way in order to be able to use the Maps tab in

StatClip. There must be two columns indicating Latitude and Longitude. StatClip will plot the points on a map, and alter its appearance depending on the chosen variables. As an example, color stratification by a certain factor could be a possibility. In the end, StatClip will be able to locate the given points on a map and display them in different ways, depending on the factors introduced by the user.

For maps in StatClip, users have the following options:

- Main Options:
  - Longitude
  - Latitude
  - Stratification Variable (optional)
- Appearance Options
  - Points' size
  - Opacity degree of the points
  - Points' color (if the map is not stratified)
  - Map title
  - Map title font size

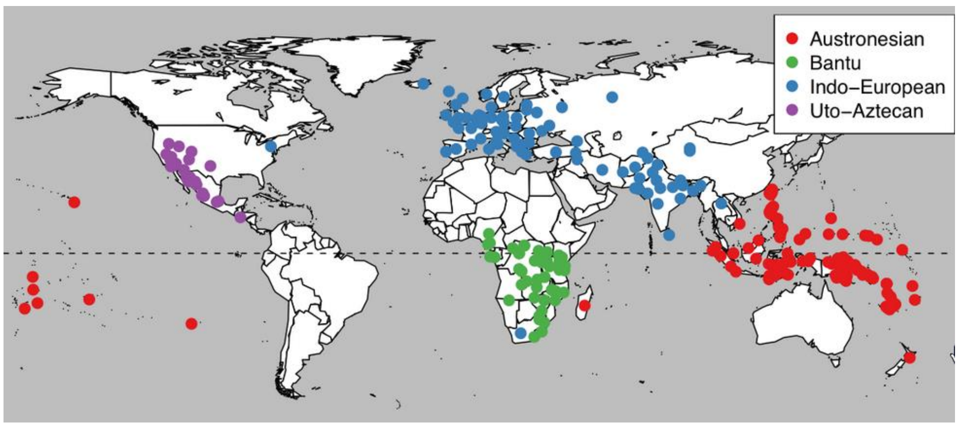


Figure 3.8. Map plot

3.1.1.3. Computations

- **Basic Operations:** This tab is thought to be used as a variable calculator. That is, allowing users to perform arithmetical operations with the variables in the working data set. Users will have to choose the variable (or variables) in which to perform the selected calculation.
- **Probabilities:** This tab has special characteristics when compared to most of the

others: no data needs to be uploaded for users to work with the Probabilities page.
It is a probability calculator; that is, allows users to calculate probabilities on various distributions. The following probability distributions are considered (with the parameters users must introduce):

- Normal:
  - Mean
  - Standard Deviation
- Binomial:
  - Number of trials
  - Success probability
- Chi-squared:
  - Degrees of freedom
- Student's t:
  - Degrees of freedom
- Snedecor's F:
  - Degrees of freedom (1)
  - Degrees of freedom (2)

Users can then select whether they want to calculate different probabilities; for example, P(X>a) or P(b<X<c). An explanatory plot is as well displayed.

- **Correlation:** used to compute the Pearson's correlation coefficient between two variables. To facilitate interpretation, a scatterplot of both variables is also drawn, together with the formal hypothesis testing to obtain the p-value.
- **Descriptive Statistics:** This tab displays the most basic statistical calculations of the selected variables of the data set. The numbers calculated help users get an idea of the variables analyzed. The predefined calculated descriptive statistics are:
  - Sample size
  - Mean
  - Median
  - Standard Deviation
  - Minimum
  - Maximum
  - Range
  - Standard Error

  Users have to select:
  - Variable (or variables, as multiple selection is allowed) to analyze.
  - Grouping variable (optional).
- **Goodness of fit**: after selecting a variable as input and the suspected probability distribution, the probability distribution plot is drawn, together with the p-value for checking the adequate fit to the distribution.

3.1.1.4. Statistics

This section includes classical hypothesis testing for one mean, two means, analysis of variance, and proportions. Nonparametric tests are also included.

- **With means/medians**: comparison of one mean with a value and two means (independent samples, 2-sample-t, and in pairs, paired-t). 1-sample median test (sign test and Wilcoxon test) and 2-sample median test (Mann-Whitney test) are included as nonparametric tests.
- **With variances**: F test and Levene test for comparison of variances, classical analysis of variances with one and two factors. Kurskal-Wallis test is also included as nonparametric procedure.
- **With proportions**: hypothesis testing for 1 and 2 proportions, based on raw data and summarized data.
- **Power and Sample Size**: developed for one and two means, this tab allows the calculation of sample sizes based on the desired difference to be detected, the power value and the significance level. A graphical representation of the result, that changes interactively, improves understandability of the procedure.
- **Regression**: based on the selection of one continuous variable as Y and a set of variables as regressors (Xs), a linear regression is performed. Graphs such as the residuals versus fitted values and normal probability plot of residuals are shown by default.

3.2. Theoretical Design: UI’s first approach

Once defined the contents of the menu, as well as some of the properties and ideas StatClip has to accomplish, the conceptual design continues towards the first approach to the User Interface.

This chapter includes the schematic design of different tabs, and the evolution of this design in the different stages of the project. To help this schematic drawing process, the software *Balsamiq Mockups 3* [20] has been used.

Balsamiq Mockups is a software designed for wireframing. A web wireframe (also known as page schematic or screen blueprint) is a visual guide that represents the framework of a website [21]. Being StatClip designed to be hosted on a website, the usage of the just mentioned software is appropriate to help in its design.

3.2.1. General frame

It was decided that the menu would be on the left side, and generally visible to facilitate the transfer from one tab to the other. The rest of StatClip's contents would be contained in the actual main panel.

The general frame designed for StatClip is thus the following (in a web browser).

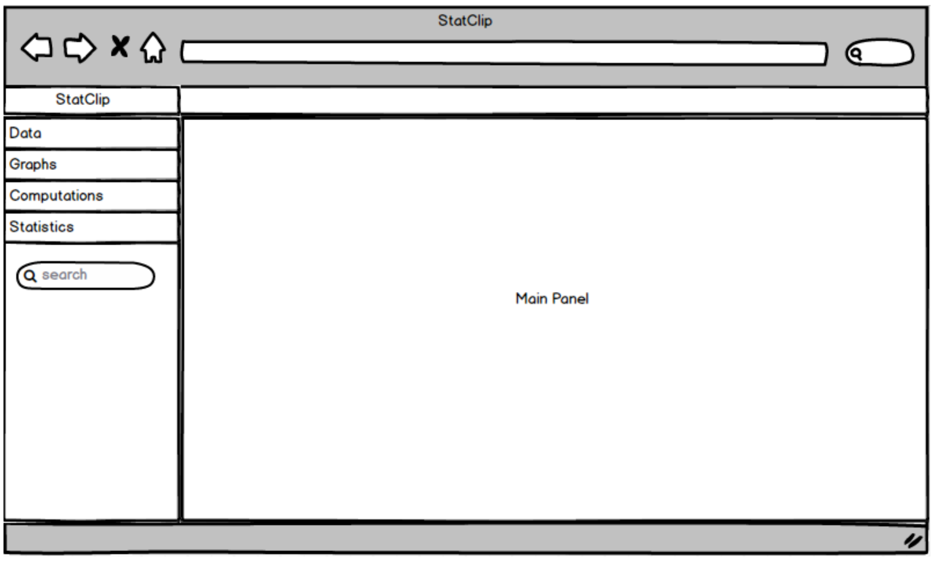


Figure 3.9. General frame

As it can be seen, the main idea behind the design is to keep it as simple as possible, leaving a big space for the user's interaction and the presentation of plots and numerical results. The menu is collapsed, as it has many options and its fully opened view would not be comfortable.

Under the menu there is a search box. As there are many different tabs, users have the possibility to write the name of the page they are looking for. Once selected, the program automatically opens the page.

The main panel is divided, in general, in two vertical spaces. The first one contains the configuration options (introduced by the user) and the other is to present the operation's results.

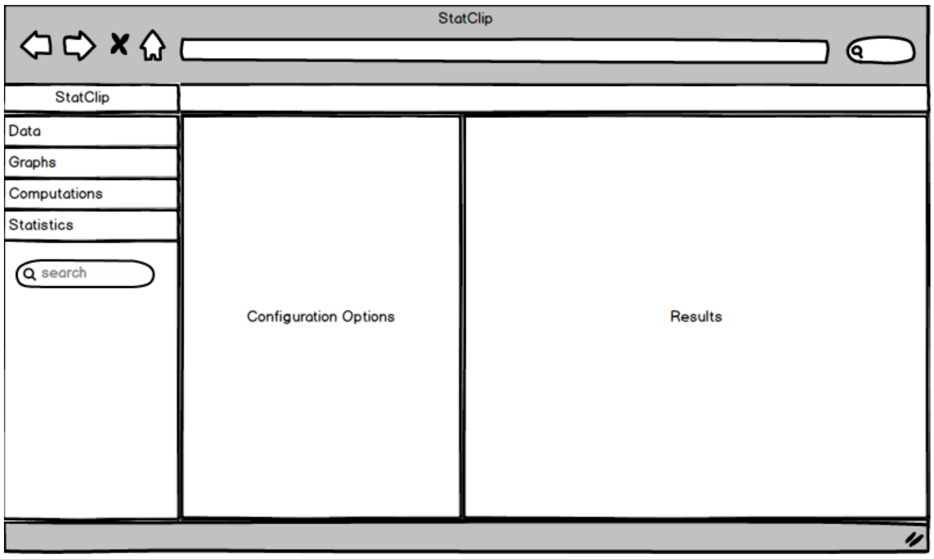


Figure 3.10. Main Panel division

3.2.2. Data

One of the discussions in the meeting held with Pere Grima, Lourdes Rodero and Lluís Marco was the distribution of the elements in the data pages. While it seemed fairly logical to have two vertical spaces in the operational tabs, the answer was not that clear in this case. The final decision, as it has been already explained, was to change the general appearance of StatClip in these two pages, to allow a wider space for the data table.

Then, the just referred Main Panel division is no longer valid for the Data pages.

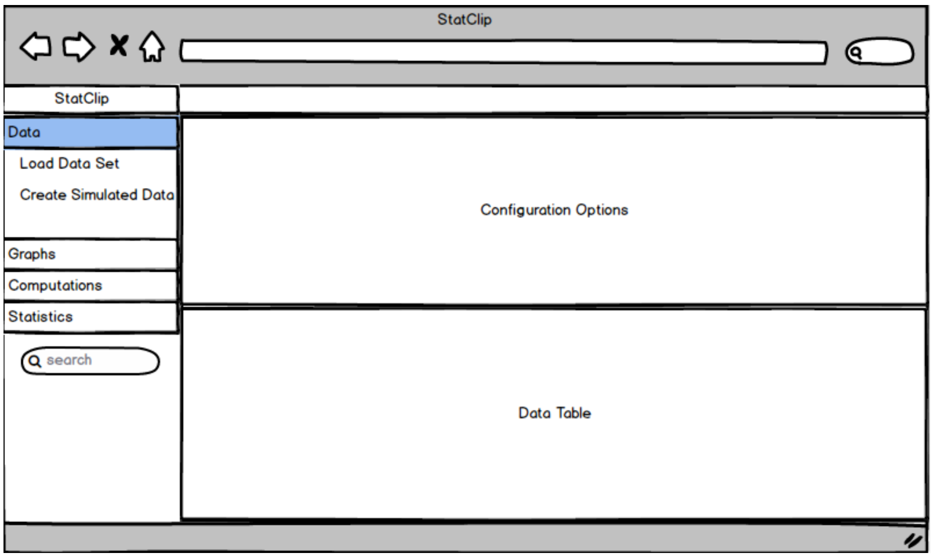


Figure 3.11. Data pages structure

3.2.2.1. Load Data Set

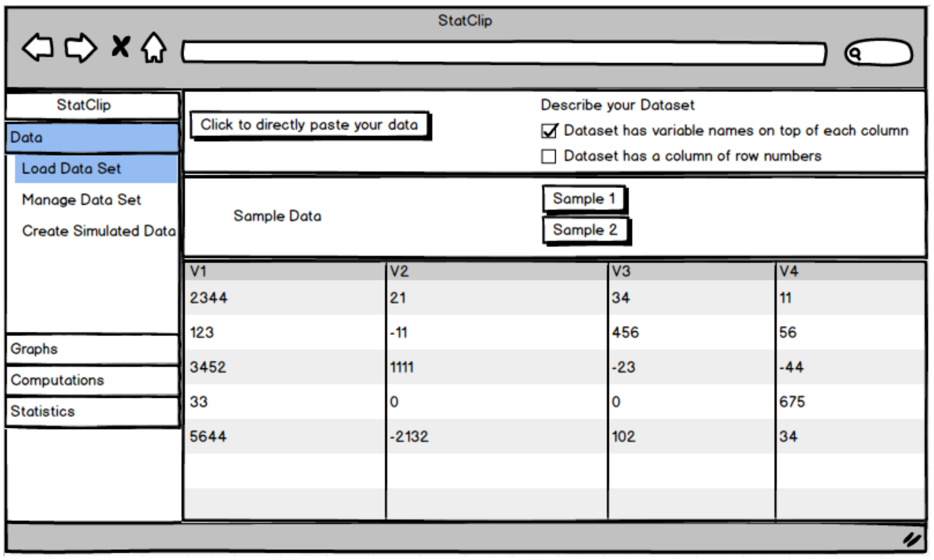
The first of the two data uploading pages contains two possibilities: uploading user's own data or loading StatClip sample data set.

As it has been explained, and due to the aim to keep StatClip as simple and fast as possible, it was decided to only let the user upload data by copying and pasting. This decision helps keep the page clean, and allows fast modification and actualization. Users only need to copy the data from, for example, an Excel file and paste it directly on to StatClip. StatClip must detect each time the user presses the paste button and get the data from the clipboard, in order to be as easy to use as possible.

Being StatClip a piece of software thought especially for students and statistics learners, we felt that it was necessary to provide some prepared datasets. This allows the different users to be able to try, play and learn with the different possibilities StatClip has. As it has been said throughout the whole document, this app has a powerful educational side, and this option facilitates in a huge way the task. Not only StatClip allows you to use your own data but it lets you use data already prepared to maximize your learning process.

It was not evident, though, how to organize the "Load Data Set" page. Knowing that the lower half was destined to see the data table, there were several options to organize the configuration buttons. The most logical ones, however, were the following:

- Two boxes (one below the other one) occupying the whole width of the main panel. Being the top one the data pasting box, and the second one the predefined data set charging one.

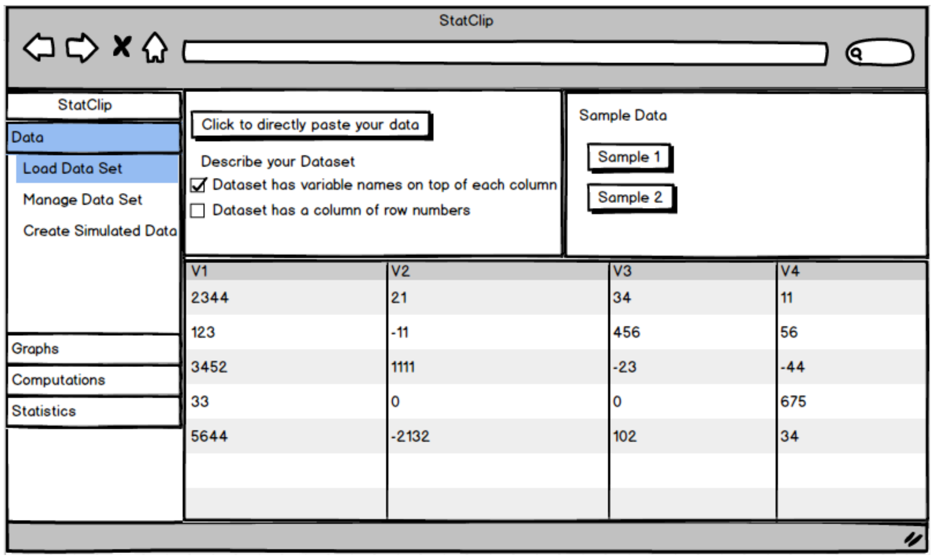


☑ Dataset has variable names on top of each column
☐ Dataset has a column of row numbers

V1	V2	V3	V4
2344	21	34	11
123	-11	456	56
3452	1111	-23	-44
33	0	0	675
5644	-2132	102	34

Figure 3.12. Load Data Set (option 1)

- The second option considered was dividing the configuration options area in two vertical boxes, as can be seen in Figure 3.13



☑ Dataset has variable names on top of each column
☐ Dataset has a column of row numbers

V1	V2	V3	V4
2344	21	34	11
123	-11	456	56
3452	1111	-23	-44
33	0	0	675
5644	-2132	102	34

Figure 3.13. Load Data Set (option 2)

It was decided to continue with the second option, for it seemed more logical, and gave the page a more balanced look. It allows to separate more the two different options, giving them a more ordered look. The data pasting option includes checkboxes to define the data characteristics. These options are:

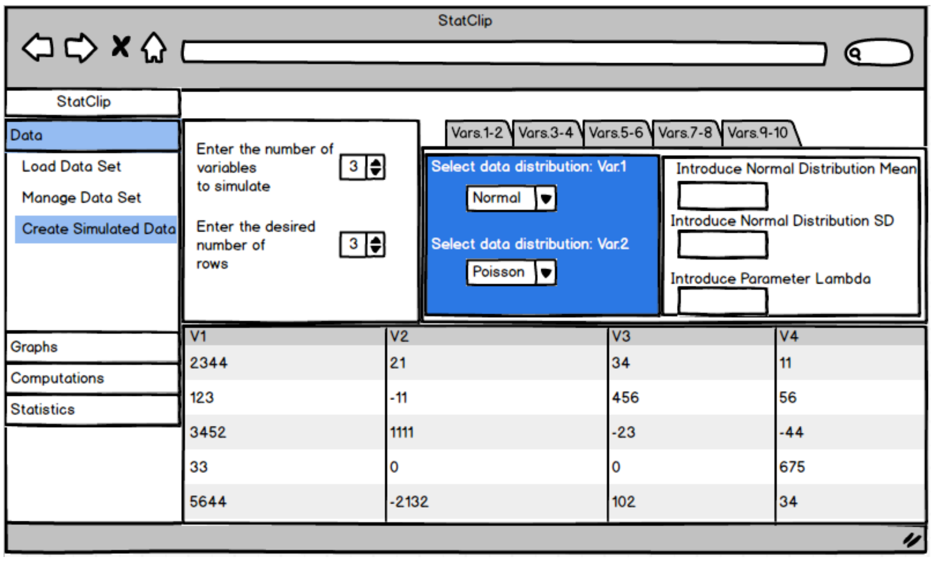
- Pasted data includes Variable Names?
- Pasted data includes a rows' names' (or numbers') column?

Those options are important, as the software could mistakenly take as data words that are not, and consequently invalidate all analytics results. In the same way, it could lose data if it took, as an example, the first row as variable names instead of considering it in the data set.

3.2.2.2. Create Simulated Data

For some calculations or comparisons it might be interesting to have the possibility to simulate data. At least, if not, it sure has an educational importance. Viewing and playing with simulated data (that is, data created simulating random numbers that follow a given probability distribution) helps understand the concept of probability distributions.

The original idea was to allow users to create a data table with a limited number of columns, in which each of them could follow a different distribution. Its structure was designed as shown in Figure 3.14.



Enter the number of variables to simulate 3

Enter the desired number of rows 3

Select data distribution: Var.1 Normal

Select data distribution: Var.2 Poisson

V1	V2	V3	V4
2344	21	34	11
123	-11	456	56
3452	1111	-23	-44
33	0	0	675
5644	-2132	102	34

Figure 3.14. Create Simulated Data first design

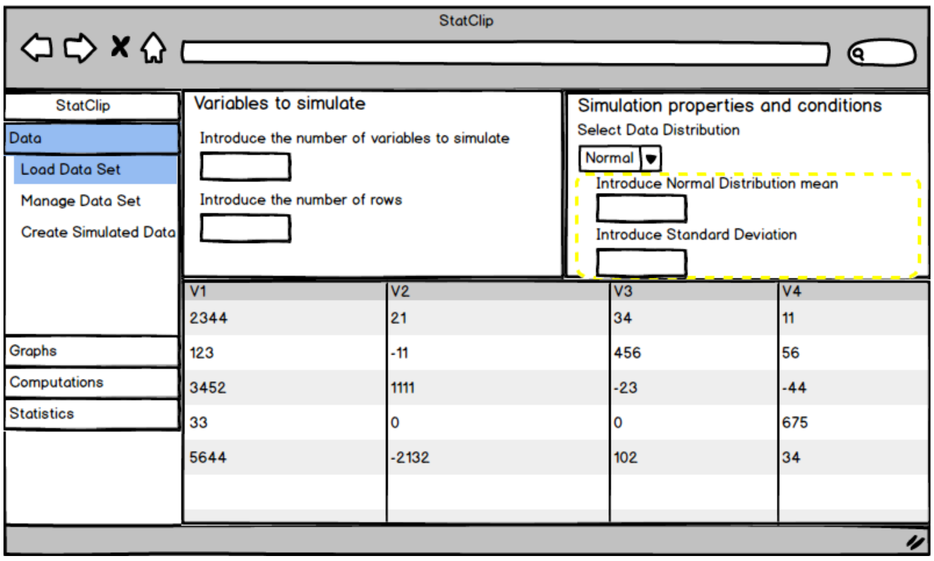
However, and as it will be explained in the Implementation chapter, technical difficulties in programming the reactivity on this item lead us to rethink the way to program it. Computing several different variables, all having different parameters might not be necessary in most cases, and adds difficulty and complications to StatClip. Moreover, the UI becomes less neat and clear.

Taking this into consideration, the design was redefined. Users can create a data table with as many columns and rows as desired, all the data following the same probability distribution.

This distributions, as it has been already said, are:

- Normal
- Binomial
- Poisson
- Exponential
- Uniform

And the conceptual design can be seen in Figure 3.15. Surrounded in yellow, the conditional part can be seen. Depending on which probability distribution has been chosen, different options appear. Figure 3.16 and Figure 3.17 show the conceptual design of the other distributions' inputs.


StatClip
StatClip
Data
Load Data Set
Manage Data Set
Create Simulated Data
Graphs
Computations
Statistics
Variables to simulate
Introduce the number of variables to simulate
Introduce the number of rows
Simulation properties and conditions
Select Data Distribution
Normal
Introduce Normal Distribution mean
Introduce Standard Deviation

V1	V2	V3	V4
2344	21	34	11
123	-11	456	56
3452	1111	-23	-44
33	0	0	675
5644	-2132	102	34

Figure 3.15. Create Simulated Data design.

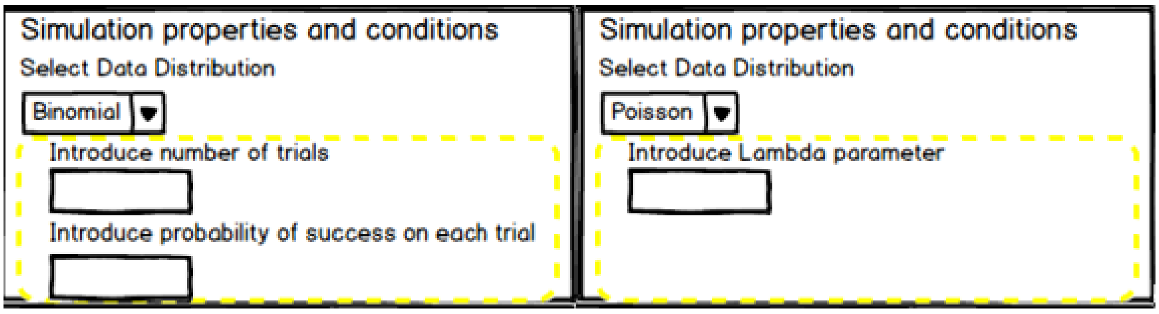

Simulation properties and conditions
Select Data Distribution
Binomial
Introduce number of trials
Introduce probability of success on each trial
Simulation properties and conditions
Select Data Distribution
Poisson
Introduce Lambda parameter

Figure 3.16. Binomial and Poisson simulation properties

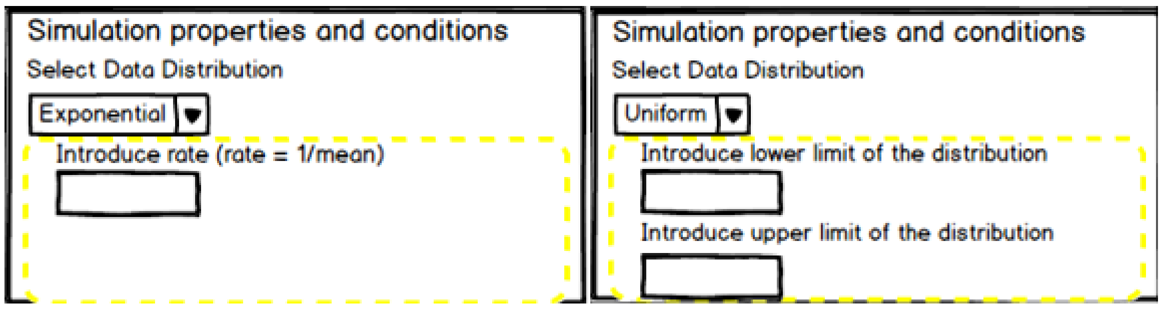

Simulation properties and conditions
Select Data Distribution
Exponential
Introduce rate (rate = 1/mean)
Simulation properties and conditions
Select Data Distribution
Uniform
Introduce lower limit of the distribution
Introduce upper limit of the distribution

Figure 3.17. Exponential and Uniform simulation properties

3.2.3. Graphs

All the graphics in StatClip will follow the general UI structure. The structure, discussed on the brainstorming meeting, separates the main panel in two vertical regions: configuration and results (in the graphics case, the actual plot).

Referring to the configuration area, there are several ways to present the options to select. As it has been said, options to plot the desired graphic are divided in “Main Options” and “Appearance Options”. Main Options are enough to provide the correct graphic (correct variables, stratification, etc.), and the selection of defaults is good enough to produce a good graph. Appearance Options are secondary, and offer the possibility to fine tune the plot to obtain the desired combination of colors and font size.

It was decided that basic and appearance options should be separated, and that this separation ought to be clearly visible. Appearance options are not necessary to produce a good graph, they appear collapsed by default – clicking on the plus sign shows the options (Figure 3.18).

The following figure shows the conceptual design of the graphics page.

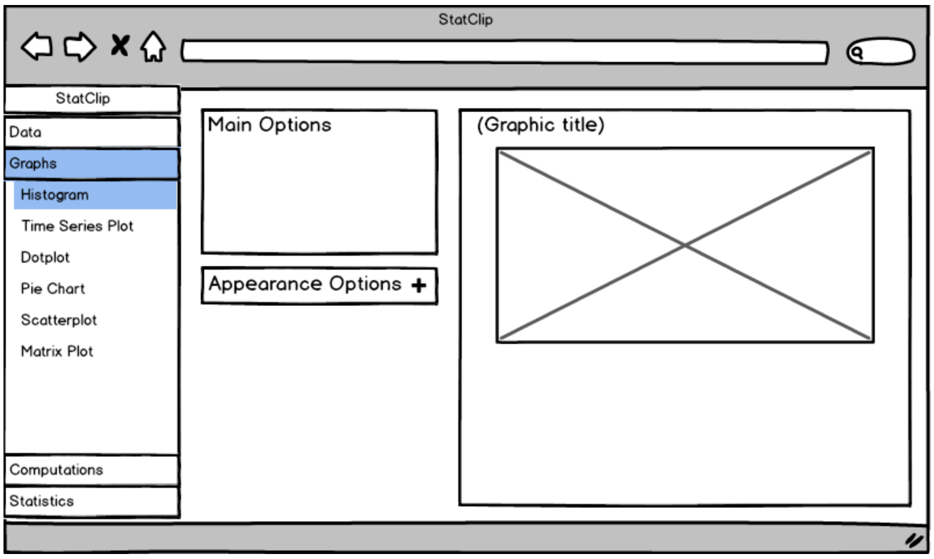


Figure 3.18. Graphs page design (Appearance Options collapsed)

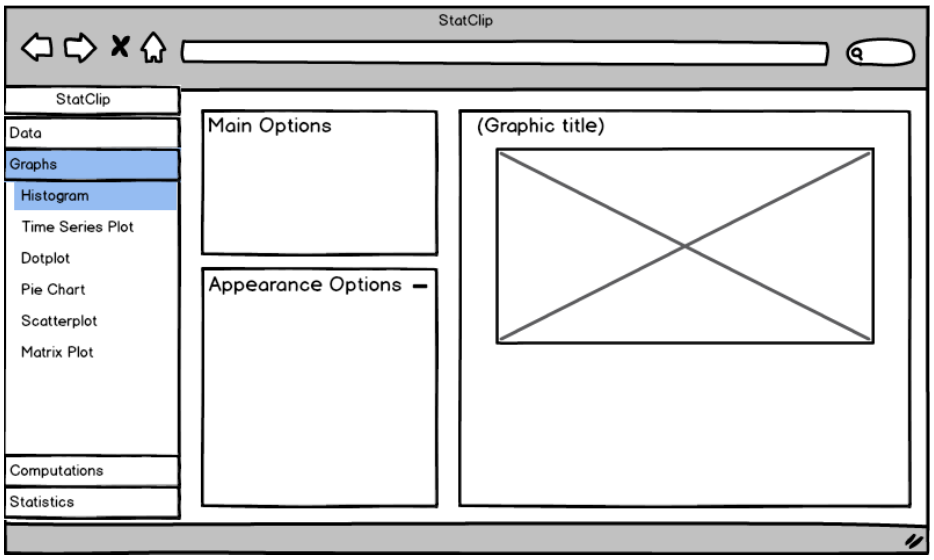


Figure 3.19.Graphs page design (Appearance Options expanded)

3.2.4. Computations and Statistics

The Computations and Statistics' tabs share some of its features. As it has been said, it was decided to locate the operations options on the left side, and the results on the fight. In this case, the exact same rule will be followed.

The only difference to consider is that, in this case, results may not only be graphical or numerical, but can be both at the same time. For example, a regression model that shows the residuals plot and the numerical results. The general structure to follow in these pages is the following ().

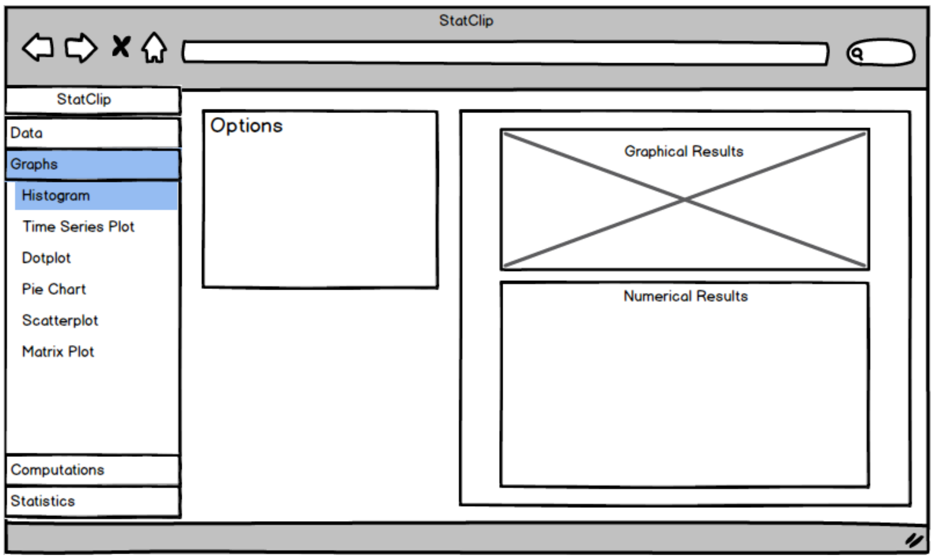


Figure 3.20. Computations and Statistics pages general design

3.3. Programming language election: R

In chapter 1 (“Statistical Software Resources”), different programming languages were explained and considered. As it was said in the introduction as well, R is the most used language for statistical programming, and therefore it was a logical option for StatClip. All needed statistical methods are already available in R, R can be freely used without any costs, and it can be integrated in other programming environments.

However, in the Industrial Engineering degree as taught at UPC, students learn to program using Python, another language hugely used to analyze data.

The decision on which language to use in programming StatClip, then, was between Python and R because, even though there are many other languages that could have been appropriate, these two were the most reasonable choices. Even before the final choice was made, I started learning R programming on my own, as it is nowadays highly important in the statistical world. Besides, I already had a knowledge of Python, so having a good understanding of the possibilities of both R and Python placed me in a good position to make a decision.

Table 3.1. R and Python comparison

	R		Python
−	I need to learn R programming, as I have no experience with the language	|	I already know Python programming, but probably not enough for implementing an application as StatClip.
+	All statistical tools needed are already available, either in the core of R or in contributed packages.	−	The core of Python is very poor in statistical tools. Perhaps some methods are available installing libraries, but it is not sure.
+	Using shiny, it is possible to create attractive user interfaces with relative ease (once knowing R and shiny programming)	−	As a general purpose language, creating an attractive user interface is possible, but probably much more difficult than using R and shiny.
+	Can be distributed for free and made available on a website	+	Can be distributed for free and made available on a website

The discovery of Shiny, an R package for app programming (described in 3.3.2), opened a great range of possibilities for user interface creation with R in the background. **Error! eference source not found.** summarizes the list of advantages (marked with a plus sign), disadvantages (marked with a minus sign) and neutral characteristics (marked with a vertical line). Based on this table, a decision was taken to use R as programming language for the application.

3.3.1. What is R?

R is a language and environment for statistical computing and graphics [22]. R is a GNU Project, very similar to S. R can run on different UNIX platforms, and as well on Windows and Mac operating systems.

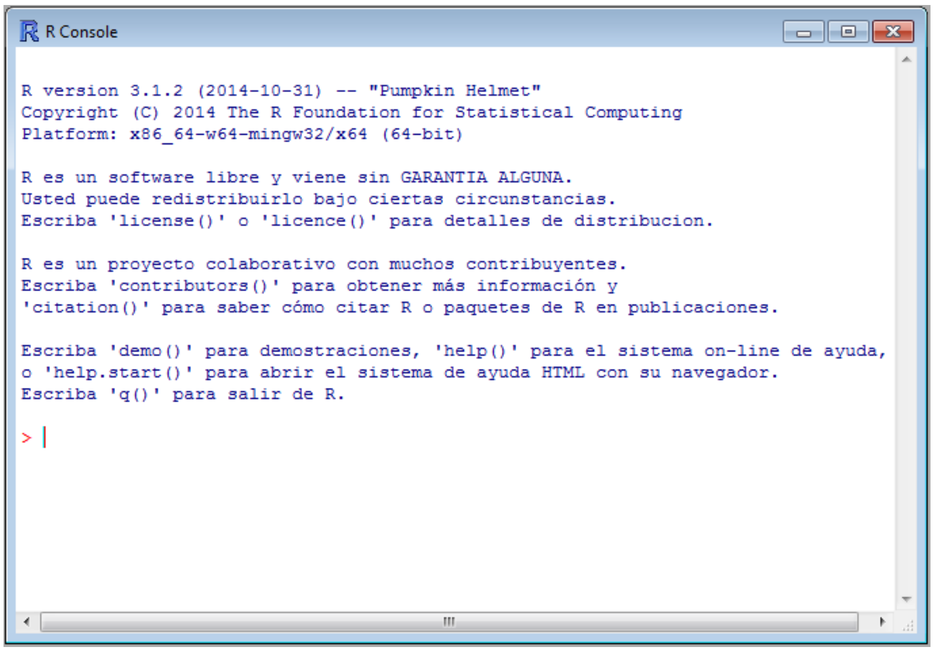
R provides a huge amount of statistical and graphical techniques, and is highly extensive. As explained by Venables, Smith and the R Core Team [23], R has:

- An effective data handling and storage facility.
- A suite of operators for calculations on arrays, in particular matrices.

- A large, coherent, integrated collection of intermediate tools for data analysis.
- Graphical facilities for data analysis and display either directly at the computer or on hardcopy.
- A well-developed, simple and effective programming language (called 'S') which includes conditionals, loops, user defined recursive functions and input and output facilities. (Indeed most of the system supplied functions are themselves written in the S language.)

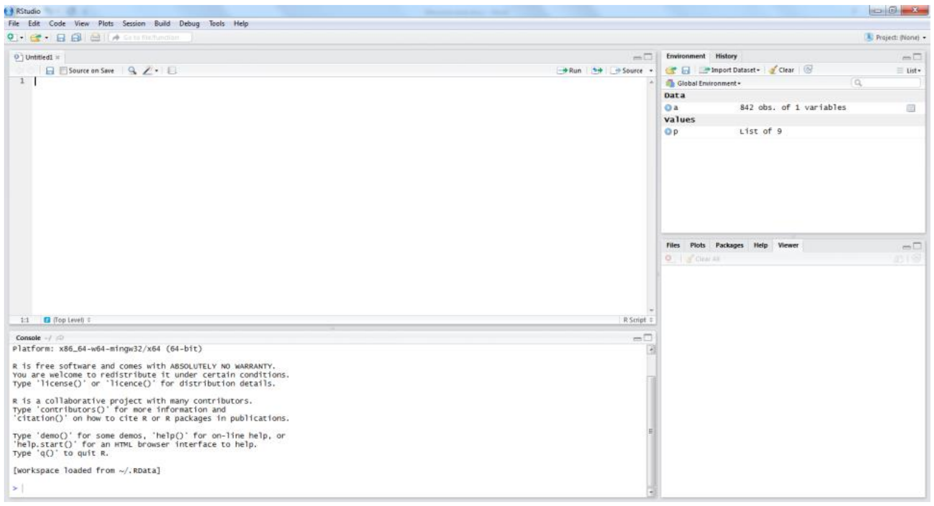
As stated in [23] *"R is very much a vehicle for newly developing methods of interactive data analysis. It has developed rapidly, and has been extended by a large collection of packages. However, most programs written in R are essentially ephemeral, written for a single piece of data analysis".*

As the default R Console for Windows (Figure 3.21) is not the most comfortable for working, to carry on the necessary programming work for StatClip RStudio has been used. RStudio is an IDE (Integrated Development Environment) for R [24]. It includes a console, a syntax-highlighting editor that supports direct code execution and tools for debugging, plotting and workspace management (see Figure 3.22).



```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

> 
```

Figure 3.21. R Console



```
Data
a                842 obs. of 1 variables
Values
p                List of 9
```

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/.RData]

>
```

Figure 3.22. RStudio

There are hundreds of online and offline resources for learning R (from books to forums or online courses), from beginner level to the most advanced techniques. Although explaining how to program in R is obviously not the objective of this project, a really brief and basic introduction on the language will be provided. Most of the examples and explanations in the following introduction to R have been extracted from the first chapter (*First Steps*) of the book *Introductory Statistics with R*, by Peter Dalgaard [25].

R's simplest possible task is to use it as a powerful calculator: from the simplest arithmetic operations to more complex calculations. As an example, we can generate 15 random values from a Normal distribution:

```
> rnorm(15)
 [1] -0.61561745 -0.22352727 -0.75485588 -1.26161939  0.86014655 -0.66729345
 [7] -0.81198964  0.15467160  0.23811019 -0.04669919  0.13783162  1.04511130
[13] -0.60252741  0.18176788 -0.81399071
```

Most times, though, it is not enough to perform the operation on the command line and see the results directly on the screen. Intermediate calculation steps have to be necessarily stored in order to perform slightly more complex operations. In R, an assignment can be:

```
> a <- 5
```

Consequently, we can perform operations like:

```
> a^2
[1] 25
```

It is important to note that R is a case-sensitive language. That is, A and a do not refer to the same variable.

One of R's advantages is that it handles vectors as single objects. Vectors can be defined using the construct c(…). If you operate with vectors of the same length, there is no problem in performing arithmetical operations just as if they were ordinary numbers. As an example, we can calculate the Body Mass Index of a group of people from vectors with their weight and height.

```
> weight <- c(60, 72, 57, 90, 95, 72)
> weight
[1] 60 72 57 90 95 72
> height <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
> height
[1] 1.75 1.80 1.65 1.90 1.74 1.91
> bmi <- weight/height^2
> bmi
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

R, as a statistical software, has many statistical calculations already built into the program. For example, to calculate the mean and standard deviation of a data vector functions mean and sd can be used:

```
> mean(weight)
[1] 74.33333
> sd(weight)
[1] 15.42293
```

R has its default way to plot data and to perform lots of different calculations. However, another one of R's strengths is the amount of packages available to install that provide methods to enlarge R's default capabilities. These packages can contain functions written in R language, libraries of compiled code (written in C or Fortran, for example) or data sets. Examples of useful packages are ggplot2 (for graphics and plots), dplyr (data filtering, sub setting, rearranging, etc.), shiny (creating web apps), and a long etcetera.

The brief and simple examples shown to the point are to use R as a calculator. However, most times we will need to write our own functions to be able to perform easily an operation multiple times, or to call it from an R script, for example. Here we create a simple function to calculate the Body Mass Index, from two given height and weight vectors.

```
#Function to calculate the BMI

body_mass_index <- function(height, weight){

  bmi <- weight / height^2

  return(bmi)
}
```

All these examples are just to illustrate, at a really beginner level, how R works. They try to be, as well, a first approach to get used to the way code is written in R, as in the following chapters the main code of StatClip will be explained.

3.3.2. Shiny

Shiny is the package that definitively turned the tables to R when deciding which language to use.

Shiny is developed by RStudio. According to its developers [26], Shiny is a package that *“makes it incredibly easy to build interactive web applications with R. Automatic "reactive" binding between inputs and outputs and extensive pre-built widgets make it possible to build beautiful, responsive, and powerful applications with minimal effort”*.

With no HTML, JavaScript or CSS knowledge, shiny allows you to create a web app writing code exclusively in R. This, together with the fact that shiny allows to create reactive bids between inputs and outputs, has made StatClip possible. However, good skills in HTML, JavaScript and CSS can enrich a lot applications built in shiny.

RStudio has developed a tutorial to get started on Shiny [27], really recommendable as an introduction to shiny programming. Here, however, there is a brief introduction on how shiny apps work.

3.3.2.1. Structure of a Shiny app

Shiny apps consist of two components:

- A user-interface script, called ui.R
- A server script, called server.R

The ui.R file controls the layout and appearance of the shiny app. The server.R script contains every instruction needed to build your shiny app. Every calculation and reactive element in the app, then, has to be programmed in the server side.

To better illustrate the structure of a shiny app we will follow a simple example, extracted from RStudio's shiny tutorial (http://shiny.rstudio.com/tutorial/lesson1/).

The following example draws a histogram, with a slider input to select the number of bins of the plot. The first script is ui.R. The app is divided in a sidebar, in which we find the slider, and a main panel, containing the plot.

```
library(shiny)

# Define UI for application that draws a histogram
shinyUI(fluidPage(

  # Application title
  titlePanel("Hello Shiny!"),

  # Sidebar with a slider input for the number of bins
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),

    # Show a plot of the generated distribution
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

The following script is the server.R. It contains every necessary instruction to make the app work. More specifically, and as it is written on the comments in the code, in this case the only reactive output is the plot itself. The renderPlot function automatically reloads the histogram at any change produced in the inputs included in the function. In this case, then, when the slider changes its value, the plot "reacts" and the histogram changes.

```
library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should re-execute automatically
  #     when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Once we execute the shiny app, the following is obtained:

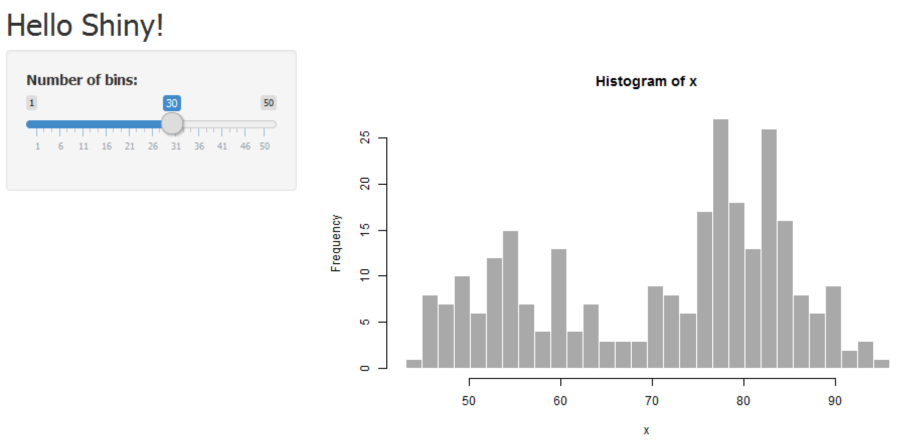


Figure 3.23. Example Shiny app

When changing the number of bins in the side panel, the histogram automatically updates. This is the most important of shiny's features. The reactivity at all the levels, that makes shiny apps extraordinarily versatile and helpful. Shiny is a great tool for programming really visual and interactive apps, and has a lot of future in educational tools.

As it can be seen, shiny helps create visually attractive and highly interactive tools with less effort than other languages. Moreover, R is behind all the computations, which gives even more power to the tool. All StatClip, then, is programmed using shiny.

4. StatClip implementation

As said in the Introduction, the scope of this project is to build a first prototype of StatClip. As it has been explained, as well, not all the functionalities have been implemented on StatClip, as it is a complex software with different processes and difficulties. StatClip will continue growing and being improved until it is a fully operative app.

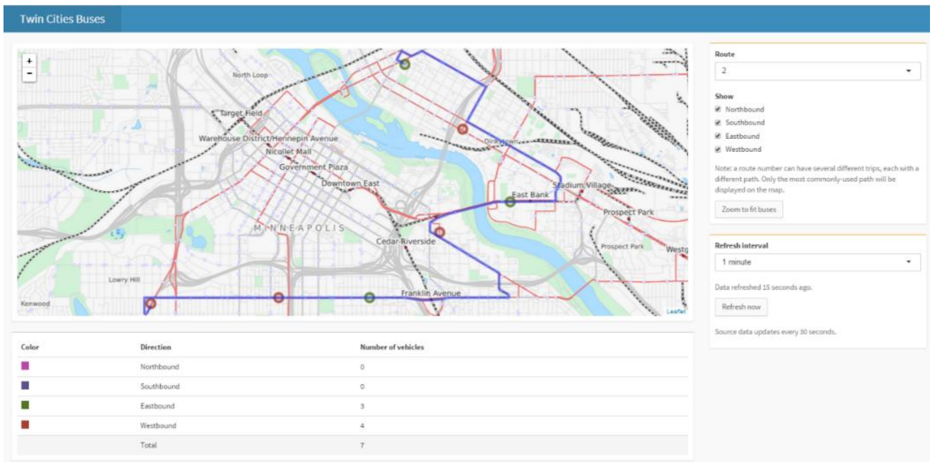
What has been programmed, furthermore, is a first version, and it consequently for sure can be improved and debugged.

All the code written for StatClip can be seen and downloaded from Github. Specifically, the web address of StatClip's repository is: https://github.com/eserrahima/statclip.

Before starting to explain the main structure and code of the app, it is necessary to comment a package that has also been crucial to be able to design StatClip: the shinydashboard package. This package provides a theme on top of shiny, allowing users to easily create visually attractive dashboards [28].

Shinydashboard helps create the side menu, and facilitates the user's navigation through the different tabs. Its basic units are boxes, in which the contents of the main body are displayed. This package completes shiny in the sense that it makes even easier to create a complex program such as StatClip.

As an example, the following dashboard extracted from the shinyapps gallery (https://gallery.shinyapps.io/086-bus-dashboard/):



Route
2

Show
- ✓ Northbound
- ✓ Southbound
- ✓ Eastbound
- ✓ Westbound

Note: a route number can have several different trips, each with a different path. Only the most commonly-used path will be displayed on the map.

Zoom to fit buses

Refresh interval
1 minute

Data refreshed 15 seconds ago.

Refresh now

Source data updates every 30 seconds.

Color	Direction	Number of vehicles
🟪	Northbound	0
🟦	Southbound	0
🟩	Eastbound	3
🟥	Westbound	4
	Total	7

Figure 4.1. Shinydashboard example

To better understand how shinydashboard works, one can read its documentation [28], or visit http://rstudio.github.io/shinydashboard/index.html for more graphical explanations.

As it has just been explained, shinydashboard facilitates the creation of a side menu (the sidebar) which will contain all the options. With the conceptual UI designed, this package is ideal.

4.1. Structure

As it has been explained in the previous chapter, all shiny apps consist of two basic files: ui.R and server.R. However, when an app is complex and it has many different options and operations, writing the code in separated files might be a good idea. Organization and structure, then, are primordial.

StatClip, as any shiny app, will have two main files: ui.R and server.R. From this two main files, different files related to the various functions of StatClip are linked and called from the main files.

The following diagrams try to make StatClip’s structure clearer.

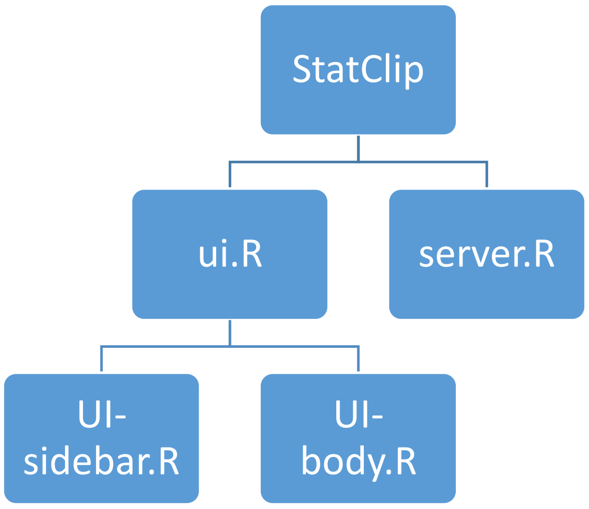


Figure 4.2. StatClip's main files

This first diagram shows the firsts’ levels division of StatClip. The user interface file calls two files, one generating the sidebar menu, and the other one the main body of the app. The server.R file calls different files, as we will see in Figure 4.4, but in a lower level.

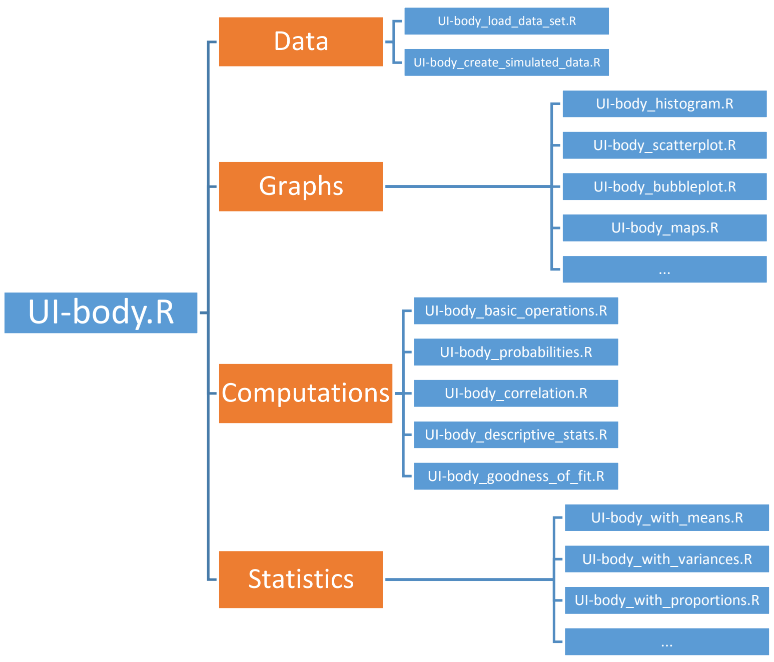


Figure 4.3. Files called from UI-body.R

The same structure is followed by the files that are called by the server.R script. The logical followed in designing this structure is really simple: dividing a complex problem in smaller and easier to handle ones. Each “problem” to solve (that is, each function StatClip does) has a user interface and some calculations and reactive functions that make it work. Therefore, each small portion of the problem, each part of StatClip (in the end, menu option) needs its own UI file and its own server file. The structure coming out of the server.R file, then, is the same than the one from UI-body.R.

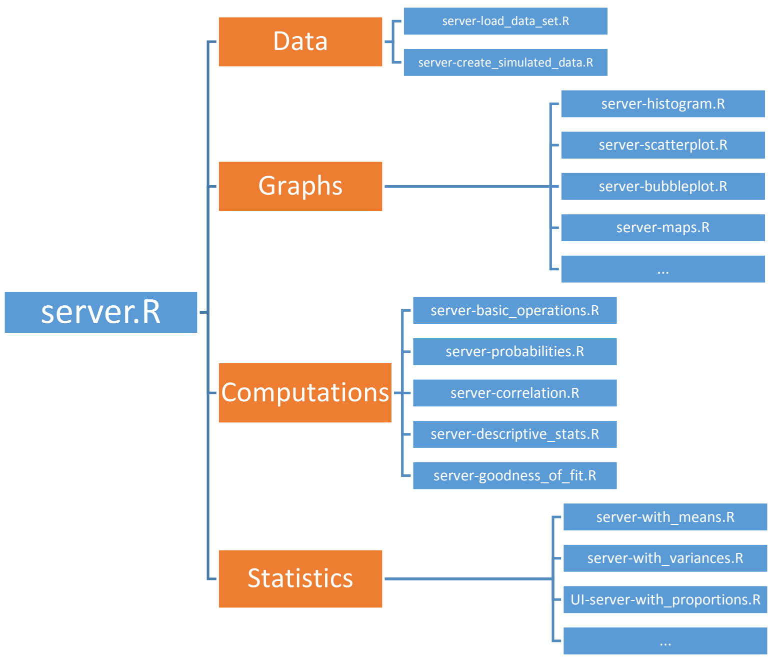


Figure 4.4. Files called from server.R

This diagrams just seen are an explanation of StatClip's architecture. It was decided to organize the program this way to facilitate the tasks of not only programming, but debugging, sharing and understanding the code. When programming such a complex application, it is a must to keep it as simple and neat as possible, for it will help find errors and continue programming the app. It helps, as well, to be able to program different tabs in a parallel way without interrupting or risking the total server or UI files, and only call the new code once it is finished.

This disaggregated architecture, then, makes the total number of files higher, but each of them is shorter in extension (most of them not longer than 100 lines) and, in consequence, easier to handle. Chapter 4.2 explains how all of this was programmed.

4.2. R Programming

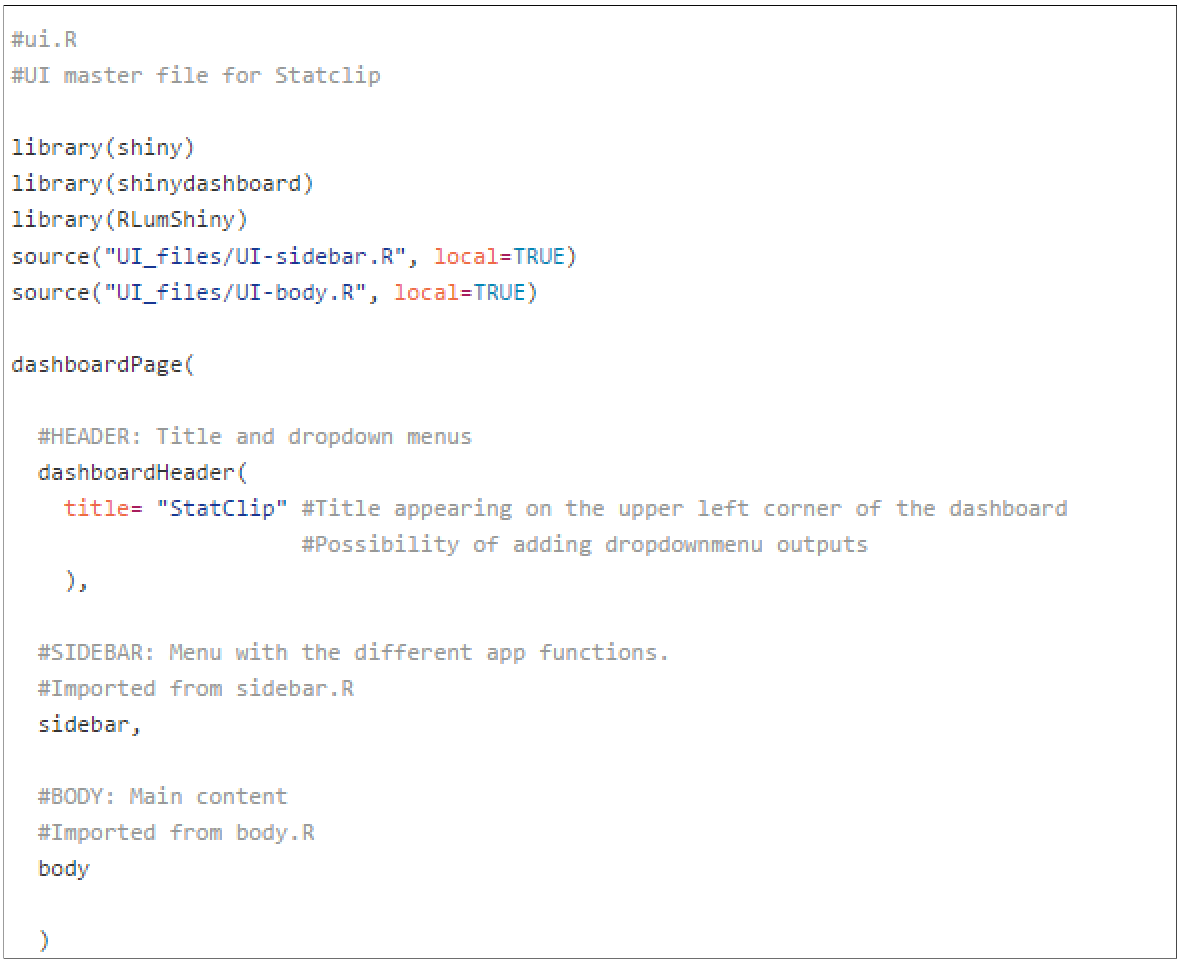
In this chapter the programming of StatClip process will be explained. It has been explained various times that not all the functionalities have been implemented yet. StatClip was started from a blank script, and it is a quite complex application. While it does not perform highly complex computations, it has a powerful and reactive User Interface.

4.2.1. User Interface (UI)

4.2.1.1. ui.R

The ui.R file defines the main structure of StatClip, together with the UI-sidebar.R and UI-body.R. Here in Figure 4.5 we can see the following:

- ui.R imports three libraries: shiny, shinydashboard and RLumShiny.
- It also calls two files: UI-sidebar.R and UI-body.R
- It constructs the dashboard page, calling the sidebar and body from their respective files.



```r
#ui.R
#UI master file for Statclip

library(shiny)
library(shinydashboard)
library(RLumShiny)
source("UI_files/UI-sidebar.R", local=TRUE)
source("UI_files/UI-body.R", local=TRUE)

dashboardPage(

  #HEADER: Title and dropdown menus
  dashboardHeader(
    title= "StatClip" #Title appearing on the upper left corner of the dashboard
                      #Possibility of adding dropdownmenu outputs
    ),

  #SIDEBAR: Menu with the different app functions.
  #Imported from sidebar.R
  sidebar,

  #BODY: Main content
  #Imported from body.R
  body

  )
```

Figure 4.5. ui.R

This, even though it can seem short, is the whole ui.R file. That is part of the beauty of dividing StatClip in different files and functions: it allows the program to only have short and easy to understand scripts.

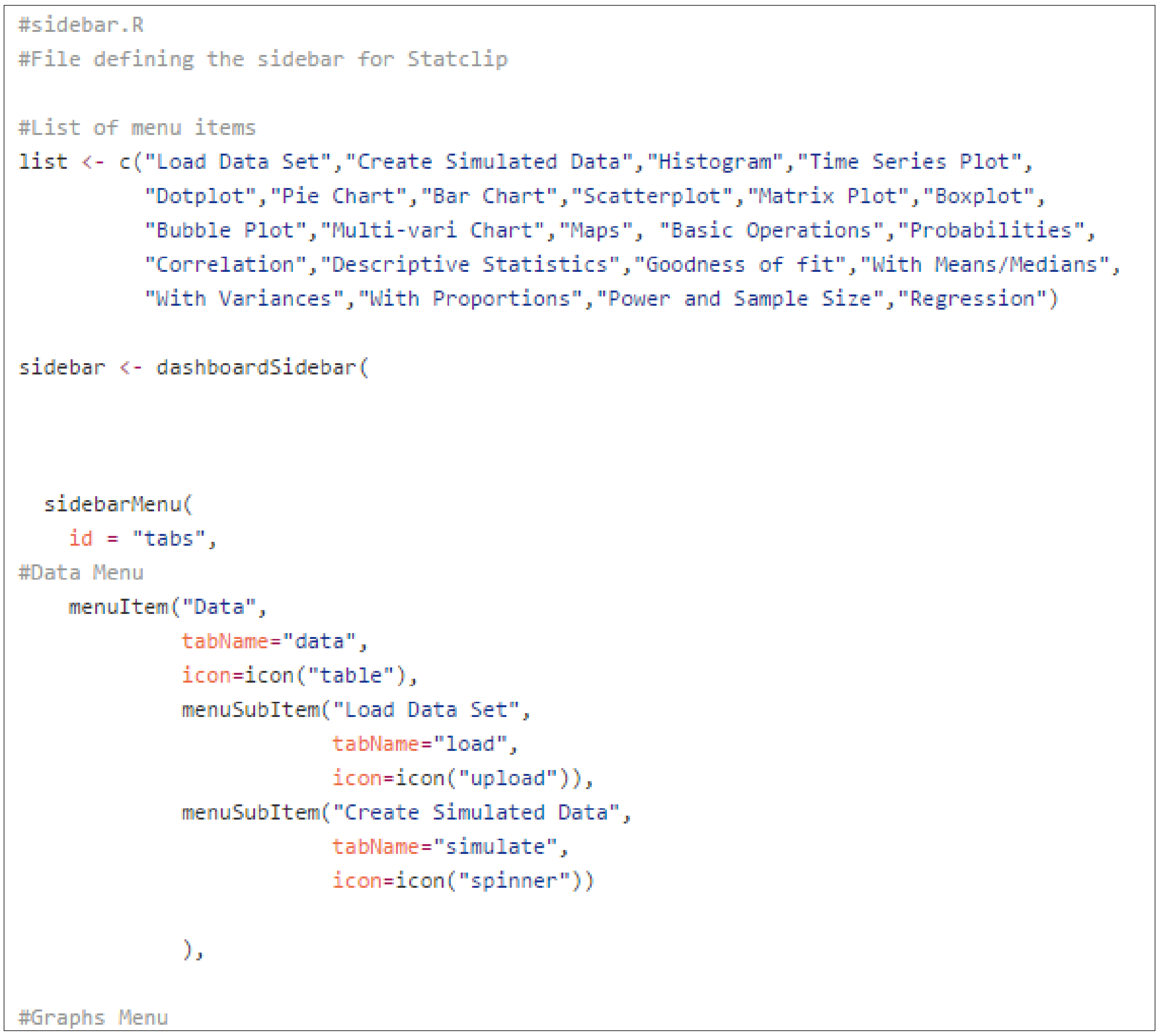
Lines preceded by # are comment lines. This means the interpreter passes over them and does not execute them. In consequence, its only purpose is to work as comments and indications to make the code easier to understand.

4.2.1.2. UI-sidebar.R

This script defines the side menu of StatClip. As explained in the theoretical design and seen in the UI theoretical approach, StatClip has a side menu always visible form which users can change pages. Each page has its own functions and works independently. The only pages that are linked to all the others are the Data ones, as the rest of functions use the updated data.

Here in Figure 4.6 we can see how the file is organized. Each MenuItem corresponds to one of the four groups that include all functionalities: Data, Graphs, Computations and Statistics. In each MenuItem we can find several MenuSubItems, which correspond to each of the pages our software has.

Figure 4.6 shows only the Data MenuItem (with its menuSubItems). The other three menu items are omitted (whole script can be found in the Github repository). We can see, as well, a list of Menu Items defined in the first place (containing the names of all the pages). This list is used by the SelectizeInput at the end of the sidebar (Figure 4.8), which is explained below.



```
#sidebar.R
#File defining the sidebar for Statclip

#List of menu items
list <- c("Load Data Set","Create Simulated Data","Histogram","Time Series Plot",
          "Dotplot","Pie Chart","Bar Chart","Scatterplot","Matrix Plot","Boxplot",
          "Bubble Plot","Multi-vari Chart","Maps", "Basic Operations","Probabilities",
          "Correlation","Descriptive Statistics","Goodness of fit","With Means/Medians",
          "With Variances","With Proportions","Power and Sample Size","Regression")

sidebar <- dashboardSidebar(



  sidebarMenu(
    id = "tabs",
#Data Menu
    menuItem("Data",
             tabName="data",
             icon=icon("table"),
             menuSubItem("Load Data Set",
                         tabName="load",
                         icon=icon("upload")),
             menuSubItem("Create Simulated Data",
                         tabName="simulate",
                         icon=icon("spinner"))

             ),

#Graphs Menu
```

Figure 4.6. UI-sidebar.R (partial)

The script just seen defines StatClip sidebar. It can be seen in Figure 4.7. The sidebarMenu function from the shinydashboard package automatically collapses the menu tabs, so that only the one selected by the user is expanded.

Below all the menu items, though, we can see the selector that we talked about in the theoretical design chapter. It allows users to change tab without looking for it on the menu. Just starting to write the item they are looking for, the program gives autocomplete options to easily find the new tab. This selector input is programmed at the end of UI-sidebar.R (Figure 4.8), and the code that gives StatClip the reactivity to change the tab when using this selector in written in server.R (Figure 4.9).

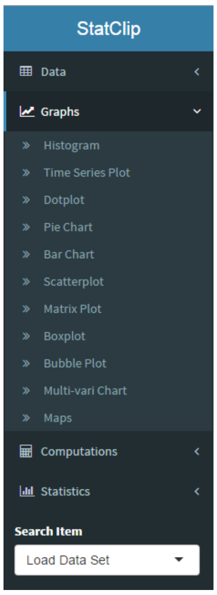


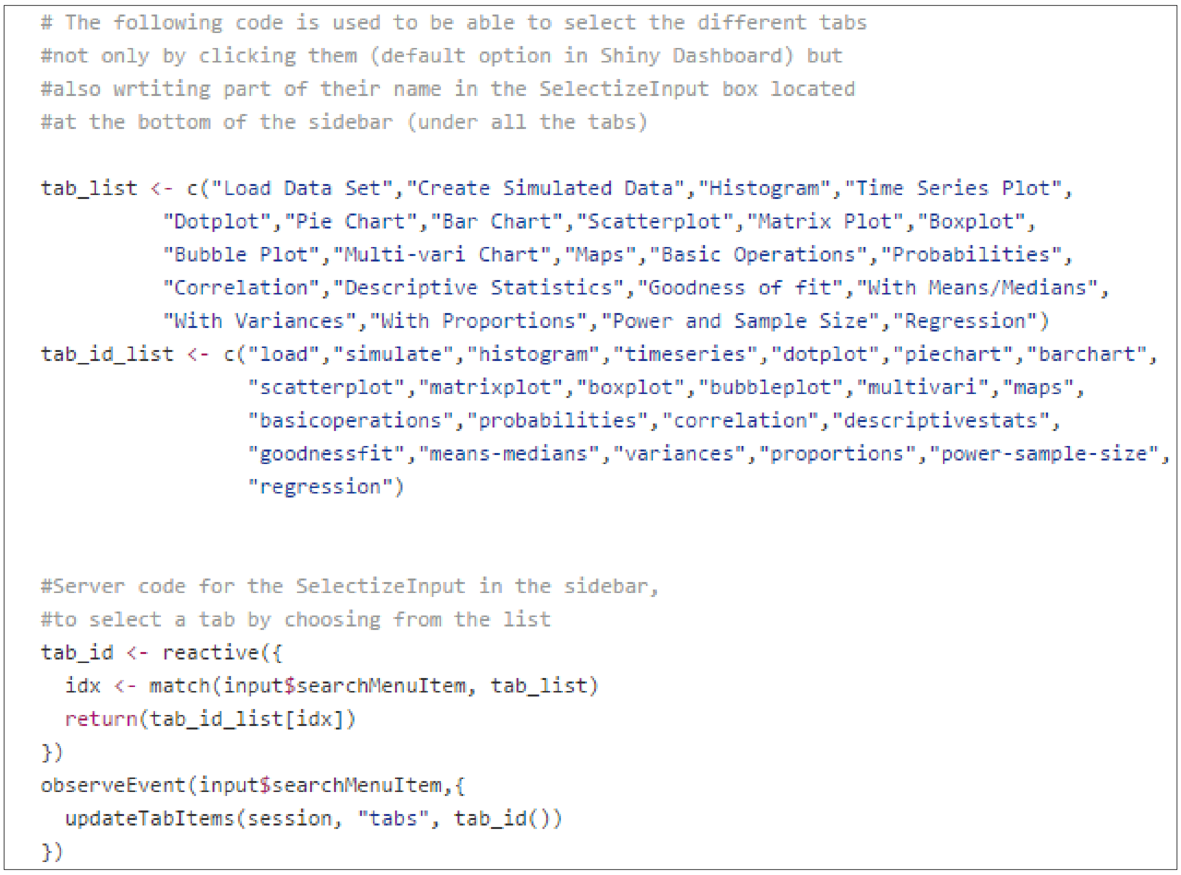
Figure 4.7. Sidebar

```
selectizeInput("searchMenuItem", label="Search Item", choices=list,
                selected=NULL, multiple=FALSE)
```

Figure 4.8. UI-sidebar.R (select input)

The code in Figure 4.8 creates the input (as can be seen, just a simple function) referred. However, the code to allow users to use the input (that is, giving the tool functionality) is slightly more complicated. It involves the creation of two lists: one with the tab names, and one with the aliases those tabs have to call the from inside the program. Each tab has a label (name displayed on the UI, the one visible to the user) and a variable name, which is used in the code to call the tab.

The code in Figure 4.9 finds the tab variable name checking the label name selected by the user. Once this is done, changes the tab to the selected one.



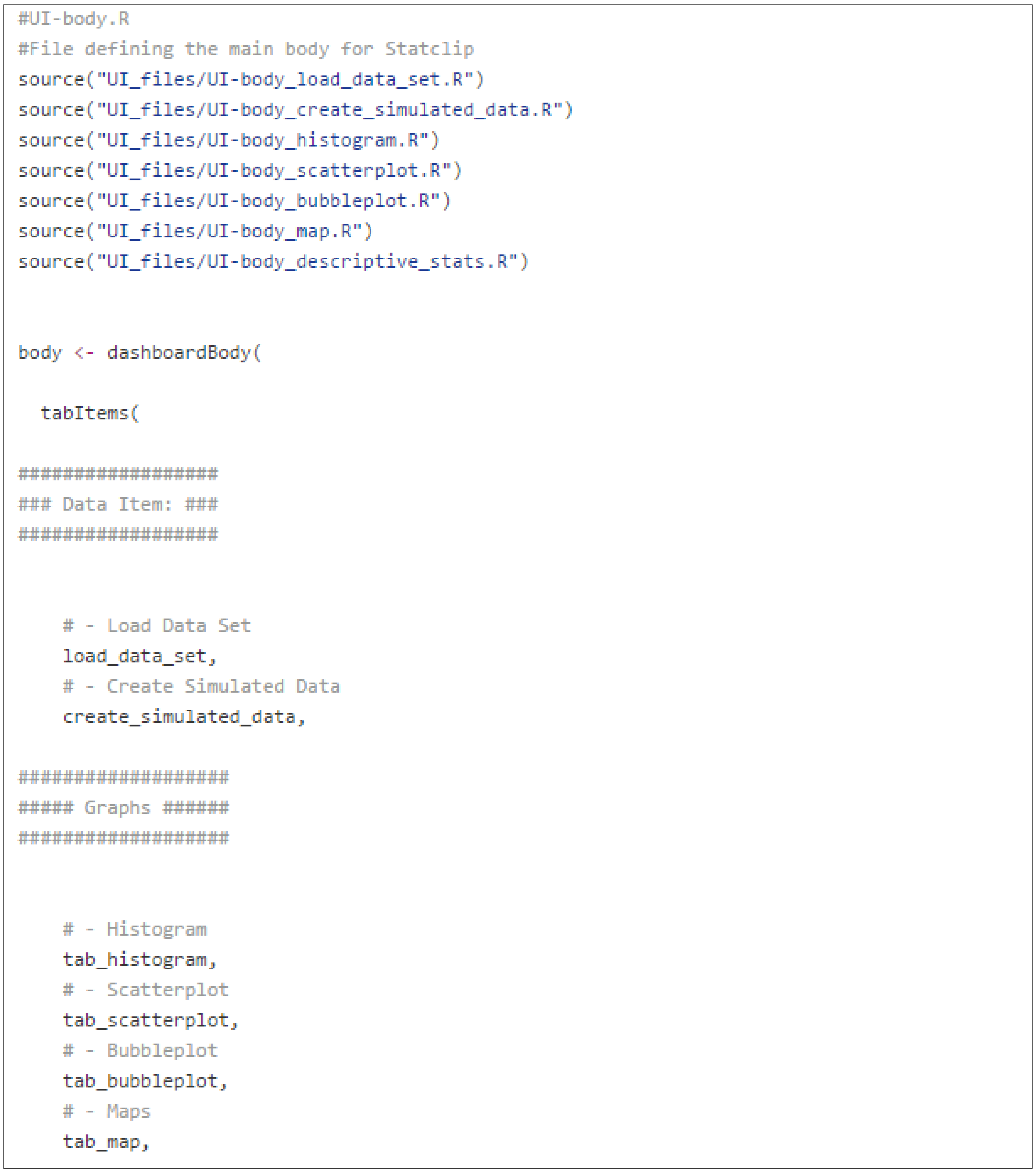
```
# The following code is used to be able to select the different tabs
#not only by clicking them (default option in Shiny Dashboard) but
#also wrtiting part of their name in the SelectizeInput box located
#at the bottom of the sidebar (under all the tabs)

tab_list <- c("Load Data Set","Create Simulated Data","Histogram","Time Series Plot",
          "Dotplot","Pie Chart","Bar Chart","Scatterplot","Matrix Plot","Boxplot",
          "Bubble Plot","Multi-vari Chart","Maps","Basic Operations","Probabilities",
          "Correlation","Descriptive Statistics","Goodness of fit","With Means/Medians",
          "With Variances","With Proportions","Power and Sample Size","Regression")
tab_id_list <- c("load","simulate","histogram","timeseries","dotplot","piechart","barchart",
                 "scatterplot","matrixplot","boxplot","bubbleplot","multivari","maps",
                 "basicoperations","probabilities","correlation","descriptivestats",
                 "goodnessfit","means-medians","variances","proportions","power-sample-size",
                 "regression")


#Server code for the SelectizeInput in the sidebar,
#to select a tab by choosing from the list
tab_id <- reactive({
  idx <- match(input$searchMenuItem, tab_list)
  return(tab_id_list[idx])
})
observeEvent(input$searchMenuItem,{
  updateTabItems(session, "tabs", tab_id())
})
```

Figure 4.9. server.R (selectizeInput code)

4.2.1.3. UI-body.R

The UI-body.R script, as it has been explained, calls the body content of each of the tabs in StatClip. The main body contents, then, are not in the UI-body file. In Figure 4.10 we can see how the file is organized.

In the first place, the script calls the necessary files (where the actual body content is stored). Then, the body is created using the function “dashboardBody”. All the tab contents are called from inside a “tabItems” function. In UI-body.R, to keep an easy to follow structure, tabs are clearly separated in the four basic groups (Data, Graphs, Computations and Statistics).



```
#UI-body.R
#File defining the main body for Statclip
source("UI_files/UI-body_load_data_set.R")
source("UI_files/UI-body_create_simulated_data.R")
source("UI_files/UI-body_histogram.R")
source("UI_files/UI-body_scatterplot.R")
source("UI_files/UI-body_bubbleplot.R")
source("UI_files/UI-body_map.R")
source("UI_files/UI-body_descriptive_stats.R")


body <- dashboardBody(

  tabItems(

##################
### Data Item: ###
##################


    # - Load Data Set
    load_data_set,
    # - Create Simulated Data
    create_simulated_data,

###################
##### Graphs ######
###################


    # - Histogram
    tab_histogram,
    # - Scatterplot
    tab_scatterplot,
    # - Bubbleplot
    tab_bubbleplot,
    # - Maps
    tab_map,
```

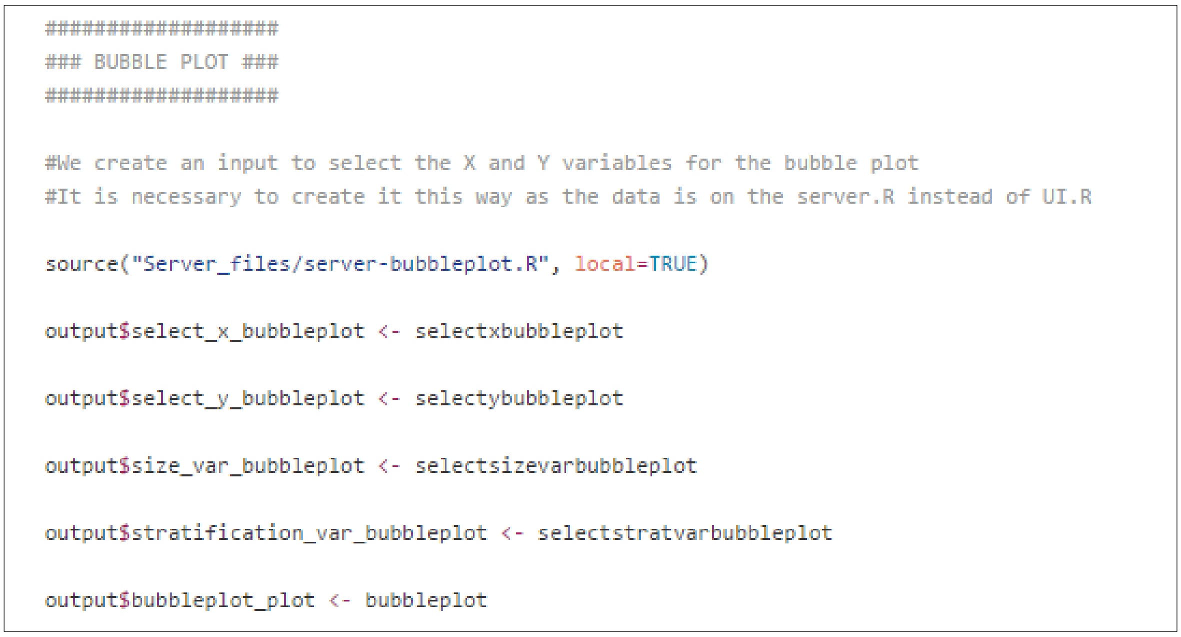
Figure 4.10. UI-body.R

4.2.2. Server

The server in a shiny app, as it has been explained, is sort of the “brain” of the program. Every calculation or reactive display comes from the code in the server.R file.

The server structure in StatClip is defined in section 4.1, and here we just show a brief example of how the main server.R script calls the function specific server files.

More specifically, in we can see the code lines necessary to call the outputs for the Bubble Plot tab.



```
###################
### BUBBLE PLOT ###
###################

#We create an input to select the X and Y variables for the bubble plot
#It is necessary to create it this way as the data is on the server.R instead of UI.R

source("Server_files/server-bubbleplot.R", local=TRUE)

output$select_x_bubbleplot <- selectxbubbleplot

output$select_y_bubbleplot <- selectybubbleplot

output$size_var_bubbleplot <- selectsizevarbubbleplot

output$stratification_var_bubbleplot <- selectstratvarbubbleplot

output$bubbleplot_plot <- bubbleplot
```

Figure 4.11. server.R (calling bubble plot's outputs)

As we can see, from the server main file we need to call each output defined in the tab's specific server file.

In the following section we will follow the inner structure of the code for two example tabs, to let everything clear and understood.

4.3. Code examples: "Load Data Set" and "Histogram"

This section will exhaustively analyze the code necessary for two different StatClip functionalities: the "Load Data Set" and "Histogram" tabs. The goal of this chapter is to give a clear vision of how StatClip works, and to help understand the coding process of each of the functionalities.

4.3.1. Load Data Set

The "Load Data Set" tab is prepared to allow users to paste their own data (directly copied from an Excel file, for example), or to use a StatClip's data set. The data sets prepared in StatClip are "Iris" and "mtcars".

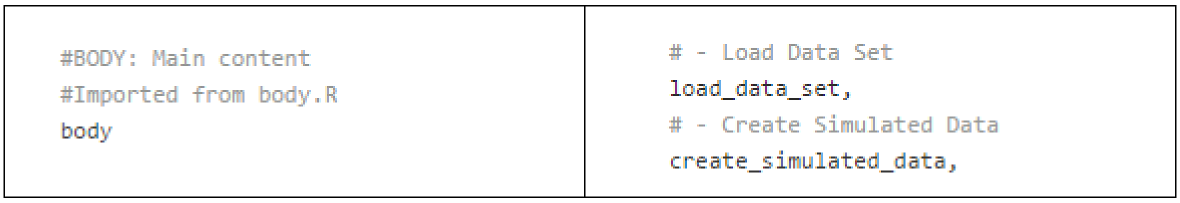
The Iris data set (known as Anderson's Iris Data set) gives the measures (in cm) of the variables sepal length, sepal width, petal length and petal width of three different iris species: *Iris setosa*, *Iris versicolor* and *Iris virginica*. There are 150 measurements in total, from 50 flowers of each species [29]. The iris dataset is implemented by default in R; it is only needed to call it from the command line.

The mtcars (Motor Trend Car Road Tests) data set is as well included by default in R (in the DataSets package). The data was extracted from the 1974 *Motor Trend* US magazine and analyzes fuel consumption and 10 aspects of the car design for 32 automobiles [30].

The User Interface general aspect was explained at section 3.2.2.1, and here we are going to see the necessary code to program this tab.

4.3.1.1. User Interface

In the first place, we have already seen that UI-body.R calls the UI-body_load_data_set.R script, which contains the necessary information to obtain the "Load Data Set" tab user interface. In Figure 4.12 (left) we can see the ui.R file calling the "body" function from UI-body.R. In Figure 4.12 (right) the lines of code that call the UI-body_load_data_set.R file from the main body file.

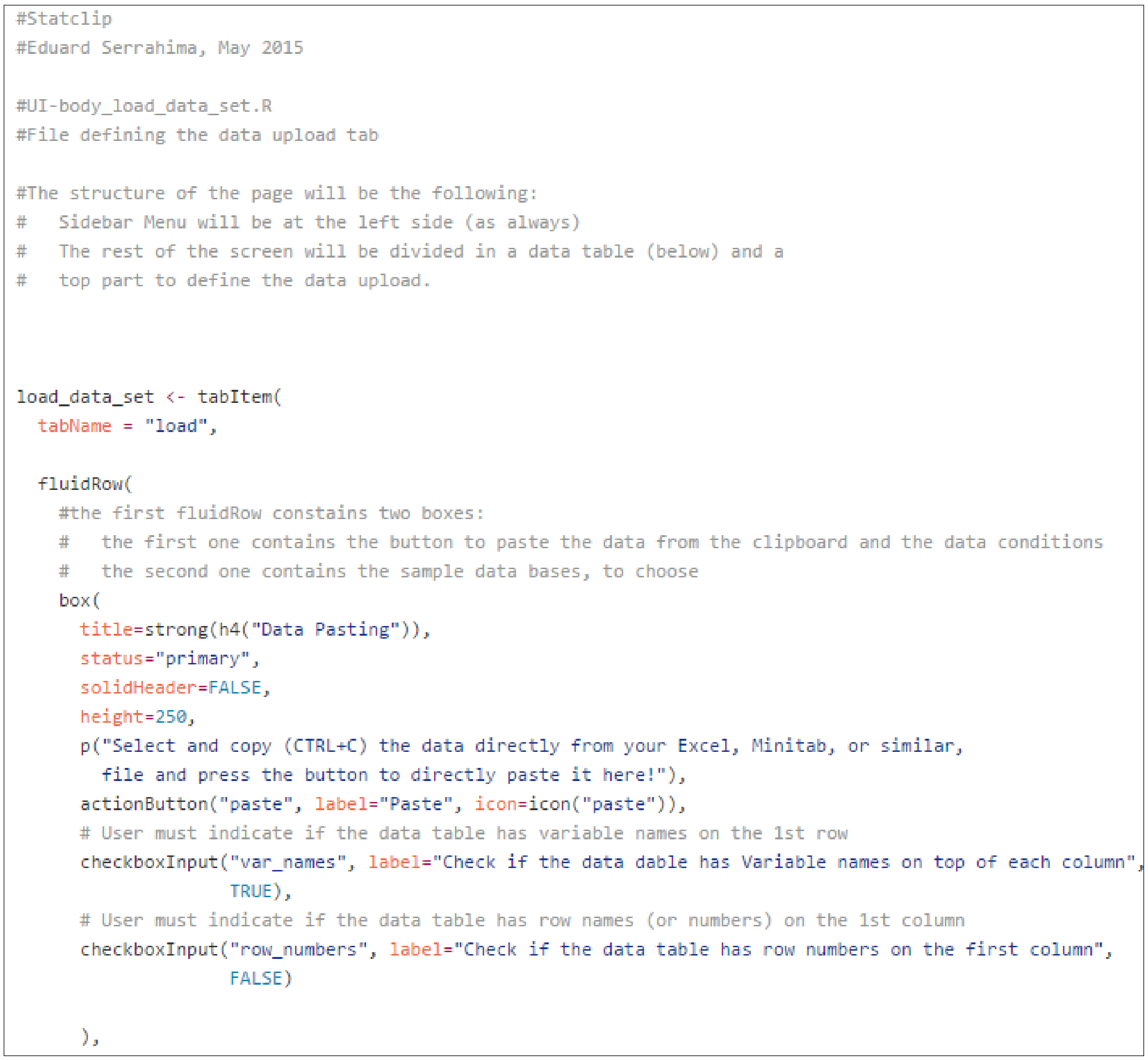


```
#BODY: Main content
#Imported from body.R
body
```

```
# - Load Data Set
load_data_set,
# - Create Simulated Data
create_simulated_data,
```

Figure 4.12. ui.R and UI-body.R imports

The UI-body_load_data_set.R script will now be exhaustively analyzed. The code will be shown divided in two parts, to allow easier explanations.



```
#Statclip
#Eduard Serrahima, May 2015

#UI-body_load_data_set.R
#File defining the data upload tab

#The structure of the page will be the following:
#   Sidebar Menu will be at the left side (as always)
#   The rest of the screen will be divided in a data table (below) and a
#   top part to define the data upload.



load_data_set <- tabItem(
  tabName = "load",

  fluidRow(
    #the first fluidRow constains two boxes:
    #   the first one contains the button to paste the data from the clipboard and the data conditions
    #   the second one contains the sample data bases, to choose
    box(
      title=strong(h4("Data Pasting")),
      status="primary",
      solidHeader=FALSE,
      height=250,
      p("Select and copy (CTRL+C) the data directly from your Excel, Minitab, or similar,
        file and press the button to directly paste it here!"),
      actionButton("paste", label="Paste", icon=icon("paste")),
      # User must indicate if the data table has variable names on the 1st row
      checkboxInput("var_names", label="Check if the data dable has Variable names on top of each column",
                    TRUE),
      # User must indicate if the data table has row names (or numbers) on the 1st column
      checkboxInput("row_numbers", label="Check if the data table has row numbers on the first column",
                    FALSE)

    ),
```

Figure 4.13. UI-body_load_data_set.R (1)

In Figure 4.13 we can see the first part of the file. Following the code order (and skipping the comments and explanations at the beginning) we see:

- Load_data_set is a tabItem function; that is, creates the user interface for one of the tabs in the sidebar. Every element that will appear on the final User Interface of this “Load Data Set” page needs to be written inside the tabItem function.
- The tabName instruction associates the tabItem function to an element in the sidebar. The name “load”, then, has to be the same in the sidebar script. If it is not, the code written inside the tabItem function will not be displayed.
- The page is the divided in two rows (see Figure 3.13). Each one is defined inside a fluidRow function. The fluidRow we can read in the code in Figure 4.13 contains the

first row of the “Load Data Set” page.

- In this first row defined, we have two boxes. Each one is contained in a box function. All the functions here referred (tabItem, fluidRow, box, etc.) are part of the shiny and shinydashboard packages.
- In the first box, we see:
    - The title. In this case “Data pasting”, as it is the box to directly paste the user’s own data.
    - The status. Shinydashboard has some predefined statuses for the boxes. Each status is a color combination. “Primary” corresponds to a blue combination.
    - “SolidHeader=FALSE”. ShinyDashboard allows users to choose if they want the header of the box filled with the selected color or not (Figure 4.14).
    - Height: The height, in pixels, of the box.
    - Text included inside a p(), which indicates that it all should be included in a paragraph.
    - An actionButton (which is just a normal press button) with the following properties:
        - “paste”: this is the identifier. In the server file, the input coming from this action button will be identified using this alias.
        - Label: The label (“Paste”) is the text that will be displayed in the button.
        - Icon: An icon can be added to the button. The icons used have been selected from a predefined list in the package.
    - A checkboxInput: This is just a check box; that is, an option with only two possibilities: selected (TRUE) or not selected (FALSE).
        - “var_names” is its identifier.
        - The label indicates the function of this input. In this case, user has to indicate if the pasted data includes a header row or not.
        - TRUE: this indicates that, by default, the check box has to be selected.
    - A second checkboxInput: The order in the code indicates as well the order in the final UI. That is, this second box will be located under the first one.
        - “row_numbers” is the identifier in this case.
        - The label indicates that users have to select the option to indicate if the first column contains row numbers.
        - FALSE: this indicates that, by default, the check box has to be unselected.

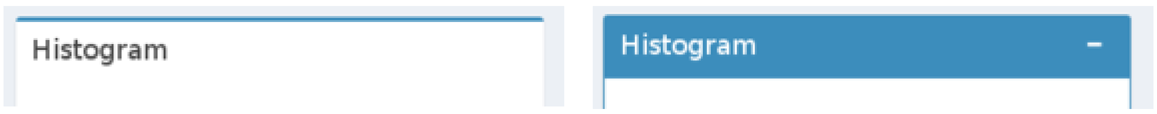
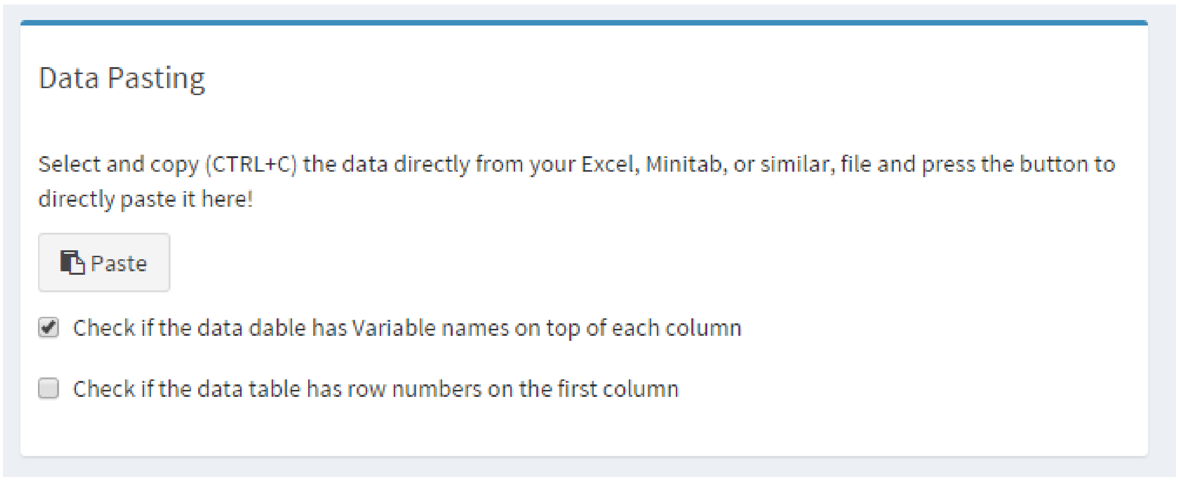


Figure 4.14. SolidHeader = FALSE (left) or TRUE (right)

Here in we can see the final appearance of the box just explained.



Data Pasting

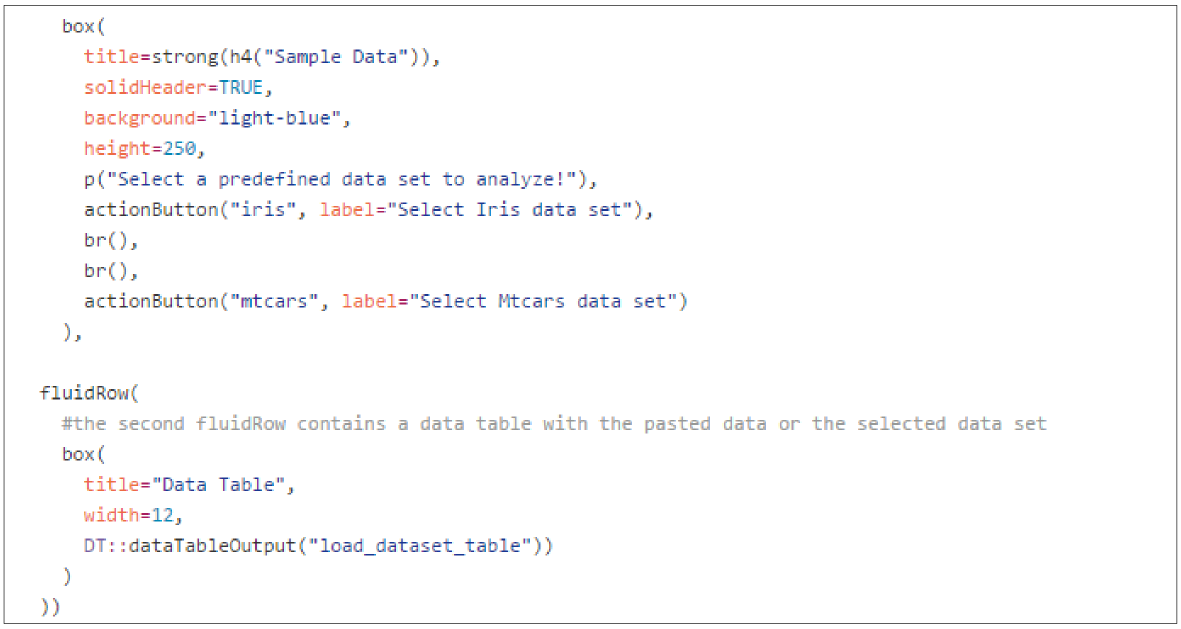
Select and copy (CTRL+C) the data directly from your Excel, Minitab, or similar, file and press the button to directly paste it here!

Paste

- [x] Check if the data dable has Variable names on top of each column
- [ ] Check if the data table has row numbers on the first column

Figure 4.15. Load Data Set. Data pasting box.

After checking the first half of the code included in UI-body_load_data_set.R, we will review the rest of the code (Figure 4.16). It includes the two left boxes: one containing the necessary to use the predefined datasets, and the second one containing the selected data (either a predefined data set or the pasted data).



```
    box(
      title=strong(h4("Sample Data")),
      solidHeader=TRUE,
      background="light-blue",
      height=250,
      p("Select a predefined data set to analyze!"),
      actionButton("iris", label="Select Iris data set"),
      br(),
      br(),
      actionButton("mtcars", label="Select Mtcars data set")
    ),

  fluidRow(
    #the second fluidRow contains a data table with the pasted data or the selected data set
    box(
      title="Data Table",
      width=12,
      DT::dataTableOutput("load_dataset_table")
    )
  ))
```

Figure 4.16. UI-body_load_data_set.R (2)

The code in Figure 4.16 includes:

- A box, still contained in the first fluid row. In this second box we find:
    - Title: Sample Data
    - solidHeader = True. Check Figure 4.14 to again see the effect.
    - Background = “light-blue”. In the first box we selected the color using a “Status”. Color can be selected defining a background. With this, the whole box will be colored.
    - A text paragraph: “Select a predefined data set to analyze!”
    - An actionButton that, if pressed, selects the Iris data set:
        - “iris” is its identifier.
        - The label (the text appearing in the button) is “Select Iris data set”.
    - A second actionButton to select the mtcars data set:
        - “mtcars” is its identifier.
        - The label is: “Select Mtcars data set”.

  In Figure 4.17we can see the visual appearance of the box just defined.
- The second fluidRow, which contains only one box. In consequence, this box will occupy all the width of the page. The box contains:
    - Title: “Data Table”.
    - Width=12. In Shinydashboard, box width is not referred in pixels, but in relative measures. 12 means occupying all the available width. In this case, being all

the page free in the second row,the box will be all the page wide.

- The actual data table. As it can be seen, the table is called with the following instruction: DT::dataTableOutput("load_dataset_table"). This means the UI-body_load_data_ser.R file is calling the output from the server file. All the table is prepared in the server side, and from the UI file there is only the need to call what has been defined.

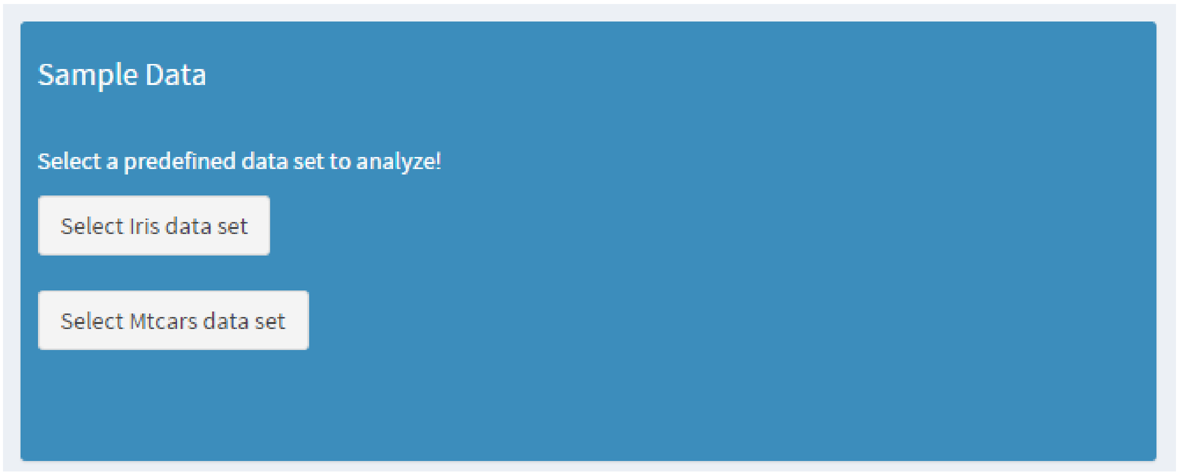
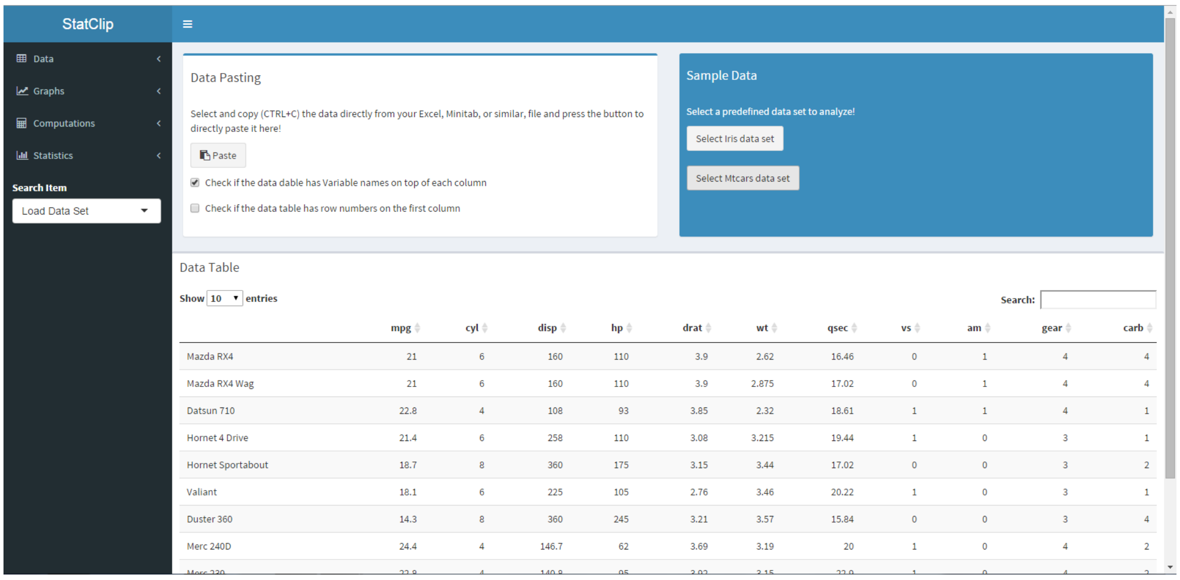


Figure 4.17. Load Data Set. Sample Data box.

And, finally, in Figure 4.18 we can see the complete "Load Data Set" page. In this case, the Data Table shows the Mtcars predefined data set.



Select and copy (CTRL+C) the data directly from your Excel, Minitab, or similar, file and press the button to directly paste it here!

[✓] Check if the data dable has Variable names on top of each column

[ ] Check if the data table has row numbers on the first column

Select a predefined data set to analyze!

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1
Duster 360	14.3	8	360	245	3.21	3.57	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1	0	4	2

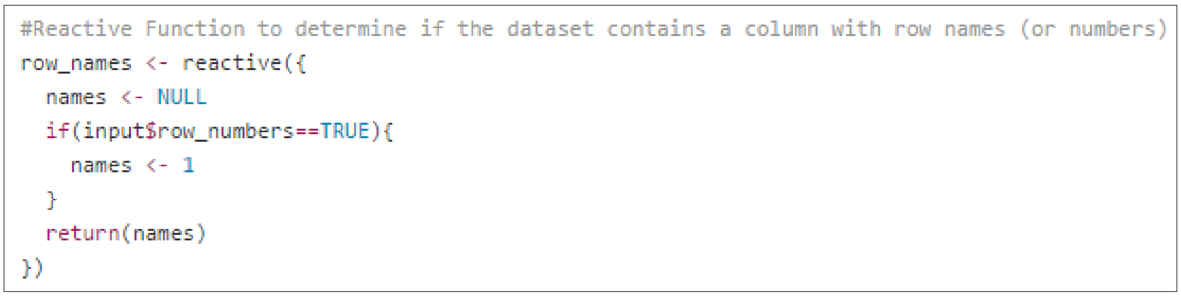
Figure 4.18. Load Data Set

4.3.1.2. Server

However, the User Interface file would just be an empty framework with no use if it was not for the server file. As an example, the table in Figure 4.18 is displayed thanks to the code in the server script.

The server code has a way higher level of difficulty. The reason behind this fact is that the server script has to define the reactive processes that display all the outputs. In a complex app such as StatClip, lots of dependencies are created and so the whole code becomes even more complicated.

For the server code, the parts will be shown and explained separately, as they involve higher complexity.

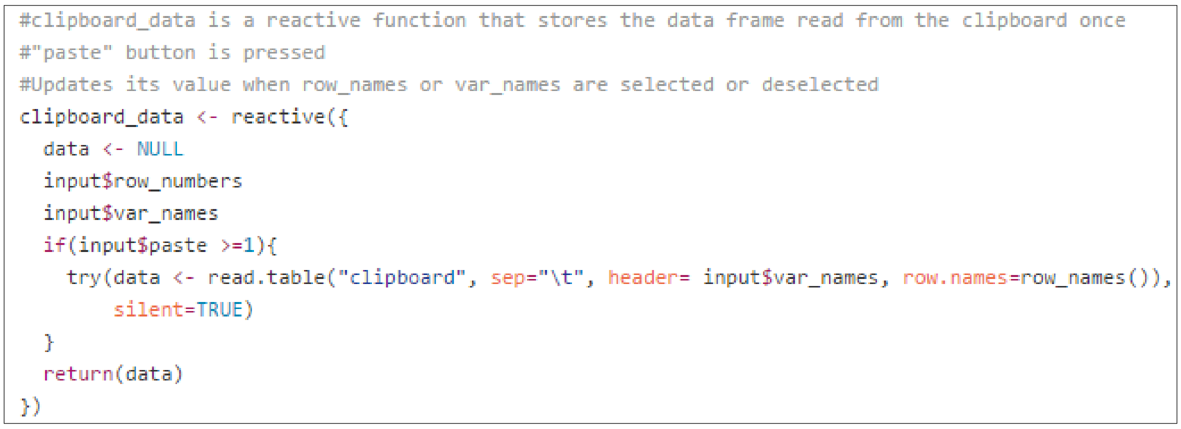


```
#Reactive Function to determine if the dataset contains a column with row names (or numbers)
row_names <- reactive({
  names <- NULL
  if(input$row_numbers==TRUE){
    names <- 1
  }
  return(names)
})
```

Figure 4.19. server-load_data_set.R (1)

The first function defined is row_names, which is a reactive function. This means that, when any of the parameters in the function changes, the value of the function is recalculated. In this case, we see:

- Variable “names” is set to NULL.
- If the check box defined in the UI files is set to TRUE, then set the variable “names” to 1. As we can see, to call an input located in the UI files (in this case, a check box to see if the first column of the data set contains row numbers or not) we need to use the structure “input$” + identifier of the input.
- Return the variable’s value.

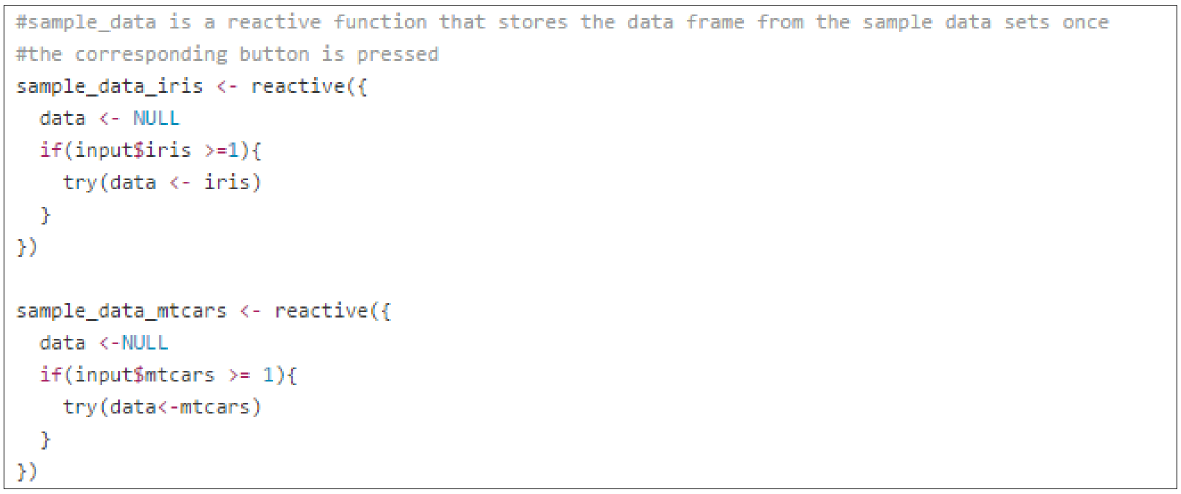


```r
#clipboard_data is a reactive function that stores the data frame read from the clipboard once
#"paste" button is pressed
#Updates its value when row_names or var_names are selected or deselected
clipboard_data <- reactive({
  data <- NULL
  input$row_numbers
  input$var_names
  if(input$paste >=1){
    try(data <- read.table("clipboard", sep="\t", header= input$var_names, row.names=row_names()),
        silent=TRUE)
  }
  return(data)
})
```

Figure 4.20. server-load_data_set.R (2)

The second part of the server-load_data_set.R file is the clipboard_data reactive function. This function stores the data directly from the clipboard. We can see:

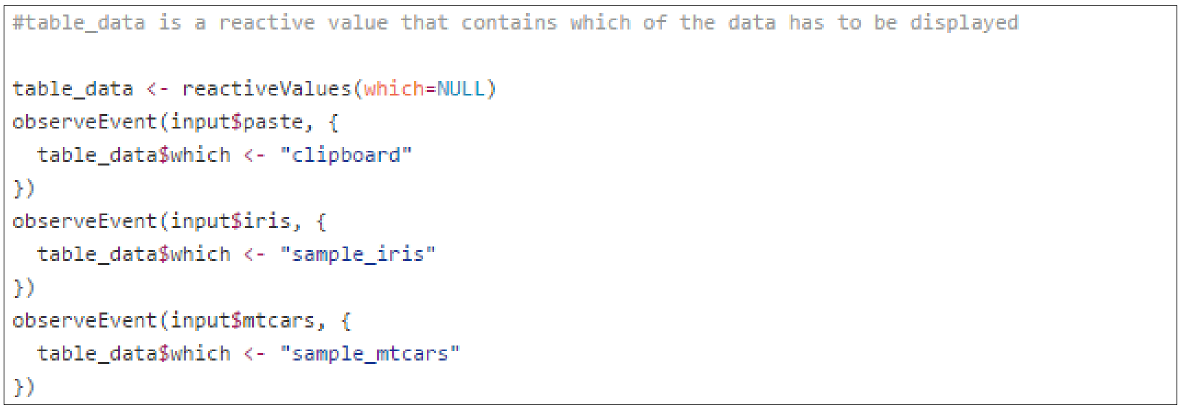
- Variable “data” set to NULL.
- We see “input$row_numbers” and “input$var_names”. These are the check box inputs that we designed in the UI files. The fact that we put them here makes the function react when their values change. That is, we make sure that we a user clicks on one of the check boxes the data in clipboard_data is updated.
- Input$paste is the button to press to paste the user’s data. Action buttons, as programmed in Shiny, have a inner value that starts at 0 and grows by one unit with each click. In consequence, this condition implies that the button has to have been clicked to get the values.
    - In this “if” loop, we assign the clipboard’s data to the variable “data”. One can see that this assignment is carried on inside a try function. This helps avoid any errors. In some computers, especially older and not powerful computers, the read.table function can take a little longer, being this sometimes an error origin. To avoid this problem, the try function only shows the results when there are no errors.
- The value of the “data” variable is returned.



```
#sample_data is a reactive function that stores the data frame from the sample data sets once
#the corresponding button is pressed
sample_data_iris <- reactive({
  data <- NULL
  if(input$iris >=1){
    try(data <- iris)
  }
})

sample_data_mtcars <- reactive({
  data <-NULL
  if(input$mtcars >= 1){
    try(data<-mtcars)
  }
})
```

Figure 4.21. server-load_data_set.R (3)

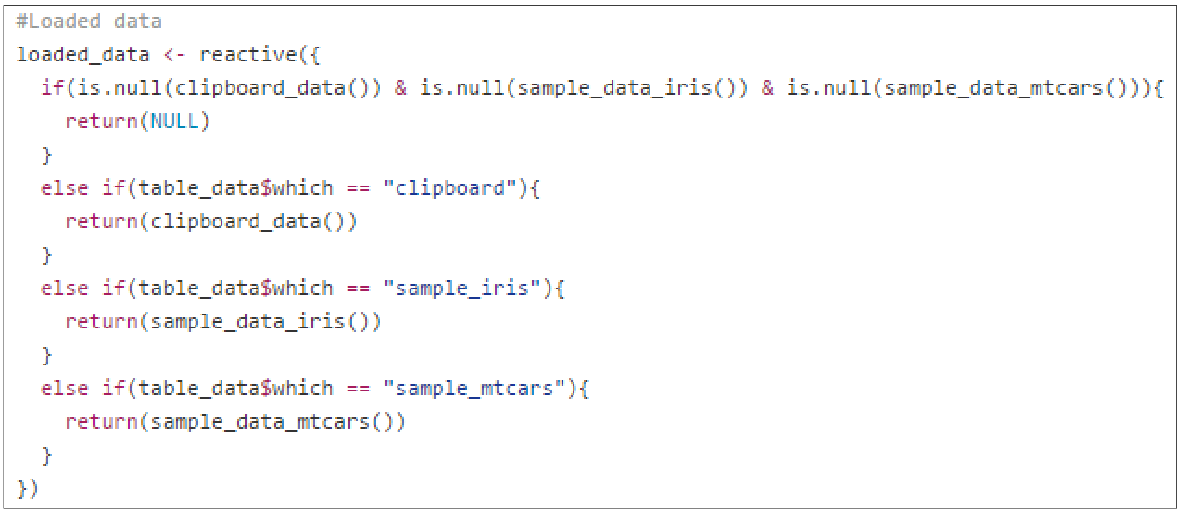
This third part includes two functions that store the sample data sets when their respective buttons are clicked. They follow exactly the same structure than the clipboard_data reactive function, which was extensively explained earlier. The only difference is that they do not need to react to anything but their own action button.



```
#table_data is a reactive value that contains which of the data has to be displayed

table_data <- reactiveValues(which=NULL)
observeEvent(input$paste, {
  table_data$which <- "clipboard"
})
observeEvent(input$iris, {
  table_data$which <- "sample_iris"
})
observeEvent(input$mtcars, {
  table_data$which <- "sample_mtcars"
})
```

Figure 4.22. server-load_data_set.R (4)

Table_data$which is an auxiliary reactive value that changes its value when any of the three action buttons in the UI is pressed. It is a string that changes between “clipboard”, “sample_iris” and “sample_mtcars”, depending on which was the last pressed button. This value is used in the following function (Figure 4.23).

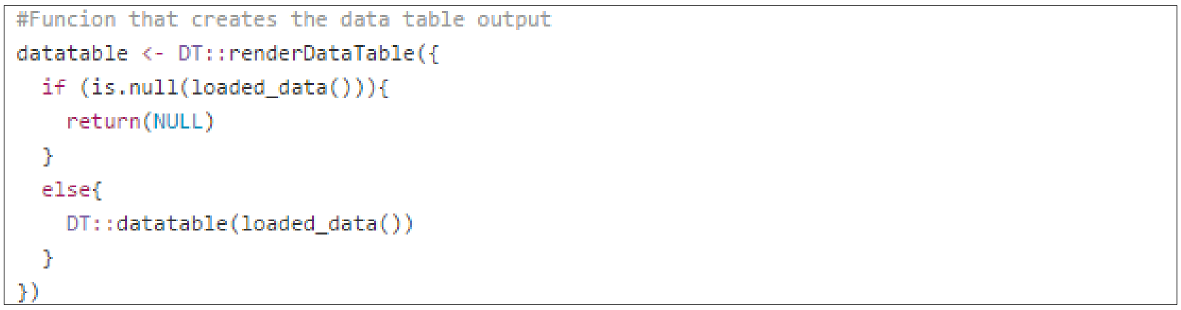


```
#Loaded data
loaded_data <- reactive({
  if(is.null(clipboard_data()) & is.null(sample_data_iris()) & is.null(sample_data_mtcars())){
    return(NULL)
  }
  else if(table_data$which == "clipboard"){
    return(clipboard_data())
  }
  else if(table_data$which == "sample_iris"){
    return(sample_data_iris())
  }
  else if(table_data$which == "sample_mtcars"){
    return(sample_data_mtcars())
  }
})
```

Figure 4.23. server-load_data_set.R (5)

The loaded_data function checks the value in table_data$which and depending on its value returns one data set or the other. It is an auxiliary function, to make all the code clearer. By this I mean that all this conditions could have been implemented in only one final function that created the output table. However, splitting the code makes it more readable and easier to follow.

Finally, the output table is created. This page only has one output, but other pages can have more of them.



```
#Funcion that creates the data table output
datatable <- DT::renderDataTable({
  if (is.null(loaded_data())){
    return(NULL)
  }
  else{
    DT::datatable(loaded_data())
  }
})
```

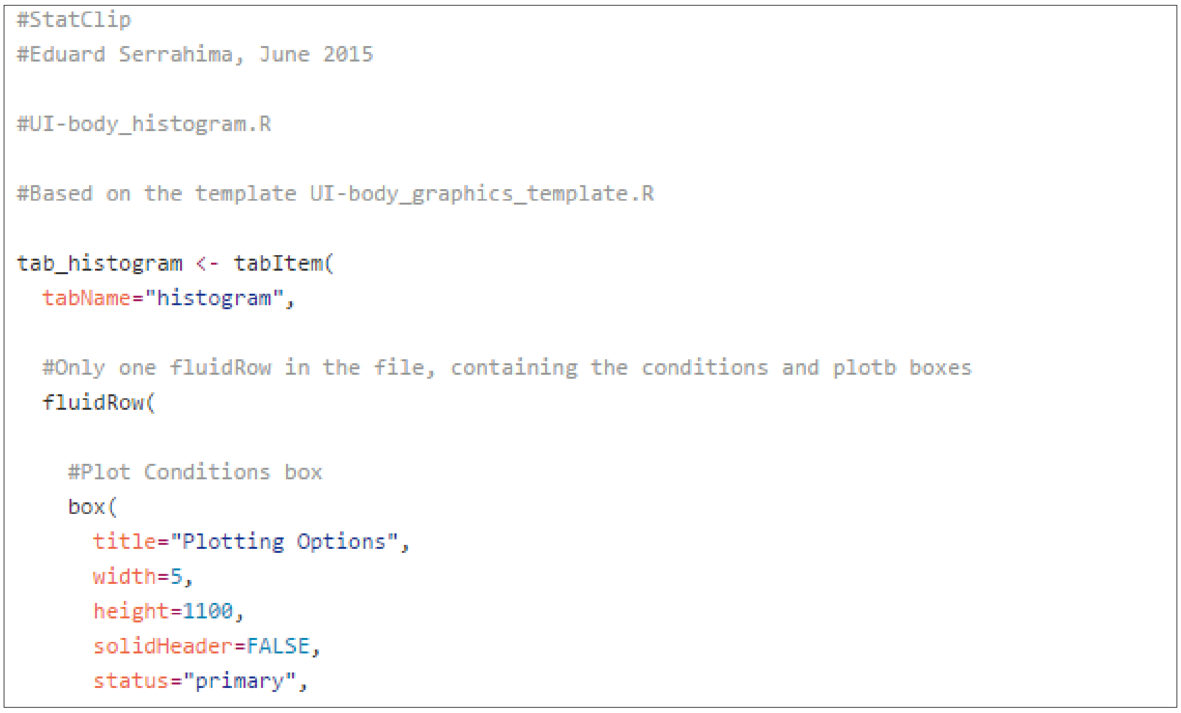
Figure 4.24. server-load_data_set.R (6)

Using the DT::renderDataTable function, we create the data table just taking the data in the auxiliary function just explained (Figure 4.23). Once this is done, the table is displayed in the UI.

4.3.2. Histogram

The second code example that will be detailed is the Histogram. The files analyzed will be UI-body_histogram.R and server-histogram.R.

4.3.2.1. User Interface



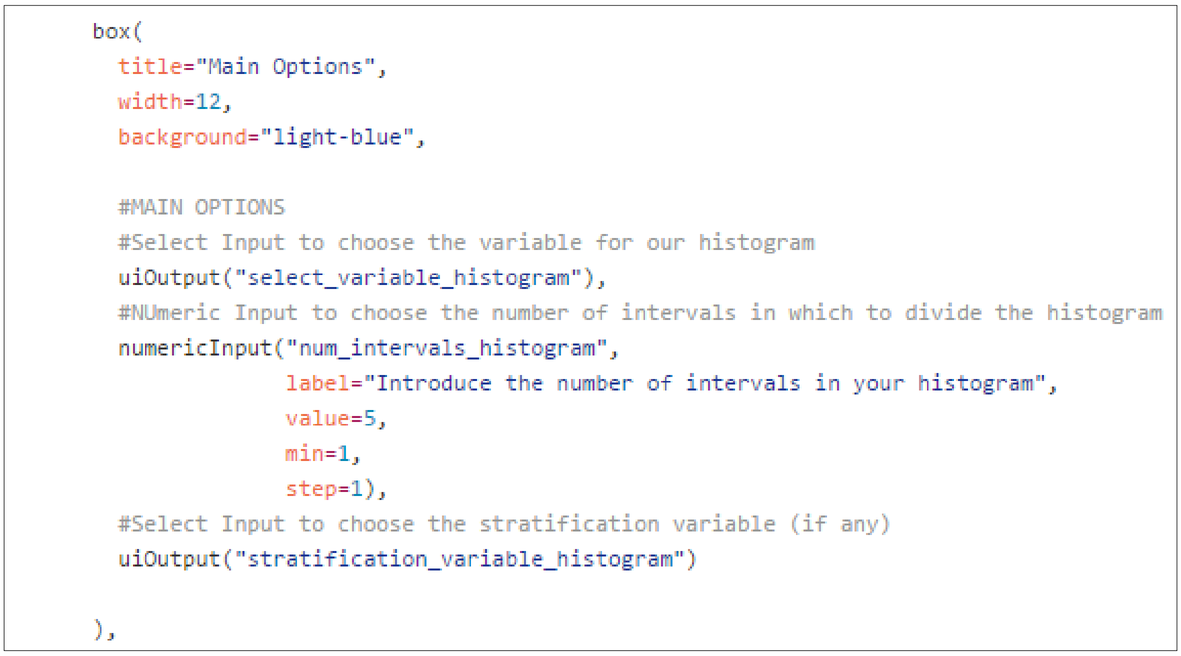
```
#StatClip
#Eduard Serrahima, June 2015

#UI-body_histogram.R

#Based on the template UI-body_graphics_template.R

tab_histogram <- tabItem(
  tabName="histogram",

  #Only one fluidRow in the file, containing the conditions and plotb boxes
  fluidRow(

    #Plot Conditions box
    box(
      title="Plotting Options",
      width=5,
      height=1100,
      solidHeader=FALSE,
      status="primary",
```

Figure 4.25. UI-body_histogram.R (1)

As we can see, the file starts with the tabItem function, with tabName "histogram". As stated in the script's comments, there is only one row in this file.

In the fluid row, we first see a box: the Plot Conditions box.

- Title: Plotting Options.
- Width = 5. This means the options' box is 5/12 of the total width of the page. The graph's box, then, will be 7/12.
- Height = 1100 pixels, enough for all the options to fit (this measure was decided trying different options until the most appropriate was found).
- SolidHeader = FALSE. Its effects have already been explained (Figure 4.14).
- Status = "primary" definesthe color, as we have already explained.

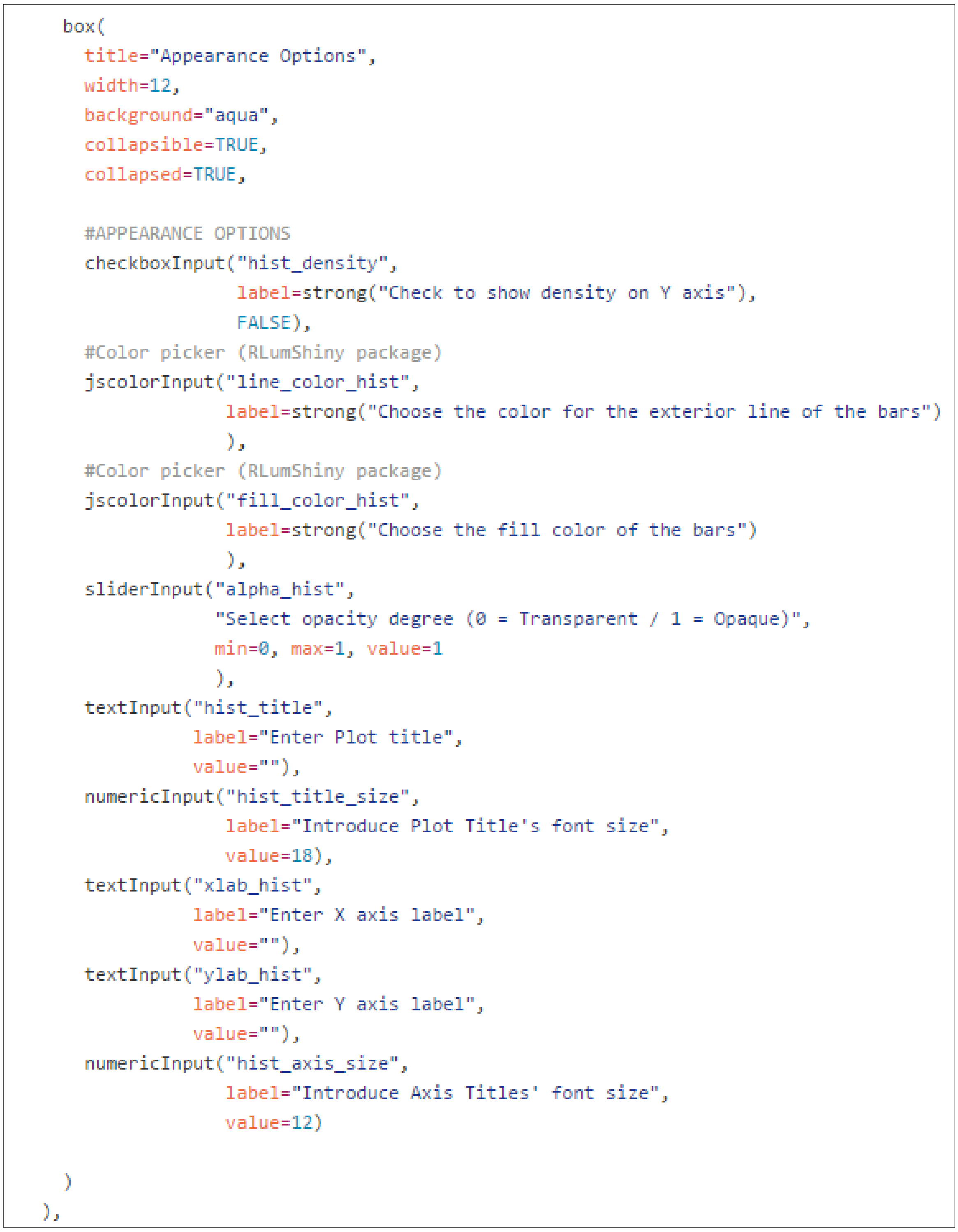


```
box(
  title="Main Options",
  width=12,
  background="light-blue",

  #MAIN OPTIONS
  #Select Input to choose the variable for our histogram
  uiOutput("select_variable_histogram"),
  #NUmeric Input to choose the number of intervals in which to divide the histogram
  numericInput("num_intervals_histogram",
               label="Introduce the number of intervals in your histogram",
               value=5,
               min=1,
               step=1),
  #Select Input to choose the stratification variable (if any)
  uiOutput("stratification_variable_histogram")

),
```

Figure 4.26. UI-body_histogram.R

In the general options box we now find another box: the Main Options Box. After defining the title, width (defined relatively, 12 means all the width of the first box) and background color, the main options are defined:

- In the first place, we can see a uiOutput. This is an output function (displays something programmed on the server side), but it displays an input. That is, what will be seen in the UI is a selectInput, but it is programmed on the server file. The reason behind this is that the list of options available on the selectInput is not fixed: the list of options is the list of variables of the working data. This makes it impossible to define the input directly from the UI file. We will see on the server explanation exactly how the input is defined.
- The second item that can be found is a regular input (numeric input, in this case). Its conditions are introduced normally.
- The third input has the same structure than the first one. Defined in the server due to its dependence on the data, we find a select input to choose a stratification variable.

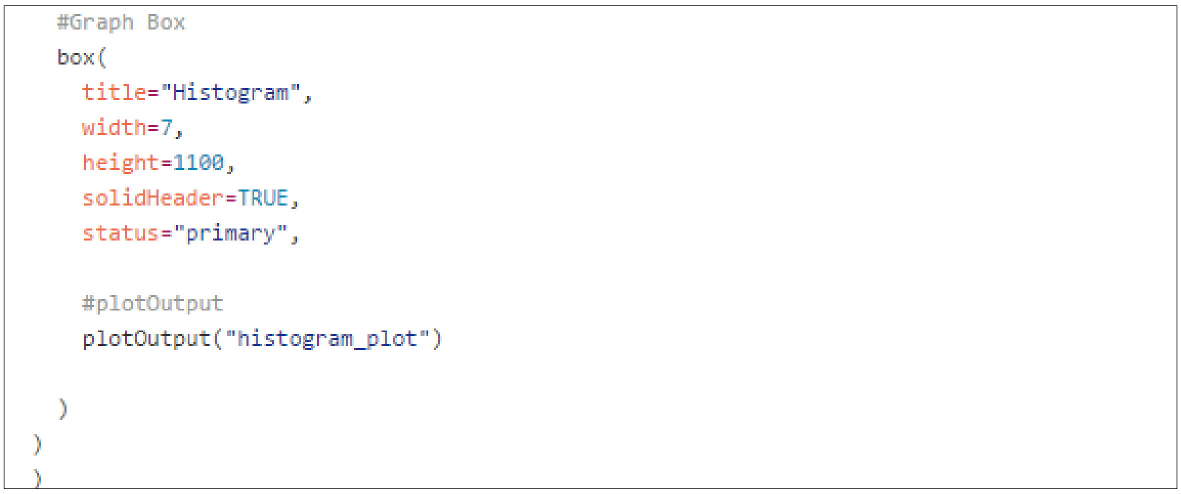


```
  box(
    title="Appearance Options",
    width=12,
    background="aqua",
    collapsible=TRUE,
    collapsed=TRUE,

    #APPEARANCE OPTIONS
    checkboxInput("hist_density",
                  label=strong("Check to show density on Y axis"),
                  FALSE),
    #Color picker (RLumShiny package)
    jscolorInput("line_color_hist",
                 label=strong("Choose the color for the exterior line of the bars")
                 ),
    #Color picker (RLumShiny package)
    jscolorInput("fill_color_hist",
                 label=strong("Choose the fill color of the bars")
                 ),
    sliderInput("alpha_hist",
                "Select opacity degree (0 = Transparent / 1 = Opaque)",
                min=0, max=1, value=1
                ),
    textInput("hist_title",
              label="Enter Plot title",
              value=""),
    numericInput("hist_title_size",
                 label="Introduce Plot Title's font size",
                 value=18),
    textInput("xlab_hist",
              label="Enter X axis label",
              value=""),
    textInput("ylab_hist",
              label="Enter Y axis label",
              value=""),
    numericInput("hist_axis_size",
                 label="Introduce Axis Titles' font size",
                 value=12)

  )
),
```

Figure 4.27. UI-body_histogram.R (3)

The second box contains the appearance options. All the inputs defined are directly coded in the UI file. Information on the different kinds of inputs can be found in RStudio’s shiny tutorial [31].

Something interesting about this second box is that, initially, it appears collapsed. This is accomplished due to two instructions introduced when defining the box’s conditions. As can be seen, “collapsible = TRUE” and “collapsed = TRUE” give the box this particular feature.

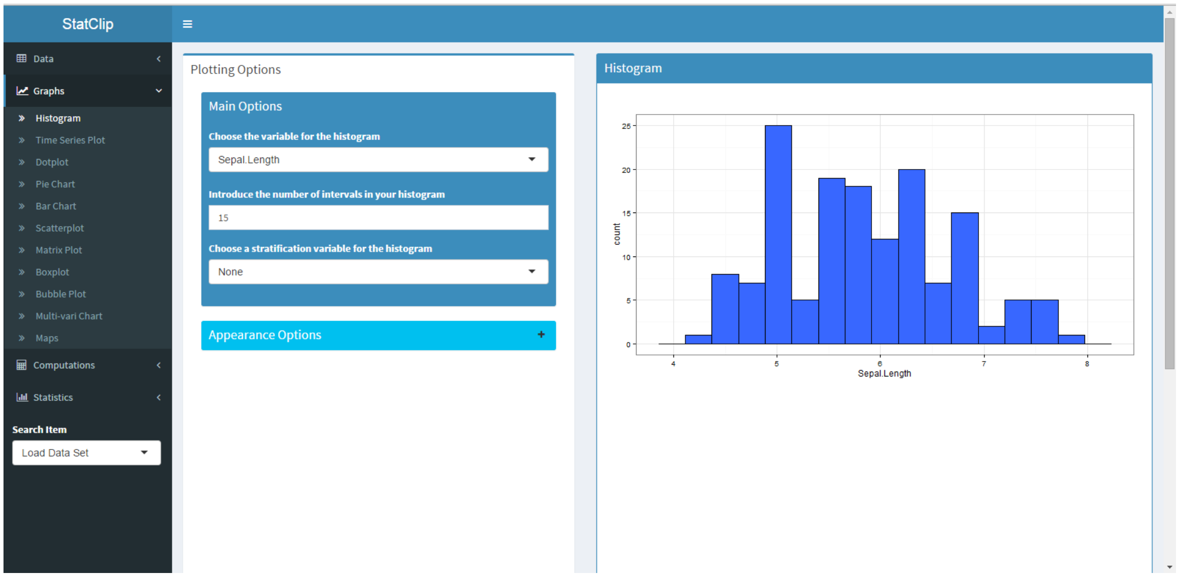


```
    #Graph Box
    box(
      title="Histogram",
      width=7,
      height=1100,
      solidHeader=TRUE,
      status="primary",

      #plotOutput
      plotOutput("histogram_plot")

    )
  )
  )
```

Figure 4.28. UI-body_histogram.R (4)

This final box is outside the Options one. It contains the actual graph. It has “width = 7”, so it occupies 7/12 of the total page width. After the regular instructions, we see a plotOutput function calling the plot designed in the server side.

The final appearance of the Histogram page can be seen in Figure 4.29.



Search Item: Load Data Set

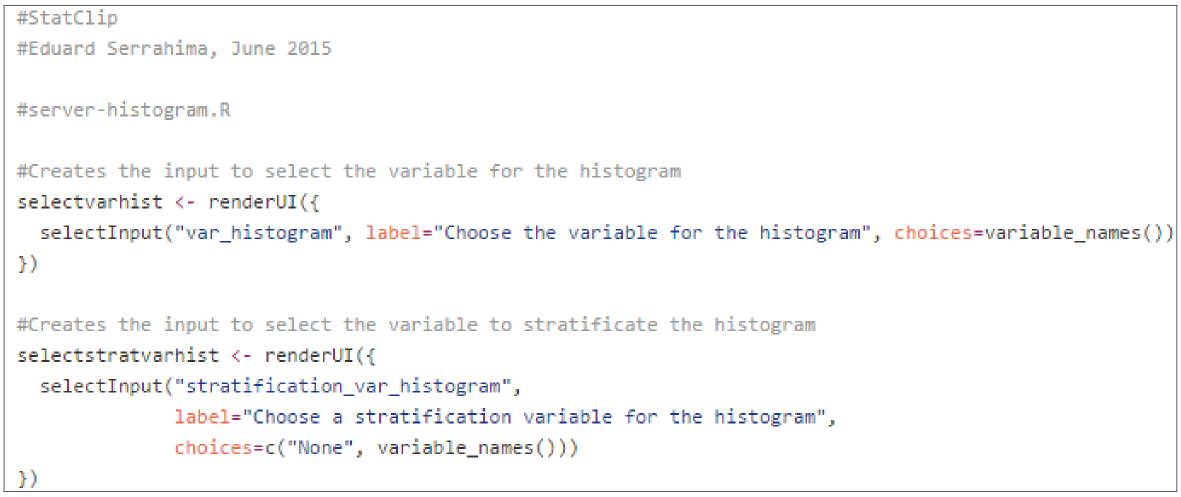
Choose the variable for the histogram: Sepal.Length

Introduce the number of intervals in your histogram: 15

Choose a stratification variable for the histogram: None

Figure 4.29. Histogram page User Interface

4.3.2.2. Server



```
#StatClip
#Eduard Serrahima, June 2015

#server-histogram.R

#Creates the input to select the variable for the histogram
selectvarhist <- renderUI({
  selectInput("var_histogram", label="Choose the variable for the histogram", choices=variable_names())
})

#Creates the input to select the variable to stratificate the histogram
selectstratvarhist <- renderUI({
  selectInput("stratification_var_histogram",
              label="Choose a stratification variable for the histogram",
              choices=c("None", variable_names()))
})
```

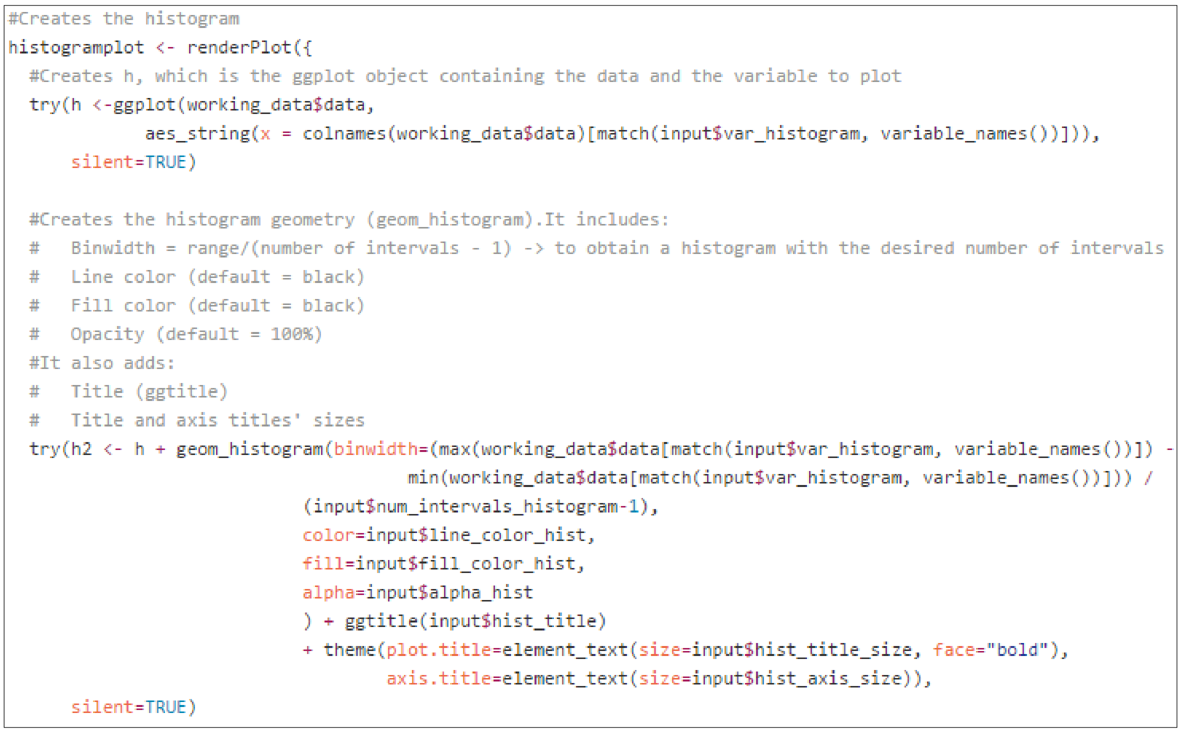
Figure 4.30. server-histogram.R (1)

What can be seen in Figure 4.30 are the functions to create the inputs for the User Interface. As can be seen, they are created in a renderUI function. This render function allows to create inside any kind of html object, while the other render functions are more specific.

Inside the renderUI functions, the inputs are created and displayed in the User Interface.

The choices in both inputs come from a function called variable_names(), which obviously contains the names of each variable in the working data set. In the second input (to select the stratification variable) the “None” option is added, to allow users to create non-stratified histograms.

In the following figures, the code that builds the actual histogram will be shown. It has been created using the package “ggplot2” [32]. This package, very powerful but not intuitive to use, bases its philosophy on adding “layers” to the plot. That is, in the first place, the ggplot object is created and, afterwards, the main features of the plot are added to the first object.



```
#Creates the histogram
histogramplot <- renderPlot({
  #Creates h, which is the ggplot object containing the data and the variable to plot
  try(h <-ggplot(working_data$data,
             aes_string(x = colnames(working_data$data)[match(input$var_histogram, variable_names())])),
      silent=TRUE)

  #Creates the histogram geometry (geom_histogram).It includes:
  #   Binwidth = range/(number of intervals - 1) -> to obtain a histogram with the desired number of intervals
  #   Line color (default = black)
  #   Fill color (default = black)
  #   Opacity (default = 100%)
  #It also adds:
  #   Title (ggtitle)
  #   Title and axis titles' sizes
  try(h2 <- h + geom_histogram(binwidth=(max(working_data$data[match(input$var_histogram, variable_names())]) -
                                      min(working_data$data[match(input$var_histogram, variable_names())])) /
                            (input$num_intervals_histogram-1),
                            color=input$line_color_hist,
                            fill=input$fill_color_hist,
                            alpha=input$alpha_hist
                            ) + ggtitle(input$hist_title)
                            + theme(plot.title=element_text(size=input$hist_title_size, face="bold"),
                                    axis.title=element_text(size=input$hist_axis_size)),
      silent=TRUE)
```

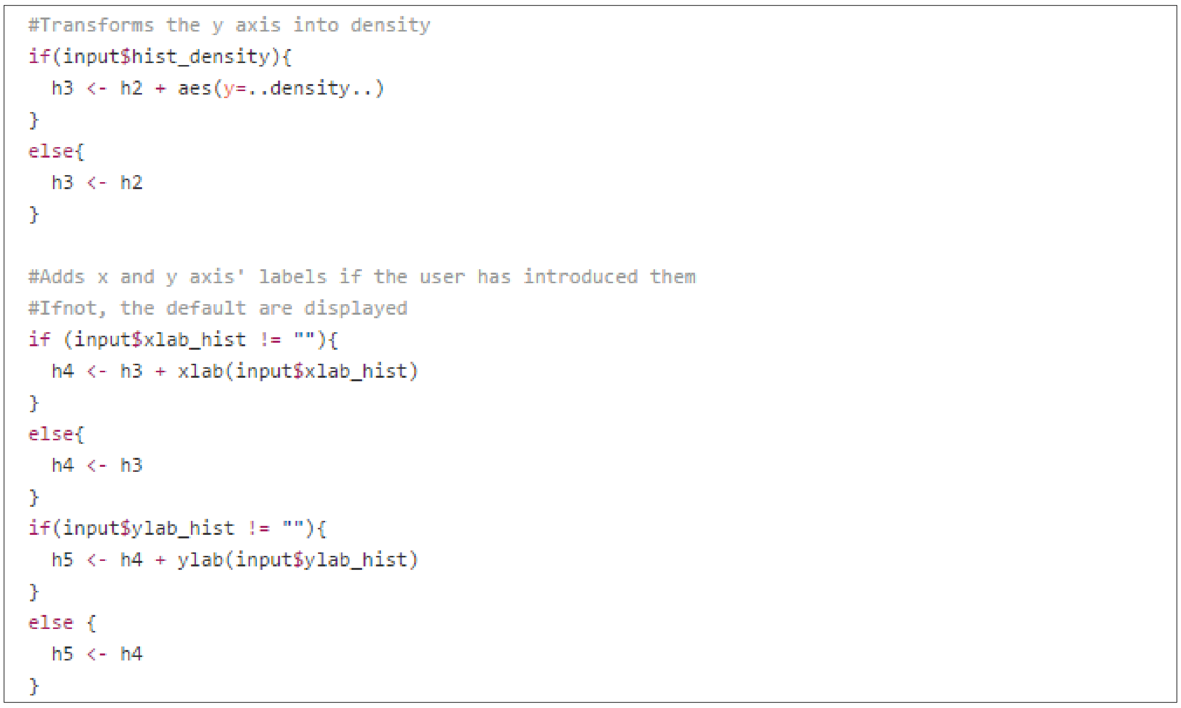
Figure 4.31. server-histogram.R (2)

In Figure 4.31 we can see:

- A renderPlot function that creates the plot to be displayed in the UI. Inside this reactive function we find:
  - We create “h”, a ggplot object containing the data to be plotted. The ggplot object contains all the data, but only plots the variable contained in the aes() function. The variable to be plotted is found matching the name coming from the input and the variable names’ list. By doing this, an index number is obtained, and the used to get the right column from the working data set.
  - Then, “h2” is created. To the original ggplot object we add:
    - The histogram geometry (geom_hist()). The bin width is calculated in the way stated in the code’s comments.
    - The title (ggtitle()).
    - A theme() to be able to interactively manipulate the title’s font size.

All the default conditions are explained in the code’s comments.

Again, try() functions have been used to try to avoid possible errors. As it has been said, this is just a prototype, and the code can be improved and debugged for a finer performance.

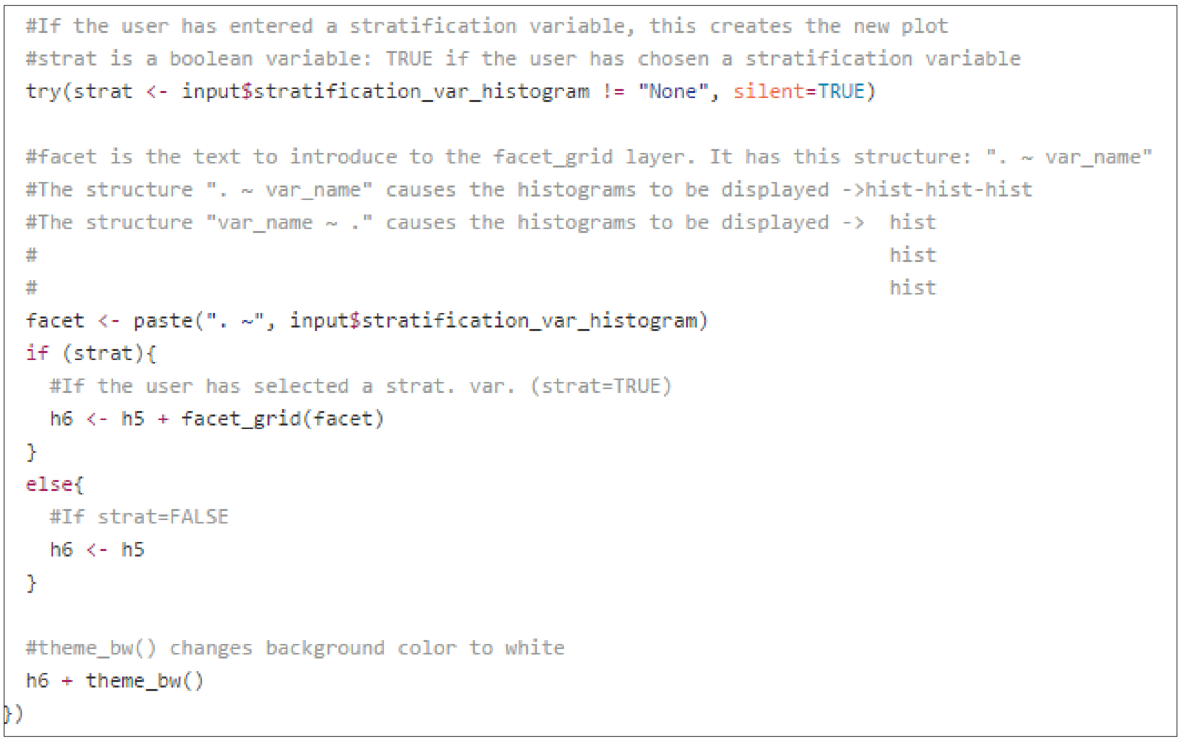


```
#Transforms the y axis into density
if(input$hist_density){
  h3 <- h2 + aes(y=..density..)
}
else{
  h3 <- h2
}

#Adds x and y axis' labels if the user has introduced them
#Ifnot, the default are displayed
if (input$xlab_hist != ""){
  h4 <- h3 + xlab(input$xlab_hist)
}
else{
  h4 <- h3
}
if(input$ylab_hist != ""){
  h5 <- h4 + ylab(input$ylab_hist)
}
else {
  h5 <- h4
}
```

Figure 4.32. server-histogram.R (3)

We can see in Figure 4.32 how the plot is modified by adding new layers. It is found:

- If the user has selected to show “density” instead of “count” on the Y axis of the histogram, an aes() function specifying this is added. If not, “h3” is equal to “h2”.
- If the user introduces Y and X axis’ names, they are added using xlab() and ylab() functions, getting to “h5”.



```
 #If the user has entered a stratification variable, this creates the new plot
 #strat is a boolean variable: TRUE if the user has chosen a stratification variable
 try(strat <- input$stratification_var_histogram != "None", silent=TRUE)

 #facet is the text to introduce to the facet_grid layer. It has this structure: ". ~ var_name"
 #The structure ". ~ var_name" causes the histograms to be displayed ->hist-hist-hist
 #The structure "var_name ~ ." causes the histograms to be displayed ->  hist
 #                                                                        hist
 #                                                                        hist
 facet <- paste(". ~", input$stratification_var_histogram)
 if (strat){
   #If the user has selected a strat. var. (strat=TRUE)
   h6 <- h5 + facet_grid(facet)
 }
 else{
   #If strat=FALSE
   h6 <- h5
 }

 #theme_bw() changes background color to white
 h6 + theme_bw()
})
```

Figure 4.33. server-histogram.R (4)

- In the first place, an auxiliary Boolean variable is created. “strat” is set to True if the user has selected a stratification variable. If not, it is set to False.
- Then, another auxiliary variable is created. “facet” adds the necessary string to the variable name for the plot to be stratified in the way we desire. In this case, and as it is explained in the comments, we add “. ~” for the stratified plots to be displayed one next to the other (Figure 4.34).
- After that, if auxiliary variable “strat” is True, a new layer is added to the plot, to create “h6”. The function facet() stratifies the histogram in the way defined in the variable “facet”.
- And, at last, a last layer is added to change the plot’s background to color white. The theme_bw() function is used for this purpose. The plot is finally finished.

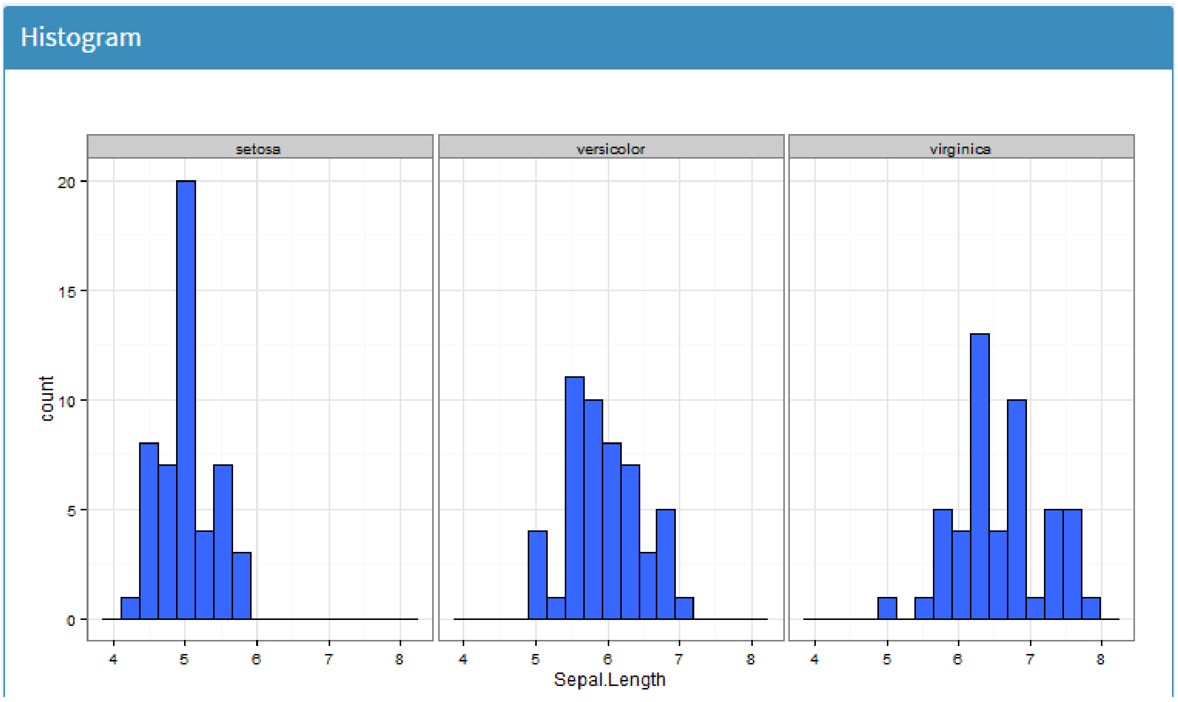


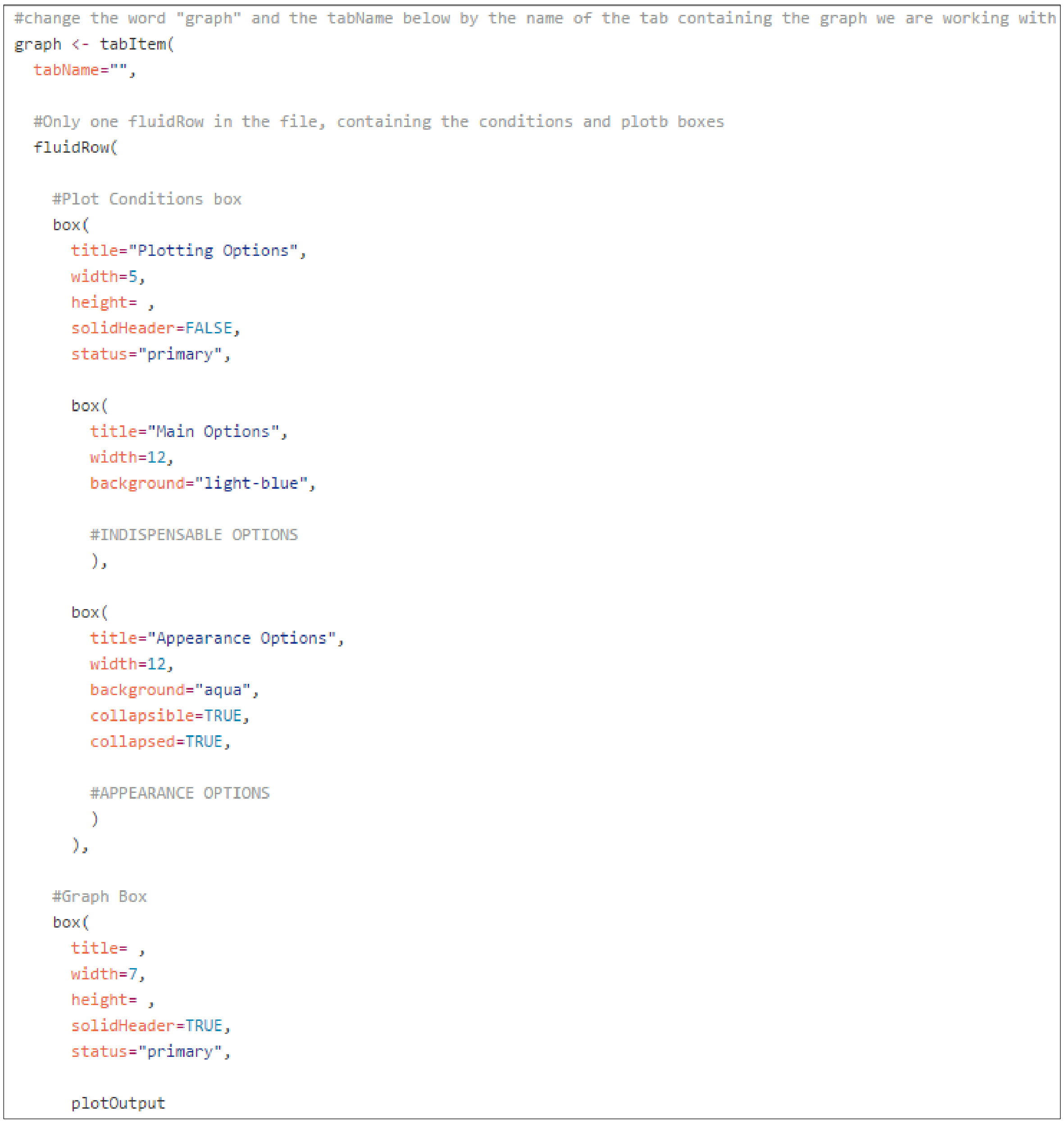
Figure 4.34. Stratified histogram

4.4. Graphics template

After commenting exhaustively these two code examples, to finish with this section it is important to explain another characteristic of StatClip's code.

As it has been said through all the project, StatClip is an app designed to be easily improved and expanded. Its modular design allows to easily add new pages without having to compromise the rest of the program.

To make this task even easier, it was decided to write a template for one of the most repeated user interfaces: graphics. The template helps avoid mistakes when designing the UI files for the newly added graphs. Even though it is simple, it can be really useful. Most of the graphs already implemented have been programmed using the UI-body_graphics_template.R.



```
#change the word "graph" and the tabName below by the name of the tab containing the graph we are working with
graph <- tabItem(
  tabName="",

  #Only one fluidRow in the file, containing the conditions and plotb boxes
  fluidRow(

    #Plot Conditions box
    box(
      title="Plotting Options",
      width=5,
      height= ,
      solidHeader=FALSE,
      status="primary",

      box(
        title="Main Options",
        width=12,
        background="light-blue",

        #INDISPENSABLE OPTIONS
        ),

      box(
        title="Appearance Options",
        width=12,
        background="aqua",
        collapsible=TRUE,
        collapsed=TRUE,

        #APPEARANCE OPTIONS
        )
      ),

    #Graph Box
    box(
      title= ,
      width=7,
      height= ,
      solidHeader=TRUE,
      status="primary",

      plotOutput
```

Figure 4.35. UI-body_graphics_template.R

5. Emotional design in StatClip: a kansei engineering approach

The visual features of StatClip were decided following parameters that seemed logical and reasonable to us. The intention was to design the graphical user interface according to the properties of the program and the users to whom StatClip is addressed.

However, StatClip needs to be more than just a functional software, and it is important that its visual appearance makes users feel what the program is intended to be: simple, powerful, fun…

To achieve this, it was decided to use emotional design techniques (more specifically, kansei engineering) to try to find which visual aspects could affect (and in which ways) the users' perception of StatClip.

5.1. An introduction to kansei engineering

This section is based on Lluis Marco's doctoral thesis, *Statistical Methods in Kansei Engineering studies* [33], especially on chapters 1, 2, 3 and 4.

Over the last decades, users of products and services have become more and more demanding. We do not only want products that satisfy our needs and work finely, but also products we like.

For many years, designers in general did not pay much attention to the emotional needs of customers, and only translated technical functionalities into parameters. However, this emotional aspects related to products or services cannot be eliminated from the equation. The massive success of some emotional products (such as the iPod, for example) has done nothing but confirm this tendency.

Decided and proved that the emotional part of a product should be taken into account, the problems arise when thinking how to incorporate it to the design of a product. Relying on the designer's intuition and creativity has been the most used option traditionally, but there are qualitative and quantitative methods to find information on how users perceive and use products and services. Most of these methods are grouped under what is called "emotional design".

Qualitative methods for emotional design, although they can give a lot of information, normally have several difficulties:

- Results from qualitative approaches depend a lot on the person leading the focus group or performing the interview.
- Qualitative approaches, especially interviews, require a lot of time. So usually only a few interviews are performed and conclusions are derived from asking a small amount of people.
- It can be difficult to obtain product design guidelines due to the fact that users are not typically thinking in a designer's paradigm.

Quantitative approaches such as questionnaires eliminate those difficulties but, obviously, have others. The main one, probably, is that they are by definition reductionist and, therefore, limited.

A quantitative method used in emotional design and mainly based in questionnaires is kansei engineering (KE). Kansei engineering's most important features are:

- In a kansei engineering study, the aim is to connect the physical properties of an object with emotions.
- In kansei engineering, there is an attempt to describe the whole range of emotions a product can convey. Not a unique response is modelled (such as the elements that make people prefer a watch over the others, for instance), but several responses (such as the elements that provoke that people perceive a watch as being modern, and elegant, and reliable, and so on). There are as many responses as necessary concepts to cover the whole range of expected emotions.
- Another important feature of KE studies is that they are based on collecting and analyzing quantitative data. Statistical or computational methodologies are usually used to find the relationship between properties and emotions.

KE is a multidisciplinary field, as it can be used in a wide variety of topics, and by scholars and professionals from diverse backgrounds. As written by Lluis Marco [33], some examples in which KE can be important may be:

- *"Engineers, designers, experts in ergonomics, usability and human-computer interaction, due to its deep link with industrial design.*
- *Neurologists, psychologists and experts in life sciences in general, because it deals with emotions and how we process them.*
- *Sociologists and economists, as emotional products have an impact in society both at a personal and social level.*
- *Statisticians, due to the need for collecting and analyzing data for formulating*

conclusions."

5.2. Model to perform a Kansei Engineering study

A model to carry on a KE study on a product can be divided in several steps [34]:

- In the first place, the product to be analyzed has to be described, as well as the people to whom is addressed and the market situation. In StatClip's case, this part has already been exhaustively explained throughout the whole project.
- Secondly, the semantic space has to be defined. This means collecting words that can describe the product emotionally. These words have to be retrieved from different sources. The initial amount of words (normally relatively large) is then reduced. In our project, affinity diagrams have been used with this purpose. The final reduced list contains what we call the Kansei words.
  In this stage, it is necessary to define the space of product properties. Some design attributes are chosen, and several possible values are considered for each attribute.
- Once the semantic and properties spaces are defined, building prototypes is necessary. These will be the products (the stimuli) for the KE study.
  Data is then collected, normally asking a group of people using questionnaires.
- In the synthesis stage, statistical methodologies are used to relate the space of properties to the semantic space. For every kansei word, product properties that affect it are found.
- To finish the process, the obtained results need to be checked and validated.

5.3. StatClip's Kansei Engineering Study

5.3.1. Semantic space definition

To get to the final list of Kansei words, it is firstly necessary to elaborate a larger list and then use methods to reduce the number of words to a more affordable one.

To set the initial list of Kansei words, it is necessary to look for words that describe the product in all sources possible, the more the better. Magazines, manuals, websites, books, etc. are good places to start on. The number of words at the end of this stage can be quite large. For Statclip, this number was 78.

Once this is done, it is necessary to reduce the list. For StatClip's case, we used affinity diagrams. They consist in grouping the words that have a similar meaning (usually an easy way to perform this task is writing down each word in a post-it and grouping the ones with a similar meaning). Once they are grouped, either the most representative word in the group is chosen or a new word that encloses all the words' meanings is picked.

In StatClip, the first affinity diagram led us to a 26 word list. We then repeated the same process to finally obtain a 16 word list. This process is represented in Figure 5.1.

Sexy	Elegant	Attractive		Attractive	Attractive
Striking	Encouraging	Impressive		Impressive	
Helps learning	Helpful	Intuitive		Intuitive	Easy to use
Assistive	With guidance			With guidance	
Friendly	Accessible	Easy to use	Straightforward	Easy to use	
Modern	Young	A whole new level		Modern	Modern
Funny	Colorful	Childish		Funny	Funny
Interactive	Dynamic	in real time		Interactive	Interactive
Powerful	Empowerful	Fast		Powerful	Powerful
Professional	Profitable	Successful		Professional	
Advanced Sopisticated	Technical	Challenging	Analytical	Advanced	
Clean Simple	Polished	Neat	Clear	Neat	Neat
Customized	Personalized			Customized	Flexible
Flexible	Extensible			Flexible	
Public	Free			Free	Free
Multi-platform Available	Anywhere Anytime	Mobile	Online	Always available	Always available
Shareable Social	Allows Communication	Collaborative	Everything linked	Social	Social
Informative	Communicative			Communicative	
Complete	Big			Complete	Trustworthy
Functional	Accurate	Good Quality		Functional	
Removes any doubt	Effective	Trustworthy		Trustworthy	
Comprehensive	Rational			Rational	Rational
Tool of choice	Adequate			Adequate	
Unique	Important			Unique	Unique
Visual	Pictorial			Visual	Visual
Automatic	Smart	Single click		Automatic	Automatic

Figure 5.1. Affinity diagrams

However, 16 words is still a large number for a kansei study, as participants have to rate each word in each image. Moreover, not all 16 words are really interesting for the study. It was decided, then, to select from these 16 words the ones that were the most relevant for StatClip.

The final 8 kansei words to use in the questionnaires were:

- Easy to use.
- Funny.
- Interactive.
- Powerful.
- Attractive.
- Flexible.
- Neat.
- Modern.

5.3.2. Properties' space definition

The properties' space determines the product factors that will be studied. When encountering the definition of the properties' space, we can find two situations: that the study is based on already existing prototypes, or that new prototypes are built specifically for the experiment. This last option, of course, gives the study much more flexibility.

For StatClip, the decision was to perform the study not on the actual software, but presenting variations of a screenshot. The page selected for the KE study was the Histogram tab.

The factors chosen for the KE study were:

Table 5.1. KE study factors

Factors	Levels
Color	Blue
	Red
	White
Collapsed Menu	Collapsed
	Non Collapsed
Results position	Left
	Right

The color factor's levels do not mean that the whole page is the same color, but that it is a combination of the level color and white.

The menu can be visible (on the left side of the page) or completely collapsed and, therefore, not visible on the “histogram” page.

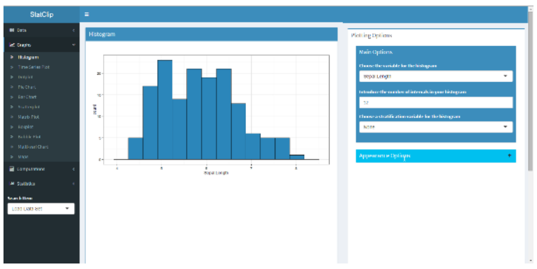
The third level is referred to the position of the results box. In this project, the order in principle established was to locate options on the left side and results on the right. This factor has two levels: one maintaining this original disposition, and a second one changing it.

As we see in the table, we have a factor with three levels and two more with two levels each. We could have, then, 12 different combinations possible. However, to find the main effects of each factor it is not necessary to try them all. In fact, it will be enough with 8 of them. The selection of the factor levels was done with the SPSS statistical Software, using the Generate orthogonal design menu option. Table 5.2 shows the final distribution.

Table 5.2. Characteristics of each prototype

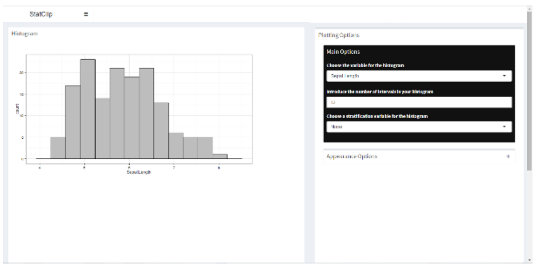
	Color	**Results box**	**Menu**
1	blue	left	non collapsed
2	white	left	collapsed
3	white	right	non collapsed
4	blue	right	collapsed
5	red	right	collapsed
6	red	left	non collapsed
7	blue	right	non collapsed
8	blue	left	collapsed

In Figure 5.2 one can see all the combinations defined in Table 5.2. These image are the ones that were shown to the participants during the questionnaire.



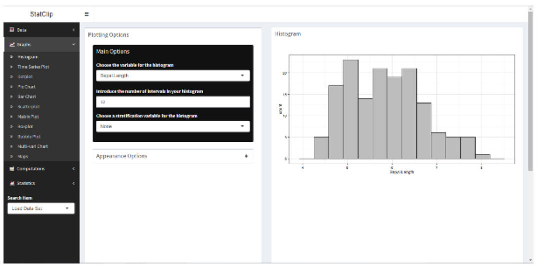
Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 1



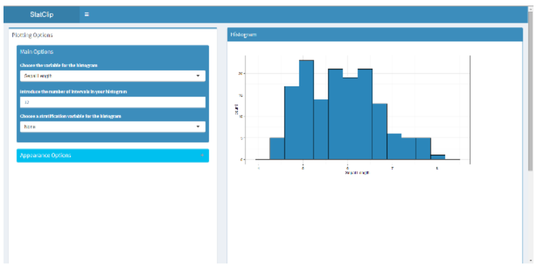
Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 2



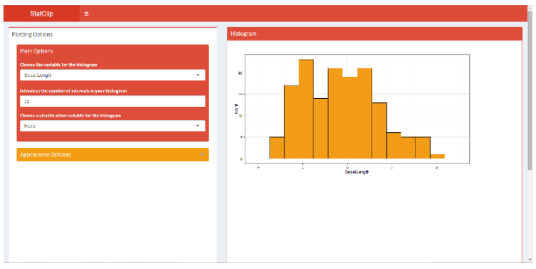
Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 3



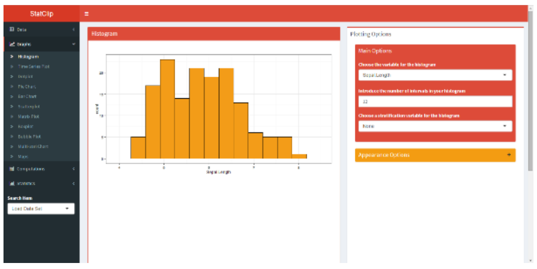
Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 4



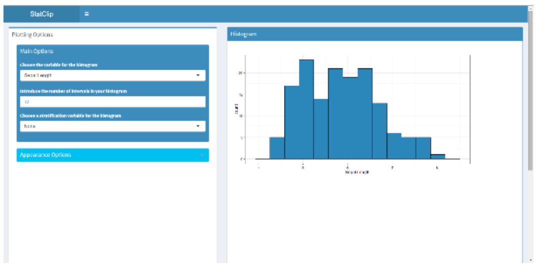
Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 5



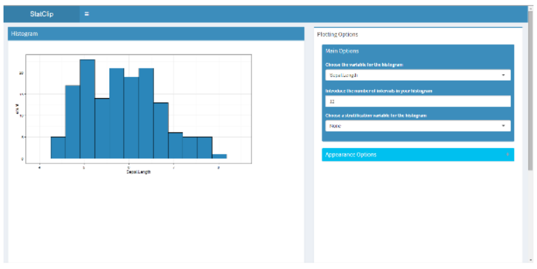
Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 6



Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 7



Choose the variable for the histogram: Sepal.Length
Introduce the number of intervals in your histogram: 12
Choose a stratification variable for the histogram: None

Image 8

Figure 5.2. KE study prototype images

5.3.3. Data Collection

Collecting the data in a correct way is really important when performing a KE study. If the data is not reliable, so will be the results obtained from it no matter how complex and advanced the statistical techniques later applied are.

In the first place, questionnaires had to be designed. For our kansei engineering study it was decided to establish a 1 to 7 punctuation for each kansei word (as can be seen in Figure 5.3).

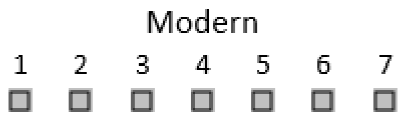


Figure 5.3. Questionnaire example

The next decision to be made was how to present the questions. That is:

- Each participant is presented with a product. He or she then rates this product on all the Kansei words.
- Each participant is presented with a kansei word. He or she then rates all products on this kansei word.

The most logical and natural way is the first one. Moreover, it accelerates the process, as participants only have to see each image once. In our KE experiment, then, the first option was used: each participant had to rate all Kansei words for one image before jumping to the following one.

Once this had been decided, it was time to collect the actual data. Kansei words were randomized for each image, so participants did not have the words in the same order for the whole process. In the same way, the display order of the images was randomized and each participant had the pictures in a different order.

Each participant did the questionnaire in front of a computer, where the images were displayed, and rated all the kansei words consecutively.

Table 5.3 shows the names, age and gender of the participants in the study.

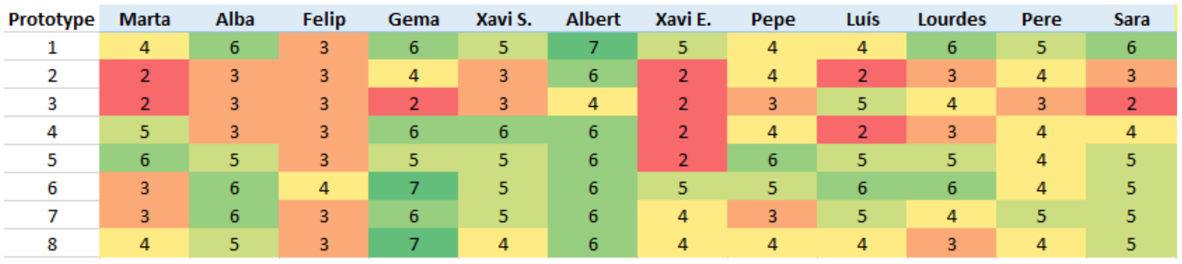
Table 5.3. Names, Age and Gender of the participants in the study

Name	Age	Gender
Pepe	21	Male
Felip	55	Male
Gema	52	Female
Marta	21	Female
Xavi	22	Male
Albert	21	Male
Xavier	21	Male
Luís	21	Male
Alba	23	Female
Lourdes	37	Female
Pere	58	Male
Sara	30	Female

5.3.4. Synthesis

Once the results are collected, a first observation already leads to some conclusions. As an example, the results for the word "Funny" (Figure 5.4). Cells are colored in a scale, from red in the lower values to green in the higher ones.

It can be easily observed that some participants had a tendency to use higher marks (for example, Albert) while others punctuated in general low marks (for example, Felip). Other participants had more variations in their results.



Prototype	Marta	Alba	Felip	Gema	Xavi S.	Albert	Xavi E.	Pepe	Luís	Lourdes	Pere	Sara
1	4	6	3	6	5	7	5	4	4	6	5	6
2	2	3	3	4	3	6	2	4	2	3	4	3
3	2	3	3	2	3	4	2	3	5	4	3	2
4	5	3	3	6	6	6	2	4	2	3	4	4
5	6	5	3	5	5	6	2	6	5	5	4	5
6	3	6	4	7	5	6	5	5	6	6	4	5
7	3	6	3	6	5	6	4	3	5	4	5	5
8	4	5	3	7	4	6	4	4	4	3	4	5

Figure 5.4. Punctuations for Kansei Word "funny"

At this stage, it is intended to find the relationship between the semantic space and the product properties, in order to extract conclusions that can help improve StatClip. Finding which properties affect the most each Kansei word, as well as the relationship between different kansei words can lead to finding which conditions would give StatClip the appearance that best suits our ideas.

In the first place, it was decided to go with a principal components analysis (PCA), which will be explained in Section 5.3.4.1.

5.3.4.1. Principal Components Analysis

The basic objective of a principal components analysis is to reduce the number of variables. This is done computing new variables that depend on the original ones. The new variables summarize the information contained by the original ones.

To find these new variables, we seek linear combinations of the original variables.

$$\begin{gathered} Z_1 = a_{11}X_1 + a_{12}X_2 + \cdots + a_{1p}X_p \\ Z_2 = a_{21}X_1 + a_{22}X_2 + \cdots + a_{2p}X_p \\ \vdots \\ Z_p = a_{p1}X_1 + a_{p2}X_2 + \cdots + a_{pp}X_p \end{gathered}$$

These combinations must accomplish:

- **Z_1, Z_2 … Z_p** are uncorrelated (orthogonal).
- Maximum **var (Z_1).**
- **var (Z_1) > var (Z_2) > … > var (Z_p)**. This condition is useful to reduce dimensionality.
- As a condition:

$$a_{11}{}^2 + a_{12}{}^2 + \cdots + a_{1p}{}^2 = 1$$

To find the unknown values (**a_{ij}**, **Z_i** and **var(Z_i)**), we must know that:

- **a_{ij}** are the eigenvectors of the covariance or correlation matrix.
- **Var (Z_i)** correspond to the eigenvalues (**λ_i**) of the correlation matrix.

It can be demonstrated that:

$$\sum_{i=1}^{p} var\ (X_i) = \sum_{i=1}^{p} var\ (Z_i) = \sum_{i=1}^{p} \lambda_i$$

However, this methodology will lead us to finding as many Z components as variables in the original set. It is important, then, to follow a criteria to select the principal components. For example:

- Keep components that capture (together) approximately around 80-90% of the variability.

- Keep as many components as eigenvalues bigger than 1 (**$\lambda_i > 1$**), for this means that the component captures more variability than the original variables.

Once this is done, we proceed to analyze the results mainly interpreting the graphics of the analysis.

In our study, a table with the mean values for each kansei word and image was created (Figure 5.5).

Color	Graph.position	Menu	Attractive	Modern	Funny	Interactive	Flexible	Powerful	Easy	Simple
blue	left	non.collapsed	5.42	5.42	5.08	5.58	5.58	6.17	5.25	4.82
white	left	collapsed	3.58	3.58	3.25	4.08	3.92	4.08	5.42	5.58
white	right	non.collapsed	3.83	4.17	3.00	5.08	5.08	5.67	4.08	4.75
blue	right	collapsed	5.00	4.83	4.00	4.42	4.00	4.42	5.50	5.33
red	right	collapsed	4.67	4.83	4.75	4.67	4.33	4.00	5.42	5.67
red	left	non.collapsed	5.00	5.00	5.17	5.83	5.58	5.67	5.17	4.33
blue	right	non.collapsed	5.58	5.92	4.58	5.58	5.58	6.27	4.83	4.50
blue	left	collapsed	4.92	4.58	4.42	4.92	4.27	4.50	5.67	5.25

Figure 5.5. Principal Components Analysis data table

Using Minitab we performed the principal components analysis. In the first place, we obtained the analysis of the eigenvectors and eigenvalues of the correlation matrix.

```
Eigenanalysis of the Correlation Matrix

Eigenvalue  5,2556  2,1171  0,3629  0,1586  0,0803  0,0214  0,0040  0,0000
Proportion   0,657   0,265   0,045   0,020   0,010   0,003   0,000   0,000
Cumulative   0,657   0,922   0,967   0,987   0,997   1,000   1,000   1,000
```

As can be seen, the two previously stated criteria led to select two principal components:

- The first two components capture 92.2% of the variability.
- There are only two eigenvalues higher than one.

To analyze the results, a good option is to use the “components plot”. It plots the original variables in a graphic which has the two principal components as X and Y axis. This leads to discover which variables are perceived as the most similar. Figure 5.6 shows the components plot for our study (the plot has been slightly rotated to improve interpretability).

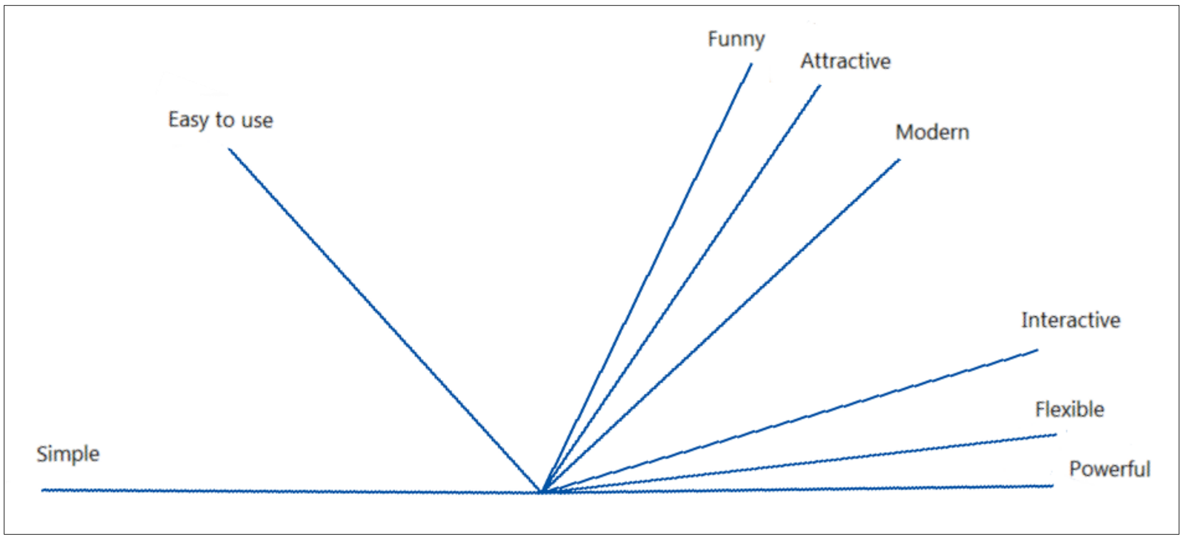


Figure 5.6. Components Plot

Looking at the graph it is possible to extract several conclusions. In the first place, we can name the principal components looking at the location of the different kansei words in the graph. The X axis corresponds to the component "Simple – Powerful", while the Y axis could be named "Visual Properties", for example, as the words close to this axis have more relationship with these properties. The graphic, then, would result:

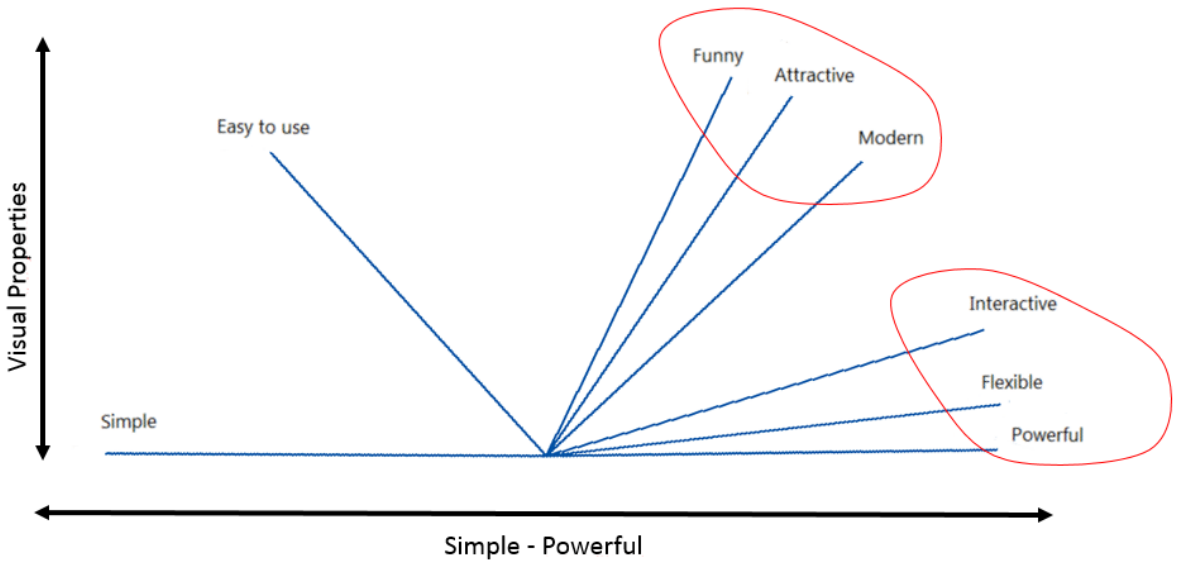


Figure 5.7. Components plot

As can be seen, the words that are close to each other in the graphic have been grouped. The fact that they are that close responds to a simple reason: they are perceived as similar.

It can be seen, as well, that the "Powerful – Flexible – Interactive" group is perceived as opposite to “Simple”. “Funny”, “Attractive” and “Modern” are as well perceived similarly, but are fairly independent from the “Powerful” group, as well as “Simple”, as they are orthogonal to them.

We can see that “Easy to use” is a word that affects both principal components. The result, when we think about it, is logical, as the “Easy to use” perception is related to both simplicity perception and visual aspects.

5.3.4.2. Quantification Theory type 1 (QT1)

An exhaustive theoretical explanation in this topic can be found in Lluis Marco’s doctoral thesis *Statistical Methods in Kansei Engineering Studies* [33], p. 161-172. Here we provide a brief introduction to the topic.

QT1 is a methodology that can be applied to try to quantify the relationship between the response (kansei words) and the properties (design elements). It is simply a multiple regression model, but with some different attributes that make it particularly appropriate for KE studies. Properties (normally called factors in factorial designs) are here named “items”, and the factor’s levels are here called “categories”.

The situation, then, is that all regressors will be qualitative, while the response (the mean of all participants’ rating to a kansei word) will be quantitative.

Being all the regressors qualitative, they are introduced in the model using dummy variables; that is, creating new variables that have value one when a product has a determined category in an item. Here you can see an example:

Table 5.4. Example of dummy variables

Graph Position	X_{11} *(left)*	X_{22} *(right)*
left	1	0
left	1	0
right	0	1
right	0	1
right	0	1
left	1	0
right	0	1
left	1	0

The idea behind QT1 is to estimate the β_{jk} coefficients in the following equation:

$$Y_i = \beta_0 + \sum_{j=1}^{R} \sum_{k=1}^{C_j} \beta_{jk}\, \delta_{i(jk)} + \varepsilon_i$$

These β_{jk} are called Category Scores in QT1. To estimate them, the usual procedure is to use the least squares method. The proposed coefficients are then those that minimize the sum of squared residuals, a residual being the difference between an observed value and the value given by the model. In consequence, coefficients can be found solving the following equation:

$$b = (X'X)^{-1}X'Y$$

However, when trying to apply the given equation, we encounter a problem. If matrix *X* is constructed using all columns corresponding to the dummy variables, the product *X'X* is a singular matrix and, therefore, has no inverse.

To solve this issue, the classical approach is to use all columns of a given factor but one. This means that, for example, for an item with two categories, only one column will be used for this item. If it had three categories, two columns would be used. Using this approach gives us a solution that has coefficient 0 for the unused columns (reference levels). This, while correct, is not comfortable to interpret the results, and makes the process more difficult.

To deal with this, one has to take into account that the normal equations form a compatible indeterminate system. That is, an equation system with infinite solutions. QT1 approach, then, is to use a solution that has no reference levels.

This is accomplished applying a mathematical transformation, with the goal to obtain a final regression equation with no reference levels and that, obviously, gives the same predictions than the original one.

The traditional output for a QT1 analysis is completed with multiple (MCC) and partial correlation coefficients (PCC). PCC's are of particular importance, as they quantify the strength of the relationship between items and Kansei words.

The square of the multiple correlation coefficient (MCC) is simply the coefficient of determination R^2 in multiple regression, which can be calculated as usually:

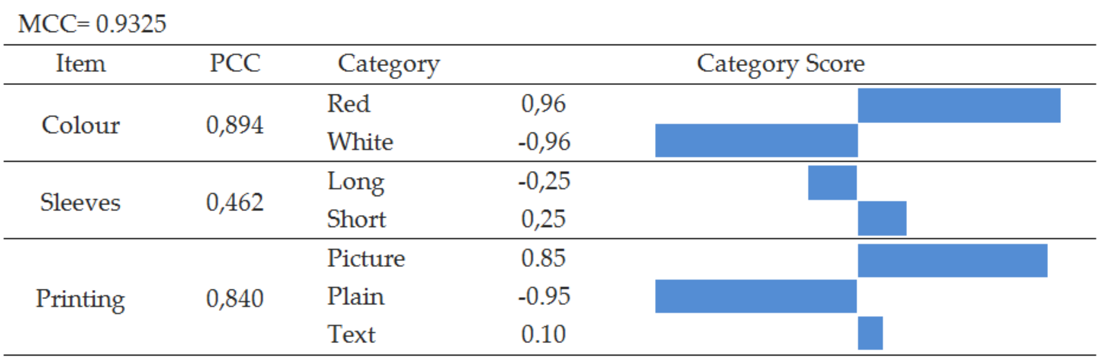
$$R^2 = 1 - \frac{SS_{err}}{SS_{tot}}$$

Where SS_{err} is the residuals sum of squares, and SS_{tot} is the total sum of squares. MCC gives information on the global contribution of the space properties to each kansei word.

The PCC can be calculated using the following procedure, extracted from Tanaka's *Review of the Methods of Quatification* [35]:

- Scores for each item are computed using the category scores b_{jk}' from the regression model. That is $W_{i(j)} = \sum_{k=1}^{C_j} b'_{jk}\, \delta_{i(jk)}$ with *j* meaning the *j-th* item and *k* the *k-th* category in the *j-th* item.
- The regression of *Y* with $W_1, W_2, \dots, W_{j-1}, W_{j+1}, \dots, W_R$ as regressors is performed and residuals (R_{es1}) are stored.
- The regression of W_j with $W_1, W_2, \dots, W_{j-1}, W_{j+1}, \dots, W_R$ as regressors is performed and residuals (R_{es2}) are stored.
- The correlation of residuals R_{es1} and R_{es2} is the PCC for item *j*.

The results are normally shown in a graphic with bars, to easily interpret them. In Figure 5.8 we can see an example.


MCC= 0.9325

Item	PCC	Category	Category Score
Colour	0,894	Red	0,96
		White	-0,96
Sleeves	0,462	Long	-0,25
		Short	0,25
Printing	0,840	Picture	0.85
		Plain	-0.95
		Text	0.10

Figure 5.8. QT1 example results (extracted from [33], p. 167)

In this figure we can easily see that "Color" and "Printing" affect the response. However, it is not that clear if "Sleeves" has any effect on it as well.

Lluis Marco proposes in his doctoral thesis [33] an improved version of QT1 that includes in its results a *p-value* indicating whether the item affects the response or not. All the process for obtaining these *p-values* is exhaustively described in the just referred doctoral thesis.

We have used this last methodology to analyze the data of our KE study. The results for the eight kansei words in our study are here presented.

Attractive

```
QT1 Analysis for ATTRACTIVE
MCC= 0.9935  MCC^2= 0.9870  p-value= 0.003674

                    PCC  F stat.  p-value
Color          0.99276 102.5225 0.001732 **
Graph.position 0.25594   0.2103 0.677717
Menu           0.93967  22.6364 0.017627 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

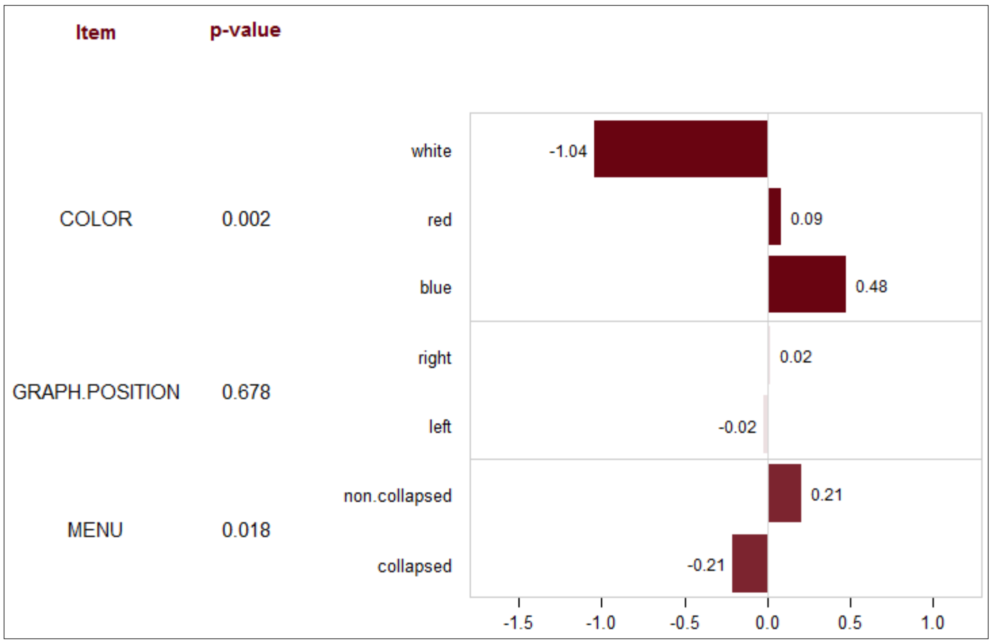


Figure 5.9. QT1 results for Attractive

We can see that, clearly, “Color” and “Menu” affect the Kansei word. More specifically, we see that selecting color Blue, the prototype is perceived as more attractive, while the opposite happens with choosing White. With the Non-collapsed menu, the prototype is as well perceived as more attractive.

Modern

```
QT1 Analysis for MODERN
MCC= 0.9719  MCC^2= 0.9446  p-value= 0.03155

                   PCC F stat.  p-value
Color          0.95972 17.5022 0.02218 *
Graph.position 0.67871  2.5622 0.20776
Menu           0.90480 13.5440 0.03475 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

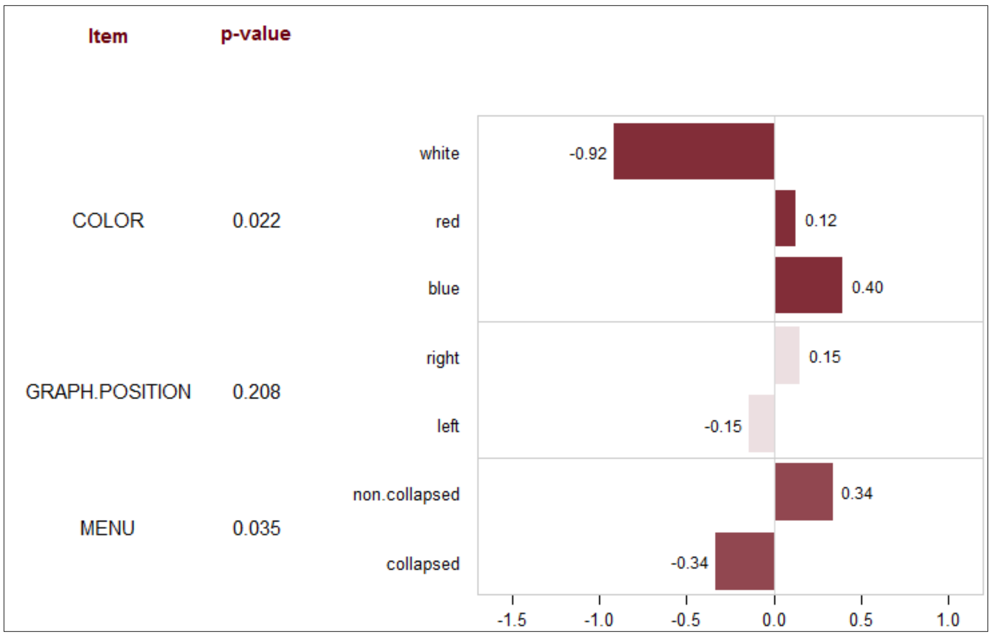


Figure 5.10. QT1 results for Modern

We can see that, again, the graph position has no effect on the response. In the “Color” side, Red and Blue are perceived as more modern (Red in a lighter way) while White is strongly perceived as not modern. The Collapsed menu is perceived as less modern than the Non-collapsed option.

Funny

```
QT1 Analysis for FUNNY
MCC= 0.9831  MCC^2= 0.9664  p-value= 0.01508

                    PCC F stat.  p-value
Color          0.98063 37.5995 0.007514 **
Graph.position 0.82126  6.2156 0.088241 .
Menu           0.78719  4.8880 0.114008
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

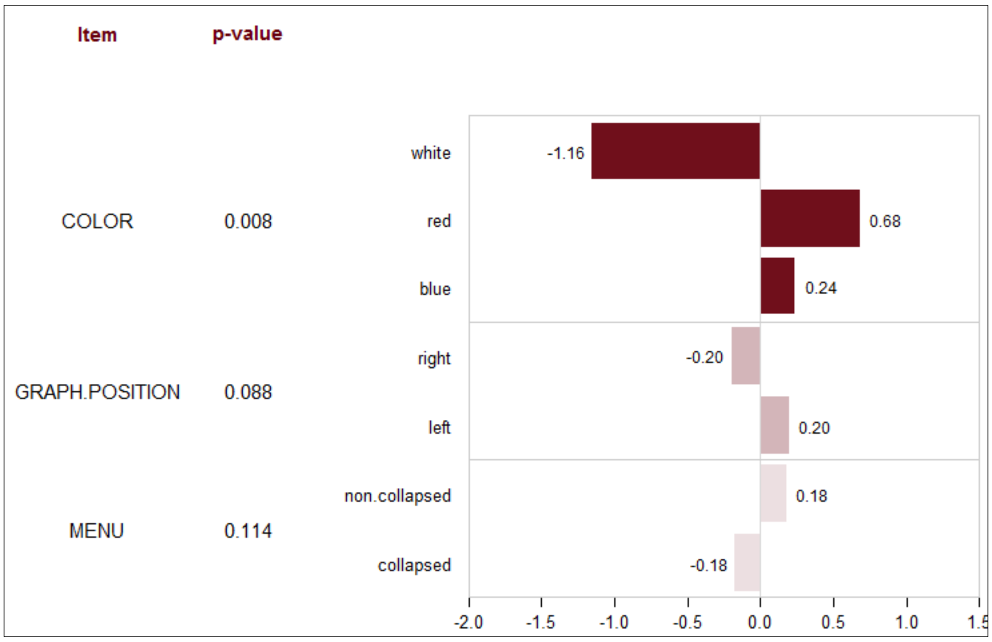


Figure 5.11. QT1 results for Funny

In this case, the “Color” is the only item that has an effect on the prototype perceived as Funny. As we can see, being the prototype Red leads to perceive StatClip’s visual appearance as funnier. Blue also has a positive effect (although smaller), and White has a big negative effect on this Kansei word. Red being the “funnier” color follows a certain logic: red is a warm color, and it transmits a sense of energy and warmness that Blue and White do not have.

Interactive

```
QT1 Analysis for INTERACTIVE
MCC= 0.9827  MCC^2= 0.9657  p-value= 0.01557

                     PCC  F stat.  p-value
Color          0.92443  8.8146 0.055458 .
Graph.position 0.61101  1.7872 0.273604
Menu           0.97769 64.9907 0.003987 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

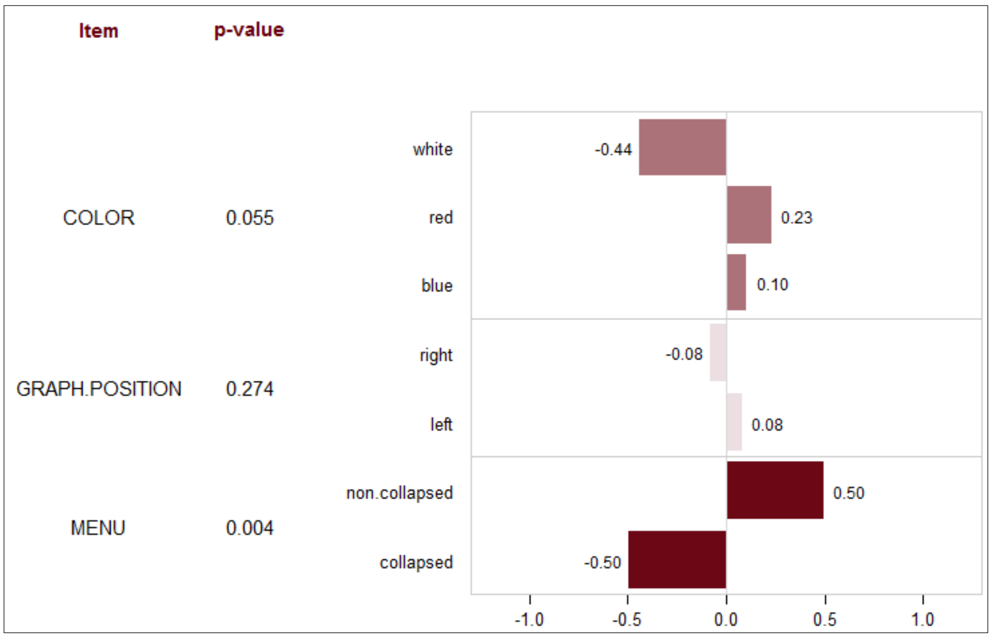


Figure 5.12. QT1 results for Interactive

In this case we see that clearly “Menu” affects the Interactive perception, and that the Graph Position has no effect on it. However, the color’s *p-value* is very close to 0.05, and could be counted as significant.

Flexible

```
QT1 Analysis for FLEXIBLE
MCC= 0.9933  MCC^2= 0.9866  p-value= 0.003835

                     PCC   F stat.   p-value
Color            0.90831   7.0727 0.0731916 .
Graph.position 0.49072   0.9515 0.4012702
Menu             0.99281 206.2408 0.0007318 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

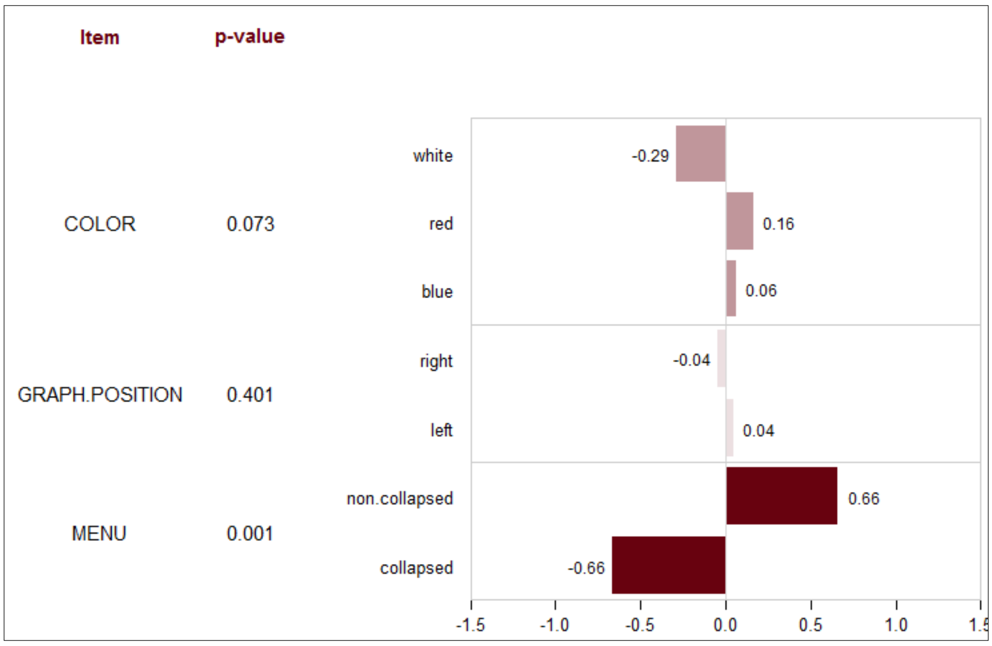


Figure 5.13. QT1 results for Flexible

The only item that has an effect on the Kansei word “Flexible” is the Menu. When Non-collapsed, StatClip is perceived as more flexible.

Graph Position and Color have no significant effect on the response in this case.

Powerful

```
QT1 Analysis for POWERFUL
MCC= 0.9986  MCC^2= 0.9971  p-value= < 0.001

                    PCC  F stat.  p-value
Color           0.98166  39.7795 0.006927 **
Graph.position 0.15703   0.0758 0.800891
Menu            0.99845 968.4354 7.29e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

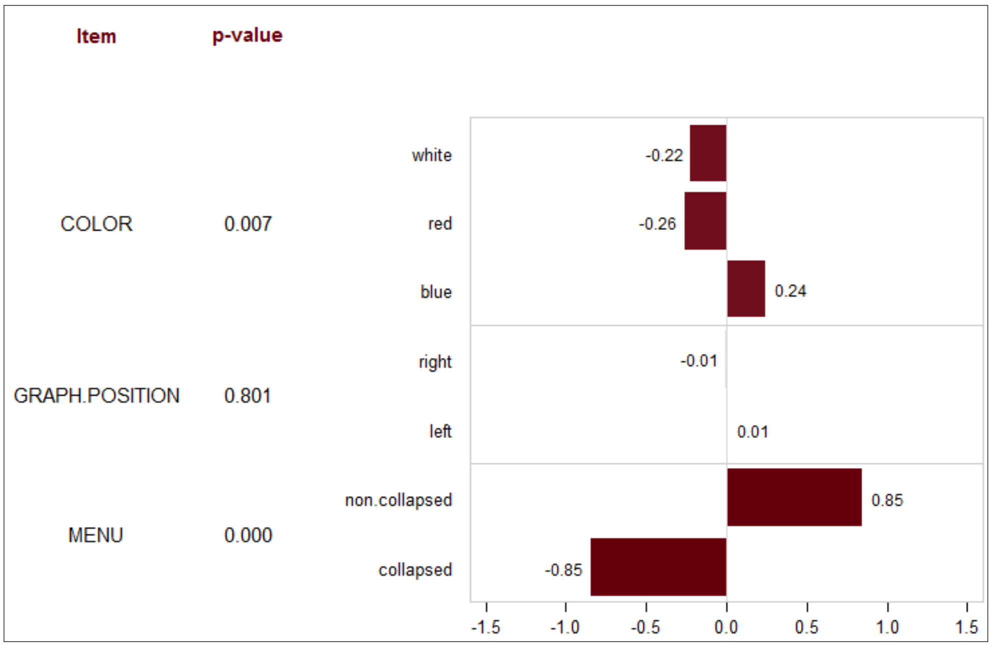


Figure 5.14. QT1 results for Powerful

Again, the Graph Position appears as not relevant for this kansei word. We can see that Blue is the color that has a positive effect on the response, while both Red and White have negative values. In the Menu item, again the Non-collapsed option has the positive effect.

Easy to use

```
QT1 Analysis for EASY
MCC= 0.9780  MCC^2= 0.9565  p-value= 0.02213

                   PCC F stat.  p-value
Color          0.92532  8.9323 0.054521 .
Graph.position 0.90482 13.5475 0.034740 *
Menu           0.95914 34.4755 0.009854 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

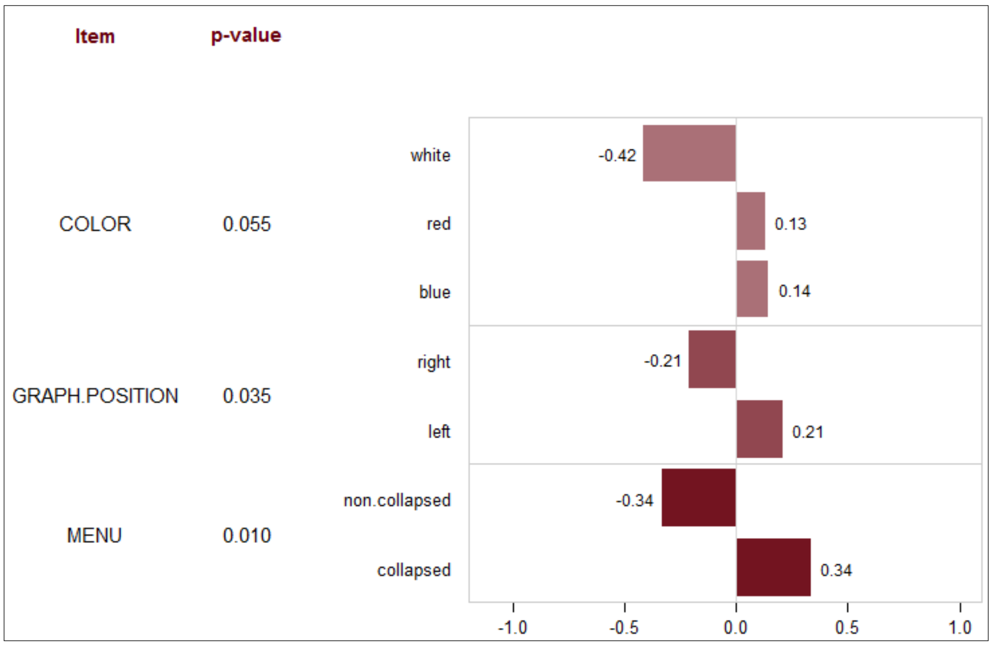


Figure 5.15.QT1 results for Easy to Use

In this case we find something not so evident. The Graph Position affects the response, being the Left option the one with a positive effect. This implies that the graph is located at the center of the page, and the options are located on the right side (the opposite as what was originally programmed). The Menu item locates its positive effect on the Collapsed option. This effect is more easily explained, as the page is perceived easier to use when the Menu is collapsed because the visual appearance is maybe simpler.

Simple

```
QT1 Analysis for SIMPLE
MCC= 0.9367  MCC^2= 0.8774  p-value= 0.09944

                   PCC F stat. p-value
Color          0.43653  0.3531 0.72825
Graph.position 0.20217  0.1278 0.74435
Menu           0.93438 20.6329 0.01998 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

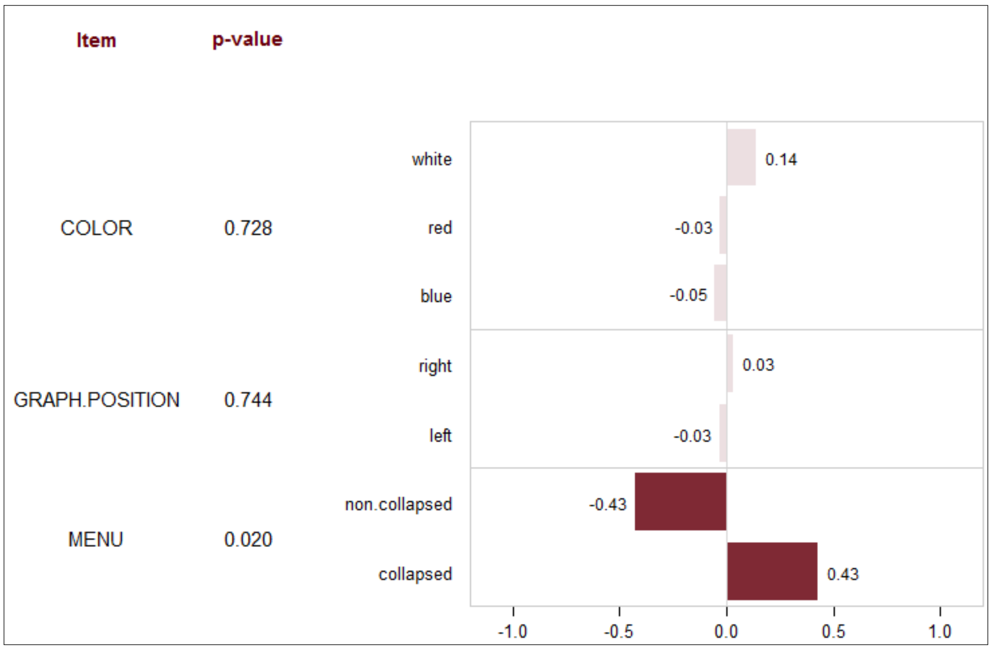


Figure 5.16. QT1 results for Simple

Clearly, in this case there is only one item that affects the kansei word. The Collapsed menu gives StatClip a way simpler appearance. The other two items (Color and Menu) have no effect on the response, as can be seen looking at the obtained *p-values*.

5.3.4.3. Results Summary

To summarize and better interpret the results, a table has been built, grouping the words as from the principal components results, and indicating its positive, negative or null effects on the responses.

Table 5.5. Summary of Results

	Color			**Graph Position**		**Menu**	
	White	Red	Blue	Right	Left	Non-collapsed	Collapsed
Funny	−	✚	+				
Attractive	−	+	✚			✚	−
Modern	−	+	✚			✚	−
Easy to Use				−	✚	−	✚
Interactive						✚	−
Flexible						✚	−
Powerful	−	−	✚			✚	−
Simple						−	✚

From the table just seen (Table 5.5), several conclusions can be extracted:

- The kansei words that were grouped in the principal component analysis turn to have really similar results on the QT1 analysis.
- For the "Attractive" group, we obtain positive effects (StatClip perceived as more attractive, more modern and funnier) with a Non-collapsed menu and using Red or Blue colors (preferably Blue, as it has a higher effect in Attractive and Modern).
- For the "Powerful" group, the positive effects are obtained with a Non-collapsed menu, and with Blue color in the Powerful kansei word.

To this point, no conflicts with the initial design of StatClip, as the results corroborate the decisions taken. However, there are two conclusions that differ:

- StatClip is perceived as simpler if the menu is collapsed.
- To be perceived as easier to use, the Graph position should be on the left, and the menu collapsed.

Even though StatClip's appearance is simpler and seems easier to use with the menu collapsed, it is considered that it is more important to have positive effects on the "Attractive" and "Powerful" groups, as they are features really important for our software.

The location of the graph, however, is only relevant for "Easy to Use", and the positive effect happens for the graph on the left position. It needs to be discussed if the originally programmed StatClip needs to change its display. The KE study results seem to indicate that it should.

Regarding the color, results show that Blue is the best color in general for all the Kansei words studied.

6. Economic cost analysis

As in any engineering project, the study of the economic view is necessary. In this section, the cost analysis will be detailed and explained.

In the first place, the elaboration of this project has involved the usage of different pieces of software. Some of the used packages have a license cost. This annual cost has been proportionally included (only taking into account the 5 months of the duration of the project) in the total cost. Secondly, the design and development costs include all costs related to the production of the actual program: the previous investigation and market studies, the conceptual design, the UI design, the actual programming and the KE study. Indirect costs are approximated to a 20% of the total software and design costs.

Table 6.1 shows the distribution of the total project cost.

Table 6.1. Project Costs

StatClip Project Costs				
Software Costs	**License annual cost**	**5 months' cost [€]**	**Usage**	**Cost [€]**
R	0	0	80%	0
Microsoft Office Professional	593	247.1	80%	197.67
Balsamiq Mockups	89	37.1	10%	3.71
TOTAL				**201.38**
Design and development Costs	**Hours [h]**	**Cost per hour [€/h]**	**Cost [€]**	
Initial documentation and investigation	50	30	1500	
Conceptual and theoretical design	25	100	2500	
UI Design	30	40	1200	
R programming	200	40	8000	
Kansei Engineering study design	10	30	300	
Kansei Engineering study	12	30	360	
Project documentation elaboration	40	30	1200	
TOTAL				**15,060 €**
Indirect Costs				
TOTAL SW + Design and development Costs	15,261 €			
+ 20%	0.2			
TOTAL	3,052 €			**2,004 €**
TOTAL COST				**17,467 €**

Conclusions

The main objective of this project was to implement an application for facilitating the learning of data analysis techniques and basic statistical concepts. A series of steps were sequentially followed to achieve this final aim: a “state of the art” studying already existing statistical software, definition of target users, compilation of users’ needs and translation into technical parameters of the application, and the added value of the emotional design study for improving user interface and perceived functionality.

However, having StatClip, our app for statistical data analysis learning, running smoothly is really where the rubber meets the road. This main objective has been successfully achieved, of course after many hours of developing code in R (which can be found at: https://github.com/eserrahima/statclip) and testing.

StatClip is now a fully functional program, and has some of it most important functionalities implemented. There is still a long way to go, but this project has developed the main structure of StatClip, and prepared the program to keep expanding and adding functionalities.

StatClip has been programmed using R, taking advantage of RStudio’s package *shiny*, and has been developed following a modular structure to facilitate the expansion, addition and modification of functionalities. A template to facilitate the programming of graphics has been created, and each one of the programmed R scripts includes explanations and comments to make the code easier to understand and modify.

The User Interface (UI) for each of the different options StatClip offers has been designed, always following the characteristics decided after the users’ needs analysis. The analysis of existing programs and the users’ needs study have been crucial to determine the features StatClip has. In the first place, the profound market study helped to understand the possibilities our software could have, and to see what was already created and developed. Secondly, and even more importantly, the users’ analysis led to define the final characteristics of StatClip. Main users (students, etc.) and operator users (teachers, professors, etc.) have different needs, and a deep analysis of them allowed to determine which ones StatClip needed to cover.

Seeking to cover those needs, and with the collaboration of Pere Grima, Lourdes Rodero and Lluís Marco, the philosophy of the program was defined. StatClip is an app designed to be held in a website. That is, it will be accessible and usable from any computer with internet access, no matter which operating system is using. Moreover, StatClip’s UI is divided in three spaces: Menu (always visible on the left), Configuration Options of the selected functionality and a Results and Plots space. As StatClip is intended to be interactive, it was decided that results would update automatically at every change on users' inputs. Each one of this features was considered in the code, and thus the result is an app with a high level of interactivity and reactivity.

Lastly, the kansei engineering study led to some interesting conclusions. Applying an emotional design technique to our software allowed to find the perception users get from StatClip's appearance. During the design process, and taking into account all the analysis previously done, it was decided that the UI would be in a blue and white color combination, that the Results box in all the functionalities pages would be on the right side, and that the Menu would be visible at all times (at the left side). The kansei engineering study proved our first ideas mostly right (StatClip is perceived as more "Attractive" and "Powerful" if blue and with visible Menu), but led to an unexpected result: StatClip is seen as Easier to Use if the results box is on the left. That is, UI being: "Menu – Results – Configuration Options" instead of the original "Menu – Configuration Options – Results". This result gives reasons to think on a change of the graphical user interface, to match what has been obtained at the kansei engineering study.

StatClip seeks to fill an empty space in statistical education. It is a simple, interactive, powerful and easy-to-use software though especially for statistics learning. The structure and architecture of the app is programmed, but there is still a lot of work to do. There are functionalities still to implement and bugs to fix, and for sure changes to make based on users' feedback and usage observation. New studies on users' views and opinions to make StatClip the ideal tool, R programming to build new functionalities and properties, as well as improving the existing ones… Another student, for example, could base his or her Final Degree Project on improving and continue developing StatClip. I also can (and will) keep working on making this app better. In the future, it would be desirable to not only work on adding functionality to the application, but also on developing training materials suitable for students at different levels. For instance, case studies including datasets and instructions on how to reach conclusions using StatClip could be very beneficial.

The ideas, development, and results of this project have been accepted as a paper for the ENBIS (European Network for Business and Industrial Statistics) annual conference. The presentation will be done in Prague in September 2015.

The door to having an open-source application for statistical data analysis learning has been opened, and its future can be very exciting. In the meantime, however, here finishes my Final Degree Project: "Design and development of an app for statistical data analysis learning".

Bibliography

[1] GUZMÁN, M. *Tendencias innovadoras en educación matemática*. [Online]. Madrid (Spain), Universidad Complutense de Madrid: 1993. [http://www.mat.ucm.es/catedramdeguzman/drupal/migueldeguzman/legado/educacion/tendenciasInnovadoras#Bibliografía; May 12th, 2015].

[2] BATANERO, C., DÍAZ, C., CONTRERAS, J., ROA, R. *El sentido estadístico y su desarrollo. NÚMEROS, Revista didáctica de las matemáticas.* [Online]. Vol. 83, 2013, pp. 7-18. [http://www.sinewton.org/numeros/numeros/83/Monografico_01.pdf; June 20th, 2015].

[3] BURRILL, G., BIEHLER, R. *Fundamental statistical ideas in the school curriculum and in training teachers*. 2011. In Batanero, C., Díaz, C., Contreras, J. ROA, R., *El sentido estadístico y su desarrollo. NÚMEROS, Revista didáctica de las matemáticas.* [Online]. Vol. 83, 2013, pp. 7-18. [http://www.sinewton.org/numeros/numeros/83/Monografico_01.pdf; June 20th, 2015].

[4] MOORE, D. S. *Teaching Statistics as a respectable subject*. 1991. In Batanero, C., Díaz, C., Contreras, J. ROA, R., *El sentido estadístico y su desarrollo. NÚMEROS, Revista didáctica de las matemáticas.* [Online]. Vol. 83, 2013, p. 7-18. [http://www.sinewton.org/numeros/numeros/83/Monografico_01.pdf; June 20th, 2015].

[5] THE R FOUNDATION. *The R project for Statistical Computing*. [Online]. [http://www.r-project.org/; March 21st, 2015].

[6] PYTHON. *About*. [Online]. [https://www.python.org/about/; March 21st, 2015].

[7] MCKINNEY, W. *Python for Data Analysis*. Sebastopol (USA), O'Reilly Media Inc.: 2013

[8] MINITAB INC. *Getting started with Minitab 17*. [Online]. [http://www.minitab.com/uploadedFiles/Documents/getting-started/Minitab17_GettingStarted-en.pdf; March 22nd, 2015].

[9] SAS INSTITUTE INC. *JMP Introductory Guide*. Version 3. Cary (USA), SAS Institute Inc: 1995.

[10] THE STATISTICAL LAB. [Online]. [http://www.statistiklabor.de/en/; March 18th, 2015]

[11] STATCATO. *Open-Source Java Software for Elementary Statistics*. [Online]. [http://www.statcato.org/statcato/index.php; March 19th, 2015].

[12] THOUGHT INTO DESIGN LTD. *Salstad Statistics*. [Online]. [http://www.salstat.com/index.html; March 28th, 2015].

[13] PATON-SIMPSON & ASSOCIATES LTD. *SOFA Statistics.* [Online]. [http://www.sofastatistics.com/home.php; March 20th, 2015].

[14] UNITED STATES CENSUS BUREAU. *Census and Survey Processing System*.

[Online]. [http://www.census.gov/population/international/software/; March 23rd, 2015].

[15] DEVELVE. *Develve*. [Online]. [http://develve.net/; March 23rd, 2015].

[16] PROJECT ENGINEERING DEPT. ETSEIB-UPC. *Tema 2: Definición del problema y análisis de necesidades.* Barcelona, UPC, p. 7.

[17] TUFTE, E. *The visual display of quantitative information*. 2nd Edition. Chesire (USA), Graphics Press: 2001, p. 15.

[18] STEINSALTZ, D. *Introduction to probability theory and statistics for psychology and quantitative methods for human sciences.* [Online]. University of Oxford: 2011, p. 9-20. [http://www.stats.ox.ac.uk/~deligian/PDF/psych/humansci1.pdf; April 23rd, 2015]

[19] MINITAB 17 SUPPORT. *What is a dotplot?* [Online]. 2015 [http://support.minitab.com/en-us/minitab/17/topic-library/basic-statistics-and-graphs/graphs/graphs-of-distributions/dotplots/dotplot/; May 2nd, 2015]

[20] BALSAMIQ STUDIOS LLC. *Balsamiq Mockups.* [Online]. 2015. [https://balsamiq.com/products/mockups/; March 14th, 2015]

[21] BROWN, D. *Communicating Design: Developing Web Site Documentation for Design and Planning*. 2nd Edition. San Francisco (USA), New Riders Press: 2011.

[22] THE R FOUNDATION. *Introduction to R. What is R?* [Online]. [http://www.r-project.org/about.html; May 21st, 2015]

[23] VENABLES, W. N., SMITH, D.M., R CORE TEAM. *An Introduction to R*. [Online]. Version 3.2.0. 2015. [http://cran.r-project.org/doc/manuals/r-release/R-intro.pdf; May 25th, 2015]

[24] RSTUDIO. *Take control of your code*. [Online]. [http://www.rstudio.com/products/rstudio/; May 21st, 2015]

[25] DALGAARD, P. *Introductory Statistics with R*. New York City (USA), Springer: 2002.

[26] CHANG, W., CHENG, J., ALLAIRE, J., XIE, Y., MCPHERSON, J. *Package 'shiny'.* [Online]. Version 0.12.1. 2015. [http://cran.r-project.org/web/packages/shiny/shiny.pdf; June 12th, 2015]

[27] RSTUDIO. *Teach yourself Shiny*. [Online]. 2014. [http://shiny.rstudio.com/tutorial/; February 23rd, 2015].

[28] CHANG, W., RSTUDIO, ALMASAEED STUDIO, ADOBE SYSTEMS INCORPORATED. *Package 'shinydashboard'*. [Online]. Version 0.4.0. 2015. [http://cran.r-project.org/web/packages/shinydashboard/shinydashboard.pdf; May 12th, 2015].

[29] R CORE TEAM. *Edgar Anderson's Iris Data*. [Online]. Version 3.3.0. 2015. [https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/iris.html; June 12th, 2015].

[30] R CORE TEAM. *Motor Trend Car Road Tests.* [Online]. Version 3.3.0. 2015. [https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html; June 12th, 2015].

[31] RSTUDIO. *Teach yourself Shiny. Lesson 3: Use Control Widgets*. [Online]. 2014. [http://shiny.rstudio.com/tutorial/lesson3/; February 23rd, 2015].

[32] WICKHAM, H. *GGPLOT2*. [Online]. 2013. [http://ggplot2.org/; 28th April, 2015].

[33] MARCO-ALMAGRO, L. *Statistical Methods in Kansei Engineering studies* (Doctoral Thesis). Barcelona, Universitat Politècnica de Catalunya: 2011.

[34] GOMEZ, A., PERALTA, E. *Aplicació de la metodologia Kansei en la valoració de sucs de fruita* (Final Degree Project, directed by Lluis Marco). Barcelona, Universitat Politècnica de Catalunya: 2007.

[35] TANAKA, Y. *Review of the Methods of Quantification. Environmental Health Perspectives.* Vol. 32, 1979, pp. 113. In Marco-Almagro, L. *Statistical Methods in Kansei Engineering studies* (Doctoral Thesis). Barcelona, Universitat Politècnica de Catalunya: 2011.