Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Statlets: Improving Statistical Analysis with R

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Michael Richard Ellers

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Chat Wacharamanotham

Registration date: *16.03.2017*
Submission date: *31.03.2017*

Contents

Abstract **xi**

Acknowledgements **xiii**

Conventions **xv**

1 Introduction **1**

1.1 R Programming Language 1

1.1.1 Job Offers 2

1.1.2 Scientific Articles 4

1.2 Problems with R 4

1.3 Contribution of this Thesis 7

1.4 Chapter Overview 11

2 Related work **13**

2.1 Popular Statistical Analysis Software 13

2.1.1 IBM SPSS Statistics 13

2.1.2 Stata and Statistica 14

2.1.3 SAS and JMP 14

2.1.4 General Purpose Programming Languages 15

2.2 Existing R Tools 15

2.2.1 R Studio 15

2.2.2 R Commander and Deducer 16

Deducer 16

R Commander 16

2.3 Flow-Based Data Analysis Software 17

2.4 Codelets 18

3 Medium-Fidelity Prototype 19

3.1 Design Process 19

3.2 Design 21

3.2.1 Background Structure 21

3.2.2 Canvas 24

3.2.3 Nodes 26

Interactions 26

User Feedback 27

3.2.4 Programming Section 29

4 First User Study 31

4.1 Study Design 31

4.2 Results and Discussion 33
4.2.1 Concept 33
4.2.2 Usability 34
4.2.3 Conclusion 37

5 High-Fidelity Prototype 39
5.1 Differences from Medium-Fidelity Prototype 39
5.1.1 Background Structure 40
5.1.2 Canvas 41
5.1.3 Nodes 42
5.1.4 Programming Section 45

6 Second User Study 47
6.1 Structure 47
6.2 Results and Discussion 48

7 Summary and Future Work 51
7.1 Summary and Contributions 51
7.2 Future Work 53
7.2.1 User Studies 53
7.2.2 Implementation 53

A First User Study 55
A.1 Questionnaire 55

A.2 Datasets 57

A.2.1 food 57

A.2.2 keyboard(unpaired) 59

A.2.3 keyboard(paired) 61

A.3 Tasks for First User Study 62

B Technical Details 65

C Second User Study 67

C.1 Questionnaire 67

C.2 Introduction to Datasets 69

C.3 Tasks for User Study 71

Bibliography 75

Index 79

List of Figures

1.1 Number of data science job offers in February 2017 on *indeed.com* 3

1.2 Compares the trends of job positions on *indeed.com* of R, Python and SAS 4

1.3 Number of scholarly articles found in 2015 . 5

1.4 Change in the number of scholarly articles from 2014 to 2015 6

1.5 Flow based interface of the high-fidelity prototype 8

1.6 Interface of the high-fidelity prototype, which includes a flow-based interface and a programming section 9

2.1 The interface of Deducer 16

2.2 The interface of R Commander 17

2.3 Number of scholarly articles found in 2015 . 18

3.1 Design-Implement-Analyse Cycle 20

3.2 Folder structure of the mock-up prototype . . 22

3.3 Background structure of the mock-up prototype 23

3.4 Canvas of the mock-up prototype 24

3.5 Example of a menu to add new nodes to the Datasets section in the mock-up prototype . . 26

3.6 Example test-node from the mock-up prototype 26

3.7 Example dataset-node from the mock-up prototype 27

3.8 States a node can have in the mock-up prototype 28

3.9 Mock-up prototype with open programming section 29

3.10 Programming section of the mock-up prototype 29

5.1 Folder-like structure of high-fidelity prototype 40

5.2 Text-input area to change the names of the background structure elements of the high-fidelity prototype 41

5.3 Canvas of the high-fidelity prototype 41

5.4 Node of the high-fidelity prototype 42

5.5 Connection menu of the high-fidelity prototype 43

5.6 Nodes from the high-fidelity prototype that are in a state of execution 45

Abstract

Statistical analysis is widely used. It influences everybody's daily life. For example, the way diseases are treated is often based on the analysis of statistical data. Therefore, it is very important that analyses are done correctly.

R is one of the most popular tools for statistical analysis [Muenchen, 2012]. The greatest advantages of R are that it is free to use and new analysis methods are usually implemented faster than in other statistical analysis tools. R is also more flexible than other tools since it is a fully functional programming language and advanced users can implement their own methods. Thus, it is a good tool for statistical analysis.

However, the workflow of programming in R can be improved. For example, users do not have an overview of the steps done during the analysis and, hence, often can not recall all of them. This is likely to cause mistakes in the interpretation of the results. The user can, for example, forget about used assumptions or data cleaning methods which have to be considered. So there is room for improvement.

This thesis discusses a new programming environment for R, called Statlets. It has an integrated flow-based user interface. That means, each step is displayed as a box and the boxes are connected by arrows to show the data flow between them. This gives the user an overview of the steps to prevent him from forgetting about any steps he has done during the analysis. It also enforces a modularised code structure to improve flexibility and reusability. Additionally, the programming is aided by a new method of interacting with inputs and outputs of a function to improve the overview of the function and the coding speed. With these methods, Statlets plans to improve the workflow.

This thesis discusses two user studies, which are conducted with prototypes of Statlets. They show that Statlets can be used to perform statistical analysis tasks.

Acknowledgements

First of all, I would like to thank my supervisor Krishna Subramanian for his consistent support with feedbacks and advises.

I thank Prof. Dr. Borchers for supervising my thesis and Prof. Dr. Chat Wacharamanotham for being my second examiner.

Special thanks to everyone who participated in the two user studies. Thank you for investing your time and for the constructive feedback.

Finally, I want to thank everyone who proofread this thesis.

Thank you!

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition: *Excursus*

Source code and implementation symbols are written in typewriter-style text.

`myClass`

The whole thesis is written in British English.

Chapter 1

Introduction

Statistical analysis becomes more and more important in a world that gathers more and more data. It is important for researchers, but nowadays also many companies create statistics to improve their decision making. Even politicians use statistics to justify their arguments.

1.1 R Programming Language

R is one of the most popular statistical analysis tools

One of the most popular tools for statistical analysis is R. R is a programming language and environment specifically designed for data analysis.

R is free, new methods are implemented quickly and it is a fully functional programming language

It has many benefits. It is open source and, hence, free whereas many other statistical analysis tools cost money. Furthermore, new methods for statistical analysis are usually implemented faster for R than for other statistical analysis software. R is also very flexible since it is a fully functional programming language and advanced users can implement their own methods.

R is popular

These benefits of R are probably the reason for its popularity. This section shows that R is popular.

However, it is not an easy task to find out how popular a software is. The most straightforward idea would be to compare the number of times a software was purchased. Though, this is not possible since some tools, like R, are free and users can download them several times and some cost money and a user does not buy the same product several times.

Nonetheless, there are many statistics that analyse different aspects of the popularities of statistical software tools. Even though none of the methods can be expected to be exact, they all demonstrate that R is among the most popular tools for statistical analysis. Hence, one can be certain that R is popular. The two most meaningful statistics are presented here.

1.1.1 Job Offers

One way to measure the popularity of a software tool is to analyse the number of job offers that include the software in their description.

Robert A. Muenchen analysed the popularity of software tools in job offers in the field of data science. ([Muenchen, 2012], [Muenchen, 2017b]). Statistical analysis is a part of data science and, hence, this analysis can give an impression of the popularity of R.

Muenchen describes an analysis of job offers in February 2017 on *indeed.com*. This website is a search engine for jobs and operates worldwide. It is the job website with the highest traffic in the United States. Thus, analysing job offers from this website is a good measure of the popularity of tools.

It was searched for each of the well-known data analysis software tools and a number of keywords that relate to data analysis were added to these searches. [Muenchen, 2017a] describes how the analysis is done in detail. So jobs that are not related to data analysis are not considered.

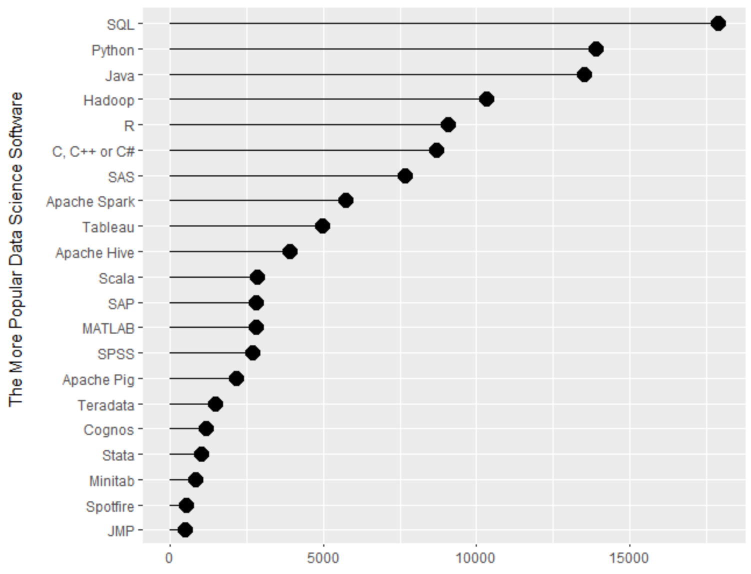


Figure 1.1: Number of data science job offers in February 2017 on *indeed.com* that include each software in their description. (Source: [Muenchen, 2017b])

The results can be seen in figure 1.1. The figure shows a list of the most popular software tools and the number of times they are used in job offers.

It can be seen that R is among the top five most popular tools in the field of data analysis. This suggests that R is also popular for statistical analysis.

[rev, 2017] compares the popularity of R, Python and SAS for statistical analysis in the last 5 years. It also investigates job offers on *indeed.com*, but in this case, only the keyword "statistics" is added to the search.

Figure 1.2 displays the result of that analysis. It shows that R is more popular than Python and SAS. Furthermore, the trend suggests that the popularity of R will increase in the future.

Concluding, the analysis of job offers suggests that R is a popular tool and its popularity is likely to increase in the future.

Job offers suggest that R is popular and its popularity will increase in the future

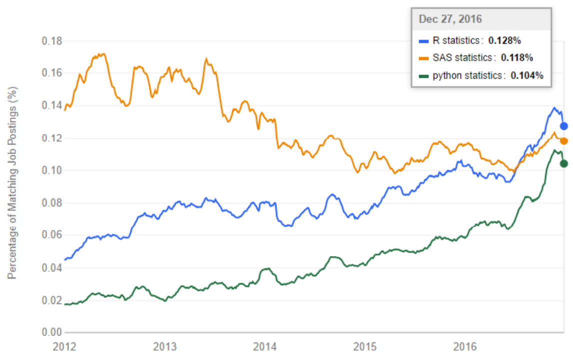


Figure 1.2: Compares the trends of job positions on *indeed.com* of R (blue), Python (green) and SAS (orange) (Source: [rev, 2017]).

1.1.2 Scientific Articles

Another method, to measure the popularity of statistical analysis tools, is to analyse which tools are used for scientific articles.

R is widely used in scientific papers

Figure 1.3 shows the number of scientific articles from 2015 that used each software package. SPSS is the most popular one followed by R and SAS. Thus, in scientific papers R is among the most popular software tools as well.

The popularity of R in scientific papers will probably increase

Figure 1.4 shows the changes in the popularity of those tools from 2014 to 2015. The popularity of R increases and of SPSS, the most used tool, decreases. Just like the analysis of job offers, the analysis of scientific papers suggests that the popularity of R will increase in the future.

Concluding, all statics shows that R is a popular tool for statistical analysis. Furthermore, the trends suggest that R will become even more popular in the future.

1.2 Problems with R

R has issues

Although R is very popular, there are problems. We identified a number of issues with statistical analysis in R:

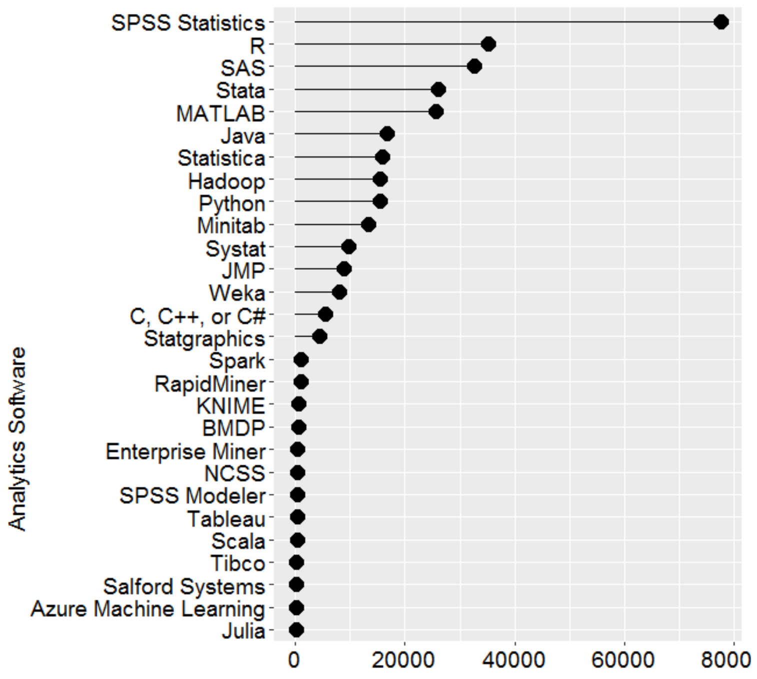


Figure 1.3: Number of scholarly articles found in 2015 that used each software ([Muenchen, 2017b]).

1. **It is difficult to organise collaboration in writing code in R.** R does not encourage a modularised code structure. Thus, it is complicated to distribute tasks between programmers. To work together developers have to agree on strict conventions of the code structure. This creates additional work. It also can lead to problems in the integration of the different parts if the conventions are not followed exactly or not designed properly.

2. **It can be difficult to integrate new methods in existing analysis procedures.** R has the benefit that new analysis methods are implemented quickly. However, this can also lead to problems. Especially researchers are always required to keep up with all new developments to be able to compare their data with other researchers. Integrating a new function in an existing analysis workflow is not always easy since it can require changes in the whole analysis procedure. For example, they can require a different preprocessing. So integrating new methods can be time-consuming.

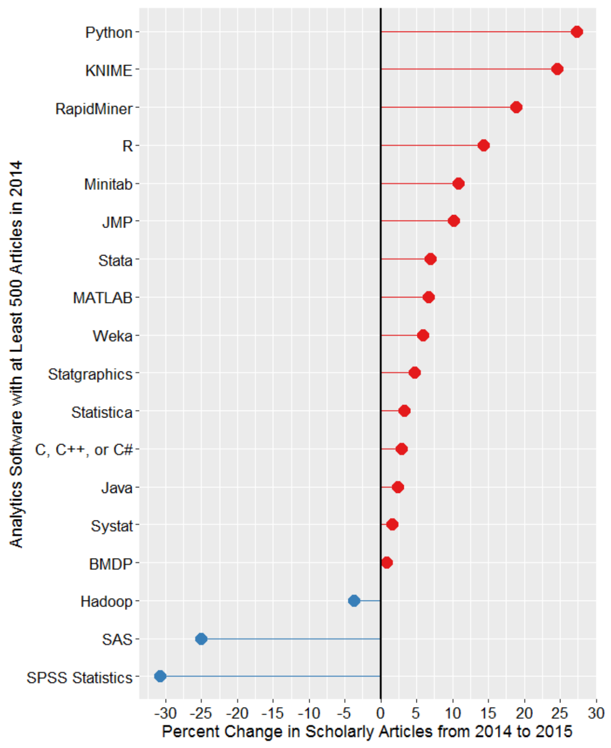


Figure 1.4: Change in the number of scholarly articles using each software from 2014 to 2015 (Source: [Muenchen, 2017b]).

3. **The use of the interactive console can lead to a number of problems.** Users can do their whole statistical analyses in the console. This seems to be convenient but encourages to experiment and hard-code values. Since the user does not have an overview of what he has done it can lead to mistakes.

   Furthermore, doing the analysis in the console it can not be reused. If the same or a similar analysis has to be done in the future the user has to write it again.

   It is also good to have the exact analysis procedure saved since in statistical analysis it is important that all steps are remembered and considered in the analysis.

   The user can, of course, convert the analysis into a file after he finished it, but this creates unnecessary additional work. It is better to directly implement the analysis procedure in a file.

4. **R is difficult for beginners.** R is a fully functional programming language and users have to learn it to be able to conduct statistical analysis with it.

5. **It is easy to forget about steps that are done during the analysis, which can lead to mistakes in the interpretation of the results.** If the analysis includes very few steps this should not be a problem. However, if the analysis is more complex, e.g. including multiple preprocessing steps, it is not easy to keep track of all of the steps.

   This can lead to mistakes in the interpretation. For example, approximations or assumptions made within the analysis can be overlooked.

1.3 Contribution of this Thesis

We developed a new concept that aims to solve the problems discussed above

This thesis demonstrates a new concept to improve the workflow when conducting statistical analysis with R. An improved workflow reduces the cost in time and resources. Additionally, it makes the analysis easier and less prone to errors. The new concept focuses on solving the problems of R that are discussed above.

The target group are people with knowledge in R and statistical analysis

The target group are people who want to conduct statistical analysis with R and have at least basic knowledge in R and statistical analysis. It is meant for fairly inexperienced users and experts alike.

We implemented the concept in the programming environment Statlets

To be able to test the new concept, we developed a programming environment that implements it. This programming environment is called Statlets. To explain how we intend to solve the problems with Statlets we have to discuss the concept first.

Statlets integrates a data flow based user interface

Statlets is a programming environment that integrates a data flow based user interface. Figure 1.5 shows a screenshot of the interface of the second prototype. Each green

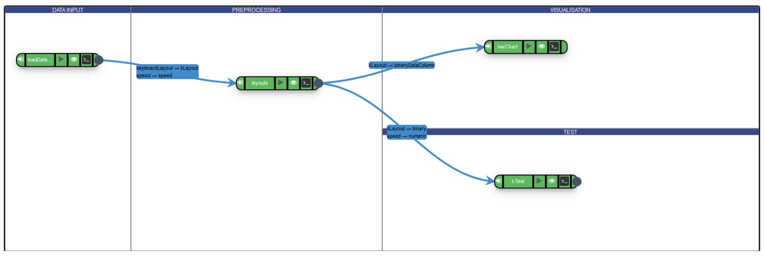


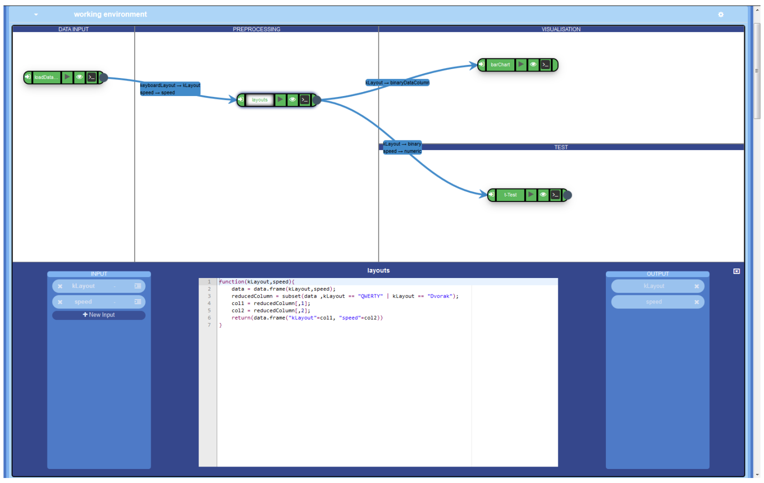
Figure 1.5: Flow-based interface of the high-fidelity prototype.

box contains some part of the code. The boxes in the graph are called *nodes* since they resemble nodes in a graph (This kind of graph that consists of nodes and edges and not the graph of a function). The connections between the nodes show which node uses the output of which other node as its input. However, a node does not have to use the whole output of its input-nodes but can select single elements like columns from these outputs. This concept creates a well-structured analysis and enforces a modularised code structure. It also makes changing the analysis structure easy.

To give an example, we discuss the analysis that is shown in figure 1.5 in detail. The left most node in the figure loads a dataset. This is then passed to the node in the middle. That is shown in the interface by the arrow that connects these nodes. The node in the middle removes rows that are not needed for the analysis. The node on the top-right displays a bar plot of some part of the data and the one in the bottom-right performs a t-Test. They both get their inputs from the output of the node that reduces the dataset. This is shown by the arrows that connect it to these nodes. However, they only use the columns that are needed for the tasks as inputs.

The innovation is that the flow-based interface is included in a programming environment

This concept of a flow-based data analysis is not new, however, the innovation is that it is a programming environment. Thus, the user programs the nodes within the tool (Figure 1.6). Other flow-based data analysis tools use predefined nodes. Most can include nodes that users coded themselves but the coding has to be done outside of the tool. A flow-based user interface integrated in the program-



```
function(kLayout,speed){
    data = data.frame(kLayout,speed);
    reducedColumn = subset(data ,kLayout == "QWERTY" | kLayout == "Dvorak");
    col1 = reducedColumn[,1];
    col2 = reducedColumn[,2];
    return(data.frame("kLayout"=col1, "speed"=col2))
}
```

Figure 1.6: Interface of the high-fidelity prototype, which includes a flow-based interface and a programming section

ming environment allows the user to see the structure of his analysis while coding. This give a better overview of the process.

Additionally, the programming section has inputs and outputs displayed next to the programming editor. The inputs are displayed on the left side of the editor and the outputs on the right side. The user can interact with them. The way it works is described in more detail in the description of the first prototype in chapter 3.

Inputs and outputs displayed next to the programming editor create a perceptual analogy to the node

It creates a perceptual analogy between the programming section and the node. The node also gets its inputs on the left and has its outputs on the right side. Additionally, this improves the overview and by interacting with the inputs and outputs the user can increase his programming speed.

With this innovative concept we want to solve the discussed problems of statistical analysis with R. In the following we describe how Statlets tackles these problems:

Statlets makes it easy to organise collaboration

1. **It is difficult to organise collaboration in writing code in R.** With Statlets it is very easy to organise collaboration since the program enforces a modularised code structure. So it is clear how to distribute tasks:

One node is coded by one person. Furthermore, programmers do not have to agree on conventions since these are defined by the programming environment.

Additionally, because single columns from the output of a node can be selected as inputs, nodes do not have to be designed around the exact structure of a dataset.

All of these advantages make it easy to merge the work of different programmers.

New methods can be integrated easily

2. **It can be difficult to integrate new methods in existing analysis procedures.** The integration of new methods is easy in Statlets. A new method can be implemented in a StatLet and integrated by connecting it to the required inputs. Furthermore, if changes in the whole workflow are required this can be done by simply changing connections between Statlets. So no major restructuring of code is necessary.

Statlets does not allow interactive use of the console to solve the problems with it

3. **The use of the interactive console can lead to a number of problems.** These problems do not occur in Statlets since it does not allow interactive use of the console. Thus, it forces the programmer to write proper functions.

The structure and the opportunity to learn step by step makes Statlets easier for beginners

4. **R is difficult for beginners.** With Statlets the user still has to learn R. Though, the beginner is supported by the provided structure. Furthermore, by using nodes that are programmed by other people, beginners can focus on a subtask, like programming one step in the analysis. So they do not have to learn all steps of a statistical analysis procedure, at once but can do it step by step.

The flow-based interface helps the user to remember the steps of his analysis procedure

5. **It is easy to forget about steps that are done during the analysis, which can lead to mistakes in the interpretation of the results.** Statlets tackles this problem by giving an overview of all steps that are done in the flow-based user-interface. So the user can see the steps and how they are connected as a visual graph with is easy to memorise for the human brain. Additionally, the user can look up the steps at a glance. So the chance that a user forgets about steps that are done in his analysis is likely reduced.

1.4 Chapter Overview

In the next section, we discuss the related work that influenced the design of Statlets. After that, a medium-fidelity mock-up prototype is discussed in detail. This prototype is used for the first user study that is presented after that. This user study focuses on usability aspects to improve the next prototype. The next prototype is a high-fidelity software prototype. This prototype is used for the second user study which is presented after the prototype. The aim of this user study is to show that Statlets can be used for statistical analysis. However, it does not proof that Statlets solves the problems that are discussed above. This validation will be part of future work. This is discussed in the last chapter after a summary is given.

Chapter 2

Related work

In the previous chapter, we discussed the motivation for this thesis. We showed that R is among the most popular tools for statistical analysis and identified problems that R has. After that, we presented the concept behind Statlets and how it intends to solve these problems.

This chapter discusses the software tools that influenced the concept and design of Statlets. First, we describe other statistical analysis software tools. Then we discuss existing tools for R. In the last section we describe Codelets, which influenced the design of the interactive input and output displays next to the programming editor.

2.1 Popular Statistical Analysis Software

Statlets aims to improve R. However, we looked at many other data analysis tools for inspiration.

2.1.1 IBM SPSS Statistics

The most used statistical analysis software is IBM SPSS Statistics. The data is displayed in a GUI and analysis methods can be selected in the menu. So SPSS is made for users

SPSS is easy to use and is designed for users without programming experience with no programming experience and it emphasises that it is easy to use. Thus, it is a good tool for beginners.

It can be difficult for experienced users to make SPSS do what they want

For a user who has advanced knowledge in statistics, it can be difficult to make the program perform the exact analysis that the user wants. There are options to integrate functionalities using other programming languages like R and Python but this is quite cumbersome and includes restarting the system. In this case, it is probably easier for users to use these programming languages directly. In addition, this also saves money since SPSS is not a free software.

2.1.2 Stata and Statistica

Stata and Statistica are similar to SPSS

As for SPSS, one has to pay for Stata and Statistica. They are quite similar to SPSS. They appear to be a bit more complicated to use but, therefore, more flexible.

2.1.3 SAS and JMP

SAS has a programming language

The SAS software has, like the tools discussed above, the option to perform analyses by selecting them in the menu. However, the strength of SAS is its own programming language, which allows for in-depth analyses.

SAS is the most similar popular tool to R

This makes the SAS software the most similar statistical analysis software to R. However, there are differences.

R is free, new methods are implemented faster and it has more methods but SAS methods are verified

R is free, while SAS costs money. In exchange, methods provided for SAS are verified while R relies on methods coded by users and, thus, the methods are not verified. Therefore, new statistical methods are implemented much faster in R. Additionally, it leads to a greater amount of available packages for R than for SAS.

JMP is for exploring the data while SAS is for in-depth analyses

JMP is like the SAS software developed by SAS Institute. Analyses can be done by selecting methods in a menu. Though, it also has its own scripting language and it can run SAS programs. However, there are differences between the SAS software and JMP. JMP focuses on exploring the data. So the user can easily switch between different visualisations to get to know the properties of the data. Though, JMP can not perform in-depth analyses like SAS.

2.1.4 General Purpose Programming Languages

General purpose programming languages can be used for statistical analysis

Statistical analysis can also be done by software that is not specifically designed for the task. General purpose programming languages can also be used.

The most popular ones are Java, Python, C, C#, C++. Matlab is not a general purpose programming language. However, it is not developed for statistical analysis either. Thus, in this context, it works similarly like general purpose programming languages.

They are often used to implement new methods

These programming languages are often used for implementing new statistical analysis methods. For example R packages are often written in C to make them efficient.

They have fewer statistical methods

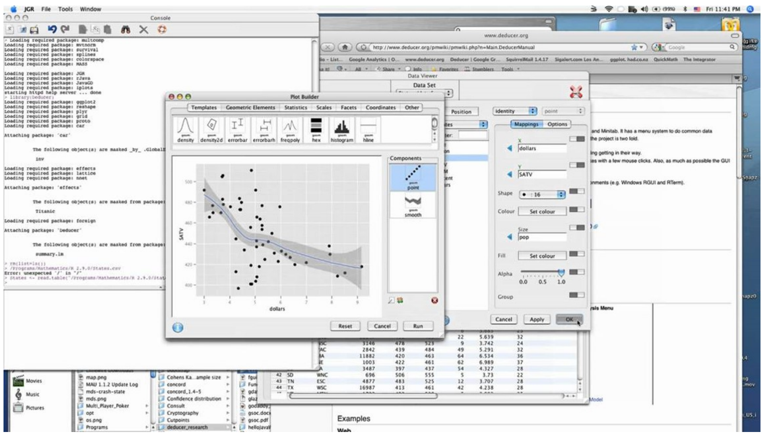
However, they have fewer existing methods for statistical analysis available.

2.2 Existing R Tools

R is not an easy programming language and there are a couple of tools that make the usage of R easier.

2.2.1 R Studio

The R Studio is an integrated development environment. It includes a code editor and debugging tools. Additionally, it has a visualisation tool to visualise datasets and graphs within the program.



```
> Loading required package: multcomp
Loading required package: mvtnorm
Loading required package: survival
Loading required package: splines
Loading required package: colorspace
Loading required package: MASS

Loading required package: JGR
Loading required package: rJava
Loading required package: JavaGD
Loading required package: iplots
starting httpd help server ... done
> library(Deducer)
Loading required package: ggplot2
Loading required package: reshape
Loading required package: plyr
Loading required package: grid
Loading required package: proto
Loading required package: car

Attaching package: 'car'

        The following object(s) are masked _by_ .Global

         inv

Loading required package: effects
Loading required package: lattice
Loading required package: nnet

Attaching package: 'effects'

        The following object(s) are masked from package

         Titanic

Loading required package: foreign

Attaching package: 'Deducer'

        The following object(s) are masked from package

         summary.lm

> rm(list=ls())
> /Programs/Mathematics/R 2.9.0/States.csv
Error: unexpected '/' in "/"
> States <- read.table('/Programs/Mathematics/R 2.9.0/Stat
>
```

Figure 2.1: The interface of Deducer

2.2.2 R Commander and Deducer

R Commander and Deducer are designed for beginners

Deducer[Fellows et al., 2012] and R commander [Fox, 2005] are designed to make using R easier for beginners and people who can not program.

Deducer

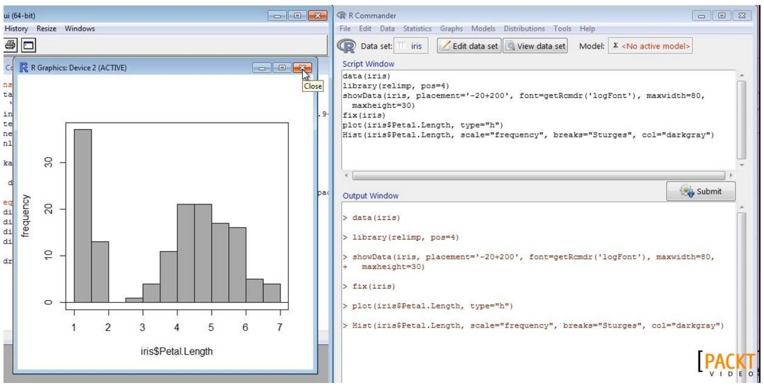
Deducer has a SPSS like user interface

Figure 2.1 shows the typical user interface for Deducer. Deducer is a GUI for R for people who can not program. A user can use basic functionalities of the language by selecting functions in the GUI. It still has a console where one can enter commands but it is primarily used for outputs and does not encourage to write commands.

R Commander

R commander supports users to learn R

R Commander, however, does that. It helps the user to improve his R skills. Figure 2.2 shows the interface of the R Commander. In this GUI the console is the focus and the user is encouraged to type commands. Additionally, R Commander supports beginners by offering a list of basic commands that can be selected in a menu.



```
data(iris)
library(relimp, pos=4)
showData(iris, placement='-20+200', font=getRcmdr('logFont'), maxwidth=80,
  maxheight=30)
fix(iris)
plot(iris$Petal.Length, type="h")
Hist(iris$Petal.Length, scale="frequency", breaks="Sturges", col="darkgray")
```

```
> data(iris)

> library(relimp, pos=4)

> showData(iris, placement='-20+200', font=getRcmdr('logFont'), maxwidth=80,
+   maxheight=30)

> fix(iris)

> plot(iris$Petal.Length, type="h")

> Hist(iris$Petal.Length, scale="frequency", breaks="Sturges", col="darkgray")
```

Figure 2.2: The interface of R Commander

2.3 Flow-Based Data Analysis Software

Developing a software with a flow-based interface, we looked at tools that use this kind of interface.

There is a number of tools that implement flow-based user-interfaces:

There is a number of tools that use flow-based interfaces

- IBM SPSS Modeler [McCormick et al., 2013]
- Orange [Demšar et al., 2013] [Demšar et al., 2004]
- RapidMiner[Klinkenberg, 2013]
- KNIME [Berthold et al., 2009]

The implementation of the flow-based concepts is similar in each tool. Each tool has a list of nodes the user can choose from. Own nodes can be created but have to be imported form outside of the tool. When the user chooses a node it is created and displayed on the canvas. For all tools but SPSS Modeler nodes can be connected by dragging an arrow from the output of one node to the input of another. SPSS Modeler creates a connection automatically when a node is created. It connects the node that was selected before the new node was created to the new node. Alternatively, the user can choose an option to connect nodes form a menu that is opened by right clicking on a node.

The realisation of the flow-based interface is similar in each tool

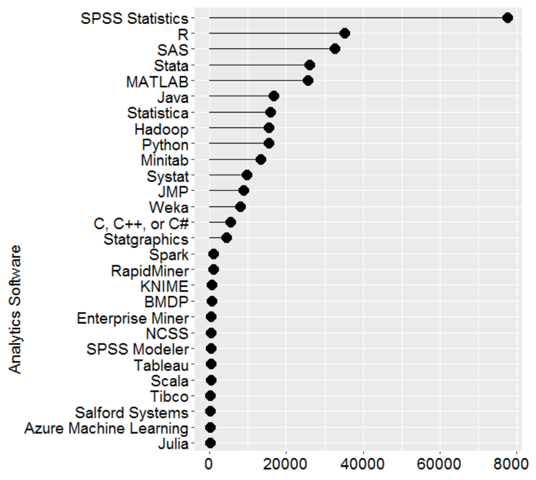


Figure 2.3: Number of scholarly articles found in 2015 for each software (Source: [Muenchen, 2017b]).

RapidMiner and KNIME gained popularity recently

As it can be seen in figure 2.3, RapidMiner and KNIME gained some popularity in recent years if one compares it to older results from [Muenchen, 2012]. Thus, it appears that the popularity of flow-based interfaces increases.

2.4 Codelets

Codelets ([Oney and Brandt, 2012]) is not related to statistical analysis. It is an extension for a code editor to write HTML code. It aims to improve the speed and user experience while coding. To do that, it links an interactive documentation to code snippets that are copied from examples from official websites. These websites usually have a good explanation of the function but without Codelets this information is lost. Additionally, it allows the user to manipulate common properties like the position of a button by choosing the position in a GUI.

It inspired the display of inputs and outputs next to the programming section and the ability to interact with them.

Chapter 3

Medium-Fidelity Prototype

Based on the related work, which we discussed in the last chapter, we developed prototypes. This chapter discusses the medium-fidelity prototype that is used in the first user study (Refer chapter 4).

The first section describes the design process to develop the prototype used in the first user study. That prototype is presented in the second section.

3.1 Design Process

The design process includes many iterations and in each new prototypes are developed.

First prototypes were paper prototypes and later ones mock-up prototypes

The first prototypes were developed as low-fidelity paper-prototypes to explore many different concepts with low development costs. Once we decided on a concept, it was implemented as a medium-fidelity mock-up prototype using a tool, called Balsamiq ([Faranello, 2012]). The developed mock-up prototype has a narrow design. That means, it simulates one sequence of tasks in-depth while other task sequences are not possible with the prototype. This al-

$$\begin{tikzcd}
 & \text{Design} \arrow[dr] & \\
\text{Analyse} \arrow[ur] & & \text{Implement} \arrow[ll]
\end{tikzcd}$$

Figure 3.1: Design-Implement-Analyse Cycle (DIA Cycle)

lowed us to do better usability testing.

Definition:
Mock-Up Prototype

MOCK-UP PROTOTYPE:
A mock-up prototype consists of a number of pictures that show different states of the interface. Each picture has a number of links to other pictures. Theses links are positioned at key elements of the user interface to simulate user input. For example, a link at the position of a button is used to simulates that this button can be pressed.

The DIA Cycle was used

Using that mock-up prototype we did many further design iterations to improve the usability. Each design iteration consisted of three steps in accordance with the DIA Cycle (Figure 3.1). First, we designed a prototype, then, implemented and finally analysed it. This procedure helped to improve the prototype at every iteration.

The prototype was analysed using Cognitive Walkthrough

To guarantee improvement in each iteration, a structured analysis was mandatory. To structure our analysis we used Cognitive Walkthrough. This method is good to analyse problems with the workflow and requires less time to perform than other methods. Thereby, more design iterations could be done. We developed typical tasks a user would perform and listed all steps needed to complete the tasks. We analysed each step by the following questions:

1. ″Does the user understand that this step is necessary to fulfil the task?″

2. ″Does the user understand how to perform that

step?"

3. "Is the feedback given to the user appropriate?"

This method helped to detect and remove design issues.

Once all detected design issues were fixed, to further improve the system, we did a user study which is discussed in chapter 4.

3.2 Design

This section discusses the prototype that was used for the first user study (Refer chapter 4).

For a better understanding, this section is divided into four subsections. The prototype lets the user perform multiple statistical analyses in the same session using a structure in the background. This structure is discussed in section 3.2.1.

Each analysis is organised on a canvas. Figure 3.4 shows an example canvas. The design and functionality of the canvas is discussed in section 3.2.2.

On the canvas there are boxes. Each box is a step in the statistical analysis. The boxes are called nodes and are discussed in section 3.2.3.

To perform a step in the statistical analysis a node contains R code that specifies what the node does. This code is written by the user in the programming section, which is discussed in section 3.2.4.

3.2.1 Background Structure

The prototype has a folder-like structure

The background structure resembles a three-layer folder structure that is visualised in figure 3.2. A session of the tool can contain a number of Workflows and each of those

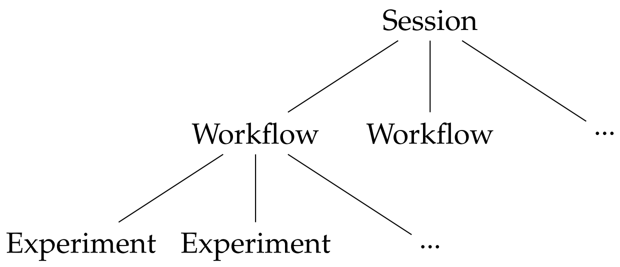


Figure 3.2: Folder structure of the mock-up prototype

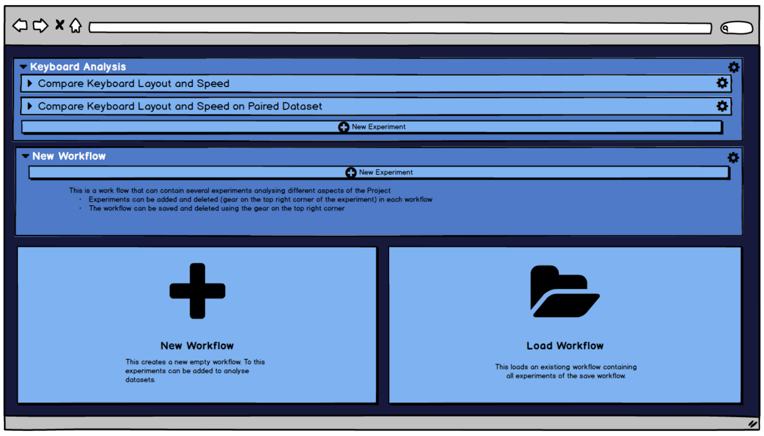
can contain a number of Experiments. This helps the user to structure his work.

If a user wants to analyse different datasets each of the analyses is done in a different Workflow. For each study the user likely wants to conduct a number of different analyses which can be done in different Experiments within the same Workflow.

To give an example, consider Max. Max is a researcher and wants to compare the typing speed of different keyboard-layouts. He conducts a user study. He splits the user study participants into groups. Each group has to type a text on one of the keyboard-layouts. To analyse the user study Max creates a Workflow for it. In it he creates an Experiment to compare the typing speeds of the layouts. However, he discovers that participants from one group are much younger than from the other groups. So he creates another Experiment in the same Workflow to investigate the effect of that. Using the results of this user study Max conducts another one. Since the study conditions are different, he does not want to confuse the user studies. So Max creates another workflow to analyse the second study.

We used a special design of the folder-like structure to save space

To realise the described folder-like structure, we chose not to display it on a side-pane, like many programming environments do. The reason is that it would take too much space. Instead, the structure is realised by nested boxes, which can be seen in figure 3.3. The whole programming environment resembles the Session folder. Each dark blue box in it is a Workflow. The Gestalt law of Closure is used here to distinguish between them. The Experiments within


Keyboard Analysis
Compare Keyboard Layout and Speed
Compare Keyboard Layout and Speed on Paired Dataset
New Experiment
New Workflow
New Experiment
New Workflow
Load Workflow

This is a work flow that can contain several experiments analysing different aspects of the Project
- Experiments can be added and deleted (gear on the top right corner of the experiment) in each workflow
- The workflow can be saved and deleted using the gear on the top right corner

This creates a new empty workflow. To this experiments can be added to analyse datasets.

This loads an existiong workflow containing all experiments of the save workflow.

Figure 3.3: Background structure of the mock-up prototype

the Workflows are also separated by closure. To help the user distinguish between Workflows and Experiments at first glance, we used different colours utilising the Gestalt law of similarity. Thus, the user can easily grasp the structure of Workflows and Experiments.

Experiments and Workflows can be extended and minimised

Furthermore, Experiments and Workflows can be extended and minimised by clicking on the small triangles in the top left corner of each. In figure 3.3 the Workflows are extended and Experiments are minimised. This extending and minimising allows the user to get an overview of the structure so that he can find the desired Experiment easily.

Experiments and Workflows can be saved, deleted and duplicated

Both Experiments and Workflows have a number of options that can be opened by clicking on the cog symbol on the top right of each. They can be saved to the local computer so that the user can load the analysis process at a later stage and distribute it to others. They can be deleted if they are not needed anymore and duplicated so that existing Workflows and Experiments can be used as a basis for new ones. These options allow for an easy navigation of Workflows and Experiments.

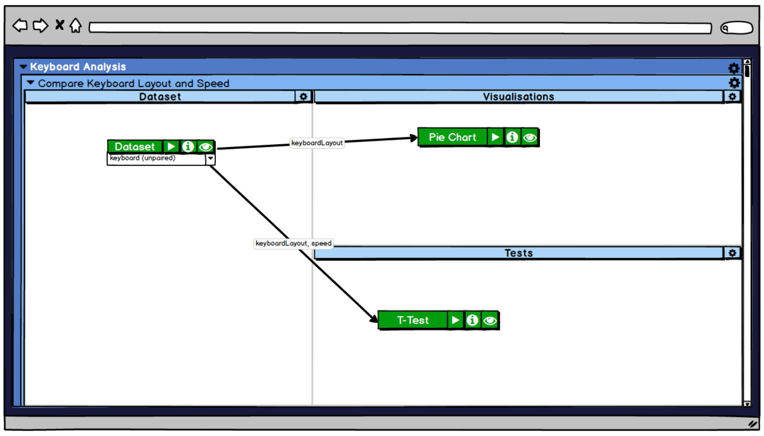


Figure 3.4: Canvas of the mock-up prototype

3.2.2 Canvas

Within each Experiment, when it is extended, the user can see a canvas. An example can be seen in figure 3.4.

Each function is saved in a node which are connected by arrows to show the dataflow

In the canvas functions for the statistical analysis are structured. Each of the nodes, which can be seen in figure 3.4, contain one function. These are connected by arrows to show which node gets inputs from which other nodes. This creates a visualisation of the data-flow and improves the user's overview.

Statistical analysis can be split into four parts

To further improve the overview, we investigated the steps that are needed for statistical analysis. It can be split into four distinct parts:

1. Loading the dataset
2. Processing the dataset (e.g. filling missing values)
3. Visualising parts of the dataset
4. Testing the dataset

For the sake of simplicity, the section for preprocessing is omitted for the user study but it is included in the software prototype.

Thus, the canvas is divided into three sections: Dataset, Visualisations and Tests. The sections are divided by lines to show their boundaries but the lines are very subtle to indicate that the nodes in the sections interact with each other.

The canvas is split into three sections

The position of each section is chosen carefully. The sections are ordered so that the user can work from left to right and top to bottom. The usual working process is that the user first loads the dataset then visualises it and then performs tests on it. Furthermore, nodes in the section Visualisations do not have any outputs needed for tests but the output of the nodes that load datasets are needed for visualisations and tests. For this reason sections Visualisations and Tests are put above each other. Overall, the chosen arrangement of the sections creates an intuitive workflow and structure.

The chosen order of the sections makes the workflow intuitive

To get the user to use the structure, nodes in their corresponding sections have specific properties to perform the tasks they are meant for:

- Datasets: Nodes load datasets, hence they do not get inputs from any other node.
- Visualisations: Nodes display graphs and other visualisations, so these do not have any outputs that can be passed to other nodes.
- Tests: Nodes that perform tests like the t-Test but also simple data analyses like mean calculations. They have outputs that can be passed to visualisations or other tests to, for example, perform Post-Hoc-Tests.

So the user has to create a node in the section that is meant for the task, he wants to perform.

To create a node in a section the user has to click on the section's title he wants to create the node in. This opens a menu where the user can decide between using any previously saved node or creating a new one. An example of such a menu for Datasets can be seen in figure 3.5. Here, the user can create a new node by clicking on "New" or he can load any node that is listed by clicking on it.

Click on a section's name to create a node

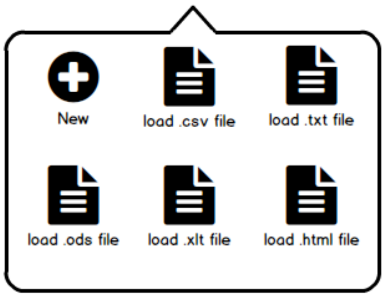


Figure 3.5: Example of a menu to add new nodes to the Datasets section in the mock-up prototype.

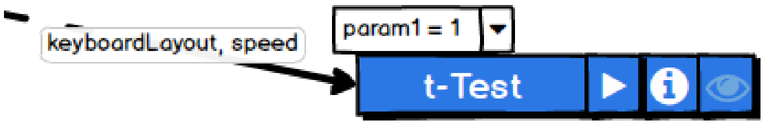


Figure 3.6: Example test-node from the mock-up prototype

3.2.3 Nodes

This section discusses the nodes. An example node can be seen in figure 3.6. At the examples of this figure, first the interactions the user can do with a node is explained and then the feedback a node gives about its state is discussed.

Interactions

A node consists of buttons, whereby each has a specific functionality

A node consists of several buttons that perform differently tasks.

The left most button contains the name. It also opens the programming section to modify the node's code which is discussed in section 3.2.4.

The next button to the right runs the node's code. After the code is executed, the user can click on the button right to it to see the feedback of the code's execution. It contains the console with the same information as the regular R console. This is primarily to detect errors in the code. If the node has run successfullyv the result can be seen by clicking on the

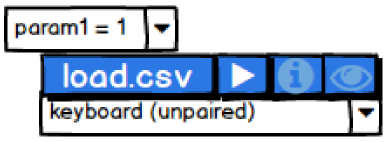


Figure 3.7: Example dataset-node from the mock-up prototype

eye-button next to it.

Besides the buttons of the node, there are two other elements the user can interact with. With these, the user can define the input values. There are two types of inputs: Inputs from other nodes and parameters which the user can define manually to allow for more general implementations of nodes. The inputs that a node gets from other nodes is specified by a label on the arrow so that the user has an intuitive connection between the input and its source. The input can be changed by clicking on the label. If parameters are used, a box above the node appears and the inputs can be changed by clicking on it.

Two types of inputs can be set on the arrows and on the box above the node

Nodes in the Dataset section do not have any nodes that are connected to them. Instead, they have another input where the dataset that should be loaded can be specified. An example input node can be seen in figure 3.7. If the user clicks on the drop-down menu below the node, he can choose a dataset that was already used or can load a new dataset form the local system.

Nodes in section Datasets have an input to choose the dataset that should be loaded

User Feedback

There are different kinds of feedback the user gets. First of all, each button on the node gives feedback whether it makes sense to click it. If this is not the case, they are disabled and can not be clicked. This prevents mistakes. On figure 3.6 the eye-button is disabled since the node has not run and it does not have any output that can be shown. But it also prevents much more time-consuming mistakes. For example, the run-button is disabled when the node's execution resulted in an error since running it again without

A button is disabled if clicking it does not make sense

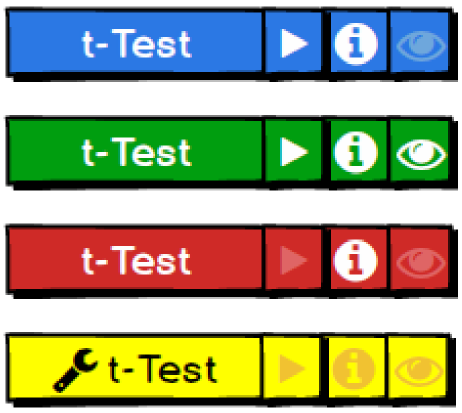


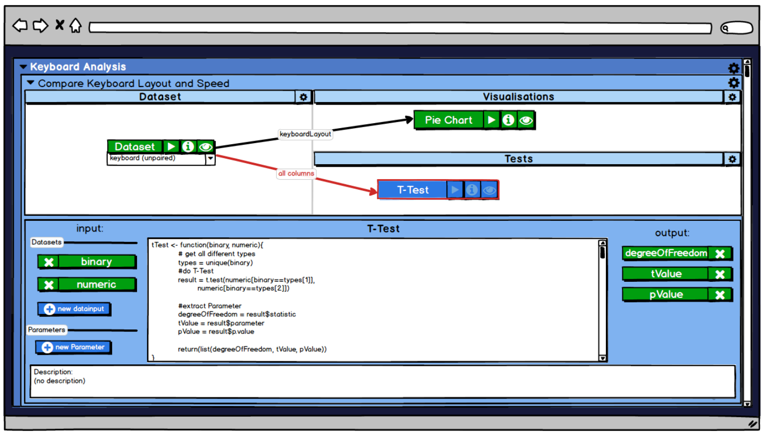
Figure 3.8: States a node can have in the mock-up prototype. Blue: ready to be executed, Green: Executed successfully, Red: Error in the execution, Yellow: Node does not have any output

any modification would not solve the problem but would cost time.

A node's colour reflects its state

To further clarify the state a node is in, we used colours to communicate it. Figure 3.8 shows an overview of the different states a node can be in. The blue state indicates that a node is ready to be executed. If that is done and it has run successfully it becomes green and if it was not successful it becomes red. If inputs or the code of these nodes are changed they become blue again showing that, to get the result of the modified node, it has to be executed again. This prototype, however, does not include a state of a running node because there is, since it is a mock-up prototype, no execution time. The node is yellow if it is newly created and running it would not give any result.

This method of indicating the state by the node's colour is simple and easy to understand but a potential problem with it is that how colours are perceived is cultural dependent. For example, in western countries red is perceived as something negative, e.g. misspelt text in an editor is underlined in red, but in the Chinese culture, the colour has a positive meaning. The frequent use of western style colour-coding in programming, however, suggests that people from other cultural backgrounds can understand it, too.



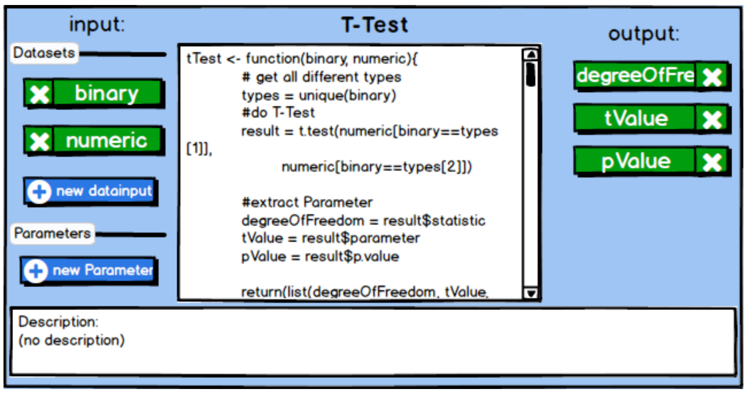
```
tTest <- function(binary, numeric){
        # get all different types
        types = unique(binary)
        #do T-Test
        result = t.test(numeric[binary==types[1]],
              numeric[binary==types[2]])

        #extract Parameter
        degreeOfFreedom = result$statistic
        tValue = result$parameter
        pValue = result$p.value

        return(list(degreeOfFreedom, tValue, pValue))
}
```

Description:
(no description)

Figure 3.9: Mock-up prototype with open programming section



```
tTest <- function(binary, numeric){
        # get all different types
        types = unique(binary)
        #do T-Test
        result = t.test(numeric[binary==types
[1]],
              numeric[binary==types[2]])

        #extract Parameter
        degreeOfFreedom = result$statistic
        tValue = result$parameter
        pValue = result$p.value

        return(list(degreeOfFreedom, tValue,
```

Description:
(no description)

Figure 3.10: Programming section of the mock-up prototype

Like the nodes, connections have states as well. Either input values are defined or not. So if not all input variables of a node are defined the arrows to it and a border around the node itself are red to indicate that.

Undefined inputs are indicated

3.2.4 Programming Section

Before the user can define input-values for a new node and execute it, the user has to write the code for that node. This is done in the programming section. Figure 3.9 shows the program with the opened programming section and figure

3.10 gives a more detailed view of the programming section.

The programming section consists of four parts

The section consists of four parts. The input area on the left, the programming editor in the middle, an output area on the right and a description on the bottom.

The input area consists of Dataset and Parameters

The input area is divided into two parts: Datasets and Parameters. The values for the inputs in the section Datasets come from other nodes and those in the section Parameters are defined manually by the user.

There are several ways to create inputs and outputs

The inputs and outputs in the interface are the same as the inputs and outputs in the function head and return statement in the programming section. There are several ways to create inputs and outputs:

- The user can write his code as he is used to and the interface updates automatically.
- The user can drag variables from within the code to the input or output section. This, however, is in the mock-up prototype technically not possible. Wherefore, it is done by double-clicking on the variable, while tooltips describe the way it is intended so the user study participants can include that in their evaluations of the prototype.
- Inputs can be created by clicking on the "new" buttons in the input section. There is no such button for outputs since the variables one wants to output are already variables in the code and dragging them to the output is more practical.

Inputs and outputs create an analogy between programming section and node

As described in the introduction, showing inputs and outputs on the side of the programming section should give the user a better overview of the function he is writing. Inputs are positioned left and outputs right because the nodes also get their inputs from the left and have their outputs on the right. This creates a perceptual analogy between the programming section and the node.

Chapter 4

First User Study

In the last chapter, we discussed the medium-fidelity mock-up prototype. This prototype is used for the first user study, which is discussed in this chapter.

The study aimed to get to know what people think about the concept and find UI flaws

The reason for this study is to get an impression of what people think of the idea and to find UI flaws to correct them in the next prototype.

We first describe the study design and then present results and discuss them.

4.1 Study Design

The study was an informal case study

The study is an informal case study. In this type of study, no explicit hypothesis is formed before the study. This allows inputs from user study participants to be analysed unbiased. This leads to a more objective conclusion.

The study had seven participants

Seven people participated that study. They were 22 to 31 years old and all but one studied in the field of computer science. The other person studied economics. However, due to difficulties of finding user study participants that have experience with R, we allowed participants who have not worked with it since is was not necessary to be able to code in R for the prototype. Likewise, we also allowed participants without knowledge in statistical analysis. Although, most participants had basic knowledge. One participant had experience with the statistical software SAS, one with SPSS and two with JMP.

There were no disturbances

The experiments were conducted in a quiet meeting room to prevent disturbance. Furthermore, they were conducted on a laptop where all possible disturbances were prevented.

The study involved four steps

The study involved four steps. At first, we explained the consent form to the user study participant and he signed it. Then we asked the participant to fill out the questionnaire which can be found in appendix A.1.

Users were given an introduction to R

In the next step, to give the participant basic knowledge of R or to remind him how R worked, we demonstrated basic functionalities of R in the command line of R Studio. It included the following steps:

1. We introduced the dataset food.csv which was used for the demonstration. It can be found in appendix A.2.1.
2. We loaded the dataset using read.csv
3. We created a pie chart using the function pie to show the distribution between the different types of food in the dataset. For that, first, sums had to be calculated for each type of food using the sum function.
4. We conducted a t-Test.

To make the demonstration appear more natural, we occasionally made mistakes on purpose. For example, we sometimes forgot to assign values or used wrong columns of the dataset. For the same reason, we took all methods that we used, e.g. the t-Test, from websites. However, we opened the websites that were used in advance to save time.

Participants performed tasks on the prototype

In the third step the datasets keyboard(unpaired).csv (Refer appendix A.2.2) and keybord(paired).csv (Refer appendix

A.2.3) were introduced and then the study participant was given a number of tasks to perform with the prototype. The task sheet can be found in appendix A.3. The task sheet also includes a description of how the t-Test works. It is used to help the user to understand how to perform a paired t-Test which was one of the tasks.

We had a discussion at the end

After all the tasks were done, we asked the participant questions about how he liked the concept. Additionally, we asked about flaws in the UI and discussed steps that the study participant did not perform well to figure out problems with the user interface.

4.2 Results and Discussion

This section presents the results of the user study. First, it discusses the feedback about the concept. Then problems with the usability of the prototype are examined. These are used to improve the high-fidelity prototype.

4.2.1 Concept

The user study participants liked the idea of the combination of a graphical flow-based interface and programming environment. However, due to the small number of participants and the qualitative nature of the results, the study can not give a conclusive answer on the popularity of the concept. Furthermore, most participants did not have much experience with statistical analysis in R. Though, all but one had good experience in programming and know how programming environments work. Thus, the results show that there are people who like the concept.

The background structure was understood well

The background structure of having nested boxes is understood well. Though there were some issues with the naming which are discussed in the next section (Refer section 4.2.2). However, one user had some initial problems understanding the purpose of that structure. This changed when he worked with the system. It suggests, however, that the

A participant had problems to understand how to utilise the background structure user would not know how to use the structure to organise his project when using the program for the first time. Users likely understand how to use the structure but might not know how to utilise it to organise their projects.

Input and output displays could not be tested

The second innovation in the design is that inputs and outputs are displayed at the side of the programming editor. This, however, could not be tested very well since the study did not include any coding. Though, the users understood the concept of it although some details were confusing. The details are discussed in the next section (Refer section 4.2.2). Thus, if the concept improves programming experience or programmers just ignore it, could not be determined in this study. But, it is a good sign that users naturally understood the concept.

The flow based canvas improves the overview when coding

The flow-based canvas was mostly perceived as a good tool for people who are not very experienced with statistical analysis since it gives a good overview of the steps in a structured way. However, many researchers do mistakes in statistical analysis ([Zuur et al., 2010], [Altman, 2000], [Daniel, 1998], [Altman, 1982], [Altman, 1980], [Dunlop and Baillie, 2009], [Cairns, 2007]). This suggests that not only beginners profit from the improved overview but also people who analyse statistics on a regular basis. Even experts who never make mistakes can profit from the overview. It reduces the thinking effort to recall the steps and makes it easier to understand analyses from other people. So the improved overview likely helps users with all kinds of experience in statistical analysis.

Participants liked the concept

Concluding, user study participants liked the concept and thought that it improves the programming experience in statistical analysis.

4.2.2 Usability

Besides the positive feedback about the concept, participants made mistakes performing tasks on the prototype. Many mistakes were done by just one user study participant and are just simple slips where the user clicked some-where he did not intend to. Though, there were a number of mistakes due to misconceptions. Most of them were done by many of the participants.

Participants had problems to change names

The first step user study participants had to do, is to create an Experiment and Workflow in the background structure and give them names. One problem all participants had, was to figure out how to change the names of Workflow and Experiment. In the prototype, this is done by clicking on the name. The intention behind it is that users could open input fields by clicking on the names and change the names in these input fields. Many participants tried to click on the cog symbol for the options to do that. However, this problem is likely solved in an implementation since, on the creation of the Workflow or Experiment, the input field would be active and a user could see how to change names.

The elements of the background structure had confusing names

The second problem was that, as mentioned above, the categories Workflow and Experiment confused some participants. This is the case because Workflow is often perceived as a single analysis. So switching the names improves the understanding.

Participants had problems to open the menu to create a node

After creating a Workflow and Experiment the participants had to create a node. To create one they had to open a menu by clicking on the name of the canvas section they wanted to create the node in. Participants had some issues to do so but many found it out after some time. So there is some need to clarify how to open the menu. It was not possible in the mock-up program, but the idea was also to open the menu on a right-click. This way to open it is more intuitive since many programs allow for actions on right-clicking. Summarising, there is some need to clarify the way the menu is opened but the functionality of right-clicking to open it makes it more intuitive.

The console button on the node is not understood

When the node was created successfully it was understood well with one exception. A few participants failed to understand that the button with the information sign on it would open the console. So the sign has to be changed for one that makes it clear.

Participants had to click too much to get results

Many study participants remarked that they had to click too much to get the results. For example, to load a dataset and view it within the program, a user has to select the dataset, click on the run button and then on the display button. For visualising something the user has to click even more. He first has to run the node that loads the dataset, then run the visualisation node and then open the results. So there is a need to reduce the amount of clicks the user has to do. Though, this can be solved by simple predictions of what the user wants to do. For example, if the user runs a node he probably wants to see the result and if the user runs a node that gets inputs from nodes that are not run these have to execute first. So there is an issue with clicking too much in the interface but it is easy to reduce the number of clicks needed.

After creating nodes and running some of them, the user study participants were asked to program their own t-Test using the programming section. They had a number of problems with that.

Participants had problems to understand different types of input

The distinction between the two input types Datasets and Parameters was confusing to many. This could be solved by finding more suitable names for the types or eliminate the distinction.

Participants overlooked error message on missing input values

Once the distinction was understood and Dataset input variables were created many users wanted to specify which column matches the inputs. However, this is not possible in the prototype due to its vertical design. This is not a problem since in a fully functional version this would be possible. Though, most forgot to specify the values for the inputs in the end. The user study participants did not understand why they could not run the node although the missing inputs were indicated by a red arrow and a red border around the t-Test node. Therefore, missing input values have to be made more obvious.

Participants did not understand how to specify inputs from other nodes

When it was understood that the input values need to be specified the participants had problems to do so. Although they had to specify the input values at the connecting arrow, many tried to set them at the inputs in the programming section. Some participants made suggestions to allow the user to enter them there. However, this is likely to lead to highly specified functions that cannot be reused in other situations. A button to open a menu at the connection is a feasible compromise. Additionally to that, how to specify the input values from other nodes has to be made clearer.

Considering all issues with the usability of the prototype, it can be said that there are many things that have to be improved. However, there is no problem that can not be solved using simple means.

4.2.3 Conclusion

The user study shows that there are many solvable issues with the prototype, but the general concept is liked by the user study participants showing that there are people who are interested in this new kind of programming environment.

Chapter 5

High-Fidelity Prototype

With the results of the first user study in mind, a high-fidelity software prototype was implemented as a website. We chose to implement it as a website because that way the distribution of Statlets is easy. A website is platform independent and can even be set up on a server to be accessible anywhere. The technical details are discussed in appendix B.

We chose to implement a website to allow for easy distribution

This chapter discusses the high-fidelity prototype. The concept is the same as for the medium-fidelity prototype. Hence, we only discuss the differences between the two prototypes.

5.1 Differences from Medium-Fidelity Prototype

The discussion of differences between the medium-fidelity and the high-fidelity prototype is structured into four subsections. First we discuss the differences in the background structure. Then we present the differences in the canvas and nodes. Finally, we discuss the changes to the programming section.

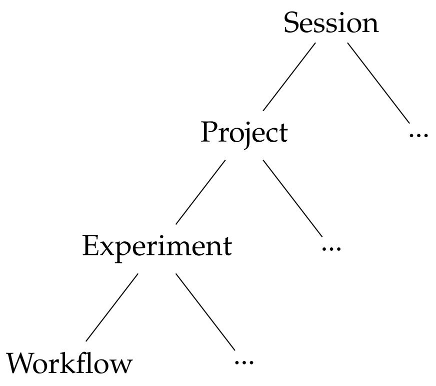


Figure 5.1: Folder-like structure of high-fidelity prototype

5.1.1 Background Structure

The background structure has an additional layer

The folder-like structure in the background is different. We added an additional layer. We also changed the names since the naming of the structure elements confused some participants. Figure 5.1 shows the new structure.

We swapped the names "Workflow" and "Experiment"

The names "Workflow" and "Experiment" are swapped compared to the medium-fidelity prototype. However, the functionalities of the sections remain the same. The user now creates an Experiment for each dataset he wants to analyse. Within the analysis of a dataset, the user creates several Workflows to analyse different aspects of the dataset.

The additional layer allows for organising projects within a session.

We added an additional layer to the structure so that the user can manage several projects in the same session of the website. Without the additional layer, it would be difficult to keep different projects separated.

Changing the name of background structure elements is likely not a problem in the high-fidelity prototype

Participants of the first user study had problems to change the names of the sections in the background structure. This is likely not a problem in the high-fidelity prototype. When a Project, Experiment or Workflow is created, a text-input area is shown. This text-input area is closed when a name is entered. Figure 5.2 gives an example of how it looks like when the text-input area is open compared to when it is closed.

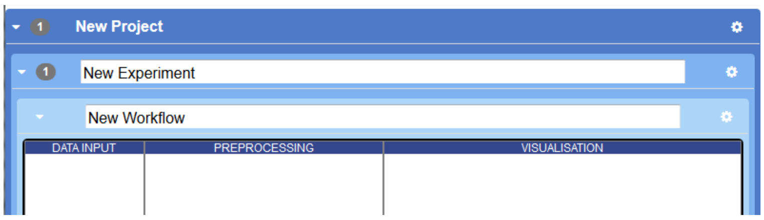


Figure 5.2: Text-input area to change the names of the background structure elements of the high-fidelity prototype. The text-input area for ”New Experiment” and ”New Workflow” are open and the one for ”New Project” is closed.

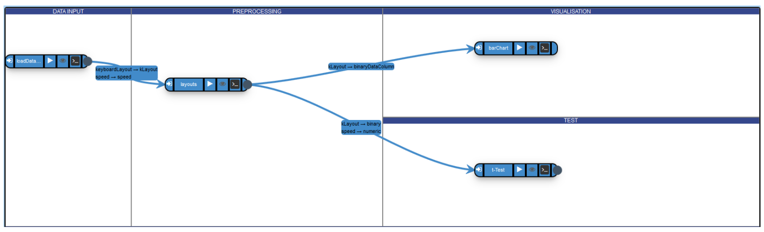


Figure 5.3: Canvas of the high-fidelity prototype

The user can reopen the input section by double-clicking the name. He can also open it in the options menu since many participants of the first user study tried to change the section's name there.

The user can reopen the input by double-clicking on the name or in the options.

5.1.2 Canvas

The only differences between the canvases of the medium-fidelity and the high-fidelity prototype are those that are already indicated in chapter 3. A picture of the canvas can be seen in figure 5.3.

As the figure shows, the preprocessing section is added to the canvas.

The high-fidelity prototype has a preprocessing section

Furthermore, the user can open the node menu by right clicking in the canvas-section that he wants to create a new node in. This is much more intuitive than clicking on the

The node menu can be opened by right-clicking.

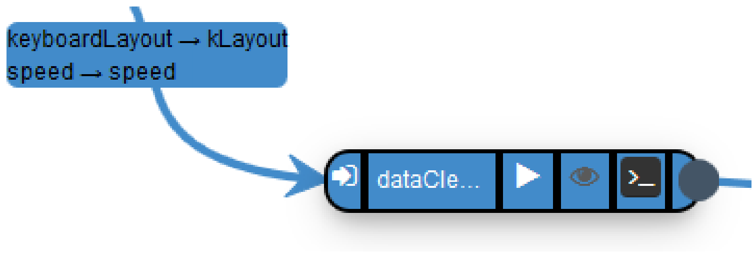


Figure 5.4: Node of the high-fidelity prototype

title of a canvas-section, which the user has to do in the medium-fidelity prototype. Additionally, on the first time a canvas is opened, a banner is displayed. This banner, alongside other information, tells the user to right-click on the canvas.

5.1.3 Nodes

There are not many differences between the canvases of the two prototypes but the nodes of the high-fidelity prototype are quite different from those of the medium-fidelity prototype. Figure 5.4 shows a node of the high-fidelity prototype.

Nodes can be dragged within their section

In the high-fidelity prototype, nodes can be dragged within the boundaries of its section. The node in the figure is a preprocessing node since it does data clearing. So it can only be moved within the preprocessing section.

The programming section is opened by double-clicking the name

To allow the node to be dragged without any of the buttons being pressed we changed the way the programming section is opened. In the high-fidelity prototype, the user has to double-click the name display of the node instead of single-clicking it.

The symbol on the button that opens the console changed

We also changed the symbol on the button that opens the console because many participants of the first user study did not understand what the button was for. The button in the high-fidelity prototype shows a standard symbol for the console. This symbol is seen in most programming en-

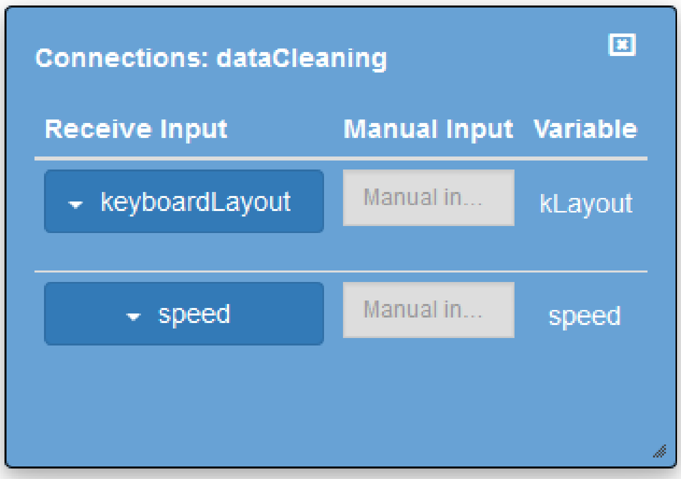


Figure 5.5: Connection menu of the high-fidelity prototype

vironments and, hence, is well known.

The medium-fidelity prototype does not have a way to connect nodes. In the high-fidelity prototype, the user can connect nodes using drag and drop. For example, the user works on a preprocessing node and wants a dataset that is loaded in a data input node as its input. He presses the left mouse button on the dot that is positioned at the right side of the data input node. Then he can drag an arrow from that dot and drop it on the preprocessing node to connect the two nodes.

The user can connect nodes using drag and drop

To use the output of another node as input, the user has to connect the nodes as described above. After that, he has to specify which part of the output should be assigned to which input variable. To do that the user can click on the leftmost button of the node. This opens the connection menu. Figure 5.5 shows an example connection menu. In this menu, the user can use the drop-down menu in the section Received Input. There he can select the column that he wants to have as an input value to the input variable that is displayed on the right side.

The output to input mapping can be specified in the connection menu

Input values can be set by using outputs of other nodes or manual inputs

The connection menu also shows a section for manual input. This section is disabled in the figure since an input is selected in the drop-down menu. If no input is selected the user can also enter the input manually. So the user has two ways to set an input value.

Without a distinction between different input types the interface is easier to understand and more flexible

The medium-fidelity prototype has a distinction between inputs that are received from other nodes and inputs that are specified by the user. However, this led to a lot of confusion among the participants of the first user study. For this reason, the distinction is removed in the high-fidelity prototype. Additionally, it is possible that an input of a node uses manual user input in one situation and input from another node in other situations. For example, consider a node that fills missing values in a dataset: The value, that is filled in, is specified by an input variable. In some cases, it is sufficient to fill the missing values with some standard value like 0. This can be entered as manual input. In other cases, the overall mean of the dataset might be better. This mean is calculated in another node and passed to the input variable. Thus, it does not make sense to separate between different types of input.

Data input nodes do not get inputs from other nodes

Data input nodes only have the option for manual input since these nodes do not get any input from other nodes. To specify the dataset that the user wants to use, he can enter the link to the dataset in the manual input.

The number of clicks needed is reduced

In the first user study, participants complained that they had to click too much to get the desired results. The high-fidelity prototype reduces the number of clicks needed using two measures, which were already indicated in chapter 4:

1. If the user clicks on the run button of a node, the results are displayed after the execution of the node.
2. If a node is run that gets inputs from nodes that have not been executed before, these input-nodes are executed first.

These measures reduce the number of clicks a user has to do and, hence, improve the workflow.

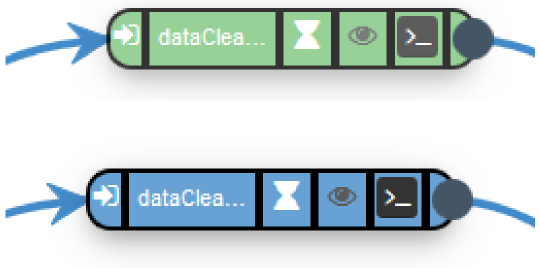


Figure 5.6: Nodes from the high-fidelity prototype that are in a state of execution. The code of the green node is executing. The blue node waits for inputs from other nodes.

Added colour-coding for nodes that are being executed

The nodes in the medium-fidelity prototype do not have any execution time since the prototype is a mock-up prototype. However, nodes in the high-fidelity prototype have execution time. To communicate that a node is executing, we added two further states to the colour-coded node states. One shows that the node is executing and the other that it is waiting for input. They can be seen in figure 5.6. To not overload the interface with too many different colours, an executing node is displayed in green but transparent and a node waiting for inputs is displayed in blue but transparent. To make clear that nodes in these states are being executed, they display an hourglass at the position of the run button.

5.1.4 Programming Section

As described above, the distinction between different input types is removed. Hence, it is also removed in the inputs of the programming section.

The inputs in the programming section have button to open the connection menu

Furthermore, the inputs, which are displayed on the left of the programming section, have additional buttons on it. If the user clicks on any button, the connection menu of the node, whose code is displayed in the programming section, opens. This is an alternative way to open the connection menu. This button is added since many participants from the first user study tried to open the connection menu there.

In the editor, the user can click on a variable. This creates an overlay above this variable. The overlay can be dragged to the inputs or outputs to add the variable to that list.

The output has to have a specific structure

The user can not program in the medium-fidelity prototype since it is a mock-up prototype. In the high-fidelity prototype the user can program. However, the output must have a specific format to work properly. An output has to have the following structure:
`data.frame(output1, output2, ... )`
However, one can also define names for columns of the output by:
`data.frame(name1=output1, ... )`
The program can also handle other outputs but only for the described output formats, it can be guaranteed that the program works properly.

Chapter 6

Second User Study

The previous chapter discussed the high-fidelity software prototype. This prototype is used in the second user study, which we will discuss in this section.

The aim of this study is to validate that Statlets can be used for statistical analysis. Additionally, the prototype is further tested for flaws in the design which should be removed in future work.

The aim of the user study is to validate that Statlets can be user for statistical analysis

We will first describe the structure of the user study. Then we will present results and discuss them.

6.1 Structure

The study is an informal case study. The question that we want to solve with the study is if Statlets can be used successfully to conduct statical analysis.

We conducted an informal case study

Two people participated in the study and they were in the same age range as participants from the first user study. The age range was 22 to 31. The field of study of both participants in the second user study was computer science. Hence, both were very experienced in programming. Additionally, they had knowledge in statistical analysis and ex-

The study had two participants that were part of the target group perience with programming in R. One also had experience with JMP. Thus, the participants in this user study were in the target user group of Statlets.

The study was conducted without disturbances

We conducted the experiments in a quiet meeting room to prevent disturbances. Furthermore, we used a laptop for the user study. The laptop had all possible disturbance sources turned off.

Participants could use the internet during the study

The internet connection remained active so that study participants could look up programming related things. This created a more realistic environment for the programming tasks.

The study involved four steps

The study involved four steps. First, we explained the consent form to the user study participant and he signed it. Then we asked the participant to fill out a short questionnaire. It is a slightly changed version of the first user study's questionnaire (Refer appendix C.1).

After that, we presented two datasets that were used for the user study (Refer appendix C.2).

Study participants were asked to perform a number of tasks on the high-fidelity prototype

Then we asked the user study participant to perform a number of tasks (Refer appendix C.3) on the high-fidelity software prototype.

After the user study participant completed the tasks, we had a discussion about the high-fidelity prototype. We discussed advantages, disadvantages and problems the user study participant had during the tasks.

6.2 Results and Discussion

Both user study participants had some problems in the beginning. However, since we did not give any introduction to the tool, this was not surprising.

The study participants had mostly different problems

Though, most of the tasks that created problems for one participant were done correctly by the other one. For example, one user study participant had problems to understand how to connect nodes with an arrow. The other participant understood that immediately but had problems to find out how to map the output of the connected node to the input of the receiving node. This, however, was done correctly by the first participant.

Both participants made slips when opening the programming section or console

Though, there were two mistakes both user study participants did. The first mistake was that both participants got confused between opening the programming section, which needs a double-click to be opened, and the console, which is opened by a single click. Both participants did occasionally do it the wrong way around. This is just a slip so it does not effect the understanding of the system but can annoy users and should be resolved.

The tooltips in the programming section have to be made more noticeably

Second, neither of the user study participants understood how overlays in the programming editor can be dragged to inputs and outputs. This is quite surprising. In both cases, the output section showed a tooltip that described how it works. When we drew attention to that tool-tip both participants understood how to drag variables to the outputs. Thus, the tooltip has to be made more noticeable.

In the end both participants dealt well with the prototype

Towards the end of the study, both participants dealt well with the prototype. They both solved the last two questions with only minor issues. Thus, over the course of the user study, the participants learned how to deal with the prototype.

This shows that the prototype and, hence, the concept of Statlets is capable of performing statistical analysis tasks. Furthermore, both participants liked the concept and said they would use a fully developed software that implements it.

Chapter 7

Summary and Future Work

The previous chapter discussed the second user study. The study showed that Statlets can perform statistical analysis tasks. This chapter summarises the thesis and discusses potential future work.

7.1 Summary and Contributions

In the first chapter, we discussed the motivation for Statlets. We pointed out that R is a popular software for statistical analysis. Then we identified problems of R:

1. It is difficult to organise collaboration in writing code in R
2. It can be difficult to integrate new methods in existing analysis procedures
3. The use of the interactive console can lead to a number of problems
4. R is difficult for beginners

5. It is easy to forget about steps that are done during the analysis, which can lead to mistakes in the interpretation of the results

Based on these problems, we introduced the concept of Statlets, which is a data flow-based user interface within a programming environment. Then we pointed out how Statlets intends to solve the problems with R that are listed above.

In the second chapter we discussed related work that influenced the concept of Statlets and affected the design of the prototypes.

The third chapter presented the medium-fidelity mock-up prototype and discussed it in detail.

This prototype was used in the first user study, which was discussed in chapter 4. We conducted an informal case study with 7 participants, who, however, did not have much experience in statistical analysis and no experience in R. The user study showed that the participants liked the concept. However, the study also discovered many design flaws.

The discovered design flaws were eliminated in the implementation of the high-fidelity software prototype. This was discussed in chapter 5. We argued how these flaws were eliminated and discussed other differences to the medium-fidelity prototype.

Chapter 6 discussed the second user study that used the high-fidelity software prototype. It was an informal case study as well. It had only two participants but these had experience in R and statistical analysis. Thus, the participants were in the target group of the Statlets. The study proofed that Statlets can be used for statistical analysis and the participants liked the idea of Statlets.

The contribution of this thesis is the development of a new concept to improve statistical analysis with R and the implementation of a prototype that can be used to test the concept. We also showed that there are people who like the idea and that Statlets can be used for statistical analysis.

7.2 Future Work

The previous section summarised the contribution of this thesis. This section describes potential future work. There are two aspects of future work: User studies to validate the tool and further implementations.

7.2.1 User Studies

The thesis demonstrated that Statlets can be used for statistical analysis and that people are interested in the concept. However, we did not test if it solves the problems that it aims to solve. To do that, further user studies have to be conducted.

7.2.2 Implementation

There are many ways to further improve the high-fidelity software prototype of Statlets.

The prototype only works for outputs that have a specific structure. Extending the prototype, so that it also works for other output formats, increases the usability.

Furthermore, an input variable of a node can only be assigned to one column from the output of an input-node. Extending the prototype, so that an input variable can be assigned to any number of columns, improves the workflow and makes integrating nodes easier.

Besides these basic improvements, Statlets can integrate a wide range of extensions. For example, it can be extended so that multiple people can work on one project at the same time.

Appendix A

First User Study

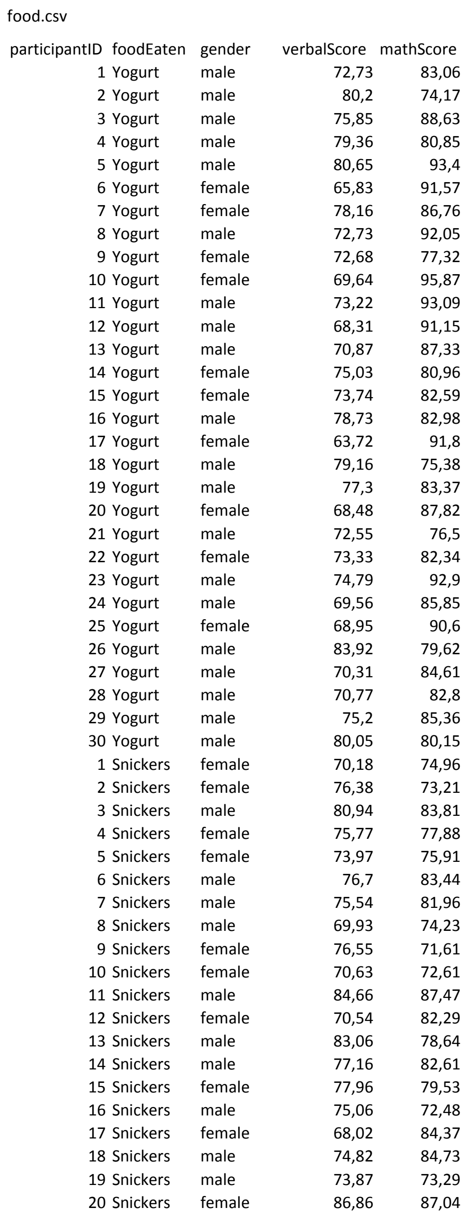
A.1 Questionnaire

Questionnaire

1. Age: ____________
2. Gender: ◯ female ◯ male ◯ Prefer not to say
3. Field of Study:
   ◯ Computer Science ◯ Psychology other: ____________
4. How experienced are you with statistical significance tests like the t-test?
   - ◯ No experience
   - ◯ Know what they do but cannot apply them
   - ◯ Can apply them but very little practical experience
   - ◯ A lot of practical experience
5. If you did statistical significance tests, what tool did you use? (multiple answers are possible)
   - ☐ SPSS
   - ☐ R programming language
   - ☐ By hand
   - ☐ Other: ____________
6. How experienced are you in programming?
   - ◯ No experience
   - ◯ I program occasionally
   - ◯ I program on a regular basis
7. How experienced are you in the programming language R?
   - ◯ No experience
   - ◯ I program occasionally
   - ◯ I program on a regular basis

A.2 Datasets

A.2.1 food



food.csv

participantID	foodEaten	gender	verbalScore	mathScore
1	Yogurt	male	72,73	83,06
2	Yogurt	male	80,2	74,17
3	Yogurt	male	75,85	88,63
4	Yogurt	male	79,36	80,85
5	Yogurt	male	80,65	93,4
6	Yogurt	female	65,83	91,57
7	Yogurt	female	78,16	86,76
8	Yogurt	male	72,73	92,05
9	Yogurt	female	72,68	77,32
10	Yogurt	female	69,64	95,87
11	Yogurt	male	73,22	93,09
12	Yogurt	male	68,31	91,15
13	Yogurt	male	70,87	87,33
14	Yogurt	female	75,03	80,96
15	Yogurt	female	73,74	82,59
16	Yogurt	male	78,73	82,98
17	Yogurt	female	63,72	91,8
18	Yogurt	male	79,16	75,38
19	Yogurt	male	77,3	83,37
20	Yogurt	female	68,48	87,82
21	Yogurt	male	72,55	76,5
22	Yogurt	female	73,33	82,34
23	Yogurt	male	74,79	92,9
24	Yogurt	male	69,56	85,85
25	Yogurt	female	68,95	90,6
26	Yogurt	male	83,92	79,62
27	Yogurt	male	70,31	84,61
28	Yogurt	male	70,77	82,8
29	Yogurt	male	75,2	85,36
30	Yogurt	male	80,05	80,15
1	Snickers	female	70,18	74,96
2	Snickers	female	76,38	73,21
3	Snickers	male	80,94	83,81
4	Snickers	female	75,77	77,88
5	Snickers	female	73,97	75,91
6	Snickers	male	76,7	83,44
7	Snickers	male	75,54	81,96
8	Snickers	male	69,93	74,23
9	Snickers	female	76,55	71,61
10	Snickers	female	70,63	72,61
11	Snickers	male	84,66	87,47
12	Snickers	female	70,54	82,29
13	Snickers	male	83,06	78,64
14	Snickers	male	77,16	82,61
15	Snickers	female	77,96	79,53
16	Snickers	male	75,06	72,48
17	Snickers	female	68,02	84,37
18	Snickers	male	74,82	84,73
19	Snickers	male	73,87	73,29
20	Snickers	female	86,86	87,04

A.2.2 keyboard(unpaired)

participantID	*keyboardLayout*	*gender*	*speed*
1	*QWERTY*	*female*	29.351469123327899
2	*QWERTY*	*female*	31.536255658614898
3	*QWERTY*	*female*	23.142238703496002
4	*QWERTY*	*female*	37.736250906521597
5	*QWERTY*	*male*	38.922907803383303
6	*QWERTY*	*female*	16.4350933129349
7	*QWERTY*	*male*	25.0069412081302
8	*QWERTY*	*female*	35.039185126200003
9	*QWERTY*	*female*	36.7668069028459
10	*QWERTY*	*female*	31.379133220879599
11	*QWERTY*	*male*	29.213378823497301
12	*QWERTY*	*male*	30.434075695010002
13	*QWERTY*	*female*	33.148329436430501
14	*QWERTY*	*male*	31.336814741787801
15	*QWERTY*	*male*	53.092222268477997
16	*QWERTY*	*male*	24.7316711957541
17	*QWERTY*	*female*	25.2512619251657
18	*QWERTY*	*male*	34.732157688555503
19	*QWERTY*	*male*	37.817765279511299
20	*QWERTY*	*male*	30.709814640848201
21	*Dvorak*	*male*	25.5005804534742
22	*Dvorak*	*male*	24.025929869944999
23	*Dvorak*	*male*	33.709664510197399
24	*Dvorak*	*female*	22.2518084126344
25	*Dvorak*	*male*	22.854318720356702
26	*Dvorak*	*male*	33.143081037138899
27	*Dvorak*	*male*	33.317359462783301
28	*Dvorak*	*male*	22.7693793558465
29	*Dvorak*	*male*	22.500426524329999
30	*Dvorak*	*male*	35.699657011792397
31	*Dvorak*	*male*	34.358168312040497
32	*Dvorak*	*female*	33.154935883434398
33	*Dvorak*	*male*	20.0172176223695
34	*Dvorak*	*male*	15.4581380606569
35	*Dvorak*	*male*	31.683192590840498
36	*Dvorak*	*female*	25.953626300393999
37	*Dvorak*	*female*	33.875135584295698
38	*Dvorak*	*male*	15.9448747430966
39	*Dvorak*	*female*	37.955648090262599
40	*Dvorak*	*male*	22.7141670941303

A.2.3 keyboard(paired)

participantID	*keyboardLayout*	*gender*	*speed*
1	*QWERTY*	*female*	29.3514691233279
2	*QWERTY*	*female*	31.5362556586149
3	*QWERTY*	*female*	23.142238703496
4	*QWERTY*	*female*	37.7362509065216
5	*QWERTY*	*male*	38.9229078033833
6	*QWERTY*	*female*	164.350.933.129.349
7	*QWERTY*	*male*	25.0069412081302
8	*QWERTY*	*female*	35.0391851262
9	*QWERTY*	*female*	36.7668069028459
10	*QWERTY*	*female*	31.3791332208796
11	*QWERTY*	*male*	29.2133788234973
12	*QWERTY*	*male*	30.43407569501
13	*QWERTY*	*female*	33.1483294364305
14	*QWERTY*	*male*	31.3368147417878
15	*QWERTY*	*male*	53.092222268478
16	*QWERTY*	*male*	24.7316711957541
17	*QWERTY*	*female*	25.2512619251657
18	*QWERTY*	*male*	34.7321576885555
19	*QWERTY*	*male*	37.8177652795113
20	*QWERTY*	*male*	30.7098146408482
1	*Dvorak*	*male*	25.5005804534742
2	*Dvorak*	*male*	24.025929869945
3	*Dvorak*	*male*	33.7096645101974
4	*Dvorak*	*female*	22.2518084126344
5	*Dvorak*	*male*	22.8543187203567
6	*Dvorak*	*male*	33.1430810371389
7	*Dvorak*	*male*	33.3173594627833
8	*Dvorak*	*male*	22.7693793558465
9	*Dvorak*	*male*	22.50042652433
10	*Dvorak*	*male*	35.6996570117924
11	*Dvorak*	*male*	34.3581683120405
12	*Dvorak*	*female*	33.1549358834344
13	*Dvorak*	*male*	20.0172176223695
14	*Dvorak*	*male*	15.4581380606569
15	*Dvorak*	*male*	31.6831925908405
16	*Dvorak*	*female*	25.953626300394
17	*Dvorak*	*female*	33.8751355842957
18	*Dvorak*	*male*	15.9448747430966
19	*Dvorak*	*female*	37.9556480902626
20	*Dvorak*	*male*	22.7141670941303

A.3 Tasks for First User Study

Tasks for User Study

1. Create a new project and experiment environment. Give them reasonable names.
2. Load and display the dataset "keyboard (unpaired)". (the dataset is saved in a .csv file)
3. Show the distribution between the different keyboard layouts in a pie chart.
4. Create a t-test to analyse whether there is a significant difference between the typing speed of the different keyboard layouts. Display the results.
5. Generalise the t-test so that it could also be used to perform a paired test. Use a variable for that to specify which form of the test is used.
6. Now we want to analyse the dataset "keyboard (paired)". That for duplicate the experiment and give it a better name.
7. Load that dataset "keyboard (paired)" and display it.
8. Change the t-Test so that it performs a paired test and show the result.

t-Test Function:

independent 2-group t-test:
$t.test(y \sim x)$
where y is numeric and x is a binary factor

dependent 2-group t-test:
$t.test(y \sim x, paired = 1)$
where y is numeric and x is a binary factor. (for an independent t-test set paired = 0)

extract values from test result: df = result$statistic, t = result$parameter, p = result$p.value

Appendix B

Technical Details

The high-fidelity software prototype is implemented based on HTML, CSS and JavaScript. Furthermore, we used jQuery ([Chaffer, 2009]) and AngularJs ([Darwin and Kozlowski, 2013]) as the basic packages.

To integrate R in the website the OpenCPU system is used ([Ooms, 2014]).

Additionally to that we used several packages for different parts of the user interface.

- The Ace code editor is used for the coding section. For an easy integration an AngularJs plugin for the Ace editor is used.
- Bootstrap is used to improve the visual design in every part of the interface
- jQuery UI ([Sarrion, 2012]) is used for all draggable and resizable elements.
- jsPlumb is used for connecting nodes

The prototype is tested on browsers Mozilla Firefox and Google Chrome.

Appendix C

Second User Study

C.1 Questionnaire

Questionnaire

1. Age: ____________

2. Gender:
   ◯ Female ◯ Male ◯ Prefer not to say

3. Field of Study:
   ◯ Computer Science ◯ Psychology ◯ Other: ____________

4. How experienced are you with statistical significance tests like t-Test or ANOVA?

   ◯ No experience
   ◯ Know what they do but cannot apply them
   ◯ Can apply them but very little practical experience
   ◯ A lot of practical experience

5. If you did statistical significance tests, what tool did you use? (multiple answers are possible)

   ☐ R programming environment
   ☐ SPSS, SAS, JMP
   ☐ Other: ____________

6. How experienced are you in programming?

   ◯ No experience
   ◯ I programmed a handful of times
   ◯ I program on a regular basis

7. How experienced are you in the programming language R?

   ◯ No experience
   ◯ I programmed a handful of times
   ◯ I program on a regular basis

C.2 Introduction to Datasets

Datasets: keyboardMale.csv, keyboardFemale.csv

participantID	keyboardLayout	typingSpeed
1	QWERTY	55.3827
2	QWERTY	61.7945
3	Dvorak	52.6922
4	Colemak	75.8594
…	…	…

C.3 Tasks for User Study

THE MEDIA COMPUTING GROUP, RWTH AACHEN

Tasks for User Study

TASKS:

1. Explore the structure of the program and set up a project that contains one statistical analysis workflow. Give names to all sections.

2. In the workflow use the saved StatLet **loadDataset** to load the dataset **keyboardMale.csv**.

3. Process the data: We only need the data of keyboard layouts "QWERTY" and "Dvorak". Reduce the data so that it only consists of rows with these keyboard layouts.

   Note: For the program to work properly the output has to have the structure:

   ```
   return(data.frame(name1=output1, name2=output2,...))
   ```

   Note: You can duplicate StatLets so it is sufficient to reduce one column at a time.

4. Use the StatLet **barPlot** to show the distribution of data between the two keyboard layouts in a bar plot.

5. Create a t-Test to analyse whether there is a statistically significant difference between the **typing speeds** of the **keyboard layouts** "QWERTY" and "Dvorak".

   Note: The system does not support returning the test result directly. Instead extract and return the results using:

   ```
   degreesOfFreedom = result$statistic
   tValue = result$parameter
   pValue = result$p.value
   ```

6. Duplicate the Workflow and change the name of the duplicate.

7. Modify the duplicate so that by changing only input parameters any 2 keyboard layouts from the dataset can be compared.
   (Layouts: "QWERTY", "Dvorak", "Colemak")

8. Using the existing structure: Load the dataset **keyboardFemale.csv** and perform a t-Test to compare the typing-speed of keyboard layouts "QWERTY" and "Colemak".

USEFUL FUNCTIONS:

For the program to work properly the output has to have the structure:

```
return(data.frame(name1=output1, name2=output2,...))
```

Select Rows with Attribute:

```
reducedData = subset(data, attrColumn == "attribute1" |
                           attrColumn == "attribute2" )
```

`reducedData` contains only rows that have any of the 2 attributes in column *`attrColumn`*.
(*`attrColumn`* can also be a column that does not appear in `data`)

Unpaired 2-Group t-Test:

```
t.test(y~x)
```

where y is numeric and x is a binary factor

Extract Values from t-Test Result:

```
degreesOfFreedom = result$statistic
tValue = result$parameter
pValue = result$p.value
```

Bibliography

popularity, 2017. URL `http://blog.revolutionanalytics.com/popularity/`. (Date last accessed 2017-02-09).

Douglas G Altman. Statistics and ethics in medical research: V–analysing data. *British medical journal*, 281 (6253):1473, 1980.

Douglas G Altman. Statistics in medical journals. *Statistics in medicine*, 1(1):59–71, 1982.

Douglas G Altman. Statistics in medical journals: some recent trends. *Statistics in Medicine*, 19(23):3275–3289, 2000.

Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime-the konstanz information miner: version 2.0 and beyond. *AcM SIGKDD explorations Newsletter*, 11(1):26–31, 2009.

Paul Cairns. Hci... not as it should be: inferential statistics in hci research. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1*, pages 195–201. British Computer Society, 2007.

Jonathan Chaffer. *Learning JQuery 1.3: Better Interaction and Web Development with Simple JavaScript Techniques*. Packt Publishing Ltd, 2009.

Larry G Daniel. Statistical significance testing: A historical overview of misuse and misinterpretation with implications for the editorial policies of educational journals. *Research in the Schools*, 5(2):23–32, 1998.

Peter Bacon Darwin and Pawel Kozlowski. *AngularJS web application development*. Packt Publ., 2013.

Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange: From experimental machine learning to interactive data mining. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 537–539. Springer, 2004.

Janez Demšar, Tomaz Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, et al. Orange: data mining toolbox in python. *Journal of Machine Learning Research*, 14(1):2349–2353, 2013.

Mark D Dunlop and Mark Baillie. Paper rejected (p¿ 0.05): an introduction to the debate on appropriateness of null-hypothesis testing. *International Journal of Mobile Human Computer Interaction (IJMHCI)*, 1(3):86–93, 2009.

Scott Faranello. *Balsamiq wireframes quickstart guide*. Packt Publishing Ltd, 2012.

Ian Fellows et al. Deducer: a data analysis gui for r. *Journal of statistical Software*, 49(8):1–15, 2012.

John Fox. Getting started with the r commander: a basic-statistics graphical user interface to r. *J Stat Softw*, 14(9): 1–42, 2005.

Ralf Klinkenberg. *RapidMiner: Data mining use cases and business analytics applications*. Chapman and Hall/CRC, 2013.

Keith McCormick, Dean Abbott, Meta S Brown, Tom Khabaza, and Scott R Mutchler. *IBM SPSS modeler cookbook*. Packt Publishing, 2013.

Robert A Muenchen. The popularity of data analysis software. *UR L http://r4stats. com/popularity*, 2012.

Robert A Muenchen. How to search for data science jobs, 2017a. URL `http://r4stats.com/articles/how-to-search-for-data-science-jobs/`.

Robert A Muenchen. The popularity of data analysis software, 2017b. URL `http://r4stats.com/popularity`. (Date last accessed 2017-02-15).

Stephen Oney and Joel Brandt. Codelets: linking interactive documentation and example code in the editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2697–2706. ACM, 2012.

Jeroen Ooms. The opencpu system: Towards a universal interface for scientific computing through separation of concerns. *arXiv preprint arXiv:1406.4806*, 2014.

Eric Sarrion. *JQuery UI*. " O'Reilly Media, Inc.", 2012.

Alain F Zuur, Elena N Ieno, and Chris S Elphick. A protocol for data exploration to avoid common statistical problems. *Methods in Ecology and Evolution*, 1(1):3–14, 2010.

Index

abbrv, *see* abbreviation

Codelets, 18

Deducer, 16
design process, 19
DIA Cycle, 20

first user study, 31–37
flow-based data analysis software, 17–18
future work, 53

general purpose programming languages, 15
Gestalt law, 9, 22, 28, 30

high-fidelity prototype, 39–46

IBM SPSS Modeler, 17
IBM SPSS Statistics, 13–14

JMP, 14–15

KNIME, 17

medium-fidelity prototype, 19–30

Orange, 17

popularity of R, 1–4
problems, 4–7

R Commander, 16
R programming language, 1–4
R Studio, 15
RapidMiner, 17

SAS, 14–15
second user study, 47–49
Stata, 14
Statistica, 14

Typeset March 31, 2017