TREESTA: A SYSTEM FOR SUPPORTING STATISTICAL ANALYSIS USING ANOVA

by

Peyvand Mohseni
B.Sc. Sharif University of Technology, Tehran, Iran, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

In the School of Computing Science

© Peyvand Mohseni 2008

SIMON FRASER UNIVERSITY

Fall 2008

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without permission of the author.
APPROVAL

Name: Peyvand Mohseni
Degree: Master of Science
Title of Thesis: Treesta: a System for Supporting Statistical Analysis using ANOVA

Examiners Committee:

Chair: Dr Jian Pei
Associate Professor

Dr. Arthur E. Kirkpatrick
Senior Supervisor
Associate Professor

Dr. John Nesbit
Supervisor
Associate Professor

Dr. Fred Popowich
Internal Examiner
Professor

Date Defended/Approved: 3 November 2008
Declaration of Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website <www.lib.sfu.ca> at: <http://ir.lib.sfu.ca/handle/1892/112>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada

Revised: Fall 2007
STATEMENT OF ETHICS APPROVAL

The author, whose name appears on the title page of this work, has obtained, for the research described in this work, either:

(a) Human research ethics approval from the Simon Fraser University Office of Research Ethics,

or

(b) Advance approval of the animal care protocol from the University Animal Care Committee of Simon Fraser University;

or has conducted the research

(c) as a co-investigator, in a research project approved in advance,

or

(d) as a member of a course approved in advance for minimal risk human research, by the Office of Research Ethics.

A copy of the approval letter has been filed at the Theses Office of the University Library at the time of submission of this thesis or project.

The original application for approval and letter of approval are filed with the relevant offices. Inquiries may be directed to those authorities.

Bennett Library
Simon Fraser University
Burnaby, BC, Canada
ABSTRACT

Existing statistical packages provide statisticians with a rich set of functions, but when it comes to organizing the work these tools simply present a linear history of the user’s activity. Treesta is a system that supports a more informed process of statistical analysis. It is a text-based interface that stands between the user and Matlab, and serves as an organizational, as well as representational, tool for the process of statistical analysis. As an organizational tool, Treesta categorizes the user’s work into a tree of workspaces, with their own variables, figures, and history. Moreover, the user can annotate each workspace. As a representational tool, Treesta depicts the whole environment and summarizes each workspace. A qualitative user study with a single experienced Matlab user confirmed our assumptions about usefulness of the tree structure, the ability to follow various paths, the annotation tool and the presentation of the structure of the environment.

Keywords: data analysis, ANOVA, statistical inference, supporting creativity, supporting problem-solving
To mom, dad and Javaneh, the most precious ones.
ACKNOWLEDGEMENTS

It is impossible to thank all the people who have had a share in who I am now, since I am pretty sure that oblivion will not allow me to name all, therefore, just to be fair, I mention those who were more relevant to this accomplishment. I am hugely indebted to my supervisor Dr. Ted Kirkpatrick, whom I truly admire not only as an estimable mentor, but also as an honest person and a supportive friend. He helped me learn the skills to excel in life and research, always encouraging me not to accept less than the best. His wisdom and spirit were invaluable along this path.

Moreover, I want to thank Dr. John Nesbit and Dr. Fred Popowich whose valuable knowledge shed light on improving this project. I also appreciate the help of two graduate students in GRUVI Lab in evaluating Treesta. In addition, I am grateful to my friends for accepting and loving me for who I am, making home seem closer. I understand the value of their never-fading friendship and support.

Most importantly, I would like to thank my family, who always love and support me and help me be a better and happier person. Their belief is the reason for me to never lose hope. I love them deeply.
TABLE OF CONTENTS

Approval .................................................................................................................. ii
Abstract.................................................................................................................. iii
Dedication ............................................................................................................... iv
Acknowledgements ............................................................................................. v
Table of Contents ................................................................................................. vi
List of Figures ....................................................................................................... viii
List of Tables ......................................................................................................... ix

1 Introduction ....................................................................................................... 1

2 Statistical Analysis ............................................................................................ 4
  2.1 Description .................................................................................................... 4
  2.2 The Process of Statistical Analysis ............................................................... 6
  2.3 Computer Support for Statistical Analysis .................................................... 8
  2.4 Current Advice for Graduate Students in Research Methods ....................... 10

3 Supporting Creativity in Computer-based Environments .................................. 12
  3.1 Definition of Creativity ................................................................................ 12
  3.2 Computer Support for Creativity ................................................................. 14
    3.2.1 External Representation ...................................................................... 18
    3.2.2 Capturing History ............................................................................. 20
    3.2.3 Supporting Reflection in Analysis ....................................................... 21
    3.2.4 Simultaneous Multiple Paths ............................................................... 23

4 Treesta's Design and Rationale .......................................................................... 28
  4.1 Main Services of Treesta ............................................................................ 30
    4.1.1 Capturing History ............................................................................. 30
    4.1.2 External Representation .................................................................. 31
    4.1.3 Supporting Reflection in Analysis ...................................................... 32
    4.1.4 Simultaneous Multiple Paths ............................................................. 32
    4.1.5 Replaying History .......................................................................... 34
    4.1.6 Dependency among Variables ........................................................... 35
    4.1.7 Sharing a Variable ........................................................................... 35

5 Detailed Description of Treesta's Design ........................................................... 37
  5.1 Two Novel Functionalities of Treesta ........................................................... 39
    5.1.1 Replaying History ............................................................................ 39
    5.1.2 Dependency among Variables ........................................................... 43
5.2 Commands Related to Viewing or Modifying a Single Workspace ..........48
5.3 Commands Related to Viewing or Modifying the Whole Environment...50
5.4 Commands for Managing Relations between Two Workspaces ..........51
5.5 Commands for Moving between Two Workspaces ..........................51
5.6 Treesta’s Implementation ..................................................................56
6 The Evaluation of Treesta ....................................................................58
   6.1 Participant ..................................................................................58
   6.2 Task .........................................................................................58
   6.3 Protocol ....................................................................................59
   6.4 Results .......................................................................................60
   6.5 Discussion ..................................................................................61
   6.6 Notes from the Pilot Study ..........................................................63
   6.7 Improvements to Treesta Suggested by the Evaluation .................64
7 Conclusion ..........................................................................................66

Bibliography ..........................................................................................69
LIST OF FIGURES

1 The Workspace of Matlab ................................................................. 9
2 The Dependency Algorithm .............................................................. 42
LIST OF TABLES

1  The Dataset with 12 Observations and a Single Focal Factor with 4 Levels........39
2  Number of the Fos Nuclei for Rats in the Three Conditions ..........................51
3  Number of the Fos Nuclei for Rats from the Second Experiment.......................52
CHAPTER 1:

Introduction

There are many computer systems that assist experts in different ways, such as writing a document, editing a picture or generating a piece of programming code. Although these tools support creating the end products, they are hardly any help in the process of designing and creating those products, and more generally solving a problem. We need a system that goes beyond supporting the creation of the final product to promoting the performance of experienced users while developing those products.

Experts in all domains follow a similar behavior, which is quite different from how novices solve problems; an example of this is reported in (Akin, 2001) about architects. Experienced users invoke different ideas, and follow several solutions, not committing to a single path early in the process. As they move forward, they refer to help from a collection of related examples, and opinions of other people working in that field. This certain behavior can be assigned the label of creativity in problem solving, and it becomes much more prominent for ill-structured problems that do not follow a well-structured structure and do not have a single correct solution. In solving this family of problems, expert users do not just follow a sequence of steps towards the final state, they rather have to consider many options, decisions and tradeoffs in choosing their approach and no software exists to help them.
An instance of the mentioned problem exists in the realm of statistical analysis. There are powerful statistical packages in the market that can carry out complex calculations in a matter of seconds. These tools provide various plots and tests to be applied on a dataset, which are otherwise impossible or very troublesome for the analyst to compute by hand. But there is the same limitation here as well; no system exists that can support the process of data analysis itself.

Since existing software represents only a list of the commands the statisticians have entered, they have to invent their own methods to organize their analysis work. The users have to somehow keep a record of their datasets, their hypotheses, and the progress they have made so far, using for instance Microsoft Word or Excel documents. I have developed a system, Treesta, which supports the expert performance in the field of data analysis. Treesta is a text-based interface that stands between Matlab and the user, adding some services to what Matlab offers.

Treesta organizes the ongoing analysis into a tree of workspaces, with the ability to associate a name, variables and figures with each workspace. The analyst can also annotate each workspace in case of the need to explain or clarify something for his own understanding. Treesta also offers commands to view and maintain the environment, share variables among workspaces, and to repeat some part of the history of commands, in order to apply the same commands on a new variable.
The structure of this thesis is as follows: some definitions and discussions on statistical analysis are presented in chapter 2, along with a list of existing systems in this area. In chapter 3 I present a theoretical background on what creativity is and how it can be supported in problem solving, followed by a set of features useful for information systems designed to support creative performance. So far, no system has been designed to support performance of the process of data analysis. Treesta brings these two areas together, applying the most relevant creativity-supporting features to support the process of data analysis.

The idea behind Treesta and its rationale is presented in Chapter 4, and the detailed description of how it works, and its set of commands is discussed in Chapter 5. To evaluate the system, I have conducted a user study with a single participant, experienced in Matlab, reported in chapter 6, which verified most of our assumptions about assisting the process of statistical analysis. Chapter 7 concludes the thesis, discussing our findings about what works in Treesta and what does not, along with the potential areas for improvement.
CHAPTER 2:

Statistical Analysis

2.1 Description

Treesta supports the process of data analysis, assisting the users to organize their work and show a more efficient performance. Therefore, it is useful to open the discussion with an overview of statistics and data analysis, followed by the areas that require data analysis, and a summary of the existing tools, and how, in spite of their great help, they have deficiencies in supporting the process of data analysis.

Authors have defined statistics in many different ways; let us begin by a community definition in Wikipedia, since it evolves based on consensus among statisticians: “Statistics is a mathematical science pertaining to the collection, analysis, interpretation or explanation, and presentation of data... It is applicable to a wide variety of academic disciplines, from the natural and social sciences to the humanities, and to government and business” (“Statistics”, 2008). Statistical data analysis is a sub-topic of statistics that deals with analyzing collected data in order to summarize information to make it more usable. It also deals with making generalizations about a population based on a sample drawn from that population.
Analyzing data generally happens after an experiment has been carried out and data has been collected. According to Kirk (1982, p. 1-2), experimental design is "a plan for assigning experimental conditions to subjects and the statistical analysis associated with the plan". He continues to mention that this plan to test a scientific or research hypothesis involves a number of interrelated activities such as; determination of the experimental conditions (independent variables) and the extraneous conditions (nuisance variables), specification of the number of experimental units required, and the procedure for assigning conditions to subjects.

Experiments like the ones mentioned above can only be devised for research questions that are likely to arrive at an answer, not for those without any feasible way to verify; for example the question "Can three angels sit on the head of a pin?" is not a suitable research question. An experiment involves manipulation of one or more independent variables (IV) in order to determine the effect of this manipulation on a dependent variable (DV). In addition to these, almost all experiments include one or more nuisance variables, undesired sources of variation in an experiment that may affect the dependent variable, and should be controlled.

The antithesis of an experiment is a non-experiment. According to Tabachnick and Fidell (2001, p. 2), a critical difference between experimental and non-experimental research is whether the researcher manipulates the levels of IVs. In non-experimental research, unlike an experiment, the levels of the IVs are not manipulated by the researcher. The researcher can define the IV (e.g., the geographic area of residence), but has no control
over the assignment of subjects to levels of it. In this type of research, it is much less certain to attribute causality to an IV. Even if there is a systematic difference in a DV associated with levels of an IV, the two variables are said to be related, but the cause of the relationship is unclear.

While analyzing data, statisticians construct and test models of their data using various inference methods. One frequently used method is analysis of variance (ANOVA), which is used to reveal the potential effect of the focal factors on the dependent variable in an experiment or a non-experiment with a categorical focal factor.

ANOVA compares two or more means to see if there are any reliable differences among them. It is a set of analytic procedures based on a comparison of two estimates of variance. One estimate comes from differences among scores within each group, and the second one comes from differences in group means. If these estimates do not differ appreciably, one could conclude that all group means come from the same distribution, and that the slight differences among them are due to random error. If, on the other hand, the group means differ more than expected, it can be concluded that they were drawn from different sampling distributions, and the null hypothesis that the means are the same is rejected (Tabachnik & Fidell, 1996, p. 37).

2.2 The Process of Statistical Analysis
Data analysis can be divided into exploratory data analysis and confirmatory data analysis, where the former focuses on discovering new features in the data (Tukey, 1977), and the latter on confirming or falsifying existing hypothesis. To the best of my
knowledge, there has never been a study on the work practices of statisticians. To find out how statisticians analyze data requires a longitudinal study, and lacking this knowledge, I quote famous statisticians. However, there are studies that have looked into the more general topics of creative problem solving which will be referred to later in the thesis.

Tukey believes that statistical analysis is not just about applying some pre-determined tests on a dataset, and proving or rejecting some hypotheses, and to concentrate on confirmation - to the exclusion or submergence of exploration – is an obvious mistake. He suggests that instead of being bound by preconceived notions and analyses, we should flexibly look around (Tukey, 1968).

He also quotes one of his friends in the same profession that: “Following a rule book for research seems to stimulate the attack on trivial problems. The great challenge is to teach investigators … not to be too tightly bound in the formulation by a preconceived model of research design” (Tukey, 1968, p. 15). This friend continues to suggest that intelligent problem formulation should be motivated, without depending on a single model of statistics, or some standard textbook solutions.

Looking at data analysis from these statisticians’ points of view, demands creativity as part of a successful analysis process. Tukey believes that a dataset should usually be analyzed in more than one way. He also recommends to look around, be flexible in handling data, and to remember that we learn by taking chances, by trial, with some
errors (Tukey, 1968). Possible ways for a statistical tool to promote these suggestions would be to support multiple paths and also reflection in data analysis. This way the analyst can simultaneously follow various solutions, and by reflecting on them regularly, decide which ones are more appropriate or more likely to lead to the most significant result.

2.3 Computer Support for Statistical Analysis

In order to analyze a dataset successfully, it is wise to get help from the systems intended to assist in analyzing data in computer environment, for example, packages like Matlab, SPSS, Maple and Mathematica. The main functionality of these tools is a rich set of statistical functions, which provide different tests, and transformations for analyzing different possible datasets. With these software packages, analysts are able to apply all these commands on their data without getting caught up in complicated calculations.

Despite their powerful mathematical engines, these systems do not provide much help in the process of data analysis. Users need to organize their work using their own methods, keeping notes about what has been done and what is still ahead. In terms of recording the ongoing job, the current software only presents a list of the entered commands, as shown in a snapshot of Matlab’s workspace, in which the commands are grouped chronologically, while they might belong to different solutions to a single problem (see Figure 1).
The second category of computer tools for data analysis includes ongoing work that tries to improve computer-based statistical analysis in other ways, such as data visualization or interaction between users and the software. These projects investigate how to present data in a manageable and efficient way so analysts can discover the underlying information like relations between factors and distributions more easily.

One such project, SocialAction, is an interactive tool supporting exploratory data analysis of social networks (Perer & Shneiderman, 2008). Visualizing output makes it easier to find patterns and outliers in the output from social network analysis; however, the overlapping nodes and tangled edges in a huge collection introduce difficulties. The strategy in SocialAction is to integrate statistics and visualization; statistics help to detect important individuals, relationships, and clusters, which will then be visualized in a
network, in turn facilitating sensemaking and discovery of distributions, trends and outliers. Users are able to filter nodes and edges dynamically.

Time-ART is another tool that is specialized in analyzing video-taped data of an experiment (Yamamoto, Nakakoji, Aoki, 2000). Using a two-dimensional positioning approach, it helps a user understand what is going on in a video-taped session. This tool allows the user to segment any portion of the video clip and position it in a free two-dimensional space. It also lets the user annotate how this portion is related to others, which portions are supporting a hypothesis, or which portions need more careful analyses.

According to Tukey an analyst should take risks, try different solutions on the data to discover new relationships (Tukey, 1968). We noticed that no existing software promotes these recommended practices in the process of data analysis. The suggestions for a better statistical package belong to a more general set of characteristics that is aimed at creative problem solving. This concept and the possible ways it can be supported by information systems are discussed in the next chapter. By implementing some of these properties, Treesta improves the process of data analysis.

2.4 Current Advice for Graduate Students in Research Methods

When it comes to data analysis, graduate students studying research methods are advised to follow some methods to keep the job organized and efficient. First, they keep the names of the variables as short as possible, so the complete name can be shown in the columns while entering the dataset in existing statistical packages. On the other hand, in
similar cases, such as programming, students are encouraged to assign meaningful names to variables, even if they are long, which results in higher readability and easier future reference. Second, in order to keep a record of their variables, research methods students try to add variables, and never remove or alter the amount of a variable.

Using Treesta, the analysts do not need to develop methods to organize and record their work. The names of the variables can be meaningful names, even if they are long, without causing any problem in printing them on the screen. Moreover, the user can change the amount of a variable, instead of adding new ones and being finally confused among so many variables. The old amounts of variables are securely saved in the ancestor workspaces and can be retrieved at any moment.
CHAPTER 3:

Supporting Creativity in Computer-based Environments

3.1 Definition of Creativity

There are a lot of research projects that look into building tools that can support creativity and innovations, called Creative Problem Solving Environments (CPSEs) by Hewett (2005). What are the traits of such systems? To answer this question, it is reasonable, first, to define creativity, and to decide whether it is possible for software to promote creativity.

Franken (1994) says that: “creativity is defined as a tendency to generate or recognize ideas, alternatives, or possibilities that may be useful in solving problems, communicating with others, and entertaining ourselves and others” (p. 396). He also mentions that: “In order to be creative, you need to be able to view things in new ways or from a different perspective. Among other things, you need to be able to generate new possibilities or new alternatives.” (p. 394).

There are different definitions of creativity; however most of them suggest bringing a new idea, method, or product to this world, through evaluating various possibilities. The end result should be feasible as a solution in that domain. Having defined creativity, let us get back to our initial concerns, which are the likelihood to support creativity and the
features of tools that can help with that. It is a concern for a group of scholars whether it is an achievable goal to support creativity. Hewett (2005, p. 5) suggests that: “While it is clearly the case that we can not command creative insights or products, there are clear indications in the psychological literature that it is possible to create conditions that will improve the possibility of creative results, even if only by avoiding conditions that are known to disrupt or to work against creativity”.

He lists several studies featuring interviews of creative individuals, to find out factors that can lead to innovation, and concludes that there are in fact several “learnable” and “teachable” techniques that can enhance creativity. Shekerjian (1990) reports that people whom she has interviewed suggest finding your talent, honing it and being willing to take risks. Gardner (1993, p. 32) adds to this his opinion that creativity in the absence of domain expertise just does not seem to happen.

Furthermore, Hewitt (2005) suggests that frequently creative products happen to be the result of normal cognitive processes that try to solve a problem harder than or different from what was expected. In this manner, newer solutions are built on previous unsuccessful solutions, and this evolution gradually leads to a creative product, which is both innovative and appropriate for the problem at hand. Therefore, even when the result may seem extraordinary, it is merely the outcome of using existing knowledge base and extending it to broader ranges.
Shneiderman (2007) adds some characteristics to this list. When it comes to creativity, he recognizes three schools. The first one is the structuralists, who believe that following an orderly method, usually composed of stages like preparation, incubation, illumination and verification, promotes creativity. The second school consists of inspirationalists, who believe that breaking away from familiar structures brings about that Aha! moment. Finally, there are situationalists, who argue that creativity is social. They try to understand the motivation of creative people, their history and relationships with mentors and teachers. Each of these schools has suggestions about characteristics of a CPSE, which will be mentioned later in this section.

Another possible way to support creativity is to encourage insight, which is a key ingredient of most creative works. Insight, as Hewitt (2005) suggests, is a flash of recognition of a new relationship or a new organization of knowledge. Csikszentmihalyi and Sawyer (1995) argue that insight requires total immersions in the problem, and communication with others. They also list some conditions under which insight is unlikely to occur, such as lack of motivation, time for reflection and opportunity to test possibilities.

3.2 Computer Support for Creativity

Returning to the topic of supporting creativity by computer-based environments, we should not forget that these are still tools, and the act of creativity is carried out by humans. And although it is impossible to guarantee creativity, the working environment should not inhibit it.
A key element to support creativity through fostering insight is an archive of the previous similar problems that can serve as analogs to the future ones (Dominowski & Dallob, 1995). However, this is not the only way, since insight can arise in many different ways as Mayer has reported (1995, p. 1). He views insight as completing a schema, reorganizing visual information, reformulating a problem, overcoming a mental block, or finding a problem analog. This diversity implies the development of environments that are tailorable by individuals.

Different researchers mention different properties for a creativity-supporting information system, and there are various systems featuring some of these properties. From these efforts to identify the properties that support creativity it is possible to extract a subset that can be useful in designing CPSEs.

- The system should let the user follow a formal, well-defined structure, as suggested by structuralists (Shneiderman, 2007).
- The system should let the user take a holistic view, that is to be able to step back and look at the whole picture, as suggested by inspirationalists (Shneiderman, 2007), and also Candy and Edmonds (1997).
- The system should let the user temporarily suspend judgment on any matter (Candy & Edmonds, 1997).
- The system should let the user explore multiple solutions, compare them and revert to previous states and goals if required, as suggested by inspirationalists (Shneiderman, 2007) and also Candy and Edmonds (1997).
• The system should let the user make unplanned deviations (Candy & Edmonds, 1997).

• The system should let the user formulate problems as well as solve them (Candy & Edmonds, 1997).

• The system should let the user reformulate the problem space as the conception of the problem or the work in progress evolves over time (Candy & Edmonds, 1997).

• The system should let the user save the history, edit it, and replay it with different parameters (Shneiderman, 2007).

• The system should automatically log the intermediate steps and paths that have led to current situations, since they can help in recapturing the choices that have been made and why they have been made, as suggested by situationalists (Shneiderman, 2007). In this way, we have access to more accurate and thorough information while sharing an interesting problem with others working in that field.

• The system should provide a library of previous works and analogs, along with multiple store and find operations and multiple access routes into information repositories (Candy & Edmonds, 1997).

• The system should have a user manual, a help library; which has a table of contents and an index. It should specify its purpose, content, and structure (Hewett, 2005).

• The system should support hyper-linkage. Humans keep a repository of general information in their memories, rather than specific details, and it would be very
helpful if they can utilize this with some environmental clues while working with the system, and this is the place that these links appear to be useful. They can provide users with examples of tool’s usages, and can cross-reference to other related items or material (Hewett, 2005).

- The system should have an online report and annotation form. With every complex system comes the problems and bugs, and it needs to be fast and easy for users to file these problems online, otherwise they will not do that. It is also helpful that this report be accompanied by an automatic file about the context in which this problem occurred, because often it is difficult to remember the exact steps leading to a specific situation. Moreover users should be able to record any commentary, suggestion, note of an improvement for an existing tool, or a new one the moment it comes to their mind (Hewett, 2005).

- The system should enable communication and collaboration, since while the Aha! moment is personal, the history leading to that is very social, as suggested by situationalists (Shneiderman, 2007).

- The system should be easy for novice users to work with, provide ambitious functionalities for expert users, and support a wide range of functionalities. This property is what Shneiderman (Shneiderman, 2007) describes as low thresholds, high ceilings and wide walls.

Let us now discuss several instances of a few of these characteristics that have been the objects of attention the most, and then for each of them some research efforts will be presented.
3.2.1 External Representation

External representations are the knowledge and structure in the environment, as physical symbols, objects (e.g., written symbols, beads of abacuses), and as physical rules, constraints and relations embedded in physical configurations (e.g., physical constraints in abacuses). In contrast, internal representations are the knowledge and structure in memory, which can be transformed into external representations through externalization.

Zhang (1997) emphasizes on the importance of external representations. He claims that these representations are not simply inputs and stimuli to the internal mind, but also memory aids, that can extend working memory, form permanent archives, and allow memory to be shared. Stenning and Oberlander (1994, p. 1) focus on graphical representations, such as Euler circles, and argue that they limit abstraction and thereby aid processibility.

Bruner (1996, p. 23) describes that externalization “produces a record of our mental efforts, one that is ‘outside us’ rather than vaguely ‘in memory’ … It relieves us in some measure from the always difficult task of ‘thinking about our own thoughts’ while often accomplishing the same end. It embodies our thoughts and intentions in a form more accessible to reflective efforts”.

There are different benefits to external representations, which are mainly the results of limitations of internal images. The most obvious one is the role that external representation plays in communication; to convey your thoughts to other people you have to depict it externally, through gestures, words, and schemas. Another functionality
results from the limitation of memory's capacity; since you cannot keep everything in your mind ready-at-hand you have to store some information outside your mind, or get external help to complete a task (e.g., using pen and paper to do a complex calculation).

There are also other subtler advantages to externalization, as suggested by Reisberg (1987). One of the benefits of external representations is that they give us access to knowledge and skills, which are otherwise unavailable to us. For example in trying to spell a rarely encountered word, one writes down two candidate spellings on a piece of paper, then judges which one "looks right". In this case, the perceptual module of one's cognitive functioning knows the correct spelling, but this is inaccessible directly. Therefore, one can gain access to this knowledge indirectly by providing input to the module and observing its output.

The second benefit is what Reisberg (1987) calls the non-intensionality of external representations. External representations are open to interpretation by others; even the creator of them has the option of setting his intention and understanding of them aside and looking at them anew. This opens up the possibility of different interpretations of the same representation. Reisberg argues that this potential ambiguity is not present in internal representations.

The final advantage of external representations is discovering omissions, as Reisberg (1987) puts it. We think of internal representations as being rich and detailed, but aspects that are not attended are undetailed and vague, and it surprises us when we discover these
areas of vagueness. As an example, imagine one is encouraged to form a mental image of a word, (e.g., “pumpkin”), and is asked to read the letters backwards (e.g., n-i-k…). This task is surprisingly difficult, even if the mental image has been very detailed and clear, because the mental image was much less complete than the corresponding picture of it.

This discussion highlighted the significance of external representations for problem solving and communication, and the same case applies in the realm of supporting creativity. Working with some kind of representation is a characteristic of the people who have produced creative results, and this representation can take forms other than pure mental. Hewett (2005) argues that professionals use tools such as drawing, making models, and taking pictures as a way of extending memory, self-education and experiment, as well as communication with others.

3.2.2 Capturing History

In the domain of capturing history we have a broad range of tools, from the simplest one like a “Save as …” command to much more complex systems.

Klemmer, Thomsen, Phelps-Goodman, Lee and Landay (2002) introduce Designers’ Outpost, a wall-scale, tangible interface for collaborative web site design. This system captures the design history, and makes it accessible through three mechanisms: a main timeline, a local timeline, and a synopsis view. The main timeline presents a visually navigable set of design thumbnail organized on a timeline. This view can be filtered by activity (e.g., by actions, or meetings), or by properties (e.g., by time or author). The
local timeline enables a designer to access the history of a single object, and the synopsis view enables post-design review of the key bookmarks.

Another example is the temporal model developed by Edwards, Igarashi, LaMarca and Mynatt (2000). In this work, the authors present two extensions over existing tools for managing and representing application timelines. The first extension represents the history as a set of atomic, incremental operations, to better support causality effects, and the second extension represents a multi-level history, which enables users to manipulate histories of objects independently from one another.

Chimera, developed by Kurlander and Feiner (1990), depicts an editable graphical history. This history allows the user to review, undo, or reapply the actions performed so far. The important events in the history of a user’s session are depicted in a chronological order based on a pictorial metaphor borrowed from comic strips.

3.2.3 Supporting Reflection in Analysis

Reflection is a notion introduced first by Schön (1983). He divides this into reflection in action and reflection on action. The former, which is sometimes described as ‘thinking on our feet’, denotes the reflective processes that take place while practicing and externalizing. The work the practitioner is involved in, talks back to him, and simultaneously the practitioner talks back to the work, by applying changes. Schön expresses this situation as follows: “The practitioner allows himself to experience surprise, puzzlement, or confusion in a situation which he finds uncertain or unique. He reflects on the phenomenon before him, and on the prior understandings, which have
been implicit in his behaviour. He carries out an experiment which serves to generate both a new understanding of the phenomenon and a change in the situation.” (p. 68).

On the other hand, reflection on action depicts the reflective processes that happen later, after the encounter, when a practitioner sees a resulting representation. In this case, the practitioner is not doing anything constructive, but simply viewing the representation and reflecting on it. The act of reflecting on action enables us to spend time exploring why we acted as we did, during which we develop a set of questions and ideas about our activities and practice.

This concept of reflection has been considered in several works including the work that has been carried out by Nakakoji, Yamamoto, Reeves and Takada (2000). Amplifying Representational Talkback (ART) is a system that uses two-dimensional positioning to support early stages in writing. The user can move objects, pieces of the document, in a two-dimensional workspace as a means of reflecting on the writing job. The authors argue that writing is not a clean top-down process where one elaborates the pre-defined structure from chapter, to section, to paragraph, to sentence. In fact, it should be viewed as a design process where the writer alternatively identifies structure and generates content.

Terry and Mynatt (2002) have explored a similar concept. Side Views is a user interface mechanism that provides on-demand and dynamic previews of commands. It allows for clarifying, comparing, and contrasting commands; generating alternative visualizations;
and experimenting without modifying the original data (i.e., "what-if" tools.). Its application is mostly rewarding in dealing with open-ended tasks, which do not have a clear, predefined sequence of steps that guarantee arriving at the final solution.

3.2.4 Simultaneous Multiple Paths

Frequently, it is useful to work on several ideas or possible solutions at the same time. This gives a more thorough understanding of the problem, and targeting different aspects of it in different solutions. This approach is most promising at the beginning of solving a problem, and for ones that do not necessarily have one single correct answer, in other words ill-structured problems.

Simon (1973) argues that ill-structuredness is a residual concept: an ill-structured problem is one that is not well-structured. A well-structured problem is defined as one that can be solved by appropriate moves that take us further towards the goal-state, one step at a time. Some of the properties of well-structured problems are (Ormerod, 2005; Simon, 1973):

1. There is a definite criterion for testing proposed solutions.
2. There is at least one problem space in which can be defined the initial state, the goal state, and any intermediate states in the attempt to solve the problem.
3. All state changes, and transitions from one state to another and the constraints upon operator selection are known in advance and can be represented.
4. There is at least one problem space in which can be represented any knowledge that the problem solver acquires in the process of solving the problem.
5. If the actual problem involves acting upon the external world, there is at least one problem space in which all transitions and effects of applying any operator reflect the laws of the external world (laws of nature).

6. All of these conditions require that the basic process involves practicable amount of computation.

Any problem that fails to satisfy one or more of these properties is decided to be ill-structured. The solutions to these kinds of problems are approximate, each has its own advantages and disadvantages, such that they can still be improved and it is up to the problem solver to decide when it is enough. Examples of ill-structured problems vary from design ones, such as a building, software system or consumer product design to a simple writing task (Terry et al., 2004).

As mentioned above, ill-structured problems do not have a single correct solution, therefore it is wise to investigate multiple, alternative approaches. Most of existing statistical packages allow for a linear problem solving approach. If users want to do otherwise, they need to keep record of their thoughts and decisions themselves. However, some projects have looked into supporting multiple solutions in other areas. One example is Parallel Pies, a user interface mechanism for image manipulation task developed by Terry, Mynatt, Nakakoji and Yamamoto (2004).

They mention that the lack of one correct solution in an ill-structured problem leads to the development of multiple potential solutions, however not much attention has been
paid to supporting the exploration of alternatives when actively operating on data. Currently users must put the solution at hand on hold, duplicate their data, and then re-establish the original context for each variation they wish to pursue. They address this problem in their work, resulting in Parallel Paths, a scheme undertaken by Parallel Pies, which allows users to easily create solution alternatives in the same workspace, manipulate the alternatives independently or simultaneously, and perform side-by-side comparisons of each.

Another work is a software system discussed by Hepting (2007). Cogito is designed from the perspective that it is possible, with appropriate support, to meaningfully explore a very large set of alternatives without becoming overwhelmed. Cogito’s architecture is such that it can act as an interface to a variety of applications, as long as they provide some form of batch interaction, making it possible to depict many different visualization patterns at the same time in the main window, giving users the chance to view and compare them simultaneously.

So far, a summary of research in the field of supporting creativity in computer-based environments was given, along with the characteristics of such systems. One might ask what role, if any, creativity has in statistical analysis.

Statistical analysis is not just about applying some pre-practiced tests on a dataset, and proving or rejecting some hypotheses. As Tukey (1968) suggests, data analysis needs to be both exploratory and confirmatory: “To concentrate on confirmation - to the exclusion
or submergence of exploration – is an obvious mistake” (p. 2). He believes that instead of being bound by preconceived notions and analyses, we should flexibly look around (Tukey, 1968).

He also quotes one of his friends in the same profession that following a rulebook for research seems to stimulate the study on trivial problems. The great challenge is to teach investigators not to be too tightly bound by a preconceived model of research design. He continues to suggest that intelligent problem formulation should be motivated, without depending of a single model of statistics, or some standard textbook solutions.

Looking at data analysis from these statisticians’ points of view, demands creativity as part of a successful analysis process. Tukey believes that one body of data should usually be analyzed in more than one way. He also recommends to look around, be flexible in handling data, and to remember that we learn by taking chances, by trial, with some errors. Noticing these, one possible suggestion would be to support multiple paths and also reflection in data analysis. This way the analyst can simultaneously follow various solutions, and by reflecting on them regularly decide which ones are more appropriate or more likely to lead to the most significant result.

In this chapter, various definitions of creativity and creative problem solving, and possible means to nurture them were presented. Then I introduced the concept of supporting creativity by computer applications, and based on the existing literature in that area summarized a list of proposed characteristics for such a system. There has never
been any project that applies the suggested properties of supporting creativity to the realm of data analysis. Treesta addresses this issue; it provides statisticians with facilities that would help them analyze their experiments' datasets in a more informed and more creative way, as will be presented in the next two chapters.
CHAPTER 4:

Treesta’s Design and Rationale

Treesta supports creativity in the field of data analysis, with the focus on analysis of variance. Existing statistical packages do not support organizing the work in a data analysis job, and that is what we address in this project. By categorizing the analyst’s activities into a tree consisting of separate inter-connected workspaces, Treesta allows for a more structured approach to a statistical analysis job. It provides a detailed auto logging of entered commands in the context of each workspace; the commands are passed through filters that check them for syntactical correctness and also parse them for special purposes. In addition, Treesta delivers other services such as replaying some part of history, as well as a set of commands to manage the whole environment.

Currently statisticians have their datasets on their computers (e.g., in an Excel or a simple text file). They start to analyze their data, using a mathematical package, to answer some questions, or prove or nullify certain hypotheses. They might also be willing to find unpredicted relations among independent and dependent variables (i.e., post-hoc analysis). Existing systems like Matlab and SPSS provide a powerful mathematical engine, and can carry on complex calculations; however other than keeping a linear list of commands, along with date and times, they are hardly any help in the organizing aspect.
Statisticians have to develop their own techniques, instead of computer tools, to keep a record of their work; they can save outputs, such as tables and figures with meaningful names. In addition, they might have a digital document or some sticky notes that contain information such as the phases in the project, what they have done, results, what should be done, the reasons behind a decision, and which calculations and results are the most significant or the most recent. If they want to try a different path (transformation and function) on a dataset, they have to duplicate it and work from there. Moreover, if they need to apply similar functions on a different set they have to do so all over again.

It is difficult for statisticians to manage and keep track of their work with this procedure; therefore, I have developed a system that addresses these difficulties. Treesta is a text-based interface that stands between the user and Matlab, and serves as an organizational, as well as representational, tool for statistical analysis.

As mentioned above, Treesta adds an organizational facility to the mathematical power of Matlab. It categorizes the user's work into workspaces, building a tree, which makes it easier to access different portions of the entire work. Each workspace has its own parent, children, variables, figures, and history. It can be assigned a meaningful name and some notes the analyst wants to say about that workspace (e.g., the reason for a certain action).

At any time, users can jump to any workspace they want and create a new child, which inherits all the variables (including the dataset). Then they are able to follow a completely different solution using these same variables, and compare their results at the end,
without the need to duplicate the problem space and keep track of multiple solutions manually. Moreover, if they want to try the same functions with a different dataset, they can use Treesta’s replay command which replays some chosen part of history. Finally the user can save the project and continue the work later.

What comes next is a summary of the main characteristics and services of Treesta:

4.1 Main Services of Treesta

4.1.1 Capturing History

Treesta is a history-capturing and representation tool. As a history-capturing tool, Treesta records users’ activities in order to give a more thorough view of what has occurred, which in turn results in a clearer path for future steps. We noticed that the existing statistical packages like Matlab and SPSS record history in a linear format which gives the information about what commands have been entered by the user and when. This method of logging is hardly successful in helping users have a clear picture of the whole analysis job, where they are at the moment, and what are the possible next steps.

Unlike other packages, Treesta stands between the user and Matlab, hears the whole communication and records syntactically correct commands, which are saved with a time stamp in association with the active workspace. An interesting command in Treesta uses the history of a workspace to discover the dependency relationships among the variables of that workspace. Moreover, the history can later be viewed or replayed. Since Treesta is intended to record and represent all this interaction, it does not allow for any undo, and delete action. It gives the possibility of branching to different paths, rerunning commands
with different parameters, but it does not let any state or command be removed from the history.

Although version control systems also save history, their history-recording is different from Treesta’s. A version control system is content-neutral and records the state of the system discretely. Treesta records the user’s activity continuously and is also content-aware. It parses the commands in order to extract some information about the ongoing analysis.

4.1.2 External Representation

The advantages of external representation were discussed in the previous chapter. This is a major property of computer systems intending to support creativity, and as one objective of Treesta is to do the same in the field of statistical analysis, it provides different representations of the ongoing project.

When users work with Treesta to analyze their datasets, they can choose to view various pieces of information about their job; such as a tree structure of the environment, which highlights the relationship between workspaces, the summary of each workspace (i.e., name, figures, variables and with what workspaces they are shared, etc.), and the dependency relationships between variables in a workspace. By viewing these pieces of information, analysts can see the big picture, the work that has been done, where they stand at the moment, and what should be done next. This way we take some cognitive load off users’ minds, and also assist them in reflecting on actions.
4.1.3 Supporting Reflection in Analysis

As mentioned in the previous chapter, reflective behavior is a significant ingredient in practices in which users are designing something, a system, a building, or an experiment. There is a similar case in data analysis where the user has to consider the available information and devise a plan to arrive at answers for the existing questions, therefore being able to sit back at any time, evaluate the situation and decide what to do next, what transformations or operations to apply.

The main answer to this requirement is externalization. By providing external representation, Treesta also supports reflection in statistical analysis. While using Treesta, users are able to step back any time during their work, and view the whole tree, different states, and the summary of each of them, including the workspace’s name, parent, children, variables, figures, and the extract of ANOVA function. Based on whether users view this information during or after a certain job, the system can provide reflection in or reflection on action. Statisticians run a command, the work talks back to them through the external representations, and they can decide on the next possible move. This way they can make more informed decisions about where they want to go from there; what functions and operations are more likely to fit as the next step.

4.1.4 Simultaneous Multiple Paths

As discussed in the previous chapter, ill-structured problems do not have a single correct answer; hence, it is beneficial to support the development of multiple solutions at the same time. In statistical analysis, the analyst would like to discover some properties and generalize them to the whole population based on the findings in the sample collection.
For instance in analysis of variance, the property is a significant difference in group means, indicating that the focal factor has an effect on the dependent variable.

When the analyst begins analyzing a dataset, she is faced with several paths to follow. She might have various hypotheses to test, for which she has to check relevant assumptions and apply functions, she might be interested in extracting some properties, summarizing or transforming data sets, or she might do all of these after cleaning the data or selecting a part of it. Lots of relevant concepts like degree of confidence, normality, homogeneity of variance, presence of outliers are not absolute, and require judgment. There are many different methods to correct any inherent problems that the data has. Using one approach, the analyst might decide that there is not a significant result, while using another one she may decide to reject the null hypothesis.

To better understand the structure of statistical analysis let us have a comparison between the ultimate example of ill-structuredness, design, and our problem. The problem space for design can not be defined in a meaningful way, since the definition should contain all kinds of structures that the designer may consider at some point in the process, all considerable material, and all organizations of design processes. For instance, consider designing a building where the architect is faced with the choice of the structures, such as a truss roof, arches, materials, like wood, metal, or the organizations, like starting with floor or a list of functional needs.
Therefore, it is almost impossible to define a problem space whose boundaries would not be breached by the intrusion of new alternatives, and as a result, the second, third, and fourth properties of well-structured problems that were mentioned in chapter 3 failed too. Compared to design, analyzing data observes some boundaries, since the available operations and transformations are finite and predefined. Hence, data analysis is in the middle of the continuum of ill-structured problems.

Treesta supports simultaneous development of multiple solutions by letting the user branch to a new child at any time. This way the user can put away the analysis so far, duplicate the data and try a different function or test on it. Our system reduces commitment to a single approach by allowing the analysts to conduct different approaches simultaneously. The analysts do not have to be concerned about saving these parallel paths themselves.

4.1.5 Replaying History

This is another important service that Treesta provides, which is especially useful when the user wants to apply similar functions to a different piece of data. Often in the middle of work, the analyst changes the dataset; he might add a focal factor, clean the data and remove outliers, or select some part of the whole data, and then wants to apply the same functions on this new dataset. Using the replay command, he does not have to re-enter all the commands, but can select a portion of history and simply replay it with the updated input.
4.1.6 Dependency among Variables

While working with Treesta, one can track dependencies between variables in one workspace. While working on the main analysis task, sometimes it happens that the analyst follows some side calculations that might be related to the job at hand, but not leading to the final result. It is useful to exclude these irrelevant parts and extract the calculations that play some actual role in the final result.

Treesta starts from the last assignment, and based on the variables to the left of the assignment operator (=) (e.g., \([p \ table \ stats] = \text{anoval}(\text{data})\)) builds a dependency structure. By benefiting from this facility, the analyst is able to see which variables have affected the ones in the last statement, thereby ignoring the unimportant ones (e.g., those involved in a calculation not leading to the main ANOVA).

4.1.7 Sharing a Variable

Another major service that Treesta offers is centred on sharing a variable, which is possible between a workspace and its child. And since in turn the child can share this variable with its child, the variable will be shared among more than two workspaces. When the value of a shared variable is altered in one workspace, the change will spread to other workspaces sharing this variable. There are two options to share a variable; the first one is at the time of creating a child, and the other one is by using the share command.

In this chapter the general idea of Treesta and how it contributes to performance support in statistical analysis were reported. Implementing a subset of characteristics required for
supporting creativity, such as external representation, supporting reflection in analysis and multiple paths, this system results in a more efficient and more informed process of data analysis. The details of design and services of Treesta are discussed in the next chapter.
CHAPTER 5:

Detailed Description of Treesta’s Design

Treesta is a text-based interface that organizes the analysis that the user is working on in Matlab. It stands between the user and Matlab, and augments Matlab’s functionality through various services and commands. The top-level unit in Treesta is a project, organized as a tree whose nodes are workspaces. Each node can have multiple children, and has exactly one parent, except for the root workspace.

Having launched Treesta, the user can either open an existing project, or create a new one. Creating a new project results in creating a new folder on the path, where every file related to this project is kept. In either case, the active workspace is always the root workspace.

Each project is associated with a folder with its name, where all the relevant files to that project are kept. All the important information about the ongoing analysis is gradually stored in these files as the analyst is working, and is used to re-open the project and access the work in the future. The first file in such a folder is called tree_structure, and keeps the tree structure of the nodes. This information includes the name of the workspace, its parent, its children, variables it shares with its parent, and a summary about ANOVA function in that workspace.
In addition to `tree_structure` file, there is another file for each workspace. While working in a workspace, the user can create variables, figures and apply Matlab commands, and all these are saved in that file, along with the date and time of the session. When the user starts to work in a workspace, its file will be opened, and the date and time will be recorded, followed by every Matlab command that is syntactically correct. When the user decides to exit that workspace, either by opening another workspace or by exiting Treesta altogether, all figures, variables, notes, and annotations that the user has entered are saved and the file is closed.

Currently figures are automatically saved when the user exits a workspace. Earlier in developing this project, Treesta would ask if the user wanted to save the figure or not, but later I decided to change this policy. First, most of the time the analysts would like to save the figures, especially the ones containing the results of an ANOVA. In addition, it is very distracting to ask this question every time users exit a workspace, because they might constantly switch back and forth between workspaces. Now when the user leaves a workspace, the figures are saved and closed, and re-opened every time the workspace is re-entered.

The level after internal representation of projects that was described is how Treesta handles commands. The current version of Treesta is focused on analysis of variance, and only one ANOVA command is allowed per workspace. After an ANOVA function has been applied in a workspace, no more commands are recorded in the file. The user can still enter Treesta’s commands and Matlab commands that display variables, but nothing
like an assignment, containing the assignment operator (i.e., '='), which has an actual
effect is permitted. If a Matlab assignment is entered, an error message is displayed and
the command is not passed to Matlab. This requirement forces some kind of order; if,
while studying the results of an ANOVA, the analyst decides to change a dataset and run
the ANOVA test on the new set, he has to do so in a new workspace. This makes it easier
to review and compare decisions and test results.

Treesta provides a set of commands that helps users maintain and navigate through their
project. These commands all start with a slash character (i.e., '/'). The purpose of the '/
character at the beginning is to distinguish Treesta's commands from Matlab's. Every
entered command not starting with '/' passed to Matlab.

Two of these commands are novel and of special importance. Replay makes it easier to
repeat a sequence of functions, and dependency provides a more complete
understanding of the variables in the workspace, leaving out the irrelevant information.

5.1 Two Novel Functionalities of Treesta

5.1.1 Replay History

/replay workspace1 [, workspace2 ...] [; variable]: this command replays a
specified portion of history, a sequence of already applied commands. Since users are
likely to consider various solutions, models and tests in their dataset, they might need to
repeat a sequence of commands, perhaps on data that is the result of transforming the
original dataset.
The arguments are the name of the desired workspace(s), optionally followed by an arbitrary variable. The commands in workspace1 workspace, excluding the irrelevant ones, are repeated in the current workspace, and in case more workspaces have been listed as arguments, a new workspace is created for each one, in which the commands from the respective workspace are run. If no variable is given as an argument, the whole history is repeated from the first command to the last one, otherwise the beginning point is after the last assignment to that variable.

The variable given as an argument indicates the starting point of the replay procedure. While analyzing data, it often happens that the user changes a dataset, for example adds or removes factors or observations, and wants to apply the same sequence of transformations on the new dataset. In these situations, he can enter the name of the dataset as the reference variable, and since replay starts immediately after the last assignment to the chosen variable, no command will alter the value of this variable, and the analysis is applied to this value.

/replay example:

To clarify replay, it is useful to give an example of how it works: An analyst has a dataset from an experiment with one focal factor with 4 levels and a single dependent variable with 12 observations per level of the focal factor (see Table 1). She would like to run an ANOVA to find out if there is any significant difference among the levels of the focal factor.
Table 1.
A dataset with 48 observations. The single focal factor has 4 levels with 12 observations at each level.

<table>
<thead>
<tr>
<th>Observations</th>
<th>Level1</th>
<th>Level2</th>
<th>Level3</th>
<th>Level4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>10</td>
<td>9</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>8</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>7</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>15</td>
<td>11</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>12</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>-2</td>
<td>10</td>
<td>8</td>
<td>22</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>37</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>9</td>
<td>-2</td>
<td>11</td>
<td>7</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>-2</td>
<td>10</td>
<td>6</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>8</td>
<td>27</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>-1</td>
<td>8</td>
<td>6</td>
<td>23</td>
</tr>
</tbody>
</table>

Here is the result of the ANOVA function on the log transformation of this dataset:

```r
> dataset = log(dataset)
> p = anova1(dataset)
```

```r
p = 2.4740e-008
```

Then she wants to compare only the first two levels of the focal factor. In order to do that she creates a new child and names it `first2levels`, and runs the ANOVA:

```r
/new first2levels
> p = anova1(dataset (: , 1:2))
```

```r
p = 8.0299e-004
```

She then notices that there are some outliers present in the original data, creates a new child, calls it `no_outliers`, loads the dataset again and removes the outliers (19, 37,
She now wants to repeat the whole work in this new dataset:

```matlab
>x = dataset(:,1);
>x = x(abs(x-mean(x)) <= 3*std(x));
>dataset2(:,1) = x;
>x = dataset(:,2);
>x = x(abs(x-mean(x)) <= 3*std(x));
>dataset2(:,2) = x;
>x = dataset(:,3);
>x = x(abs(x-mean(x)) <= 3*std(x));
>dataset2(:,3) = x;
>x = dataset(:,4);
>x = x(abs(x-mean(x)) <= 3*std(x));
>dataset2(:,4) = x;
>dataset = dataset2;
>replay space_0, first2levels; dataset
```

This command results in the repetition of the history of workspaces `space_0` and `first2levels` on the new dataset. First ANOVA is applied on the new dataset in the current workspace, `no_outliers`, \( p = 0 \), and then a new child, `space_3` is created and another ANOVA is applied on the first two columns \( p = 1.2008e-008 \)

Finally, she views the structure of the workspaces:

```bash
>structure
```
For very small $p$ values, Treesta simply shows that they are less than 0.01, which is easier to read. In the future, this significance level should be specifiable by the user.

Replay does not replay irrelevant commands, those commands that do not have any effect on the final variables in the last command in the history list. While analyzing data, it may happen that the analyst views the amount of the variables or some histograms just to have a visual check or temporarily pursues a side calculation. There is no need to repeat these commands, and in order to achieve this, replay uses a dependency algorithm to remove these irrelevant commands from the repeat list. This algorithm traces the dependency links among variables in a workspace. Next, we explain how dependency works.

### 5.1.2 Dependency among Variables

/dependency [workspace]: This command extracts the dependency relationships between the variables in a workspace; that is for each variable which other variables determined its current value. If the user has entered the name of workspace as an argument, the dependency algorithm is applied in that workspace, otherwise the
current workspace is considered as the target. The dependency algorithm is summarized in Figure 2.

Dependency (history of commands)

1. Initialize important_vars list //the list of important variables
2. Initialize dependencies list //the list of dependencies among variables
3. Reverse the order of commands in the history
4. Add them to a vector called Commands_Vector
5. Read the last command in vector Commands_Vector (usually the ANOVA command)
6. Parse it to find the variables
7. Mark the command as important
8. Add the variables on the left and right of '=' to important_vars
9. Add the variables on the right of '=' to dependencies of variables on the left of '='
10. For each remaining command in vector Commands_Vector
11. Parse it to find the variables
12. If the variables on the left of '=' are important
   i. Mark the command as important
   ii. Add the variables on the right of '=' to the important_vars and also to the dependencies list of the variables on the left of '='
13. Merge the dependencies list for this command with the ones so far
14. End

Figure 2. The dependency algorithm
Treesta opens the file for the specified workspace, starts from the last command and goes backward to the first one, keeping a list of important variables during this algorithm. The result of dependency is organized such that each variable is on a separate line, and each variable is followed by a list of variables affecting its value. In the following example, the user enters a sequence of commands, and then runs dependency:

(1)    >a = 12;
(2)    >b = [2 3 5; 8 7 9];
(3)    >c = a+10;
(4)    >d = b*10;
(5)    >p = anova1 (d)
(6)    p = 0.5637
(7)    >/dependency
(8)    Variable dependencies are as follows:
(9)    p d
(10)   d b
(11)   b

The result is based on the order of variables' appearance in the reversed history. Here it shows that p depends on d, which depends on b. Finally, b is the source and is not related to any other variables. It should be noted that the variables a and c do not have any effect on p, and therefore are removed from this list. And if the user chooses to repeat the history of this workspace, lines 1 and 3 will not be replayed.
The dependency algorithm uses a parse which recognizes the assignment operator (\texttt{=}) and identifiers, a collection of letters, digits and \_\_\_s starting with a letter. Variables, which are important for us, are a subset of identifiers; therefore, Treesta checks to see whether the found identifier is actually a variable. First Treesta makes sure the identifier is not a Matlab keyword by using the Matlab function \texttt{iskeyword}. If the answer was false (shown as 0 in Matlab), the identifier is examined to be a Matlab or a user-defined function or a variable, using Matlab function \texttt{exist}. 0 indicates an undefined variable, 1 an existing variable, and 2 specifies a Matlab or user-defined function.

The first variables to enter the list of important variables are the ones on the left side of the \texttt{=} in the last assignment (\texttt{p} in this example). It should be noted that in Matlab it is possible to have multiple variables on the left side of \texttt{=} . These variables are followed by the ones on which they depend, which are the variables on the right side of the assignment operator (\texttt{d} in the above example). For each important variable there is a list of influencing variables, which is also updated as the routine moves on through the commands (in this example \texttt{d} is a variable which affects \texttt{p}). At the end of the operation, we have the information about which variables influenced which ones, containing only those that determined the values of the variables in the last assignment.

There is a downside to the way Treesta only records the commands that are entered via the command prompt, therefore ignoring the content of any M-files, saved pieces of programs that can be called later. This becomes troublesome while extracting the dependency links in a file where a function with side effect exists. Side effects alter a
global variable, without this being specified in the function header, as shown in the following example:

Function r = test (a)

global GVar;

r = a *100;

GVar = a+1;

GVar is defined to be global in the main workspace. If we call the function:

var1 = test (10)

besides changing the value of var1, the value of GVar would change too.

If a function with side effects exists in the file being scanned for dependency relations, Treesta is unable to detect the side effects on the global variable, and whether it is affected by any other variables. Since user-defined functions are typically saved in M-files, in order to avoid typing them each time they are needed, this weakness should be addressed in future improvements of Treesta, by opening and parsing the content of the M-file, when faced with a function call.

Global variables are generally considered bad practice by programmers, precisely because of their non-locality. They can be accessed from anywhere in the code, making it difficult to isolate units of code for the purposes of understanding and testing a program. They can lead to problems of naming, for example, a local variable with the same name as a global variable shields the global one from being accessed. Global variables lead to side effects which are hard to understand and predict. Another problem associated with
using global variables is a race condition in multi-threaded programs. A race condition occurs when various pieces of code need to update a global variable and it is difficult to make sure they do that in turn and leave the variable in a valid state.

The complexity estimation of the dependency algorithm is $O(Kn)$, where $K = 3 + v$; in which $n$ is the number of commands, or file lines, and $v$ is the number of important variables (i.e., those influencing the variables assigned to by the last command). The $3n$ part is related to reading the file and recording the commands in a vector for further analysis, and the $nv$ part shows that in a loop, each command is parsed and at the end of each iteration, the resulting dependency list is merged with the result so far, and the order of this last activity is proportional to the number of variables. In a typical analysis, especially since the size of a workspace does not grow too much because of the constraint to a single ANOVA, the CPU time of the algorithm on a typical contemporary machine is in the order of milliseconds.

Having discussed replay and dependency, the remaining commands are presented along with a short description of each. The commands are grouped according to their functions:

5.2 Commands Related to Viewing or Modifying a Single Workspace:

- `/space`: Prints the name of the active workspace.
- `/rename newname`: Changes the name of the active workspace to newname.
• */%note */%/: Adds a note. The user enters "*/%" and starts writing the notes for that workspace. The note can extend for several lines, and is ended by "*/%".

• */%: Views the notes the user has entered in that workspace. This command is distinguished from opening a new note; because it is not followed by words, but by a carriage return.

• */figureList*: Prints a list of the saved figures of the current workspace. Figures are not saved until the user exits that workspace or runs a save command (see below), therefore newly created figures are not included in this list.

• */summary*: Summarizes the active workspace, including its name, parent, children, variables and saved figures. For each variable it is mentioned whether it is shared and with which workspaces (data (shared with: workspace 1, workspace 2)). In addition, a summary of the ANOVA function is printed if one has been run in this workspace. This summary includes the number of focal factors, and how many levels each has. It also includes the number of observations under each cell, and finally $F$ and $p$ values for each focal factor. Finally all the notes for the workspace are printed.

• */history [workspace]*: Views the history of commands in the specified workspace, or the current workspace if no argument has been entered. This command helps the user to review what he has done before. For example, prior to replaying a previous workspace, the analyst can view the history of that workspace and make sure if the list of commands are in fact useful at the moment, applied on this variable. This command views the complete list of commands, including the commands that will be stripped out by dependency.
5.3 Commands Related to Viewing or Modifying the Whole Environment:

- **/structure**: Depicts the structure of the tree of workspaces, representing the parent-child and sibling relationships, by indentation. Each workspace is displayed on a separate line, followed by the list of variables it shares with its parent, and the list of \( p \) values from ANOVA functions in that workspace. This information provides a quick summary of the whole structure. By running this command, the user can tell which workspaces are sharing which variables with their parents. Also it becomes obvious whether a workspace already has an ANOVA function, and if so what are the resulting \( p \) values. The final piece of information is an asterisk (\('*\)'), denoting the active workspace.

- **/globally Matlab-assignment-command**: The Matlab command should be an assignment, indicated by containing an assignment (\( '=' \)) sign; otherwise an error message regarding a false assignment is generated. First the command is run in the current workspace as a regular command, after that Treesta runs the assignment in all workspaces that it is effective. This means in all the workspaces that at least one of the variables on the left side of the assignment operator (\( '=' \)) exists, therefore changing the value of these variables in the whole environment, similarly and simultaneously. As it should be clear, no new variable is created in the workspaces other than the current one. The final note on this command is that it will not be run in any workspace that has an ANOVA function.

- **/save**: Saves the project.

- **/exit**: Saves the project, and shuts down Treesta.
5.4 Commands for Managing Relations between Two Workspaces:

- `/new [child] [, variable1, ...]`: Creates a new child. If a name has not been entered, a default name will be assigned to the new workspace. Since workspace names are unique, there should not be a workspace with the entered name. The user can also specify a list of variables to be shared with the child workspace.

- `/share variable1 [, ...]`: Shares the entered variables with the parent workspace. Together with the sharing option at the time of creating a new child, Treesta delivers two ways to access this facility. This command is useful whenever the user forgets to share a variable while creating a child, or he decided to do so later. If the variable does not exist in the current workspace, Treesta gives an error message.

- `/unshare variable1 [, ...]`: Unshares this list of variables with the parent workspace. This command is useful in situations that the user does not want to affect this certain variable in other workspaces, therefore choosing to break the sharing pact. If the user tries to unshare a variable which does not exist or was not shared in the first place, the system will generate an error message.

5.5 Commands for Moving between Two Workspaces:

- `/goto workspace`: Jumps to the mentioned workspace.

- `/up`: Moves to the parent workspace
• `/down`: Moves to the child workspace, which is possible only when the active workspace has one child. In other cases it results in an error message without changing the active workspace.

• `/root`: Jumps to the root workspace.

Now that Treesta's design and command list have been explained in more detail, it is beneficial to include an example of a real data analysis project using Treesta.

**The analysis of a dataset from an experiment on rats**

This experiment was conducted on a sample of rats, in order to find out the relationship between a certain kind of protein in the rats' brains and their sexual response (from an assignment in the Design of Experiments course (PSYC 910), by Dr. Michael Maraun, SFU, Fall 2007). The Fos protein is a DNA binding protein and it is believed that the generation of Fos is associated with sexual response. To find out about that a researcher randomly assigned 21 rats to three conditions with the proviso that 7 rats appear in each condition. The three conditions were:

1. The rats were kept in a glass case and allowed only visual contact with the larger community of rats (in particular no odor clues).

2. The rats were allowed social contact (including odor clues), but were not allowed to copulate.

3. The rats were allowed to socialize (including copulate) in the larger community of rats.
Following a period of three hours, each of the 21 rats was killed and the number of Fos nuclei present in its inferior reticular nuclei was counted (see Table 2). The experimenter would like to check if there is a significant difference among these three groups ($\alpha = 0.05$):

<table>
<thead>
<tr>
<th>Table 2.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of the Fos nuclei for rats in the three conditions</strong></td>
</tr>
<tr>
<td>Rats</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

The user starts Treesta, and names the project `FosProject`. She enters the dataset, and calls it `FosData`. Then she runs an ANOVA on this dataset:

```r
> p = anova1(FosData)
p = 0.0183
```

Since the $p$ value is 0.018 which is less than $\alpha$, one could conclude that the number of Fos nuclei is in fact related to the sexual response; however the researcher wants to be more certain, thus she decides to increase the number of observations. She repeats the experiment using another 21 rats, which results in the set shown in Table 3:
Table 3.

Number of the Fos nuclei for rats in the three conditions from the second experiment

<table>
<thead>
<tr>
<th>Rats</th>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>11.2</td>
<td>28.0</td>
</tr>
<tr>
<td>2</td>
<td>12.5</td>
<td>34.6</td>
<td>33.6</td>
</tr>
<tr>
<td>3</td>
<td>16.8</td>
<td>17.3</td>
<td>45.8</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>55.8</td>
<td>34.0</td>
</tr>
<tr>
<td>5</td>
<td>13.8</td>
<td>29.4</td>
<td>25.0</td>
</tr>
<tr>
<td>6</td>
<td>22.4</td>
<td>15.0</td>
<td>90.1</td>
</tr>
<tr>
<td>7</td>
<td>18.1</td>
<td>14.3</td>
<td>27.8</td>
</tr>
</tbody>
</table>

She renames the workspace, and records some notes, so she can easily understand what has been done:

`>rename 7observations`

Name changed to 7observations

`>/% The p value (0.018) suggests that there is a difference among these three groups, but I would like to be more certain, so I run the experiment again with another 21 rats %/`
Then she creates a new workspace, calls it `14observations`, and adds the new observations to `FosData`. She then remembers the `/replay` command, but she wants to view the command history of the previous workspace, just to make sure none of the commands causes any problems here:

```
> /goto 7observations

Active space changed to 7observations

> /history

FosData (:, 1) = [14.7 5.3 0 3 27.52 12.47 24.4];
FosData(:,2) = [29 10.9 14.46 16.94 14.1 23.7 69.7];
FosData(:,3) = [49.2 27.94 26.38 25.5 63.5 31 82.65];
FosData
p = anova1 (FosData)
/rename 7observations
/%
/new 14obesrvations
/history

She decides to replay this history, only with the new `FosData`:

```
> /replay 7observations; FosData

p = 2.0046e-004
```
She then adds some notes:

```matlab
/* I have added 7 more observations to each condition, and the result is more significant (p = 0.0002) */
```

Finally, she views the structure of the environment:

```
> /structure

7 observations (p = 0.0183)
* 14 observations (p = <0.01)
```

This example illustrated a simple analysis done using Treesta, featuring various Treesta’s commands. The commands of Treesta, extensions to Matlab’s functions, help maintain the project, through presenting the structure of the tree and also the summary of each workspace. Moreover, Treesta provides some extra commands such as replay and dependency, which were described in detail. I believe this system matches the workflow of analysts more closely and in order to verify that I have conducted a user study which will be summarized in the next chapter.

5.6 Treesta’s Implementation

Treesta is implemented using the Java programming language, using the JMatLink package to connect to Matlab. The main class, directMat opens the engine to Matlab and maintains the workspaces. Workspace is the class containing all the attributes and functions for the workspace object. Parser is the class that parses commands in the history, extracts the dependency relationships among the variables and replays the
selected commands. AnovaSearch is responsible for detecting and summarizing ANOVA commands. Two other classes help maintaining the files that store the information about the project. The total code consists of 3,700 lines.
CHAPTER 6: 

The Evaluation of Treesta

To initially evaluate Treesta I have conducted a user study with a single participant. This test was mainly intended to test the idea behind this system, allowing the users to follow multiple solutions and providing means of reflection on their work, and also as a means for assessing Treesta’s usability. Along with taking notes by the experimenter, the session was audio recorded. Before the actual study, I asked a graduate student, moderately familiar with statistical analysis and Matlab, to work with Treesta for an hour in a pilot study, in order to examine the approach and tools, and to make sure they work in the real session.

First, the actual study is reported, followed by a summary of the pilot test.

6.1 Participant

The participant was a computer science graduate student, experienced in statistical analysis and Matlab.

6.2 Task

The participant used his own data, which we believe was advantageous, because he knew the experiment, and was actually interested in analyzing the data and finding out the results. The purpose of his experiment was to compare two input devices; therefore, it
had one focal factor with two levels. There were four dependent variables: total time, acquisition time, idle time and manipulation time. The participant was interested to find out if there was a significant difference between the two input devices based on the recorded data for dependent variables.

6.3 Protocol

The actual study was divided into two sessions: the first one would give the participant a chance to work freely with Treesta, get to know the commands and their functions, and the second one would be the actual audio-taped study session.

In the familiarization session, where no data was gathered, I answered the user’s questions and gave instructions when needed (e.g., when he was not sure about the functionality of a certain command). At the beginning of the second session, the participant was informed about audio taping and signed a consent form. Then he was given the task description orally and was asked to think aloud.

At the end of the session, I had an interview with the participant, during which I asked him questions about the description of his actions and the reasons behind them, how Treesta changed his work process and how he evaluated different features of the system. Finally, he was encouraged to mention any usability problems and suggestions he could think of.
6.4 Results

Our findings are from the participant’s answers to the interview questions as well as the observations during the evaluation session.

He found the tree structure and the ability to follow various paths interesting and useful. He commented that it imposes a structure on the workflow and makes it easy to find and refer to each individual workspace, especially if they have meaningful names. Moreover, he said that since you have to create a workspace before you go and do the analysis, it makes you think about your next step in a more structured way; you set some goal about what you want to achieve in this workspace, and you work around until the goal is achieved. Besides the opportunity to try multiple solutions, the user found the notion of making notes pretty useful too, because it makes it possible to write down what you think at the moment, including the reason behind a certain behavior of the data, or some reminder for a further analysis. He also liked the /structure command, since it provided an overview of the work that had been done, as well as a summary of ANOVA in each workspace.

Besides these positive remarks, the participant found a single ANOVA per workspace quite a limiting property. For instance, when he wanted to make a minor change in the dataset and run ANOVA again, he preferred to do so in the same space. Finally, having had worked with Treesta for a while, he followed Treesta’s model; whenever he wanted to do more analysis after an ANOVA, he would create a new child, which would inherit
the most updated version of the dataset, in this was he could simply continue his analysis on the data in the new workspace.

In addition, he mentioned that creating a new child is a bit slow. This delay is due to the fact that Treesta opens a new engine to work with Matlab for each workspace. I found this choice better than working with the same engine the whole time, since each workspace has its own variables and figures, probably with similar names, and preventing Matlab from overriding them would be troublesome in a single engine. Moreover, the routines to solve this problem might have been even slower.

6.5 Discussion

One of the positive feedbacks received from the participant was related to the design of the study and the familiarization session. The first session helped him utilize the system better, since he knew how Treesta was designed to work. The participant imported the whole dataset in the root workspace, but he did not start analyzing there. He mentioned that having had the experience of the first session he now knows that after running an ANOVA test, no more calculation is allowed in a workspace, therefore he prefers to leave the root workspace as a reference point, so he can go back to it and apply commands there. For analyzing each dependent variable, he created multiple children, which inherited the dataset from their parent and assigned meaningful names to them, so each branch was corresponding to a single dependent variable. He later mentioned that creating various workspaces helped organize his work. It was also useful since Treesta provides a summary of the ANOVA function in each workspace.
During his work, the participant checked the structure of his environment regularly, to keep an overview to see what he has done so far. This behavior is an example of reflection in action. When he had questions or doubts, he would look at the data, or view Matlab’s plots, like histogram plot, to find out what was going on. He also made use of Treesta’s comment command, making notes to explain the reason behind some outcome or as a reminder to continue his analysis on a variable, if it was behaving unexpectedly or strangely. For example while analyzing acquisition time, despite the fact that the means of two groups were really close, the $p$ value was pretty small. This result intrigued a question in the user’s mind, and he decided to write that down as a note in Treesta, move on to see other results and come back to this variable later.

The actual study showed that the ability to follow various solutions is helpful. The participant mentioned that organizing the work into various workspaces, with meaningful names helps referring to an exact point in the analysis, as he tended to view the structure of the environment and jump to a certain workspace, where he analyzed the data more carefully. He also made use of taking notes, whenever he had some questions, or wanted to explain something. He did not use the /replay command, because he had transformed the data beforehand, therefore there was no need to change the dataset and reapply the previous Matlab commands again, but otherwise he would have used this command.
6.6 Notes from the Pilot Study

Prior to the actual study, I ran a pilot study, to examine the testing approach. The participant, a computer science graduate student, moderately familiar with statistical analysis and Matlab, was provided with a dataset consisting of a focal factor with 4 levels and 12 observations under each level. He then was asked to analyze this dataset and find out whether the focal factor has any effect on the dependent variable. He then was instructed to check for the existence of outliers. As the last step of the study, he was asked to add a new dataset including 8 more observations to the main dataset and analyze it again.

Besides testing the experiment approach, the pilot session also gave us some useful feedback about the system. First, the participant suggested the name /rename for one of Treesta's commands, which initially was /space. The functionality of this command is to change the name of the active workspace; however he commented that /space is misleading since one assumes that as a result control jumps to the mentioned workspace, not that it changes the name of the current workspace. Second, by observing his questions and reactions, the complicated concept of sharing variables was removed for the pilot test, since there is hardly any use for it in a basic evaluation.

The most important outcome from the pilot study, which was in accordance with my assumptions, was that Treesta is not quite useful for a novice user. The participant, who was not an experienced analyst, followed just one path instead of trying various solutions. He also did not try reflecting on his actions using commands like /summary and
/structure. Although the ability to organize the work, assigning meaningful names to workspaces, and saving and restoring the whole project is beneficial to all users, the more advanced features, such as replay, and sharing, are more likely to be utilized by expert statistical analysts. Furthermore, we believe that even the main opportunity that Treesta provides, the simultaneous parallel paths, is more advantageous for expert statisticians. Akin has observed such behavior among architects (Akin, 2001). He reports that novice architects follow one solution until they face some problems, or a dead end, whereas more experienced architects tend to employ a mixed strategy of depth-first and breadth-first, pursuing more than one solution at the same time. Referring to this observation, the possibility of pursuing multiple solutions in Treesta benefits the experienced analysts more than the novice ones.

6.7 Improvements to Treesta Suggested by the Evaluation

The first option to improve Treesta is to design a graphical user interface. It could display the overview of the structure of the workspaces and some information related to the current workspace (e.g., the list of the variables) permanently, as well as menu entries for accessing frequently used activities. Such an interface can also display the list of the entered commands in the active workspace, similar to how Matlab does. This is an alternative to the existing /history command, and provides the user with a constant perception of the activities in this workspace, a more convenient choice than entering the /history command frequently.

We might also remove the restriction of one ANOVA per workspace, or at least allow assignments in the workspace after entering an ANOVA command. This removes the
sometimes-frustrating necessity to create a new child to continue the analysis after ANOVA. However, it will influence other commands, such as `/dependency`, as well, since it is unclear which ANOVA command is the starting point, which can be clarified by asking the user. Other suggestions of the participant, such as keeping track of the time spent in workspaces and saving the position of figures on screen, can also be considered.

In this study, the participant did not apply an ANOVA command in the root workspace, so he would be able to apply commands and assignments there, however this behavior results in a complication. If the value of a variable in a workspace is altered after the children of the workspace have inherited it, then the history of commands in the two workspaces do not show how the variable in the child workspace was derived from the parent workspace. In addition, if after changing the variable, a new child were created, the two children would not be consistent in the value of that variable. Future improvement of Treesta should consider solving this problem, for instance through informing the user.
CHAPTER 7:

Conclusion

In this thesis, I introduced supporting performance of the process of statistical analysis. Although there are many systems that help users create products, none is focused on improving the process that leads to those end products. Problem solving is not just following a sequence of steps, guaranteed to result in the correct solution. Many problems do not generally have a single best answer. In dealing with such cases, the experienced practitioner generates several ideas, and follows various paths instead of just one, and a performance support system might improve the performance by helping in creating, comparing and reflecting on these solutions.

The same issue exists in the field of data analysis. While many statistical packages help analysts carry out calculations, they do not support the process of analyzing the data. Having in mind the properties for a performance support tool, I have developed a system that features the following capabilities: capturing history, representing externally, supporting reflection in/on action, and allowing the development of multiple solutions. Treesta helps analysts organize their analysis works and results in a more informed data analysis.

This system organizes the analysis work into a tree of workspaces; each having a name, a set of figures and variables. The user can also annotate the workspace, whenever there is
a need to explain or remind something. Treesta, which stands between Matlab and the user, augments Matlab’s services by offering a set of commands that help maintain the environment, by presenting the structure of the tree and summary of each workspace. Some of these commands serve more functionalities such as viewing the dependency links among variables in a workspace, and replaying a portion of history.

We believe that by helping users organize and annotate their work, and also offering other mentioned services, Treesta matches the work flow of analysts much more closely than the existing statistical packages. To evaluate this, a user study, with a single participant, was conducted, confirming our main assumptions as he found the following features useful: tree structure, the possibility to follow multiple paths, the annotation tool and the presentation of the structure of the environment.

Evaluating Treesta is a big project, since Treesta’s goal is to improve the quality of the process of data analysis and the confidence of statisticians, and assessing this does not include evaluating the speed of analysis, and answering the wrong question correctly. We have taken one step in assessing the theory base and major features of the system. In future, we should conduct a more thorough study, with an expert data analyst, covering a broader range of the functionality (e.g., sharing variables). We should ask the participant to use Treesta in a real analysis project, which might go on over several days, and include closing and reloading the environment a few times. It will give us more feedback about the contribution of our work, how it can be improved and will point out the usability problems.
Another improvement is related to the Treesta’s summary of activities. Performing many tests on the same dataset is controversial, and in future versions, Treesta might include multiple comparison corrections while reporting the $p$ values. We could also consider implementing more characteristics of a computer system that supports creativity, like including a user manual with a guide on how to use a command along with some examples, and a library of relevant problems.

Treesta is the first system that supports performance of the process of data analysis, combining data analysis and the concept of creative problem solving. Featuring several characteristics for supporting creative problem solving in computer systems, it results in a more informed and more convenient data analysis. A user study conducted with a single participant confirmed Treesta’s main assumptions and goals. Treesta’s services, such as capturing and representing history and allowing multiple solutions, are not limited to statistical analysis, but in fact helpful in the general domain of supporting creative problem solving.
BIBLIOGRAPHY


