

6-24-2016

# Supporting Web-based and Crowdsourced Evaluations of Data Visualizations

Mershack B. Okoe

*Florida International University*, [mokoe001@fiu.edu](mailto:mokoe001@fiu.edu)

**DOI:** 10.25148/etd.FIDC000757

Follow this and additional works at: <https://digitalcommons.fiu.edu/etd>

 Part of the [Graphics and Human Computer Interfaces Commons](#)

---

## Recommended Citation

Okoe, Mershack B., "Supporting Web-based and Crowdsourced Evaluations of Data Visualizations" (2016). *FIU Electronic Theses and Dissertations*. 2566.

<https://digitalcommons.fiu.edu/etd/2566>

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact [dcc@fiu.edu](mailto:dcc@fiu.edu).

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

SUPPORTING WEB-BASED AND CROWDSOURCED EVALUATIONS OF  
DATA VISUALIZATIONS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Mershack Bortey Okoe

2016

To: Interim Dean Ranu Jung  
College of Engineering and Computing

This dissertation, written by Mershack Bortey Okoe, and entitled Supporting Web-based and Crowdsourced Evaluations of Data Visualizations, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

---

Christine Lisetti

---

Peter J. Clarke

---

Malek Adjouadi

---

Mark A. Finlayson

---

Jian Chen

---

Radu Jianu, Major Professor

Date of Defense: June 24, 2016

The dissertation of Mershack Bortey Okoe is approved.

---

Interim Dean Ranu Jung  
College of Engineering and Computing

---

Andrés G. Gil  
Vice President for Research and Economic Development  
and Dean of the University Graduate School

Florida International University, 2016

© Copyright 2016 by Mershack Bortey Okoe

All rights reserved.

## DEDICATION

To my wife Cynthia, and my parents Grace and Emmanuel for their endless support and believe in me.

## ACKNOWLEDGMENTS

Without the support and guidance of numerous people, this dissertation would not have been possible.

First, I would like to thank my advisor Radu Jianu for his guidance, mentorship, and support throughout this program. I greatly appreciate Radu's commitment to ensuring I obtained the necessary skills as a researcher. From brainstorming ideas together, to debugging codes together, Radu's involvement throughout the research process provided me with great research experiences and thoughtful lessons. I am very grateful for his influence as a supporter, mentor, challenger, and sounding-board through this research journey. I could not have imagined a more effective advisor.

Second, I would like to thank my committee members Christine Lisetti, Peter J. Clarke, Malek Adjouadi, Mark A. Finlayson, and Jian Chen for their time and effort in helping me improve this dissertation. I thank them for their useful comments and valuable feedback which helped made this dissertation better.

I am very grateful for professors in the School of Computing and Information Sciences whose classes and ideas had a major influence in my vision of Computer Science as a whole: thanks to Mark Weiss, Tao Li, Alex Pelin, and Geoffrey Smith.

There are several amazing colleagues in the School of Computing and Information Sciences who I was fortunate to surround myself with. I would like to thank my fellow VizLab mates Sayeed Safayet Alam, and Nahid Ferdous for their friendship, and for being good collaborators on research projects. I am grateful for the friendship and company of several current and past colleagues of the School

of Computing and Information Sciences including: Hasan Mahmud, Banik Madhusudan, Ugan Yasavur, Mahmudur Rahman, Mingming Guo, Kishwar Ahmed, and Melita Jaric.

I am also very grateful to the staff of the School of Computing and Information Sciences: thanks to Olga Carbonell for her good cheer and for making the bureaucratic aspect of the university seem tractable, Donaley Dorsett and Maureen Braham for their cheerful and friendly conversations, and Tiana Solis for the numerous conversations and encouragements.

I would also like to thank my amazing friends Albert and Ruby Yeboah-Forson for their friendship and for proofreading this dissertation. Thanks to Albert for his companionship and level-headed advices on issues related to school and life.

Finally, I would like to express my deepest gratitude to my family. Many thanks to my parents, Grace and Emmanuel for their love and endless believe in me. Special thanks to my wife, Cynthia, for being a great companion, loving, understanding, and for keeping me sane through this adventure. Many thanks to my brothers and sisters for their support and encouragements throughout this process.

ABSTRACT OF THE DISSERTATION

SUPPORTING WEB-BASED AND CROWDSOURCED EVALUATIONS OF  
DATA VISUALIZATIONS

by

Mershack Bortey Okoe

Florida International University, 2016

Miami, Florida

Professor Radu Jianu, Major Professor

User studies play a vital role in data visualization research because they help measure the strengths and weaknesses of different visualization techniques quantitatively. In addition, they provide insight into what makes one technique more effective than another; and they are used to validate research contributions in the field of information visualization. For example, a new algorithm, visual encoding, or interaction technique is not considered a contribution unless it has been validated to be better than the state of the art and its competing alternatives or has been validated to be useful to intended users. However, conducting user studies is challenging, time consuming, and expensive.

User studies generally requires careful experimental designs, iterative refinement, recruitment of study participants, careful management of participants during the run of the studies, accurately collecting user responses, and expertise in statistical analysis of study results. There are several variables that are taken into consideration which can impact user study outcome if not carefully managed.



Hence the process of conducting user studies successfully can take several weeks to months.

In this dissertation, we investigated how to design an online framework that can reduce the overhead involved in conducting controlled user studies involving web-based visualizations. Our main goal in this research was to lower the overhead of evaluating data visualizations quantitatively through user studies. To this end, we leveraged current research opportunities to provide a framework design that reduces the overhead involved in designing and running controlled user studies of data visualizations. Specifically, we explored the design and implementation of an open-source framework and an online service (VisUnit) that allows visualization designers to easily configure user studies for their web-based data visualizations, deploy user studies online, collect user responses, and analyze incoming results automatically. This allows evaluations to be done more easily, cheaply, and frequently to rapidly test hypotheses about visualization designs.

We evaluated the effectiveness of our framework (VisUnit) by showing that it can be used to replicate 84% of 101 controlled user studies published in IEEE Information Visualization conferences between 1995 and 2015. We evaluated the efficiency of VisUnit by showing that graduate students can use it to design sample user studies in less than an hour.

Our contributions are two-fold: first, we contribute a flexible design and implementation that facilitates the creation of a wide range of user studies with limited effort; second, we provide an evaluation of our design that shows that it can

be used to replicate a wide range of user studies, can be used to reduce the time evaluators spend on user studies, and can be used to support new research.

## TABLE OF CONTENTS

CHAPTER	PAGE
1 INTRODUCTION .....	1
1.1 Thesis Overview .....	4
1.2 Original Contribution of the Study .....	6
2 LITERATURE REVIEW .....	7
2.1 Background on user studies .....	7
2.1.1 Controlled user studies.....	9
2.1.2 Independent and dependent variables .....	10
2.1.3 Trials.....	11
2.1.4 Examples of controlled user studies.....	11
2.1.5 Typical processes involved in controlled user studies.....	15
2.1.6 Typical protocol used in running controlled user studies.....	16
2.1.7 Within and between subjects study design .....	17
2.1.8 Learning effects.....	18
2.1.9 Controlling fairness and other factors that affect study validity .....	19
2.2 Tasks adapted in user studies .....	19
2.2.1 Recruiting participants.....	21
2.2.2 Web-based and crowdsourced user studies .....	22
2.2.3 Amazon mechanical turk .....	23
2.3 Research trends that motivate this thesis .....	24
2.4 Related work .....	26
2.5 Summary.....	28
3 PROBLEM DEFINITION AND METHODOLOGY .....	30
3.1 Research Problem .....	30
3.2 The need for a framework design that facilitates a wide range of controlled user studies. ....	30
3.2.1 The Existing Problem: .....	32
3.3 Research goals .....	33
3.4 Research Questions .....	33
3.4.1 High-Level Questions.....	33

3.4.2	Middle-Level Questions.....	34
3.4.3	Lower-level questions.....	34
3.5	Research methodology.....	36
3.5.1	Designing a flexible user friendly online framework (VisUnit) that semi-automates the processes of visualization user studies and supports many types of user studies.....	37
3.5.2	Evaluating the effectiveness and efficiency of VisUnit's design.....	38
3.6	Summary.....	39
4	GRAPHUNIT: A FRAMEWORK TO SUPPORT THE EVALUATION OF GRAPH USER STUDIES.....	41
4.1	Architecture.....	42
4.1.1	Extending GraphUnit with new datasets and tasks.....	44
4.2	Configuration of user studies.....	46
4.2.1	Connecting a visualization.....	46
4.2.2	Configuring.....	46
4.2.3	Putting studies on Mechanical Turk (AMT).....	49
4.3	Running the user study.....	49
4.4	Optional methods.....	50
4.5	Analyzing study results.....	52
4.6	Evaluation.....	53
4.6.1	Study I - Evaluating node link diagrams vs. matrix diagrams.....	53
4.6.2	Evaluating multiple ways to represent edge directionality in node link diagrams.....	55
4.6.3	Study III - Configuring available visualization for a user study.....	58
4.7	Summary.....	63
5	VISUNIT - A FRAMEWORK THAT FACILITATES EVALUATION OF DATA VISUALIZATIONS.....	64
5.1	Designing VisUnit.....	64
5.2	VisUnit Architecture.....	66
5.3	Setup Interface to support the design of user studies.....	69
5.3.1	Specifying independent variables and experimental conditions.....	69
5.3.2	Specifying experimental design.....	73
5.3.3	Introductions.....	74

5.3.4	Standardized tests.....	74
5.3.5	Pre-study survey questions .....	75
5.3.6	Specifying post-study survey questions .....	75
5.3.7	Specifying study tasks.....	76
5.3.8	Post-study survey questions .....	77
5.3.9	Viewer dimensions .....	77
5.3.10	Running a demo and saving study designs for future use .....	78
5.4	Study Management Interface.....	79
5.5	User study interface to automatically manage study runs .....	80
5.5.1	Study-Run Manager .....	81
5.5.2	Choosing appropriate experimental conditions .....	82
5.5.3	Presenting introductions.....	83
5.5.4	Presenting standardized tests .....	84
5.5.5	Presenting pre-study survey questions .....	84
5.5.6	Presenting study tasks and training .....	84
5.5.7	Presenting post-study survey questions.....	90
5.5.8	Providing task translations .....	90
5.5.9	Providing in-situ task translations.....	91
5.6	Results Interface to support analysis of study results.....	92
5.6.1	Support filtering and cleaning of results data .....	92
5.6.2	Provide graph summaries of the results.....	94
5.6.3	Provide statistical analysis of the study results .....	96
5.7	Extending VisUnit with new tasks and datasets.....	99
5.7.1	Creating new tasks and survey questions.....	99
5.7.2	Creating new task instances .....	106
5.7.3	Adding new datasets .....	108
5.8	VisUnit storage.....	109
5.9	Summary.....	109
6	VALIDATION: EVALUATION OF VISUNIT.....	110
6.1	Evaluating the effectiveness of VisUnit.....	111
6.1.1	Ziemkiewicz et al. [91].....	116

6.1.2	Laidlaw et al. [29] .....	117
6.1.3	Heer et al. [12].....	117
6.1.4	Haroz et al. [88].....	119
6.1.5	Robertson et al. [38].....	120
6.1.6	Jianu et al. [15].....	122
6.1.7	Discussion of the limitations of VisUnit.....	123
6.2	Evaluating the efficiency of VisUnit.....	116
6.2.1	User study setup .....	125
6.2.2	User study results.....	129
6.2.3	Survey involving researchers who conduct controlled user studies .....	129
6.2.4	Survey results.....	131
6.2.5	Statistical comparison of user study results and baseline survey results .....	133
6.3	Using VisUnit to support research studies .....	135
6.3.1	Ecological validity in quantitative user studies – a case study in graph evaluation.....	135
6.3.2	Original study .....	136
6.3.3	A discussion of ecological validity .....	136
6.3.4	User study .....	140
6.3.5	Results .....	142
6.4	Summary.....	143
7	CONCLUSION AND FUTURE WORK.....	145
7.1	Summary.....	145
7.2	Lessons learned in this research .....	146
7.3	Contributions.....	148
7.4	Future directions .....	149
	REFERENCES.....	152
	VITA .....	169

## LIST OF TABLES

TABLE	PAGE
2.1: A table showing a within-subject design and a between-subjects design. ...	17
2.2: Low-level tasks that are commonly used in the visualization user studies...	20
5.1: Statistical Analysis that are performed depending on the experimental design of the user study and the number of experimental conditions involved in the study.....	98
5.2: Types of effect sizes that are performed depending on the statistical analysis. ....	98
6.1: The properties of a list of user studies published in InfoVis that can be supported by VisUnit .....	113
6.2: Results table showing the times in minutes that that it took study participants to create tasks, create task instances, and design user studies and see demos .....	129
6.3: Survey results table showing the times researchers expected graduate students to use in designing the user study. ....	132
6.4: Survey results table showing the time researchers reported to spend on designing user studies (Q3), running user studies (Q4), and analyzing the results of user studies (Q5). ....	132

## LIST OF FIGURES

FIGURE	PAGE
2.1: Three different types of directed edges: tapered (a), curved (b), and arrow-head (c). .....	11
2.2: Two different representations of graph data, node-link (a) and matrix (b). ..	12
2.3: Six different visualization methods for displaying 2D vector data. Image taken from [29]. .....	13
2.4: Three of the stimuli used for judgment tests in Heer et al. [12]. Image taken from Heer et al. [12]. .....	14
2.5: One of the stimuli used for judgment tests in Heer et al. [30]. .....	14
2.6: The typical processes that are involved in user studies. ....	16
2.7: Examples of Latin squares for 2, 3, and 4 conditions. ....	19
3.1: An overview of research questions that were approached and answered in this thesis. ....	35
3.2: Overview of research methodology: First, an investigation was done to determine how to design a framework to support user studies of a specific visualization type (i.e. graph network). Second, requirements for different visualization types were analyzed. Third, VisUnit was designed, implemented and evaluated. ....	36
4.1: Architecture of GraphUnit. ....	42
4.2: An example of a task file. ....	44
4.3: Options of quantitative tasks that can be used for the evaluation. ....	45
4.4: An interface for configuring a user study. ....	47
4.5: An example of a study specification file. ....	48



4.6: An example of a user study, showing three stages: instruction about task, training, and study. ....	51
4.7: The graph generated for the results of the user studies. Error bars are standard errors. ....	58
5.1: Architecture of VisUnit .....	68
5.2: Study setup page Part I. ....	70
5.3: Study setup page part2. ....	71
5.4: Study setup page part3 .....	72
5.5: An XML file showing details of a designed user study. ....	78
5.6: A Page for managing existing user studies. ....	80
5.7: The introduction stage of the user study. VisUnit loads the introduction file into an iframe. ....	86
5.8: A short instruction about the tasks and training involved in the user study. ....	87
5.9: Presenting training tasks. Participants can check the accuracy of their answers to quantitative tasks and the study is not timed. ....	88
5.10: Presenting actual study tasks. A countdown timer is started and the viewer window is hidden when the countdown timer gets to zero. ....	89
5.11: Results page (Part 1), showing a results table of the summarized quantitative results, with two rows filtered. ....	94
5.12: Results page (Part2). Showing the number of completed studies, bar charts of the accuracy and completion time of the quantitative tasks, and links to statistical analyses. ....	95
5.13: An interface for creating new tasks. ....	102
5.14: An example of an XML file for tasks. ....	106
5.15: An example of an XML file for task instances. ....	108

6.1 : Radial tree visualization involved in the designed user study .....	127
6.2 : Collapsible indented tree involved in the designed user study.....	128
6.3 : A bar-chart showing the mean time of the user study results (withVisUnit) vs the mean time of the survey results (withoutVisUnit). ....	133
6.4 : Lexicographically ordered AMs (left) cannot reveal graph structure in the same way that a clustered AM (center) and an NLD (right) can .....	137
6.5 : Available interactions: hovering/selecting a node highlights it and its edges green/red; hovering a link highlights it and its end-points green. The images illustrate a task 5a instance .....	142
6.6 : Accuracy and time results for the three tasks. ....	143

## LIST OF ABBREVIATIONS AND ACRONYMS

2D	Two Dimension
3D	Three Dimension
AJAX	Asynchronous Javascript
AMT	Amazon Mechanical Turk
ANOVA	Analysis of variance
CSV	Comma separated values
GRID	Icons on a Regular Grid
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTML 5	Hypertext Markup Language version 5
InfoVis	Information Visualization
JIT	Icons on a Jittered Grid
JSON	Javascript Object Notation
LIT	Icons using a layer that borrows from oil painting
TSV	Tab separated values
XML	Extensible Markup Language

## 1 INTRODUCTION

Information visualization (InfoVis) is the use of visual representations of abstract data to amplify human cognition [1]. One of the main goals of InfoVis is to enable the quick absorption of large amounts of data by leveraging the powerful human visual system. InfoVis researchers increasingly work on algorithms, visual designs, and interaction techniques for visualizing and analyzing data. The scientific InfoVis process necessarily involves evaluating the effectiveness of such algorithms, designs, and interaction techniques. These investigations are mostly conducted with some form of user-driven evaluation, since the effectiveness of a visualization system is typically measured in terms of its ability to help users extract information or insight from it. As such, user-driven evaluation forms a key element in InfoVis research and human-centered visualization designs [2, 3, 4].

User-driven evaluations, also known as user studies in human centric experiments, are fundamental to validating research contributions in the field of visualization. For example, a new algorithm, visual encoding, or interaction technique is not considered a contribution unless it has been validated to be better than the state of the art and its competing alternatives or has been validated to be useful to intended users. Similarly, a design study (which involves designing a visualization system to support the workflow of a domain expert), is considered a contribution if it is validated to efficiently support the intended workflows of the domain expert. User studies are also instrumental in informing the choices visualization designers make between wide array of visualization types and

interaction techniques. For example, a network or relational data can be displayed as either a node-link or a matrix, and each method may support different tasks with different degrees of effectiveness. With the help of a user study, alternatives can be weighed to discover which one is more effective in supporting the goals of the designer. Finally, user studies can highlight problems with visual designs. For example, a user study can be used to pinpoint visualization properties that are problematic to users, which user tasks are inadequately supported by a given visualization, and which changes can be made to improve user experience.

Despite all the listed advantages, conducting user studies is challenging, time consuming, and expensive [2, 3, 5, 6]. These challenges may be a contributing factor to the disproportionately small number of formal user studies done in the infoVis community. For example, Lam et al. [7] revealed that only 42% of 850 papers published in the major visualization venues (between 2002 and 2012) reported an accompanying evaluation. Building infrastructure to facilitate the evaluation of visualizations is seen as a step towards making user studies more feasible for evaluators [8] [9] [10].

The goal of this dissertation is to reduce the overhead involved in designing and running controlled user studies of web-based data visualizations so evaluations can be done more easily, cheaply, and frequently to rapidly test hypotheses about visual designs. This goal was pursued with the following two objectives and contributions:

**Contribution 1:** Design of a framework (VisUnit), that can reduce the overhead involved in designing and running web-based user studies, is user

friendly, and is sufficiently flexible to support a wide range of user study types. Prior to this work, such a framework did not exist, and there were no design recommendations for implementing one.

To this end we explored requirements for the design from previous user studies in the information visualization literature. Key technologies were explored and leveraged. A prototype framework (GraphUnit) that facilitates graph user studies was designed, implemented, and evaluated. Several user studies were ran to gather more requirements for different user study and visualization types; and VisUnit was designed and implemented. VisUnit is offered as an open source framework to be used and extended by the visualization community, and is also offered as an online service to provide real time support and time saving functionalities for user study evaluators.

**Contribution 2:** An evaluation of the effectiveness and efficiency of using such a framework (VisUnit) to support a wide range user study types.

To this end we demonstrated the effectiveness of the design, VisUnit, by showing that it can be used to replicate a wide range of existing user studies. Specifically, 84% of 101 controlled user studies published in IEEE Information Visualization conferences between 1995 and 2015 can be replicated with VisUnit. These user studies involve graphs, multidimensional visualizations, trees, 2D areas, and temporal visualizations. We also conducted a user study involving 5 computer science students to demonstrate the efficiency of the design. Students were asked to design user studies to evaluate alternate visual encodings using freely available web visualizations. On average, it took participants one hour to

design a study. In summary, the study portrays the efficiency of the design by showing that evaluators can use VisUnit to design user studies in less than an hour, run studies and analyze study results within a day.

An important merit of our work is recognizing that current research trends create the opportunity to significantly simplify and stream-line the evaluation of data visualization systems. First, current advances in web development such as HTML5, D3 [11], and WebGL, have prompted a migration of visualizations towards the web with increasing number of visualizations being targeted at web-based users. Second, crowdsourcing has been established as an effective tool for evaluating visualizations [12, 13, 14, 15]. Third, the visualization community has standard design guidelines [2, 3, 5, 6, 16, 17, 7] and task taxonomies [18, 19, 20, 21, 22, 23] to support the evaluation of visualizations. These advances create two opportunities: First, the overhead of evaluating visualizations can be lowered by semi-automating the processes involved in designing and running user studies, by managing study participants on the web and collecting their data automatically. Second, the size of study participants can be expanded, and studies can be targeted to users of diverse or specific demographics and expertise. This dissertation investigated a design of a framework to help user study evaluators harness these opportunities.

## **1.1 Thesis Overview**

The dissertation is organized into 7 chapters. Introduction and motivation for the study are contained in Chapter 1, while the conclusions and contributions of

the study are presented in Chapter 7. Chapter 2 provides background information, literature review and related work necessary for laying the basic foundation of the entire study. Chapter 3 presents problem definition and methodology used in this dissertation while providing information on the key questions to be addressed.

Chapter 4 introduces the design, implementation, and evaluation of a prototype framework, GraphUnit, that semi-automates the process of designing, running, and analyzing results of graph user studies. GraphUnit simplifies the process of designing and fielding controlled quantitative user evaluations of web-based graph visualizations. This prototype provided a solution for a smaller problem that helped fine-tune the general design requirements and also helped test hypotheses on a smaller scale. The design and primary results contained in this chapter have been published in the Computer Graphic Forum (CGF) journal. GraphUnit is currently available as open-source software at <http://vizlab.cs.fiu.edu/graphunit/>

Chapter 5 describes the novel design and implementation of VisUnit, a framework that semi-automates the process of designing, running, and analyzing results of a wide range of user studies that involve many visualization types. VisUnit is flexible, user-friendly and allows evaluators to design user studies with their own tasks and datasets, automatically manages the run of user studies. VisUnit automatically generates statistical analyses for finished studies, and provides functionalities to support the cleaning of results data. The detailed architecture and design of VisUnit are intended for submission to the IEEE Transactions on Visualization and Computer Graphics (TVCG) journal.



Chapter 6 evaluates VisUnit's effectiveness by showing that it can be used to replicate several existing user studies in the visualization literature, and also describe an example of a research work that was supported by VisUnit. The efficiency of VisUnit was tested by evaluators with advance degrees in the field who are familiar with information visualization and who are unaffiliated to this project. VisUnit is currently available as open-source software at <http://vizlab.cs.fiu.edu/visunit/>

## **1.2 Original Contribution of the Study**

The major original contributions of this dissertation, which can be found in Chapters 4, 5, and 6 discusses the significance of this work primarily to the field of information visualization. First, this dissertation introduced a novel design that facilitates web-based user studies and supports a wide range of user study types. Secondly, an implementation of an open-source system and an online service that offers time saving functionalities to evaluators was introduced. Thirdly, we provide evaluations to demonstrate the efficiency and effectiveness of our design to support a wide range of user studies, and we provide an example of a research work that was supported by our design. Ultimately, this research offers a new way of designing web-based infrastructure to facilitate user studies, and shows the potential impact of VisUnit to the visualization community.

## **2 LITERATURE REVIEW**

This chapter presents a detailed background information on user studies, research trends that motivated this dissertation, as well as previous works related to this dissertation.

### **2.1 Background on user studies**

Information visualization (InfoVis) is the study of visual representations of abstract data to amplify human cognition [1]. InfoVis researchers study algorithms, visual designs, and interaction techniques for visualizing and analyzing data.

This thesis targets evaluation studies involving real users that allow visualization designers to obtain empirical evidence of the usability of their designs [2]. User studies play an important role in data visualization research because they allow us to measure the strengths and weaknesses of different visualization techniques, provide insight into what makes one technique more effective than another, demonstrate the practical use of new techniques, and inform refinement and redesigns of techniques [2, 5].

User studies are fundamental to validating research contributions in the field of visualization. For example, a new algorithm, visual encoding, or interaction technique is not considered a contribution unless it has been validated to be better than the state of the art and its competing alternatives or has been validated to be useful to intended users. User studies are also instrumental in informing the choices visualization designers make between the wide array of visualization types

and interaction techniques. For example, a network or relational data can be displayed as either a node-link or a matrix. Knowing which visual encoding is better for a specific case will depend on the task that the visualization will be used to perform, the data, or the domain. With the help of a user study, alternatives can be weighed to know which one is more effective in supporting the goals of the designer.

However, conducting user studies is challenging, time consuming, and expensive [2, 3, 5, 6, 17]. Challenges faced by evaluators include finding the right variables to evaluate, picking the right tasks and datasets, choosing the right methodology, and being rigorous in procedure and data collection. Evaluators also face challenges in recruiting participants and analyzing study data [1, 7]. These challenges may partly explain a widening gap between technique development and their independent formal evaluation in the visualization community. For example, the most recent comprehensive graph drawing survey cited about 100 papers on techniques and only about 30 papers on design and evaluation studies together [15, 24]. Lam et al. [7] also revealed that only 42% of 850 papers published in the major visualization venues (between 2002 and 2012) reported an evaluation. The work presented in this dissertation helps reduce the overhead involved in conducting controlled user studies and allows evaluators to easily design study protocols, field such studies with online crowdsourcing, and receive appropriate analyses of the study results.

In visualization research, user studies can broadly be grouped into three categories [2]: controlled studies - which compare two or more designs

quantitatively; usability evaluations - which are used to identify problems users encounter with a design; and case studies - which are used to observe how users use designs in their natural environment. Case studies and usability evaluations both answer evaluation questions with subjective human responses with the goal of deriving information that can be used to improve a given visualization system.

Controlled user studies on the other hand are quantitative studies aimed at producing generalizable and reproducible results [7]. This dissertation focused on controlled user studies because they follow an established methodology, are challenging to conduct, can be successfully run with web-based or crowdsourced participants, and are common in the visualization community [7, 25].

### **2.1.1 Controlled user studies**

Controlled user studies are quantitative studies used to compare two or more visualizations or interactive techniques by measuring human performance (typically accuracy and time) on simple tasks abstracted from real life tasks. For example, a controlled study can be employed to determine if a node-link visualization is better at depicting graph connections than a matrix visualization by allowing study participants to repeatedly identify if two selected nodes are connected in each visualization and comparing their performance in the two visualizations.

In controlled user studies, evaluators follow a rigorous process of developing hypotheses, identifying independent variables, choosing tasks that users will perform, measuring dependent variables such as performance time and accuracy, and using statistics to declare confidence in the results [17]. Ultimately, due to the

relatively high precision in controlling experimental factors and measuring dependent variables, study results of controlled user studies can be generalized to a larger population [17].

### **2.1.2 Independent and dependent variables**

Independent variables are characteristics related to the properties of a visualization system that can be manipulated or controlled in an experiment for a change in user behavior [26]. Independent variables are manipulated across at least two levels of the characteristic. In visualization, an independent variable can take the form of a visualization type or visual encoding, (e.g. different visual encodings for representing multidimensional data, such as parallel coordinate plots vs. star plots), an interaction technique (e.g. different ways of performing a selection), or a dataset (e.g. a small dataset vs a big dataset). The levels of an independent variable are known as "test conditions" or "conditions".

Dependent variables on the other hand are human behaviors that can be measured while users are interacting with the independent variables. The common dependent variables used in visualization experiments are task completion time and task accuracy (or error rate). More broadly, dependent variables can be any measurable human behavior that can provide an insight into the difference in strength for any two test condition such as answer re-entries, number of interactions used, or amount of time spent in training [26].

### 2.1.3 Trials

Typically, each user task is repeated with several task instances, each of them is known as a trial, and the mean of the trials is taken as the user's performance on the user's task. For example, for the connectivity graph task "Are the two highlighted nodes connected?", there will be several trials or task instances, with each trial having two different highlighted nodes.

### 2.1.4 Examples of controlled user studies

Below are examples of controlled user studies in visualization.

**Example 1** - Holten et al. [27] investigated how different types of directed edges (Figure 2.1) affect user performance on graph tasks. The independent variable used was "directed edge", and the test conditions of the independent variable were "tapered edge", "curved edge", and "arrow-head edge". The measured dependent variables were accuracy and completion time.

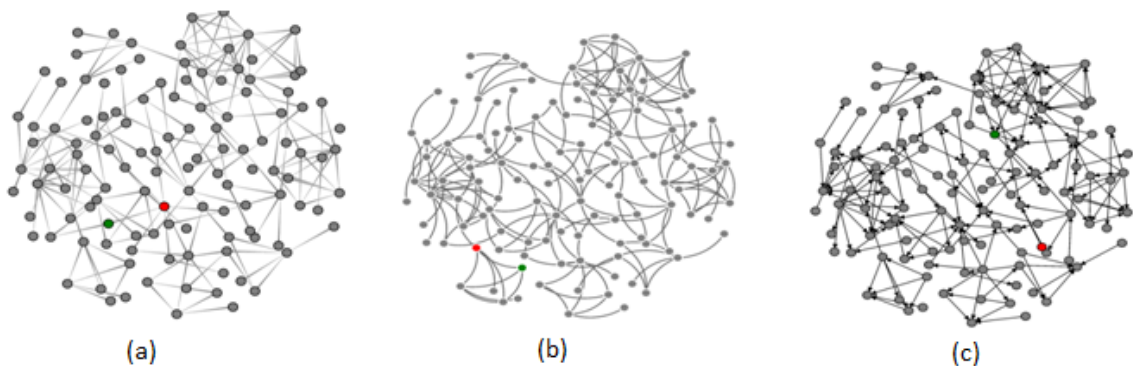


Figure 2.1: Three different types of directed edges: tapered (a), curved (b), and arrow-head (c).

**Example 2** - Ghoniem et al. [28] investigated the readability of two different graph representations (Figure 2.2) on graph tasks. The independent variable used was "graph representation", and the test conditions of the independent variable were node-link, and matrix. The measured dependent variables were accuracy and completion time.

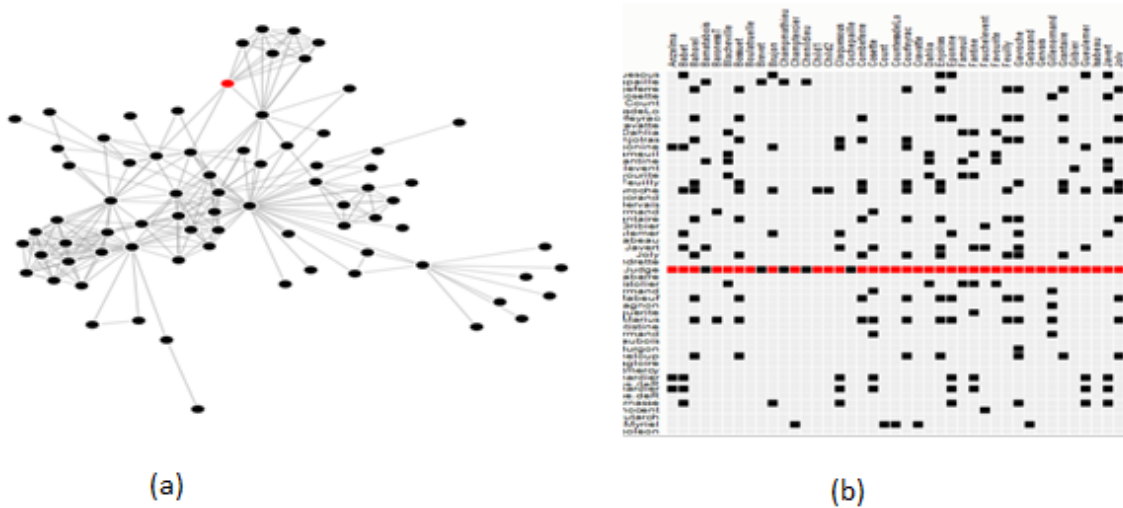


Figure 2.2: Two different representations of graph data, node-link (a) and matrix (b).

**Example 3** - Laidlaw et al. [29] compared six visualization methods used for displaying two dimensional vector data (Figure 2.3) using tasks related to fluid mechanics. The independent variable used was "visualization methods for 2D vector data", and the test conditions were icons on a regular grid (GRID), icons on a jittered grid (JIT), icons that borrow concepts from oil painting (LIT). The measured dependent variables were accuracy and completion time.

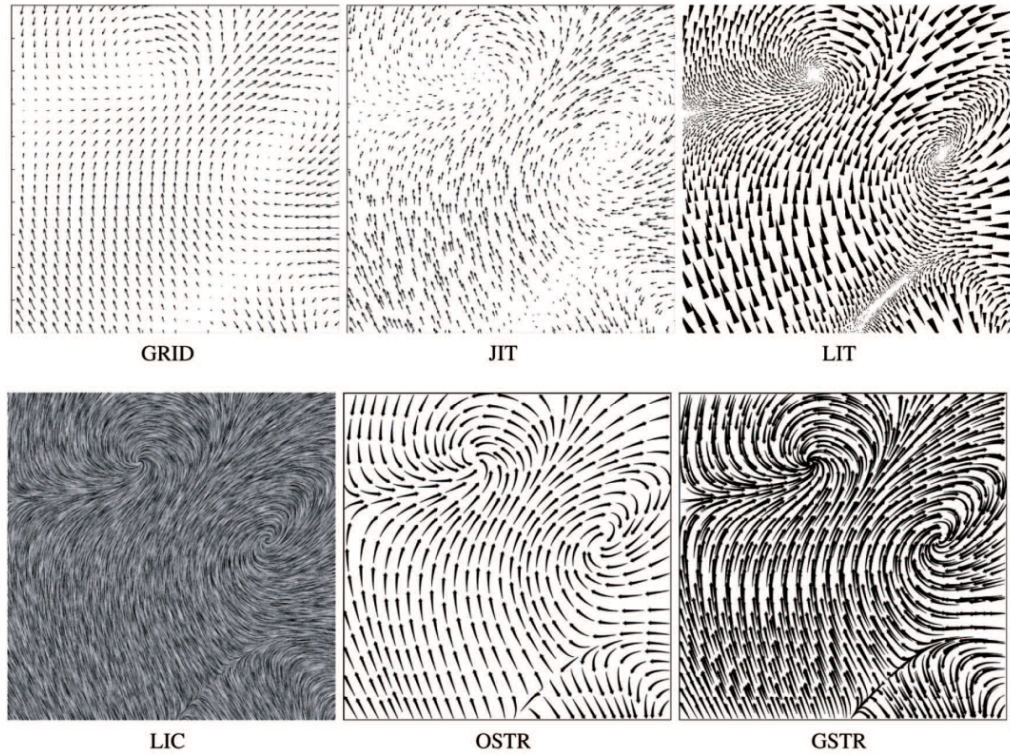


Figure 2.3: Six different visualization methods for displaying 2D vector data. Image taken from [29].

**Example 4** - Heer et al. [12] compared judgment types corresponding to different types of visual encodings (Figure 2.4) using judgment tasks. The independent variable used was "types of judgments", and the test conditions were "judgment based on position along a common scale", "judgment based on length", and "judgment based on "angle". The measured dependent variables were



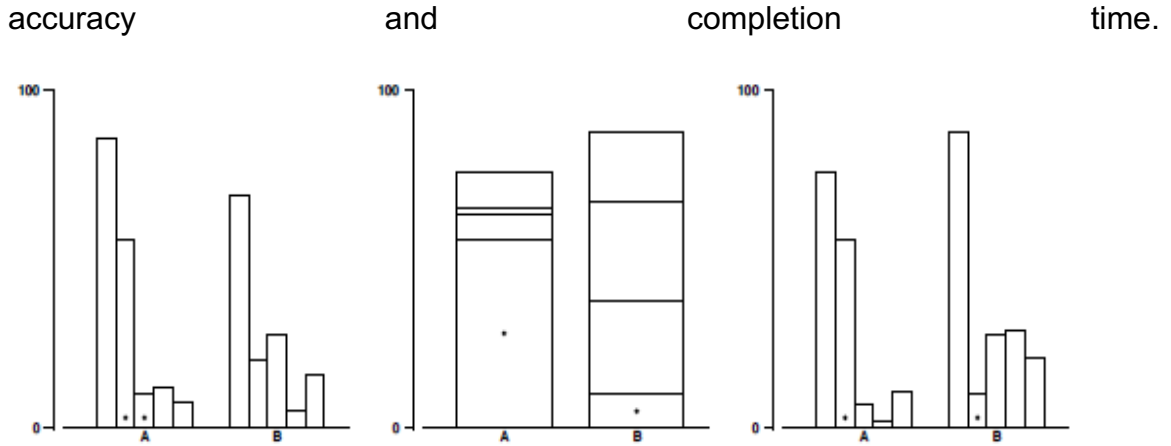


Figure 2.4: Three of the stimuli used for judgment tests in Heer et al. [12]. Image taken from Heer et al. [12].

**Example 5:** Heer et al. [30] evaluated the effect that chart size and layering have on perceptions of time series visualizations (Figure 2.5) using judgment tasks. The independent variable used was "chart type - size type", and the test conditions used include 3 (charts) x 4 (chart sizes). The measured dependent variables were accuracy and completion time.

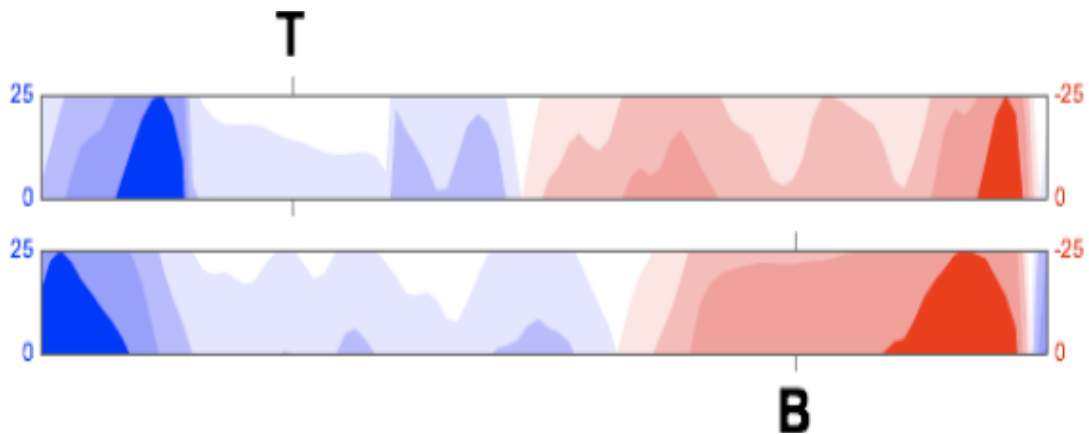


Figure 2.5: One of the stimuli used for judgment tests in Heer et al. [30].

### 2.1.5 Typical processes involved in controlled user studies

The methodology of quantitative empirical evaluations has been around for centuries and they include developing hypothesis, identifying independent variables, controlling other factors of the study, measuring dependent variables, and applying statistics to the study [31, 32]. Controlled user studies in visualization generally follow this established methodology [17].

The processes commonly used by evaluators can be grouped into 5 stages [13, 29, 33, 34, 35, 36, 37, 38]:

**Stage 1:** Develop hypothesis - identify the independent variables for the study (i.e. visualizations or datasets or combination of visualization and dataset), identify the dependent variables, and identify the tasks to be performed.

**Stage 2:** Design the study - specify the independent variables, specify the dataset(s) and tasks to be used, specify the dependent variables, and specify the experimental design of the study (i.e. within-subject or between-subject).

**Stage 3:** Run pilot studies with participants - have participants perform the study to identify problems with the study design.

**Stage 4:** Run actual study with participants - have participants perform tasks with the appropriate test conditions, and measure and save dependent variables.

**Stage 5:** Filter results and analyze results - clean the data and perform statistical analysis of the results.

Problems discovered and observations made during pilot studies are addressed iteratively to improve the user study design. The actual study is run after successful run of pilot studies. Afterwards, results are analyzed with statistics to

determine the effects that the independent variables have on the dependent variables.

Aside stage1, where the evaluator makes decisions on the visualization to evaluate and the data and tasks to use for the user study, stages 2 through stage 5 can be automated. This dissertation therefore focuses on how to facilitate or automate these processes listed in figure 2.6.

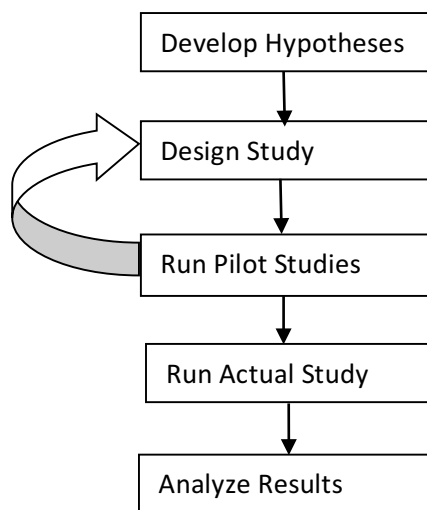


Figure 2.6: The typical processes that are involved in user studies.

### 2.1.6 Typical protocol used in running controlled user studies

A typical controlled study protocol has pre-study, actual-study, and post-study activities. *Pre-study* activities include providing an introduction of the experiment to participants, having participants sign consent forms, and gathering demographic data (e.g. age, gender, and experience). *Pre-study* activities also include demonstrating the tasks with examples, and allowing participants to perform practice trials of the tasks they will be performing in the study. Additionally, *pre-study* activities sometimes include providing participants with standardized tests

that measure their characteristics, such as color-blindness test, perceptual speed test, and visual working memory test. *Actual-Study* activities include having participants perform the tasks with the appropriate test conditions, measuring and saving dependent variables related for the tasks. *Post-study* activities include gathering data on the experience and opinions of participants using a questionnaire. For example, users can be asked to rate their experience on a Likert scale and provide their comments and preferences among competing test conditions.

### 2.1.7 Within and between subjects study design

Studies can either be designed as within-subjects or between-subjects. The Table 2.1b below illustrates how test conditions are assigned to participants. In within-subjects (also known as *repeated measure*), each participant performs the study tasks with all the test conditions, one after the other as illustrated in Table 2.1a. In between-subjects studies, each participant performs the tasks with only one of the test conditions being evaluated (Table 2.1b).

Table 2.1: A table showing a within-subject design and a between-subjects design.

Participant	Test Condition	
1	A	B
2	B	A

a) Within-subjects design

Participant	Test Condition
1	A
2	B

(b) Between study design

Using a within-subjects or a between-subjects study design comes with advantages and disadvantages [26]. A key advantage of within-subjects studies is that it requires fewer study participants because subjects are tested on all conditions. As a result, behavioral differences of subjects have less impact on the variance of the study data because subjects are likely to exhibit their performance behavior across all conditions. On the other hand, within-subjects studies require longer study duration, and results can be compromised by learning effects or fatigue.

### **2.1.8 Learning effects**

Learning effects occur due to the order in which conditions are presented, for example if participants performed tasks with condition A before performing the same tasks with condition B, they may perform better with condition B due to experience gained from condition A. However, there are protocols to minimize learning effects such as *counterbalancing* where subjects are first placed in groups and the order of conditions are presented differently to each group using a *Latin Square*. For example, if there are two conditions in the study (condition A and condition B), subjects can be equally assigned to two groups (Group 1 and Group 2), Group 1 members will perform tasks with condition A before condition B, and Group 2 members will perform the tasks with condition B before condition A. An example of a Latin square ordering for 2, 3, and 4 test conditions is illustrated in Figure 2.7. A more robust Latin square has also been recommended by other

researchers which include having a balanced table in which conditions follow each other equally [26]

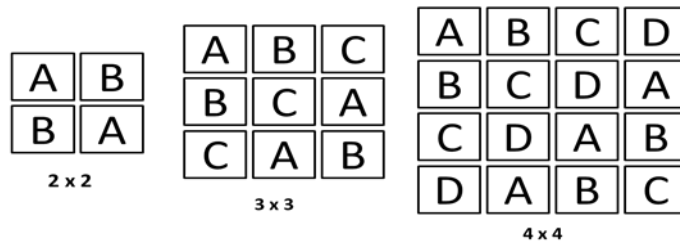


Figure 2.7: Examples of Latin squares for 2, 3, and 4 conditions.

### 2.1.9 Controlling fairness and other factors that affect study validity

The validity of user studies can be affected by several factors, these include: providing unfair experiences across participants, having inconsistent instructions and tasks across participants, providing unequal training to participants, overstressing participants, and learning effects [5, 17]. If these factors are not controlled properly, they can lead evaluators to make erroneous conclusions about studies. Example includes, finding a relationship between the independent and dependent variables when there is none (type I error) and not finding a relationship when there is one (type II error) [17]. These types of errors indicate that performing valid user studies require skill, carefulness and rigorous work.

## 2.2 Tasks adapted in user studies

One important component of user studies is the tasks users perform. Tasks commonly used in user studies range from low-level domain-independent taxonomy tasks such as the taxonomies of Amar et al. [39], Lewis et al. [40], and

Zhou et al. [41], to compound and domain-specific tasks that build on these low-level tasks such as the graph taxonomy [18], and the network evolution taxonomy [21]. Table 2.2 shows the low-level tasks commonly used in visualization user studies. The focus of this work is on how to present these tasks to users and how to accurately receive and validate responses of these tasks.

Table 2.2: Low-level tasks that are commonly used in the visualization user studies.

<b>Low-Level Tasks</b>	<b>Description</b>
Retrieve value [39]	Find attributes of a set of data cases.
Filter [39]	Find data cases whose attribute values satisfy a set of conditions.
Compute Derived Value [39]	Compute an aggregate numeric value representation of a set of data cases.
Find extremum [39]	Find data cases that have an extreme value of an attribute
Sort [39]	Rank a set of data cases according to some ordinal metric.
Determine Range [39]	Find the span of values of an attribute within a set of data cases.
Characterize distribution [39]	Characterize the distribution of a quantitative attribute's values over a set of data cases.
Find anomalies [39]	Identify any anomalies within a set of data cases with respect to a given relationship or expectation.
Cluster [39][40][41]	Find clusters of similar attribute values within a set of data cases.
Correlate [39][40][41]	Determine useful relationships between the values of two attributes of a set of a data cases.
Scan [18]	Quickly review a list of items.
Set Operation [18]	Perform set operations on sets of data cases E.g. intersection.
Locate [40][41]	Find a data case that you know about.

Identify [40][41]	Find a data case that was not necessarily known previously.
Distinguish [40] [41]	Find differences in attribute values between sets of data cases.
Categorize [40] [41]	Find divisions that a set of data cases can be sorted by.
Distribution [40]	Describe the overall pattern of a set of data cases.
Rank [40][41]	Find the order of a set of data cases based on values of an attribute.
Compare [40]	Compare a set of similar data cases based on some attributes.
Associate [40][41]	Find the relationship between sets of data cases.

### 2.2.1 Recruiting participants

In lab-based user studies, participants are usually solicited through several mediums such as email, word-of-mouth, phone calls, and wall notices [26]. Participants have to be scheduled on days and times within the week that is favorable to them. Due to this, studies that require large number of users can run into weeks, and even months. Participants are also compensated for their time. Generally, the recommended compensation is at least the minimum wage rate to be able to attract enough prospective participants [12].

Ideally, participants should be drawn at random from a broad population. However, the common practice in lab-based studies is that, participants are selected from an available group of people such as undergraduate and graduate students, and work colleagues [26, 13]. Results from such studies can compromise the external validity of a research in cases where the population used for the study is too different than the intended population [26].



Recruiting and managing diverse and large number of participants for lab-based studies is challenging and can lead to user studies lasting for weeks and months. As such, most lab-based user studies are used as a final validation of research projects. A framework that reduces the effort required in running user studies will enable user studies to be done within hours or days, which will enable designers and researchers to frequently evaluate competing visual designs.

Additionally, recruiting domain experts for experiments is a challenge, because domain experts are rarely free to participate in lab-based user studies. A framework that automates the management of user studies on the web will also provide an opportunity to get access to busy participants such as domain experts who can perform the study at their own free time.

### **2.2.2 Web-based and crowdsourced user studies**

The web provides tremendous opportunities for empirical researchers to perform experiments and evaluate ideas quickly, as such web-based experiments is increasingly popular in several research fields [12, 15, 42, 43, 44, 45]. Furthermore, advancements in web technologies such as HTML5, D3 [11], and WebGL, has also made it possible to increasingly develop interactive web-based visualizations and perform web-based user studies. A web-based user study provides the opportunity of having access to many participants from different parts of the world. In addition, it provides the opportunity to distributes studies easily through email requests, posting on forums, and sharing on social media platforms.

Crowdsourcing user studies refers to the process of recruiting a group of web-based participants to perform tasks that require human effort. Crowdsourcing takes many forms such as gamification [46], wisdom of the crowd [47], and peer-production science [48]. Crowdsourcing platforms such as Amazon Mechanical Turk provide an infrastructure for deploying experiments and recruiting diverse user populations to participate in the study. This dissertation provides a design that supports both web-based user studies and user studies performed with crowdsourced participants.

### **2.2.3 Amazon mechanical turk**

Amazon Mechanical Turk (AMT) [49] [50] is a paid crowdsourcing platform where Requesters (employers) post microtasks known as Human Intelligence Tasks that are completed by Workers (employees). Payments can be as low as \$0.01 and rarely exceeds \$1, but AMT recommends rewarding user effort based on the minimum wage. Requesters can also choose to reward good work with bonuses and choose not to pay Workers that perform very badly.

Workers are anonymous to requesters, and they can be pre-screened or filtered with "Qualifications" such as level of experience, and country of residence. The Qualification feature opens opportunities for performing research with specific user groups and the possibility of performing longitudinal user studies with selected class of users [50].

The low cost and relative ease of recruiting large number of participants makes Amazon Mechanical Turk an attractive platform for large scale experimentation.

AMT has been used to successfully run research experiments in information visualization and several areas of Computer Science such as HCI [51], Computer Vision [52], information Retrieval [53], and Natural Language Processing [54].

This research provides a design that leverages crowdsourcing platforms such as Amazon Mechanical Turk to support the recruitment of participants for web-based user studies.

### **2.3 Research trends that motivate this thesis**

This dissertation leverages four recent research trends in information visualization research. First, guidelines and protocols for fielding evaluation studies effectively are becoming increasingly standardized [2, 3, 5, 6, 7, 16, 17]. For example, Carpendale provided guidance on the different quantitative and qualitative evaluation approaches [17], Munzner presented the appropriate evaluation methodology to use for different design choices [16], and Lam et al. [7] provided an overview of current evaluation practices in visualization.

Second, online crowdsourcing platforms such as Amazon Mechanical Turk (AMT) have been shown to be valid for running evaluative visualization research [12, 13, 14, 15], and crowdsourcing has several advantages over on-site experimentation. These includes easy access to a diverse population of participants, low cost of experiments, and fast iteration between hypothesis formation and hypothesis testing [13, 55]. Heer et al. [12] replicated previous laboratory studies of spatial encoding and luminance contrasts on AMT to show that results obtained online can match results obtained in laboratory studies.

Kosara et al. [13] successfully leveraged AMT to replicate a previous lab study on how visual metaphors affect users' understanding of node-link and treemap diagrams. More recently, Jianu et al. [15] used AMT to evaluate how four different node-link visualization methods display group information, and Boukhelifa et al. [14] employed AMT to investigate how sketchiness can serve as a visual variable to encode data uncertainty in information visualization. Following the same trend in HCI, Komarov et al. [56] re-implemented three previous experiments on user interface designs both in the lab and on AMT, and did not find any significant difference in accuracy, time, and consistency between the two settings. All such studies were specific and manually set up. This research leverages AMT to semi-automate the evaluation of visualization user studies.

In addition, advances in web technologies such as Asynchronous Javascript and XML (AJAX), HTML5, WebGL, Data Driven Documents (D3) [11], and Processing [57] have caused a transition of visualization development from desktop to the web, and as such, an increasing number of visualizations are prototyped and developed directly to run in web-browsers [58, 59, 60].

Finally, standard task taxonomies and datasets have been organized in the visualization community for the evaluation of specific types of visualizations. These task taxonomies include Lee et al. for graphs [18], Saket et al. for group-level graphs [19], Valiati et al. for multidimensional visualizations [20], Ahn et al. for network evolution analysis [21], and Fekete et al. for tree visualizations [22]. Benchmark datasets include datasets used for the InfoVis contests 2003 - 2005 [23].

These four trends make it now feasible to stream-line user study evaluation by assembling user studies that conform to standard evaluation protocols and uses taxonomy tasks linked to benchmark datasets; fielding such user studies online using web-visualizations; and using crowdsourcing to automatically recruit study subjects. The proposed methodology of integrating these four research trends is novel, even though efforts to simplify the design of experiments exist.

## **2.4 Related work**

There are previous systems that have worked on simplifying how controlled user studies are designed and run in data visualization and HCI. TouchStone [61] is a platform for designing and running lab-based user studies. It facilitates the process of creating new experiments and extending existing experiments. However, it was targeted at HCI experiments that evaluate pointing and navigation interaction techniques; and it supports tasks that require answers through interaction but not other types of inputs such as text or numbers.

The Hierarchical Visualization Testing Environment (HVTE) [62] is a testing environment for running comparative studies of hierarchy browsers. It launches predefined tasks, and records users answers and completion times. However, it was built on top of a Java framework for visualizing hierarchies and it is tightly coupled to that framework.

EvalBench [63] is a software library that supports controlled lab-based visualization user studies. It provides commonly used evaluation methods and functionalities that evaluators can use to simplify their work such as generating

answer widgets, and logging user answers and interactions. However, EvalBench requires additional implementation effort from evaluators. Evaluators have to extend and implement interfaces, and modify the source code for the design of a study. Implementation efforts for the use of EvalBench can be as much as 800 lines of codes (LOC).

Experimentr [64] is a light-weight library that aims to support module-based evaluation of web-based visualization systems. It is an unpublished work-in-progress, and currently provide helper functions and sample modules that evaluators can use. Experimentr [64] however, do not support the design and running of experiments.

A More recent work is VEEVVIE [65] which supports the analyses of result data of visualization and virtual reality user studies. They provided tools such as heatmaps, parallel coordinates, and other widgets to support the exploration of results data. Their motivation for simplifying the data analysis process is close to the motivation of this research. Although VEEVVIE provides visual exploration of the data, the tool presented in this study automatically generates graphs and appropriate statistical analyses of the data which is absent from VEEVVIE.

This research work is also closely related to research efforts to provide infrastructure that facilitate experimentation in human computation experiments, social experiments, and website usability experiments. For example, Bakshy et al. [66] provided a language to simplify web-based randomized field experiments; their language provides a library to separate experimental design from application logic, however, it is not user friendly, and requires a significant amount of code

writing from evaluators. TurkServer [67] provides a framework that supports designing and running human computation and social experiments on Amazon Mechanical Turk, but it also requires a significant amount of code writing on both client and server sides. CrowdStudy [68] also provides a framework that supports designing and running website usability experiments with online study participants. This research work differs from these efforts by focusing on supporting processes involved in visualization user studies.

This dissertation was also inspired by TurkIt [69], a toolkit that leverages crowdsourcing for iterative text editing tasks, and CrowdDB [70], a system that uses crowdsourcing to answer queries that cannot be otherwise answered by traditional database systems. However, these systems do not support visualization user studies.

## **2.5 Summary**

This research work differs from the above efforts of simplifying and facilitating user studies and experiments. First, this research work provides a design and implementation of a framework that facilitate a wide range of web-based user studies, which differs from TouchStone [61] that was targeted at lab based HCI experiments, HVTE [62] that was targeted at specific hierarchical browsers, and EvalBench library [63] that provides reusable functionalities to support lab-based visualization studies. Second, this research work provide a high degree of simplicity and automation derived from the use of standardized protocols which differs from EvalBench [63] that requires significant implementation effort from

evaluators. Third, this research work provides a design that enables evaluators to design user studies with a user-friendly interface, automatically manage the run of web-based user studies, and automatically analyze results of user studies with minimal effort.



### 3 PROBLEM DEFINITION AND METHODOLOGY

This chapter presents the research problem, describes the research goals in detail, presents the research methodology, and describes the research questions that were addressed.

#### 3.1 Research Problem

The problem investigated in this research was *how to design an online framework that can reduce the overhead involved in conducting controlled user studies involving web-based visualizations*. Currently such a framework does not exist and there is no clear design guideline in the literature on how to design and build such a framework that is user-friendly and flexible to support many types of studies and tasks.

#### 3.2 The need for a framework design that facilitates a wide range of controlled user studies.

User studies are commonly used in information visualization to validate research contributions and validate effectiveness of design decisions. For example, imagine the following scenarios:

*Scenario 1:* John has developed a new visual encoding to represent multidimensional data. John will have to perform a user study to compare his new visual encodings with the state of the art visual encodings such as parallel coordinate plots and start plots in order to validate the effectiveness of his design.

*Scenario 2:* Kate have designed a novel interaction technique for performing overview+detail. Kate will have to compare her new interaction technique with the state of the art such as "pan and zoom" and fisheye lens to validate the effectiveness of her design or to identify the unique functionalities of her design.

*Scenario 3:* Mark is developing a visualization system for a given data, domain, or task; Mark want to choose the right visualization or the right visualization properties that will be more effective or useful for his specific domain and tasks. First Mark can use guidelines in the visualization literature and choose the right visualization and/or visualization properties that apply to his condition. In situations where there are no specific guidelines, Mark can perform user studies to choose the right visualization and the right visual properties for his specific condition. After the visualization is done, Mark can also conduct a user study to see how best his solution supports the intended tasks.

As can be seen in the above scenarios, user studies are central to information visualization researchers as well as visualization designers. Apart from its importance in evaluating finished designs, user studies are also essential to support decisions that visualization designers make during the developmental stages of a visualization design. During a visualization design, designers make a lot of decisions on variables such as size, value, area, surface, volume, texture, color, orientation, and shape [71] [72]. Each of these variables have different expressive power depending on the data, domain, or intended tasks. As such using user studies to choose the right variables enables evaluators to make more informed decisions.

### **3.2.1 The Existing Problem:**

User studies generally requires careful experimental designs, iterative refinement, recruitment of study participants, careful management of participants during the run of the studies, accurately collecting user responses, and expertise in statistical analysis of study results. There are several variables that are taken into consideration which can impact user study outcome if not carefully managed. Hence the process of designing user studies, running pilot studies, and successfully running the actual studies can take several weeks to months.

Due to the huge amount of time and expertise required for user studies, it is not surprising that user studies have been used predominantly to evaluate finished designs. But finished designs can be inherently complex, and as noted by Walenstein [73] and Tory et al. [3] they can also be problematic and error prone. For example, uncontrolled features in a given visualization system can dominate results and lead to unexpected study outcomes. As such, it will be beneficial to have user studies focus on comparing the effectiveness of design ideas and on hypothesis that are easily testable. However, given the amount of time required to run user studies, running many different studies to evaluate different design ideas and hypothesis can be time consuming and challenging.

Based on these limitations, having a design that facilitates user studies will save evaluators considerable time on designing and running user studies. Such a framework will also enable evaluators to run many different user studies to test

different design ideas and hypothesis involved in designing visual encodings, interactive techniques, or domain based visualization design.

### **3.3 Research goals**

The main goal of this research project was to lower the overhead of evaluating data visualizations quantitatively through user studies. To this end, the study implemented the research goal with the following objectives: (1) Design an online framework (VisUnit) that semi automates the processes involved in the evaluation of visualization user studies, which is flexible to support many tasks and visualization types. (2) Evaluate the effectiveness (ability to support a wide range of user studies and efficiency (ability to save evaluators time) of such an online framework in supporting visualization user studies.

### **3.4 Research Questions**

To achieve the research objectives, three different levels of research questions categorized as *high-level*, *middle-level*, and *low-level* (Figure 3.1) were addressed.

#### **3.4.1 High-Level Questions**

On the high-level, the following two research questions were addressed. (1) "How can we design a framework that is user-friendly and flexible to semi-automate web-based user studies of many visualization types and task types?". Answering this research question is important to solve the main research problem. (2) "Can this framework be effective and efficient to facilitate a wide range of user

studies?". Answering this question is important to evaluate the effectiveness and efficiency of the solution to the research problem.

### **3.4.2 Middle-Level Questions**

On the middle-level, smaller research questions were explored to help achieve the high-level research questions. These research questions include: "How can we support the design of web-based user studies?", "How can we support the automatic running of user studies?", "How can we support the analysis of user study results?", "How can the framework be made extensible with task types, task instances, and datasets?", "How can this framework design be made effective?", and "How can this framework design be made efficient?"

### **3.4.3 Lower-level questions**

On the low-level, low level design and implementation questions were explored to help address the research questions at the middle-level. These questions include how to design the architecture of VisUnit, how to design interfaces, and how to design and implement research solutions in general.

### High Level Questions

How can we design a framework that is user-friendly and flexible to semi-automate web-based user studies of many visualization types and task types?

Can this framework be effective and efficient to facilitate a wide range of user studies?

---

### Middle -Level Questions

How can we support the design of user studies?

How can this framework be made effective?

How can we support the automatic running of user studies?

How can this framework be made efficient?

How can we support the analysis of user study results?

How can this framework be made extensible with new task types, task instances, and datasets?

---

### Low-Level Questions

What are the Low-level design and implementation issues that need to be solved to answer the intermediate and high level questions?

Figure 3.1: An overview of research questions that were approached and answered in this thesis.

### 3.5 Research methodology

An overview of the methodology of this research work is shown in Figure 3.2.

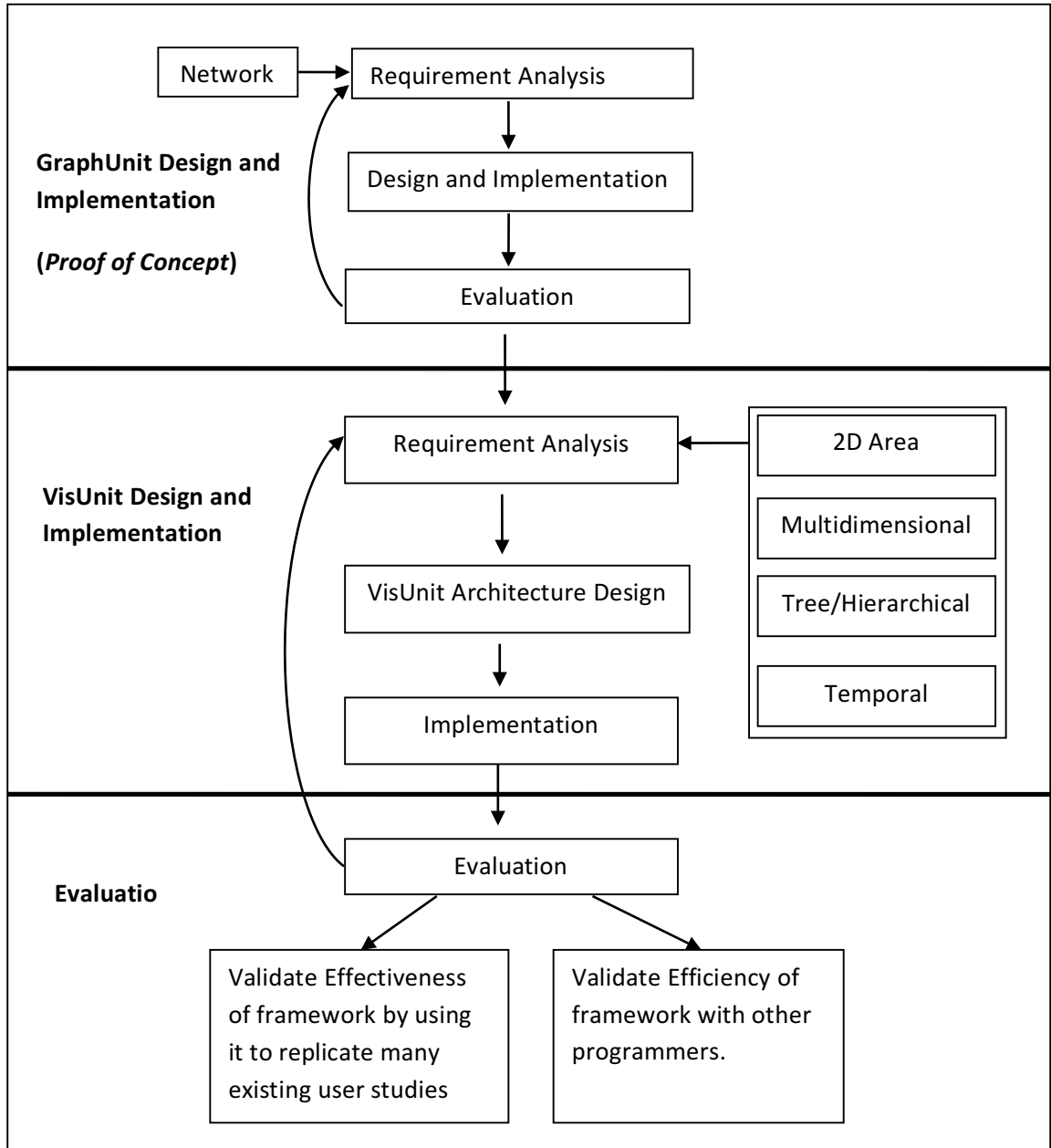


Figure 3.2: Overview of research methodology: First, an investigation was done to determine how to design a framework to support user studies of a specific visualization type (i.e. graph network). Second, requirements for different visualization types were analyzed. Third, VisUnit was designed, implemented and evaluated.

### **3.5.1 Designing a flexible user friendly online framework (VisUnit) that semi-automates the processes of visualization user studies and supports many types of user studies.**

This research problem was approached in two phases. The first phase of this research investigated a smaller problem, *"how to design a framework that semi-automates the processes involved in conducting user studies of a specific visualization type"*. Starting with a smaller problem allowed us to gather design requirements, and to test design decisions and hypothesis on a smaller scale. To this end, an investigation was done to determine how to design a framework that facilitates user studies involving *graph networks*. Graph networks were chosen as the starting point because graphs are commonly used in visualization and several other domains, and graph tasks are well understood. To solve this research problem, the visualization literature was explored for requirements involving graph user studies. The requirements gathered include processes involved in conducting user studies and fundamental properties of graph user studies. Based on these requirements, an open-source framework (GraphUnit) was designed, implemented and evaluated. GraphUnit facilitates graph user studies by semi-automating the design, run, and result analyses of graph user studies. The design and implementation of GraphUnit is described in *Chapter 4*.

The second phases of this research investigated *"how to design a framework that can facilitate the processes involved in conducting a wide range of user studies that involve many visualization types"*. In addition to the requirements gathered from the design of GraphUnit, we surveyed the visualization literature to



understand common practices in conducting user studies in the visualization domain, and to define the range of user study designs that VisUnit has to support. Based on these requirements, an open-source framework (VisUnit) was designed and implemented. VisUnit is flexible, user-friendly, and facilitates a wide range of user studies involving many different visualization types. Ultimately, VisUnit was designed to allow evaluators to design user studies with benchmark tasks and datasets as well as their own tasks and datasets. VisUnit allows evaluators to design user studies using a user-friendly interface; automatically manage the run of user studies; automatically generate statistical analyses for finished studies, and provide functionalities to support the cleaning of results data. The design and implementation of VisUnit is described in *Chapter 5*.

### **3.5.2 Evaluating the effectiveness and efficiency of VisUnit's design**

During the course of designing GraphUnit and VisUnit, several evaluations were performed to test the effectiveness and efficiency of design choices. Such evaluations served as a guide to fine-tune VisUnit's design and improve its support for a wide range of user study types.

The effectiveness and efficiency of GraphUnit and VisUnit were also formally evaluated after their design and implementation. To demonstrate the effectiveness of the design, we describe how the methodology of 84% of 101 controlled user studies published in IEEE Information Visualization conferences can be replicated on VisUnit. These user studies involve graphs, multidimensional visualizations, trees, 2D areas, and temporal visualizations.

The efficiency of the design was also demonstrated by showing that evaluators can use VisUnit to design user studies in less than an hour; and run studies and analyze study results within a day. Specifically, a user study was performed involving 5 graduate computer science students who are familiar with visualizations. Students were asked to design user studies to evaluate two different representations of tree data using freely available web visualizations. On average, it took participants one hour to design a study, and place those studies on Amazon Mechanical Turk. The evaluation of GraphUnit is described in *Chapter 4*, and the evaluation of VisUnit is described in *Chapter 6*.

### **3.6 Summary**

The problem investigated in this research was how to design an online framework that can reduce the overhead involved in conducting controlled user studies involving web-based visualizations. Currently there is no guideline on how to design such a framework that is user friendly and flexible. The goal of this research is to lower the overhead involved in performing quantitative user studies. The goal was achieved with two sub-goals: designing a flexible online framework that semi-automates the processes involved in running user studies; and evaluating the effectiveness (ability to support a wide range of user studies) and efficiency (ability to save evaluators time) of the framework. The first phase of the research investigated a smaller problem, "how to design a framework that can facilitate graph user studies?". The solution to this problem is discussed in Chapter 4. The second phase of the research investigated "how to design a framework

that can facilitate the processes involved in conducting a wide range of user studies that involve many visualization types". The solution to this problem is presented in Chapter 5. Finally, the evaluation of the research is presented in Chapter 6.

## **4 GRAPHUNIT: A FRAMEWORK TO SUPPORT THE EVALUATION OF GRAPH USER STUDIES**

This chapter presents the first phase of this dissertation. In this phase, we show the designed, implemented, and evaluated GraphUnit, a framework that semi-automates the process of designing, running, and analyzing results of graph user studies. GraphUnit was designed after exploring the visualization literature for requirements based on the fundamental properties of graph user studies and common processes involved in graph user studies. GraphUnit supports graph user studies by offering a user-friendly interface for designing user studies, and leverages crowdsourcing and a set of evaluation modules based on a graph task taxonomy. Graphs play an important part in several domains such as neuroscience [75], social sciences [76], software engineering [78], and genomics and proteomics [77]. Graph visualization research provides novel and effective ways to understand networks through effective visual encodings and interactions and user studies are commonly used to evaluate graph research outcomes.

Here we introduce a novel framework that allows visualization designers to quickly configure user studies for web-based graph visualizations, uses crowdsourcing to conduct the user study, and automatically returns the appropriate statistical analyses of the study's results.

The design of GraphUnit focused on five issues: defining tasks and datasets, connecting a visualization to our evaluation service, configuring user studies,

running user studies, and analyzing user study results. These five key issues are detailed in the following sections.

## 4.1 Architecture

The architecture of GraphUnit is shown in Figure 4.1. GraphUnit conceptually consists of three main modules (i.e. Study Setup, Study Manager, and Result Analyzer), and a library of graph related datasets and tasks. The Study Setup module handles user study configurations and it consists of a setup interface and a setup manager. The Study Manager module handles the execution of user studies and it consists of a study manager and a study interface. The Result Analyzer module handles the analysis of user study results and it consists of a results manager and a results interface. The architecture is shown in Figure 4.1.

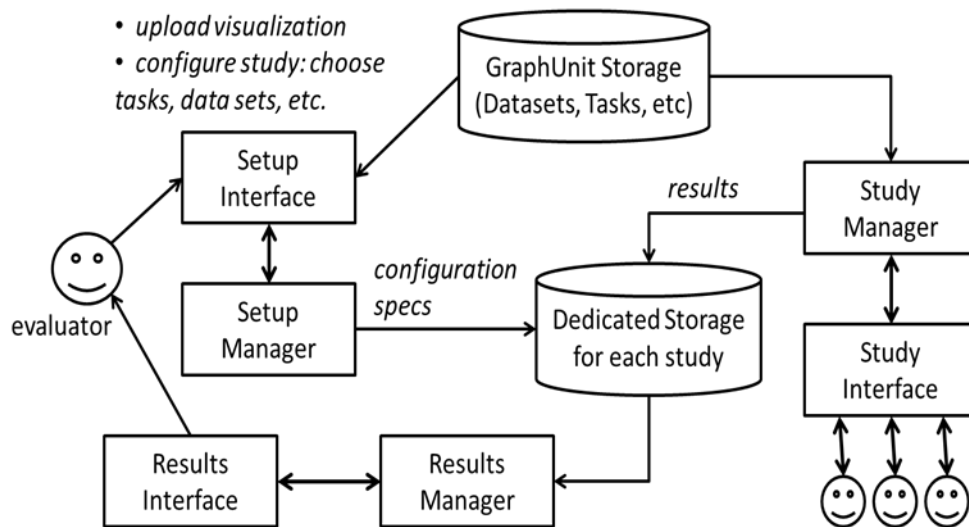


Figure 4.1: Architecture of GraphUnit.

The interface is used by evaluators to upload their visualizations on GraphUnit's server and to configure user studies. Configuring a user study involves specifying which uploaded visualizations should be used as conditions, selecting datasets and tasks from GraphUnit's default libraries, and configuring the study

protocol: type of study (i.e. within or between); number of users. The setup manager uses this information to create a configuration specification file. The setup manager then creates a dedicated directory for the user study, and loads to that directory the configuration specification file and other files uploaded by the evaluator.

The Study Manager is activated once an online user accesses the deployed user study. The manager loads the study's specification file, and creates the necessary infrastructure for conducting the experiment. The manager then oversees the actual user study by assigning participants to conditions, presenting tasks to participants through the study interface, and saving results to text files in the study's dedicated directory.

The Result Analyzer loads these results, summarizes and graphs them using D3 [11], generates statistical analyses that are appropriate for the study design using R [134], and presents these results to the evaluator.

GraphUnit stores its own library of datasets and tasks in raw text and XML format in a dedicated directory structure. Specifically, GraphUnit stores interconnected data definitions and task definitions. A data definition includes both the actual data, and task instances defined on that data for each type of task that GraphUnit supports. Task instances are instantiations of a general type of task (e.g., "are two nodes connected?") on a particular dataset (e.g., "are nodes A and B connected?"). As such, for each dataset we define an XML file that contains a list of specific data elements required to create instances of that task (e.g., specific

pairs of nodes for a neighbor task). An example of such an XML file is shown in Figure 4.2.

As shown in Figure 4.3, GraphUnit supports quantitative tasks adapted from the graph task taxonomy of Lee et al. [18]. It also contains several graph datasets of varying sizes and complexities which were derived from two larger networks — a book recommendations network (which was also used by Jianu et al. [15]), and a co-starring network derived from the internet movie database (IMDB).

```
<?xml version="1.0"?>
<taskFile>
  <answertype>options</answertype>
  <option>yes</option>
  <option>no</option>
  <question>
    <node>Daniel radcliffe</node>
    <node>Ralph Fiennes</node>
    <answer>Yes</answer>
  </question>
  <question>
    <node>Billy Dee Williams</node>
    <node>Mark Hamill</node>
    <answer>Yes</answer>
  </question>
  <question>
    <node>Peter Falk</node>
    <node>Dennis Hopper</node>
    <answer>No</answer>
  </question>
</taskFile>
```

Figure 4.2: An example of a task file.

#### 4.1.1 Extending GraphUnit with new datasets and tasks

The online version of GraphUnit allows studies to be configured using only data and tasks that are stored on GraphUnit's server.

However, evaluators can install their own version of GraphUnit and gain control over what these datasets and tasks are. To extend GraphUnit with a new dataset, the actual data need to be added first in JSON format or as lists of edges. Then, a new task-instance file (XML) needs to be created for that data for every task that GraphUnit supports, or at least for tasks that the dataset will be used for. Thus, the complete definition of a GraphUnit dataset will consist of both the actual graph data, and a series of XML files, each listing instances of one particular task type defined on that dataset (e.g. a list of node pairs for the graph connectivity tasks). Similarly, to extend GraphUnit with a new task type, this task needs to first be defined in an XML file by specifying the generic question that subjects will be asked, and the type of answer they will be able to provide. Then, new XML files need to be created for each existing dataset in GraphUnit, or at least for those datasets that will be used, to specify task-instances for the newly created task type on those datasets.

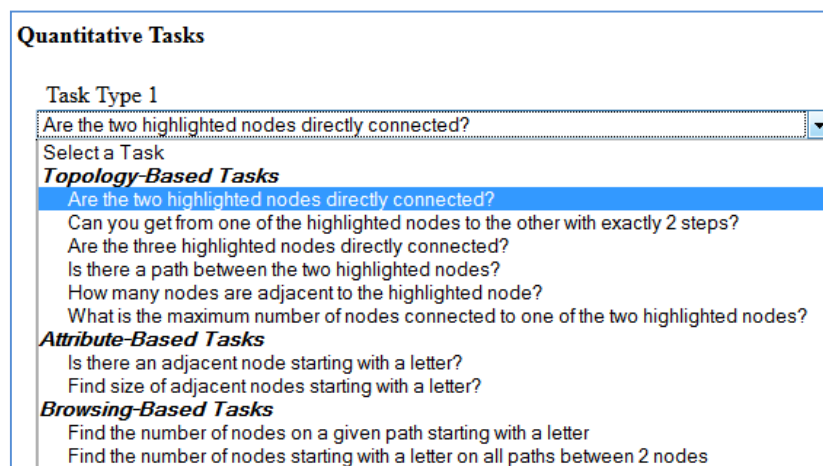


Figure 4.3: Options of quantitative tasks that can be used for the evaluation.



## **4.2 Configuration of user studies**

This section describes how evaluators can configure user studies.

### **4.2.1 Connecting a visualization**

Evaluators are required to augment their web visualization by implementing an interface of JavaScript methods that allow GraphUnit to control their visualization. Specifically, they were asked to provide methods for loading a dataset into their visualization (`setDataset`), and highlighting nodes in the visualization (`selectNodes`). A few optional interface methods allow developers to customize tasks and messages that are shown to the subjects during the study, and will be described later. Once the visualization implements these interfaces, developers can upload them, together with supporting files, to GraphUnit. At that point, they become accessible by GraphUnit, and can be linked to tasks and datasets that GraphUnit provides.

To evaluate visualizations that cannot be uploaded to GraphUnit's host server, for instance because they require significant additional resources such as a database, the evaluator needs to install their own copy of GraphUnit. This is relatively simple as GraphUnit is a small Java servlet application that requires no special libraries or database dependencies.

### **4.2.2 Configuring**

To configure user studies, evaluators use the simple web form shown in Figure 4.4. This form allows them to upload one or several visualizations and their supporting files, specifying which uploaded visualizations should be used as

conditions in the study, selecting one of GraphUnit's datasets to be used in the evaluation, and selecting tasks that will be evaluated.

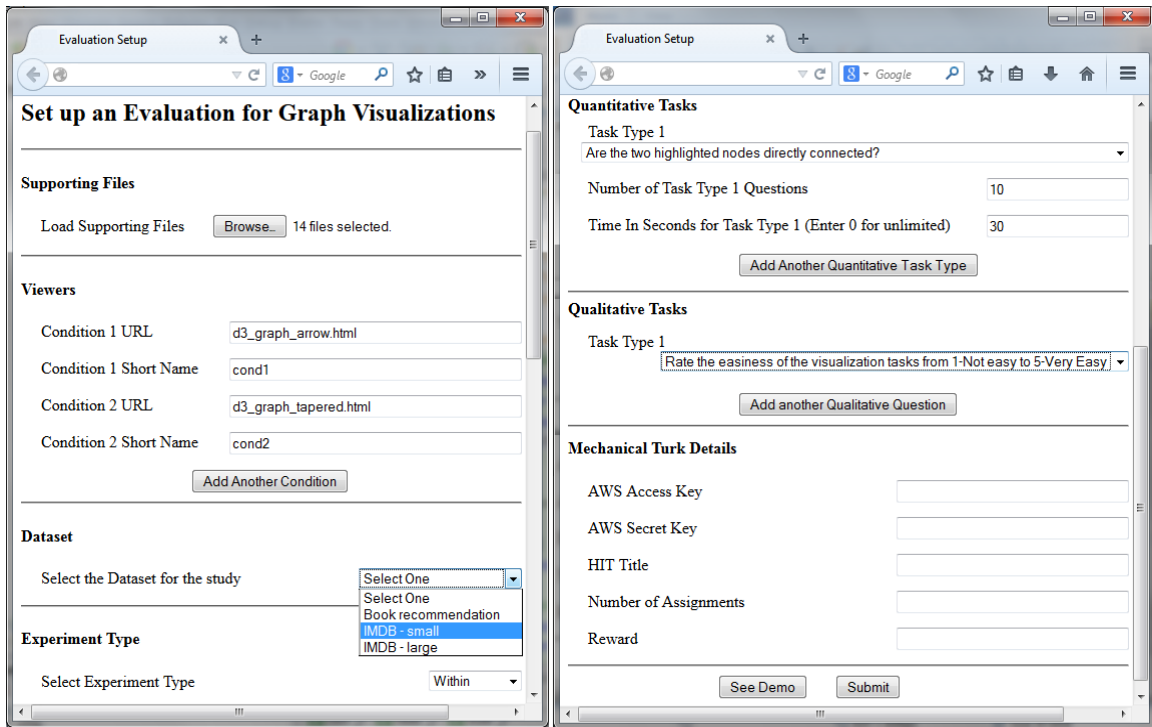


Figure 4.4: An interface for configuring a user study.

In accordance to Lee et al.'s taxonomy [18], quantitative tasks include: topology tasks (e.g. Are two highlighted nodes directly connected?), attribute tasks (e.g. Is there an adjacent node starting with a given letter?), and browsing tasks (e.g. Find the number of nodes on a given path that starts with a given letter). Figure 4.3 shows available options for quantitative tasks. Additionally, for each type of task evaluators select, they need to specify the number of instances and maximum allowed time for that task. For example, a study can be configured to contain 20 instances of the task "Are two highlighted nodes directly connected?" and allow subjects 10 seconds to complete each task instance. Optionally, studies

may also include qualitative questions such as "Rate the easiness of the visualization tasks from 1-Not easy to 5-Very Easy" or "What problem did you have with the visualization?".

Once the configuration is complete, GraphUnit generates a study specification XML file (Figure 4.5), and uploads it to the dedicated study repository. The study manager will use that specification to create instances of the study. One such instance is shown to the evaluator as a preview demo, at which point the evaluator can deploy the study or edit it. Deployment can be done either through Mechanical Turk or by sending the study URL to a dedicated group of online users.

```
<?xml version="1.0"?>
<study_specification>
  <dataset>imdb_small</dataset>
  <studyname>study8</studyname>
  <experimenttype>Within</experimenttype>
  <condition>
    <conditionurl>d3_graph_arrow.html</conditionurl>
    <conditionshortname>cond1</conditionshortname>
  </condition>
  <condition>
    <conditionurl>d3_graph_tapered.html</conditionurl>
    <conditionshortname>cond2</conditionshortname>
  </condition>
  <task>
    <name>neighbor_one_step</name>
    <question>Are the two highlighted nodes directly connected?</question>
    <size>3</size>
    <time>20</time>
  </task>
</study_specification>
```

Figure 4.5: An example of a study specification file.

### 4.2.3 Putting studies on Mechanical Turk (AMT)

GraphUnit has a default binding to the Amazon Mechanical Turk platform, and it can configure and place tasks (HITs) on this platform automatically for evaluators who own AMT developer accounts, without requiring them to interact with the platform separately. GraphUnit will request evaluators to provide their AMT login credentials, a HIT title, the number of assignments, and the reward for the HIT. Using this information, GraphUnit will dynamically configure an appropriate AMT HIT. Specifically, GraphUnit instructs AMT to create a HIT with a short description of the study and an external link to the study hosted by GraphUnit. Evaluators without developer accounts will still be able to use our system but will have to configure AMT hits manually using the study link provided by GraphUnit.

## 4.3 Running the user study

**Assigning subjects to conditions:** For a between-group study we ensure the number of participants per visualization condition is uniform. Each new participant is presented with a condition with the least count of study completions.

**Ordering of conditions:** For a within user study, we use a Latin square to organize the study conditions in such a way that all possible orderings of the visualization conditions are performed by a uniform number of participants, and that learning effects are minimized.

**Protocol:** The studies follow three stages: introduction, training, and study. The default introduction page provides a short graph primer. During the training stage samples of each evaluated task are shown and study participants are

allowed to check the correctness of their answers. Subjects are then walked through the actual study.

As exemplified in Figure 4.6, the user study interface is partitioned into two sides. A large panel on the left hosts the visualization being evaluated. A smaller panel on the right shows the text for each question, a timer which informs the subject of the time allotted for a task, and allows subjects to provide answers and to navigate through the study. For each question, a blank white screen hides the visualization when the time allotted to complete that task expires. For studies run on AMT, we provide study participants with a mechanical turk code once they complete the study.

#### **4.4 Optional methods**

A few optional interface methods can be implemented by evaluators to customize how the study is presented to online users, and ensure that subjects can properly understand each visualization and tasks associated with it. Custom introduction: Instead of our default graph primer, evaluators can use an introduction page that is tailored to the evaluated visualization. To do that, they need to override the *getIntroduction* function to return a customized introductory HTML file. At the beginning of a user study, GraphUnit will check the existence of this method and, if it exists, will use the HTML it returns to replace the default introduction.

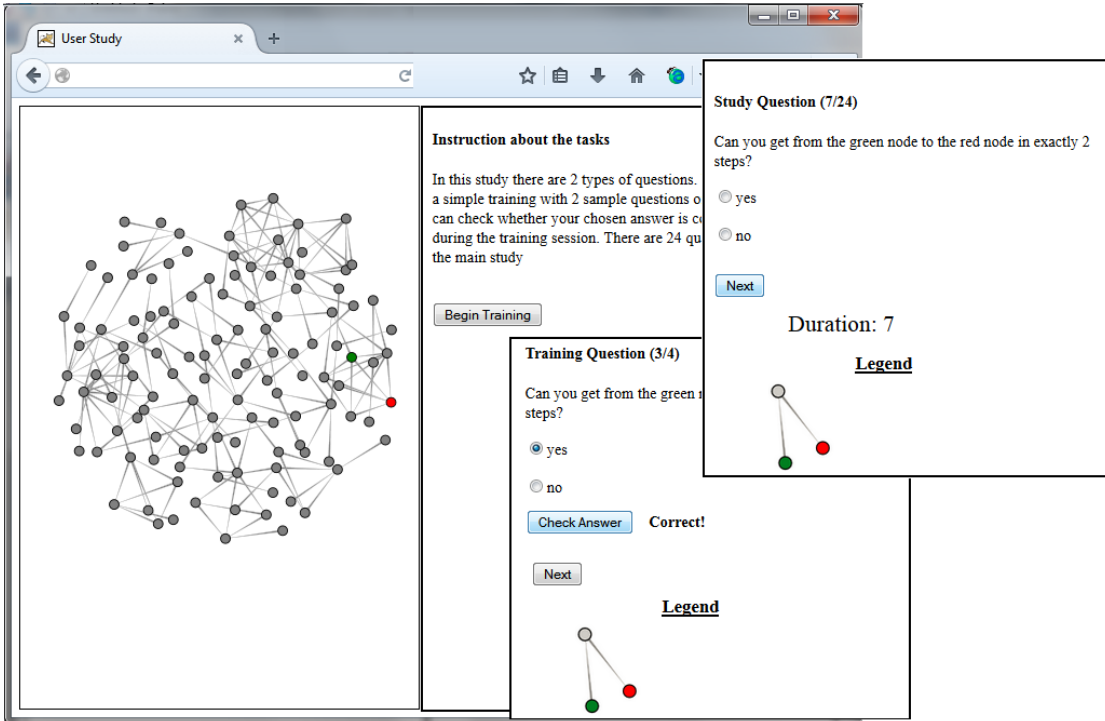


Figure 4.6: An example of a user study, showing three stages: instruction about task, training, and study.

**Task translations:** Evaluators can customize how a task is phrased to users, by configuring their visualization to "translate" GraphUnit's graph taxonomy tasks. This allows each visualization to use a nomenclature that matches its appearance and that subjects can relate to. For instance, a node link visualization can "translate" the neighbor question into: "Are two highlighted nodes directly connected?", while a matrix representation may ask the same question as: "Is there a black colored box at the intersection of the highlighted row and column?". Evaluators can provide task translations by implementing the *changeQuestion* function.

## 4.5 Analyzing study results

**Statistical analysis:** GraphUnit uses R [134] to provide statistical analyses of the data it collects from online users. GraphUnit results are summarized for accuracy and time. Each analysis starts with Shapiro-Wilk normality tests for accuracy and time distributions for each evaluated task across all conditions. The time or accuracy distribution for a given task is classified as normal only if results are deemed normal for that task across all conditions. Depending on the number of conditions and the study type (between-group or within-group), we perform the appropriate statistical analyses as follows.

For a between-group study with exactly two conditions, we perform either an independent t-test, if our results are sampled from a normal distribution, or a Wilcoxon rank-sum test in the case of a non-normal distribution. For a between-group study with more than two conditions, we perform an independent Anova if the result conforms to a normal distribution, and a Kruskal-Wallis for non-normal distributions.

For within-group studies with two conditions, we perform either a paired t-test for normal distributions, or a Wilcoxon signed-rank test for non-normal distributions. Finally, for within-group studies with more than two conditions, we perform a repeated measure Anova for normal distributions, or a Friedman test for non-normal distributions.

If a result is found to be significant across more than two conditions, GraphUnit follows up with a post hoc analysis. For between-group studies, we perform a TukeyHSD for normal distributions, while for non-normal distributions,

we use a Wilcoxon rank-sum test to compare pairs of the conditions followed by an adjustment of the resulting p-values with a Bonferroni correction. For within-group studies with normally distributed results, we perform paired t-test comparisons on all condition pairs and adjust the resulting p-values with a Bonferroni correction. Finally, for within-group studies with a non-normal distribution of results, we use a Wilcoxon signed-rank test to compare pairs of the conditions and adjust resulting p-values using Bonferroni correction.

**Raw data:** We also provide two types of raw data for time and accuracy in CSV format: a summarized raw result where averages of a user's performance on each task is recorded, and a basic raw data where performance on individual questions of tasks are recorded. Evaluators can download this data to run additional analyses.

## **4.6 Evaluation**

We demonstrate GraphUnit's effectiveness by showing how it can be used to replicate published graph evaluation studies with minimal effort. Moreover, we show how two visualization researchers could configure user studies of their own graphs quickly.

### **4.6.1 Study I - Evaluating node link diagrams vs. matrix diagrams**

We configured a user study similar to the study published by Ghoniem et al. [28], comparing node-link diagrams to matrix visualizations. For this study, we



used a freely available matrix visualization of a network, and a freely available undirected graph visualization. We configured these visualizations for the user study as follows.

First, we introduced the following functions. (1) *set-Dataset* - we ensured that both visualizations were able to load GraphUnit's data and display the visualization when this function was called. (2) *selectNode* - the visualizations received an array of node names through this method and were responsible for highlighting them in the visualization. The node-link visualization implemented the *selectNode* method by coloring the nodes red, while the matrix visualization highlighted entire rows. (3) *changeQuestion*, which translated a question based on the visualization type. Since we intended to evaluate the "How many nodes are connected to the highlighted node?" taxonomy task, the *changeQuestion* method left the question unchanged in the node-link visualization but translated it into "How many black boxes are on the row highlighted red?" in the matrix representation.

After implementing these functions in both visualizations, we configured the user study on the Study Setup page by loading the visualizations, selecting a dataset from the available options, choosing a between-group design, and selecting to evaluate 20 instances of one task ("How many nodes are connected to the highlighted node?") and allowing 20 seconds for each instance. It took us approximately 30 minutes to complete the configuration including time used in augmenting the visualizations with the necessary functions. The StudySetup module deployed this study and automatically placed it on AMT. We ran this study with 112 AMT users and we reimbursed each user \$0.5 for their time.

The Study Manager module instantiated the tasks for each user that accessed the study, showed users either the node-link or matrix graph, presented a custom introduction page with information on how to perform the task with the node-link or matrix visualization, provided a training session using 2 questions for the task, presented the actual tasks, and saved user responses to file. The Result Analyzer was used to interpret the study's results. GraphUnit result analysis: First, a Shapiro-Wilk test showed that the data was not normally distributed (accuracy p-values were 0.13 and  $<0.001$ ; time p-values were 0.02 and 0.39). A Wilcoxon rank sum test showed significant difference between node-link graphs and matrix for both accuracy (p-value $<0.001$ ) and time (p-value=0.03). The mean accuracy for the node-link graph was 0.52 (SD = 0.14), and the mean accuracy for the matrix was 0.85 (SD = 0.26). The mean time for the node-link graph was 7 seconds (SD = 2), and the mean time for the matrix was 6.3 seconds (SD = 1.7).

This result is consistent with the result obtained by Ghoniem et al. [28] and shows that for tasks that involve estimating node degree, matrix visualizations perform significantly better in accuracy and time compared to node-link visualizations.

#### **4.6.2 Evaluating multiple ways to represent edge directionality in node link diagrams**

We replicated Holten and Wijk's study on representing edge directionality in node link diagrams [27]. We created graph visualizations that used three types of edge representations evaluated by the original study: tapered edges, arrow-head

edges, and circular edges. On the Study Setup page, we configured the study as within-group, used a small dataset with approximately 100 nodes and 175 edges, and selected two types of quantitative tasks: "Are the two highlighted nodes directly connected" and "Can you get from one of the highlighted nodes to the other in exactly two steps".

However, to replicate the study as it was initially fielded, our visualizations translated these questions into: "Can you get from the green node to the red node using only one step?", and "Can you get from the green node to the red node in exactly two steps?". We chose to evaluate four instances of each of the two tasks, and allowed a five seconds response time for the first, and ten seconds for the second.

The total number of questions was 24 (8 questions per condition). The study was configured in just 15 minutes, excluding the time required to implement the visualization. The StudySetup module deployed this study and automatically placed it on AMT. We ran this study with 62 AMT participants and rewarded each participant with \$0.55. Similarly, to the previous study, the StudyManager module instantiated the tasks, presented a custom introduction page, presented a training session involving 2 questions per task, and allowed users to perform the two tasks with one visualization at a time using a latin square ordering of conditions. GraphUnit result analysis: First, a Shapiro-Wilk test showed that the accuracy and time data for the "one-step connection" task (task1) and the accuracy data of the "two-step connection" task (task2), were not normally distributed (all p-values were < 0.01), but the time data for task2 was normally distributed (all three p-values

were  $> 0.1$ ). Second, a Friedman's test showed that the accuracy data of task1 ( $p$ -value $<0.001$ ), and the accuracy data of task2 ( $p$ -value $=0.01$ ) were statistically significant across all three conditions. Third, a post-hoc analysis for the accuracy data of task1 revealed significant difference for arrow vs. circular ( $p$ -value $<0.001$ ), and tapered vs. circular ( $p$ -value  $< 0.001$ ), while a post hoc analysis for accuracy of task2 revealed significant difference for arrow vs. circular ( $p$ -value $=0.001$ ).

Fourth, an Anova test showed that the time differences in task2 were significant across the three conditions ( $p$ -value $=0.002$ ) and a post hoc analysis revealed significant difference for arrow vs. tapered ( $p$ -value $=0.001$ ). The graphs generated for the study are shown in Figure 4.7. These results are consistent with those of Holten et al. [27] in showing that the circular edge performed significantly worst in accuracy for the two graph tasks, and there was no significant difference between the arrow edge and the tapered edge. However, the arrow edge performed better than the tapered edge in overall accuracy, and the tapered edge performed better in overall time. This contrasts with Holten et al.'s results which showed that tapered edges out-perform arrows in both accuracy and time. Several reasons might have contributed to this: first, we limited users to a maximum of 10 seconds for each question, while there was no clear limit to the time used by Holten et al.; second, we used an instance of a real IMDB dataset, whereas Holten et al. used randomly generated datasets; third, Holten et al. did not specify the length of the edges, the stroke-size used, the size of the arrow head or the steepness of the tapered edges, and as such the dimensions used in our study may have differed from theirs.

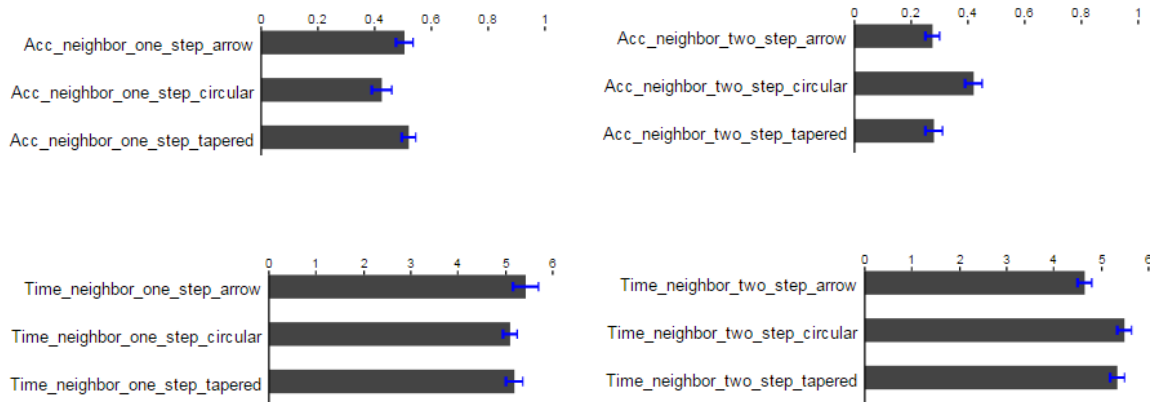


Figure 4.7: The graph generated for the results of the user studies. Error bars are standard errors.

### 4.6.3 Study III - Configuring available visualization for a user study

Finally, we tested how long it would take a visualization researcher to configure a simple user study using GraphUnit. We asked two graduate students unaffiliated with our project and familiar with data visualization concepts to configure user studies of freely available D3 node link diagrams.

First, we provided them with instructions on how to augment the visualization with required functions, and how to configure a user study. They then downloaded the visualizations from D3’s website. We asked one student to configure a study that evaluates two options of node size (5 and 10). For this, they had to create two versions of the graph visualization, each with a different node size. Similarly, we asked the second student to configure a study that evaluates two options of edge size (2 and 4). The first student required approximately 40 minutes to configure the required study, while the second user was able to read instructions, modify the code, configure the study, and view a demo in 35 minutes.

#### 4.4 Discussion

**Moving evaluation from "after" design to "in" design:** Evaluations are used predominantly after visualization development, to test or validate new designs. We hypothesize that a cheap, semi-automated, and low overhead method of performing user studies can pave the way to a more widespread use of quantitative user evaluations, in particular as a way to choose between alternative designs during the design process. In other words, quantitative evaluations could become part of the design and implementation process rather than a way of validating a finished system. Munzner [16] and Sedlmair et al. [79] advocate that designers should not rely on techniques they feel comfortable with, but rather choose techniques that serve the application domain well, and design multiple testable prototypes in short iterations. However, choosing the technique and design that is best for a particular domain and application is a difficult decision since often multiple designs are possible for the same combination of data and tasks. For example, networks may be represented both as node link diagrams and as matrices, and both representations support a wide range of tasks. Similarly, viewing group information can be done either using Bubble Sets [80] or Line Sets [81]. In such cases only a quantitative evaluation can reveal which design is optimal for a particular data and combination of tasks. Similarly, evaluating a visualization system qualitatively with domain experts, or even using an insight based methodology, can reveal only whether a design allows its users to perform the tasks they require, but cannot determine whether the tasks can also be performed efficiently.

**Promoting benchmark testing and study reproducibility:** GraphUnit can help promote study reproducibility by standardizing user study protocols. Visualization researchers can evaluate their own or another visualization and publish GraphUnit's configuration specification along with their results. Other researchers could use that specification file to run the same protocol on a newly developed visualization, and, to some degree, their results would be comparable to the previous results.

Moreover, GraphUnit can help popularize the idea of benchmarks in visualization. While benchmark tasks and datasets have been proposed [18, 23], the additional effort of creating data loaders, and setting up and fielding user studies, makes it unlikely that these resources can be widespread. Their integration into GraphUnit could help promote their transition to becoming accepted benchmarks while increasing their user base. We intend to keep the task taxonomy that GraphUnit relies upon up to date with research advances.

**Access to study participants:** Having access to numerous and diverse subjects for user studies has the potential to strengthen the support for statistical findings. We also hypothesize that since low-level data-reading tasks are mostly domain-independent, naive subjects could be used to quantitatively evaluate some aspects of domain specific visualization applications. Specifically, some domain specific workflows could be reduced to generic data reading tasks without significant loss in semantic information, and ultimately be evaluated on naive crowds to determine a visualization's ability to support basic data reading and manipulation tasks efficiently, if not necessarily its ability to produce high-level

insights. Moreover, disseminating studies online has the potential to allow evaluators to reach a sufficiently large crowd of domain experts to perform quantitative evaluations of domain specific visual applications. Such hypotheses require formal evaluation.

**Flexibility:** Our design is both structured and flexible. It is structured in that it provides a single simple form that can be used to configure all user studies, in that all studies follow a similar design protocol (training, identical interface), and in the way results are analyzed. However, our design allows experimenters to create a wide range of designs by choosing how their visualization's interface methods are configured.

For instance, experimenters can control how questions are phrased for particular visualizations (section 3.4). This raises an interesting question: does phrasing a task differently across conditions introduce an unwanted bias in subjects' results? We believe that unintentional biases can also occur when tasks are phrased identically, especially when evaluating visual encodings that are significantly different. For example, we argue that naive users will more easily translate a question such as "Are two nodes connected?" into a visual task in node-link diagrams than in matrices, since node-link diagrams are closer to naive users' mental model of a network. Experienced users of matrix visualizations however, may translate connectivity tasks into their matrix equivalent without effort. Thus, an evaluation that phrases tasks identically in these two visualizations may inadvertently capture a task translation component that is more predominant in



naive users than experienced users. As such, GraphUnit leaves this study design choice at the evaluator's discretion.

Perceptual studies often show blank screens or intermediate screens between or before actual tasks [27]. Evaluators can achieve such effects by hiding their visualization for a few milliseconds when a question is passed to it. The interface that GraphUnit relies on to communicate with evaluated visualizations can be extended to allow more such flexibility, while maintaining the structure of the main configuration options. Finally, GraphUnit can be extended with additional tasks and datasets as described in section 3.1.

**Improving quality of collected data:** GraphUnit does not currently control the quality of data provided by online users. However, we will evaluate the opportunity of extending GraphUnit with one or multiple of the following quality control capabilities. First, we will require each dataset to specify a limited number of control questions for each type of task that GraphUnit can evaluate. Such control questions will be designed to be easy enough that any well-intentioned participant can solve. Evaluators will have the option to ask GraphUnit to intersperse such control questions with actual tasks, and discard data from users who fail to answer control questions correctly. Second, we will allow evaluators to specify a percentile, and discard results that are below that percentile. Third, we will allow GraphUnit to take advantage of AMT's ability to only recruit users whose general acceptance rate is 95% or better. Finally, the Cognitive Reflection Test has been shown to make users more engaged if shown at the beginning of a user study [82] and we will consider adding it as an option in GraphUnit.

## 4.7 Summary

GraphUnit simplifies the process of designing and fielding controlled quantitative user evaluations of web-based graph visualizations. Visualization designers can field a user study by simply connecting their web-visualization to GraphUnit, selecting tasks they want to evaluate and datasets that they want those tasks on, and configuring the study protocol using a simple web form. GraphUnit will then automatically deploy the study online, use Mechanical Turk to attract participants, collect user responses and store them in a database, and analyze incoming results automatically using appropriate statistical tools and graphs. We showed that GraphUnit can be used to create and deploy previously published graph evaluation studies in a matter of minutes, and we discussed the potential of this method to guide graph visualization design by facilitating quick feedback elicitation, to evaluate and choose between competitive designs, and to evaluate graph visualizations for research purposes.

GraphUnit is currently available as open-source software at <http://vizlab.cs.fiu.edu/graphunit/>

## **5 VISUNIT - A FRAMEWORK THAT FACILITATES EVALUATION OF DATA VISUALIZATIONS**

This chapter presents the second phase of this dissertation. In this phase we generalized GraphUnit into VisUnit, a framework that semi-automates the process of designing, running, and analyzing results of a wide range of user studies evaluating many visualization types. To design VisUnit, we leveraged requirements gathered from the design and evaluation of GraphUnit, and we explored the visualization literature to gather requirements based on standard guidelines, standard processes, and fundamental properties of user studies that involve common visualization types (i.e. multidimensional, temporal, tree, and 2D area) [74].

VisUnit allows evaluators to design user studies with their own tasks and datasets, automatically manages the run of user studies, and automatically provides statistical analysis and management of results data.

### **5.1 Designing VisUnit**

The design of VisUnit focused on the following design issues: (1) How can we support design of user studies that evaluate diverse static and interactive visualizations? (2) How can we automate the running of user studies as much as possible? (3) How can we support the analysis of user study results data? (4) How can VisUnit be made extensible with new task types, task instances, and datasets?

In answering these questions, we identified the following design requirements for VisUnit after analyzing the methodologies of controlled user

studies published in the IEEE Information Visualization (InfoVis) conference since 2004 [12, 13, 15, 28, 29, 30, 38, 37, 99, 100, 87, 88, 89, 90, 91, 92, 93, 101, 102]:

- ***Simplified user study design*** - The framework should allow evaluators with limited expertise in designing human centric studies to design and edit user studies that evaluate diverse static and interactive visualizations.
- ***Support for different task types, input types, and response types*** - The framework should support the different types of tasks, task inputs and methods of accepting user responses.
- ***Automatic run of web-based user studies with support for different types of tasks*** - The framework should automatically manage web-based participants through the stages of a given user study, and accurately collect and save responses of participants.
- ***Simplified results analysis*** - The framework should enable evaluators to quickly see result summaries, raw result data, and the appropriate statistical analyses of user studies using a user-friendly interface.
- ***Simplified data cleaning*** - The framework should enable evaluators to easily perform data cleaning activities using a user friendly interface.
- ***Easily extensible with more tasks and data*** - The framework should enable evaluators to use their own tasks and datasets to design user studies that evaluate diverse static and interactive visualizations.

In the following sections, we describe how the architectural design of VisUnit helps implement these design requirements and answer these research questions.

## 5.2 VisUnit Architecture

To make sure VisUnit facilitates a wide range of user studies, we provide a framework design that decouples the visualizations being evaluated from the resources that are used for the user study (such as tasks and datasets); and also decouples both the visualization and its resources from the engine and interface that is used to run the study.

The architecture of VisUnit is shown in Figure 5.1. We focused on the following design solutions.

(1) How to provide a simplified user-friendly interface that people who have visualizations to evaluate (evaluators) can use to design user studies even if they have little expertise. VisUnit's architecture provides a user interface design (**Setup Interface**) that allows evaluators to design user studies using their own tasks and datasets as well as benchmark tasks and datasets. We discuss how requirements of study designs can be fulfilled with the setup interface in section 5.3 (*Setup interface to support the design of user studies*).

(2) How to manage previously designed user studies. VisUnit's architecture provides a user interface design (**Study Management Interface**) that allows evaluators to manage user studies that they create. Evaluators can use this user interface to edit, run, and access results of user studies that they design. Section 5.4 (Study Management Interface) provides a description of how requirements for user study management can be realized with the Study Management Interface.

(3) How to automatically manage study runs with participants. VisUnit's architecture provides a user interface (**Study Interface**) that automatically

manages study participants during the run of studies. This user interface is used by a VisUnit process (**Study Run Manager**) to walk participants through the study from the beginning of the study to the end of the study. We discuss how VisUnit uses this user interface to fulfill the requirements for running web-based user studies in section 5.5 (User study interface to automatically manage study runs).

(4) How study results can be analyzed and presented to evaluators. VisUnit's architecture provides a user interface (**Results Interface**) that is used to present results of user studies completed by participants. VisUnit uses this interface to present raw results, graph summaries, and statistical analysis of study results to evaluators. This user interface also provides functionalities that evaluators can use to filter and clean the results data. In section 5.6 (Results interface to support analysis of results) we describe how VisUnit uses the Results Interface to fulfill requirements of results analysis.

(5) How to allow the creation of new tasks, task instances and datasets. VisUnit provides interfaces to allow evaluators to create new tasks, task instances, and datasets. In section 5.7, we discuss how VisUnit allows requirements of tasks and datasets to be provided for new data.

(6) How to manage data and resources for evaluators. VisUnit's architecture includes a storage that is used to manage data and resources for evaluators (**VisUnit Storage**). This storage is used to manage user studies, visualizations, datasets, tasks, and task instances for evaluators. This storage is also used to manage benchmark datasets and benchmark tasks and task instances for VisUnit. We describe the storage of VisUnit in section 5.8 (VisUnit Storage).

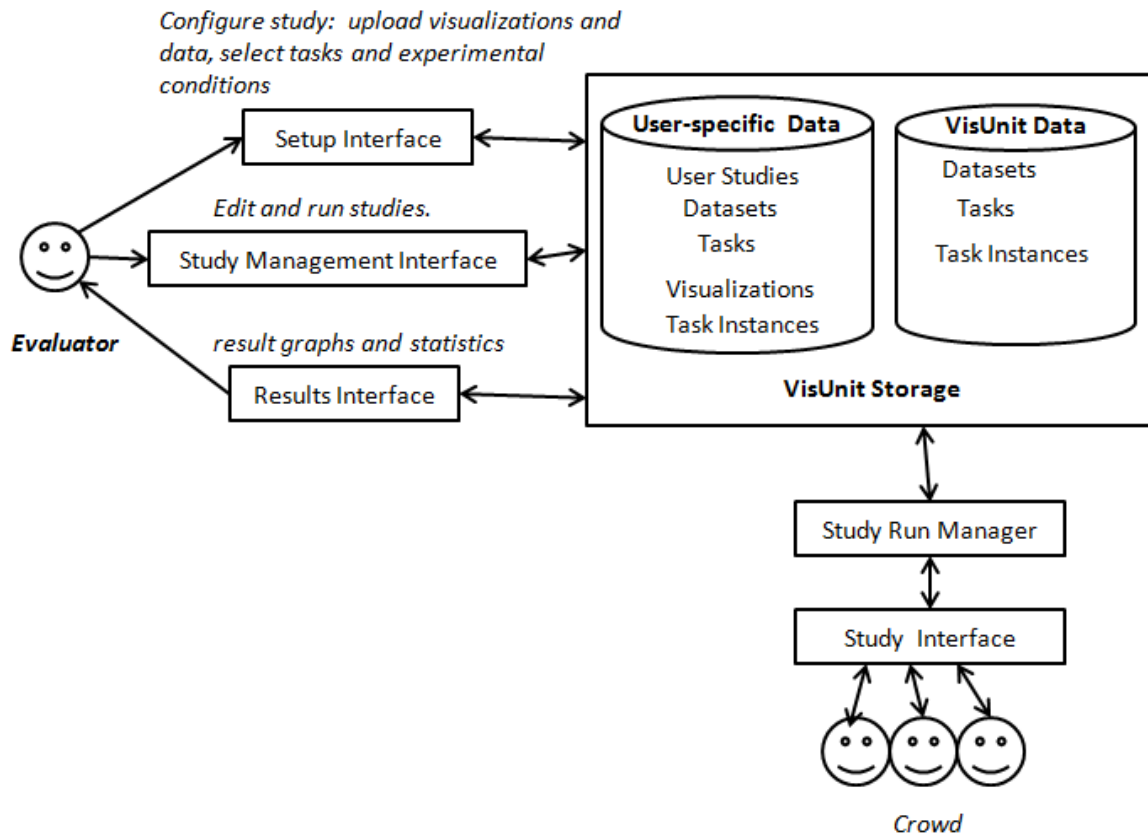


Figure 5.1: Architecture of VisUnit

### **5.3 Setup Interface to support the design of user studies**

The setup interface (Figure 5.2, Figure 5.3, and Figure 5.4) can be used by evaluators to configure user studies. The design of the setup interface took into consideration typical designs properties of user studies. These are listed below and discussed in detail in the sections below:

1. *Independent variables and experimental conditions*
2. *Experimental design types*
3. *Introductions*
4. *Standard Tests*
5. *Pre-study survey questions*
6. *Study tasks and Training*
7. *Post-study survey questions*
8. *Viewer dimensions*

In the sections below, we discuss how the setup interface of VisUnit can be used to control these properties.

#### **5.3.1 Specifying independent variables and experimental conditions**

User studies generally use two main independent variables: (a) Visualizations - evaluators can specify one or more visualizations as experimental conditions; the visualizations can be static or interactive. (b) Datasets - Evaluators use one or more datasets as experimental conditions, and the datasets can have different formats; however, some user studies do not use datasets.



## Design a user study involving Data Visualizations

---

### Create or Update files

Create or Update Viewer Directories

Create or Update Tasks

Upload or Update Datasets

---


### Study Name

Name of Study

study47

---

### Viewers

	Viewer Directory 	Condition URL	Condition Shortname
1.	Select One ▾	Select An Uploaded Web-page ▾	cond1
2.	Select One ▾	Select An Uploaded Web-page ▾	cond2

Create a new Viewer Directory



### Experiment Design Type for the visualizations

Select the experiment design type for the visualizations

Select One ▾

---

Figure 5.2: Study setup page Part I.

### Viewer Dimensions

Viewer Width	<input type="text" value="860"/>	Viewer Height	<input type="text" value="800"/>
--------------	----------------------------------	---------------	----------------------------------

### Dataset

Add A New Dataset or Update an Existing Dataset Files

	Dataset 	Format	
1.	<input type="text" value="Select One"/>	<input type="text" value=""/>	



### Experiment Design Type for the datasets

Select the experiment design type for the visualizations

### Pre-Study Activities

	Introduction URL	For which condition? 
1.	<input type="text" value=""/>	<input type="text" value="Select One"/>



Figure 5.3: Study setup page part2.

**Pre-Study Tasks**

Task Type	
1.	Select a Task

+

---

**Actual Study Tasks**

Task Type		# of Questions	Time (sec)
1.	Select a Task		

+

**Number of Training Samples**

Specify the number of training Samples

---

**Post-Study Tasks**

Task Type	
1.	Select a Task

+

Figure 5.4: Study setup page part3

**Requirement:** *The setup interface should allow evaluators to specify one or more experimental conditions for visualizations and datasets.*

**Solution:** *Specifying Visualizations* -The setup interface allows evaluators to upload the supporting files of the visualizations they want to evaluate into viewer directories of VisUnit. Evaluators can then specify one or more viewer conditions for the study. For each viewer condition, the evaluator will select the viewer directory that contain the visualization, specify the name of the visualization file and specify a short name that will be used to identify the visualization condition.

*Specifying datasets* - For studies that involve datasets, the setup interface allows evaluators to upload and select one or more datasets for the study. For each dataset condition, the evaluator will also select the format of the dataset.

### **5.3.2 Specifying experimental design**

There are two main types of experimental designs that evaluators use. These are between-subjects designs and within-subjects designs. Experimental designs are specified for visualizations, if there are more than one experimental conditions for the visualization. Similarly, experimental designs are specified for datasets if the study involve more than one dataset.

Within-subjects experiments generally randomize the order of the conditions to minimize learning effects. However, some within-subjects study intentionally used a fixed order of the conditions. For example, some studies allow all study participants to perform the study with a simple dataset condition before performing the study with a complex dataset condition.

**Requirement:** *The setup interface should allow evaluators to specify the different types of experiments: between-subjects, within-subjects, and within-subjects (fixed order).*

**Solution:** The setup interface allows evaluators to specify the experimental design type for the visualization conditions and the experimental design type for the dataset conditions.

The evaluator can select from a list one of the following options: "within-subjects", "between-subjects", and "within-subjects-fixed-order". The evaluator will

select the “within-subjects-fixed-order” option if the order of conditions in a within-subjects study is expected to be fixed.

### 5.3.3 Introductions

User studies include introductions which are used to brief participants about the study and to provide them with examples of the tasks they will be performing in the study.

**Requirement:** *The setup interface should allow evaluators to provide introductions for the user study.*

**Solution:** The setup interface allows evaluators to specify the name of an HTML file that contains an introduction to the study. Evaluators will select from a list whether to use the introduction file for all visualization conditions; or in cases where different visualization conditions deserve different introductions, the evaluator can select the visualization condition for each introduction file.

### 5.3.4 Standardized tests

Some user studies employ standardized tests which are used to measure certain characteristics of participants such as color-blindness tests and perceptual speed tests.

**Requirement:** *The setup interface should allow evaluators to specify one or more standardized tests for the study.*

**Solution:** The setup interface allows evaluators to include standardized tests to the study. For each standardized test, evaluators will specify the name of

an HTML file that contains the standardized test, and specify a name of an interface in the HTML file which can be called to return the responses provided by participants.

### **5.3.5 Pre-study survey questions**

Most user studies include one or more pre-study survey questions that are used to gather demographic information about study participants (such as gender, educational level, and experience). Responses to pre-study survey questions are collected with GUI widgets (such as text boxes and multiple-choice options depending on the type of answer).

**Requirement:** *The setup interface should allow evaluators to specify one or more pre-study survey questions that study participants can respond to.*

**Solution:** The setup interface allows evaluators to create and select one or more pre-study questions for the study. Evaluators will specify the type of answer when creating the questions.

### **5.3.6 Specifying post-study survey questions**

Optionally, evaluators can include post-study questions. These questions will include questionnaire questions that are asked at the end of the study such as asking users to rate their experience on Likert scales, and asking users to provide comments and feedbacks.

### 5.3.7 Specifying study tasks

User studies include study tasks which are the main tasks that participants perform in the study. There are two main types of study tasks: tasks that have ground truth, and tasks that do not have ground truth. There are two main way of collecting task responses: by using GUI widgets (e.g. textboxes and multiple choice options), and by interacting with visualizations (e.g. clicking on objects in the visualizations). Each study task has one or more trials, and trials can be timed or not. Participants are usually trained with sample trials of each of the study tasks that they will performing.

**Requirement:** *The setup interface should allow evaluators to create and select study tasks that have ground truth or not, and study tasks that can be answered either through GUI widgets or through interaction with visualization. The interface should allow evaluators to specify the number of trials of the task, and the timing for each task. It should allow evaluators to specify the number of trials that participants will be trained on.*

**Solution:** The setup interface allows evaluators to create and select study tasks for the study. For each selected task, evaluators will specify the number of trials (i.e. number of task instances) and the timing per trial. Section 5.7.1 describes in detail how tasks are created.

*Specifying training size* - The setup interface allows evaluators to specify the number of sample trials that will be used to train users.

### 5.3.8 Post-study survey questions

Many user studies include post-study survey questions. These questions are asked at the end of the study to gather subjective feedback from participants. For example, asking participants to rate their experience on Likert scales, and asking participants to provide comments and feedback.

**Requirement:** *The setup interface should allow evaluators to specify one or more post study survey questions that participants can respond.*

**Solution:** The setup interface allows evaluators to create and select one or more post-study questions for the study. Evaluators will specify the type of answer when creating the questions.

### 5.3.9 Viewer dimensions

Evaluators expect user study participants to see similar dimensions of visualizations being evaluated. In web-based studies where participants have the flexibility to use different devices with different screen dimensions (smart phones, tablets, laptops), the performance of users can be affected by the screen dimensions of their device.

**Requirement:** *The setup interface should allow evaluators to specify the minimum screen dimensions of devices that participants can use to perform the study.*

**Solution:** The setup interface allows evaluators to specify the minimum screen dimensions (width and height) of devices that should be allowed to perform the study. For example, if the dimensions of the viewers are 860 width x 800 height,



devices whose screen dimensions falls below these dimensions will be prevented from participating in the study.

### 5.3.10 Running a demo and saving study designs for future use

The setup interface allows evaluators to see a demo of the configured study before saving it. When a user study is saved, VisUnit saves the details of the study in an XML file as shown in Figure 5.5.

```
<?xml version="1.0"?>
<study_specification>
  <dataset>receipe_small</dataset>
  <datasetFormat>.json</datasetFormat>
  <studyname>study1</studyname>
  <experimenttype>Between</experimenttype>
  <trainingsize>3</trainingsize>
  <condition>
    <conditionurl>
      viewer3/matrix__Long_intro_IdenticalQn6.html
    </conditionurl>
    <conditionshortname>cond1</conditionshortname>
  </condition>
  <condition>
    <conditionurl>
      viewer2/nodeLink_LongTrainnig6_G5.html
    </conditionurl>
    <conditionshortname>cond2</conditionshortname>
  </condition>
  <viewerwidth>860</viewerwidth>
  <viewerheight>800</viewerheight>
  <task>
    <name>allNeighborsOfANode</name>
    <question>
      Given a highlighted node, select all its neighbors
    </question>
    <size>3</size>
    <time>29</time>
  </task>
</study_specification>
```

Figure 5.5: An XML file showing details of a designed user study.

## 5.4 Study Management Interface

The Study Management Interface (Figure 5.6) can be used by evaluators to manage previously designed user studies (existing user studies). The study management interface was designed based on the following requirement.

**Requirement:** *Allow evaluators to manage previously designed user studies. Evaluators should be able to see demos, see results, publish, edit, copy, and delete studies.*

**Solution:** The Study Management Interface allows evaluators to perform the following activities with a click of a button.

*Show Demo* - Evaluators will be able to see demos of previously designed studies.

*Publish studies* - Evaluators will be able to publish previously designed studies. To publish a study, the evaluator will be provided with the URL of the study, and they will have to either enter their Amazon Mechanical Turk (AMT) developer account information for the study to be automatically placed on the AMT crowdsourcing platform for them or place the URL of the study on Mechanical Turk themselves by creating an external HIT (a HIT where users will follow a given URL to perform the study). Evaluators can also share the URL of the study with participants through emails and social media.

*Show Results* - evaluators will be able to see the results of designed studies that has been completed by participants.

*Editing studies* - evaluators will be able to change parameters of previously designed studies.

*Copy studies*- evaluators will make copies of the configuration of previously designed studies.

*Delete studies* - Evaluators can delete studies by clicking on a button.

<input type="button" value="create New Study"/>	
Study Name	Actions
study1	<input type="button" value="Demo"/> <input type="button" value="Publish"/> <input type="button" value="Results"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>
study2	<input type="button" value="Demo"/> <input type="button" value="Publish"/> <input type="button" value="Results"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>
study3	<input type="button" value="Demo"/> <input type="button" value="Publish"/> <input type="button" value="Results"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>

Figure 5.6: A Page for managing existing user studies.

## 5.5 User study interface to automatically manage study runs

The User Study Interface is the interface that study participants interact with for the duration of the study. It is used by the Study Run Manager (described in the section below) to present the different stages of a user study to participants.

When presenting introductions and standard tests to participants, the user study interface presents the respective page in full screen. For subsequent stages of the study, the User study interface is partitioned into two panels- a large panel on the left and a small panel on the right. The large panel is used to host the visualizations (viewer window) and the small panel is used to present questions,

instructions and study controls (control window). The viewer window is resized using the dimensions specified by the evaluator, participants that do not have the required screen dimensions are prevented from taking the study.

The User Study Interface has an advance button that study participants can use to navigate through the stages of the study.

### **5.5.1 Study-Run Manager**

The Study-Run Manager is a VisUnit process that automates user study runs from beginning of the study to the end of the study. It interacts with the user study interface, and saves task responses to file. The Study-Run Manager was designed to fulfill the following requirements of study run processes which are described in detail in the following sections.

1. Choosing appropriate experimental conditions
2. Presenting Introductions
3. Presenting Standard tests
4. Presenting pre-study survey questions
5. Presenting study tasks and training
6. Providing task translations
7. Providing in-situ task instructions
8. Presenting post-study survey questions

### 5.5.2 Choosing appropriate experimental conditions

**Requirement:** *Participants of the study should be presented with an appropriate experimental condition of the study.*

**Solution:** The Study-Run Manager chooses the appropriate experimental condition for each study participant as follows.

***Experimental conditions in studies with only one experimental design:***

In a simple study design where there is only one experimental design (i.e. viewers or datasets), participants will be tested on only one condition in the between-study, and participants will be tested on all conditions in the within-study.

***Experimental conditions in studies with two experimental designs:***

When the study involve more than one viewer and more than one dataset, then the study is a factorial design involving 2 independent variables (i.e. viewers and datasets). In a between-subjects factorial design (where both experimental designs are between-subjects), both independent variables will be manipulated between the participants. For example, if the study involves two viewers (VisA and VisB) and two datasets (DsA and DsB), the study involve 4 experimental conditions (i.e. VisA-with-DsA, VisA-with-DsB, VisB-with-DsA, and VisB-with-DsB) and participants will perform the study with one of the conditions. In a within-subjects factorial design (where both experimental designs are within-subjects), both independent variables will be manipulated between the participants. For example, if the study involves two viewers (VisA and VisB) and two datasets (DsA and DsB), participants will perform the study with all four conditions (i.e. VisA-with-DsA, VisA-with-DsB, VisB-with-DsA, and VisB-with-DsB).

However, if one of the experimental designs is within-subjects and the other is between-subjects, then the experimental design of the study is mixed factorial. For example, if the study involves two viewers (VisA and VisB) and two datasets (DsA and DsB), and the experimental designs is between-subjects and within-subjects respectively, then participants will either perform the study with the conditions "VisA-DsA and VisA-DsB" or "VisB-DsA and VisB-DsB". Similarly, if the experimental design for the viewers is within-subjects and the experimental design for the datasets is between-subjects, then participants will perform the study with the conditions "VisA-DsA and VisB-DsA" or "VisA-DsB and VisB-DsB".

**Ensuring uniformity of Conditions:** To ensure the experimental conditions of the study are uniformly completed by study participants, the Study Run Manager ensures that new participants are assigned with the experimental condition (or an order of conditions in a Latin square in the case of within-subjects studies) that has the least sum of completed studies and ongoing studies. We categorize a study as an ongoing study when a participant has gone past the training stage; because, we realized from experience that considerable amount of crowdsourced participants abandon the study during the introduction and training stages of the study. A study is categorized as a completed study when the participant has performed all tasks and their responses have been saved.

### 5.5.3 Presenting introductions

**Requirement:** *Study participants should be provided with an introduction of the study.*

**Solution:** At the beginning of the study, the Study-Run Manager presents the introduction of the study by loading the introduction file specified by the evaluator into an iframe on the User Study interface (Figure 5.7).

#### 5.5.4 Presenting standardized tests

**Requirement:** *Evaluators should be able to specify and include standardized tests in their user studies.*

**Solution:** For studies that involve standardized tests, the Study-Run Manager loads the HTML files that contain the standardized tests, and receives the user responses to the tests by calling the interface methods specified by the evaluator (as described in section 5.3.4).

#### 5.5.5 Presenting pre-study survey questions

**Requirement:** *Participants should be presented with one or more pre-study survey questions.*

**Solution:** For studies that include pre-study survey questions, the Study-Run Manager presents the pre-study survey questions one at a time to participants, and creates the appropriate GUI widget to receive user responses.

#### 5.5.6 Presenting study tasks and training

**Requirement:** *Participants should be trained with sample trials of the tasks they will performing. They should be presented with the trials of the study tasks after the training, and they should be timed for each trial. Depending on the answer*

*type (described in section 5.7.1), the responses of participants should be accurately collected from GUI widgets or from interface methods specified by the evaluator. For tasks that have ground truth, their accuracy should be validated.*

**Solution:** The study run manager presents participants with training tasks and study tasks.

**Presenting training tasks** - Before training tasks are presented to participants, participants are presented with an instruction page that provides them information on the number of tasks and number of trials per task that they will be performing in the study, and the number of trials they will be training with (Figure 5.8). During the training stage, participants are presented with sample trials of the tasks that they will be performing in the study. Tasks performed by participants in the training stage are not timed. This is to give participants enough time to understand how to solve the tasks. Participants can also check the correctness of their responses with a click of a button as shown in Figure 5.9. If the experimental condition of the visualizations is within-subjects, participants will perform the training trials with the entire visualization conditions one after the other. In between-subjects studies, participants will perform the training tasks with only the visualization condition that they will be using in the actual study.

**Presenting study tasks** - For each of the tasks involved in the study, participants will be presented with the number of trials specified by the evaluator (Figure 5.10). Depending on the properties of the task (as described in section 5.8.1), participants will be asked to respond to tasks either through GUI widgets or by interacting with the visualization (interface-response-type). For tasks that



expects responses through GUI widgets, the appropriate widget is created for the task trials. A visible countdown timer is started for each trial, and the visualization is hidden once the countdown timer gets to zero.

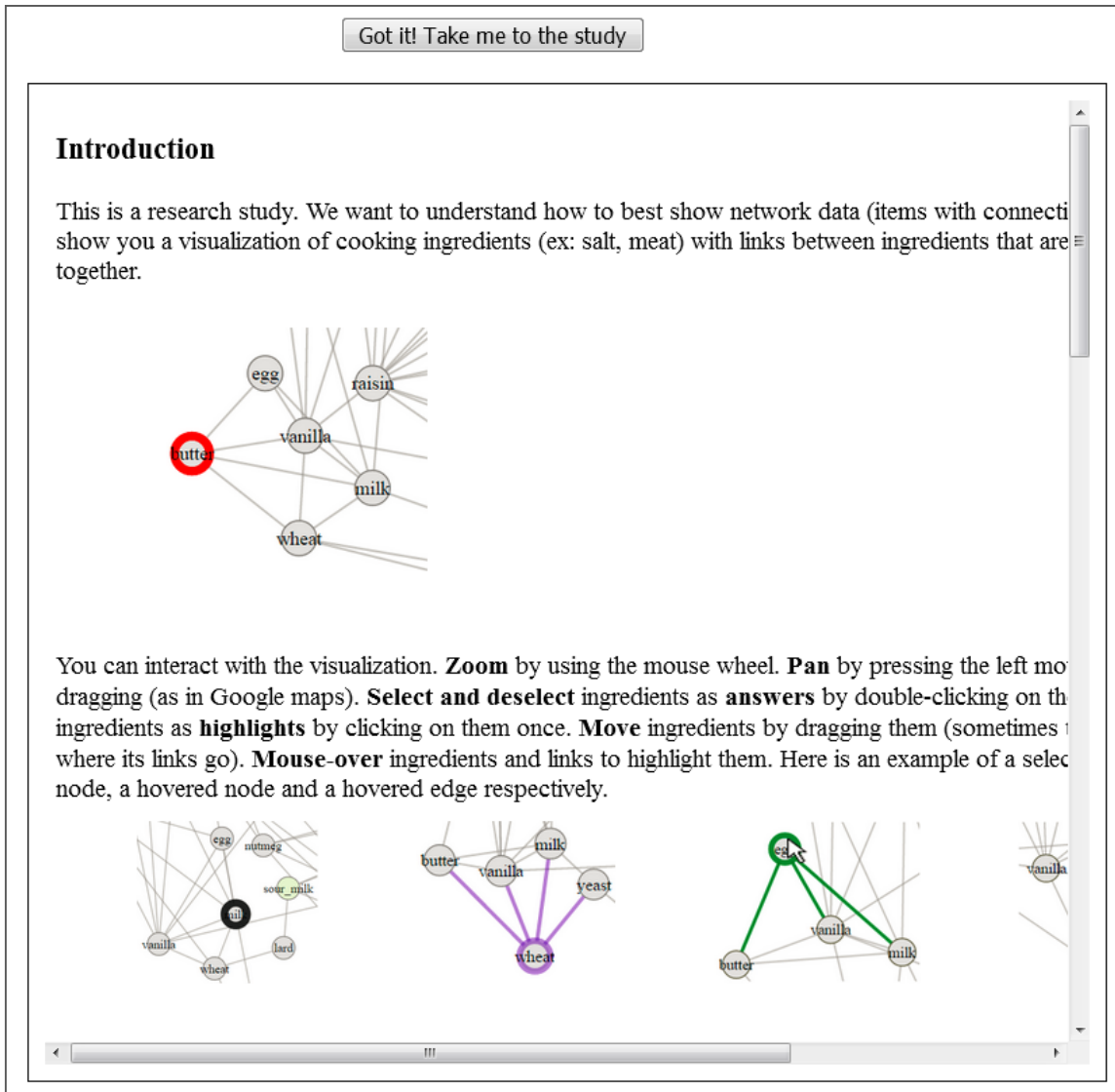


Figure 5.7: The introduction stage of the user study. VisUnit loads the introduction file into an iframe.

The image is a screenshot of a user interface for a study. On the left, there is a large, dense network graph with nodes representing various food items and ingredients. The nodes are color-coded: orange (e.g., kiwi, orange, lime, prawn, elderberry, avocado, chickpea, cilantro, blueberry, pepper, green tea, sumac, pineapple, lemongrass, mint, papaya, cumin, pear, chayote, yogurt, soybean, holy\_basil, palm, pumpkin, mackerel, quince, lentil, ginger, shrimp, sake, jasmine, pork, salmon, sea\_algae, red\_wine, lobster, shallot, pimenta, cod, lovage, okra, flower, beer, haddock, cognac, scallop, balm, herring, sherry, white\_wine, root, turnip, tarragon, corn\_grit, smoke, tam\_arind). The nodes are interconnected by a complex web of thin grey lines. On the right, there is a white rectangular box with a black border. At the top of this box is the heading "Instruction about the tasks". Below the heading is a paragraph of text: "In this study there are 2 types of questions. You will be given a simple training with 2 sample questions of each type. You can check whether your chosen answer is correct or not during the training session. For the main study, there are 8 questions in total. Thank you for participating in this study." At the bottom of the box is a button labeled "Begin Training".

**Instruction about the tasks**

In this study there are 2 types of questions. You will be given a simple training with 2 sample questions of each type. You can check whether your chosen answer is correct or not during the training session. For the main study, there are 8 questions in total. Thank you for participating in this study.

Begin Training

Figure 5.8: A short instruction about the tasks and training involved in the user study.

**Training Question (1/8)**

Are the highlighted nodes connected to cider?

(**Hint:** Interaction might help to make sure - grab and drag a node around, zoom-in/zoom-out, pan)

Yes

No

Check Answer

Next

**Interaction Hints:**

- Zoom with mouse wheel, pan by dragging.
- Double click to select answers, double-click to unselect answers
- Single click to highlight, single click to unhighlight.
- Grab and move a node if unsure where its links go.
- Hover on a node or link to see its connections.

Figure 5.9: Presenting training tasks. Participants can check the accuracy of their answers to quantitative tasks and the study is not timed.

**Study Question (2/16)**

Are the highlighted nodes connected to orange?

(Hint: Interaction might help to make sure - grab and drag a node around, zoom-in/zoom-out, pan)

Yes

No

Next

Duration: 4

**Interaction Hints:**

- Zoom with mouse wheel, pan by dragging.
- Double click to select answers, double-click to unselect answers
- Single click to highlight, single click to unhighlight.
- Grab and move a node if unsure where its links go.
- Hover on a node or link to see its connections.

Figure 5.10: Presenting actual study tasks. A countdown timer is started and the viewer window is hidden when the countdown timer gets to zero.

After each trial, participants will click on an advance button to move through the tasks. When the advance button is clicked, VisUnit records and saves to file, the response, the accuracy of the response and completion time of participants.

**Validating accuracy of responses:** For tasks that have ground truth, VisUnit automatically validates responses of participants if the responses were

provided through GUI widgets. Alternatively, if evaluators specified their own method for validating responses (section 5.7.1), then VisUnit will use that method for response validation. For responses that were provided through interaction with the visualization, VisUnit will access the response and its accuracy by calling interface methods previously specified by the evaluator (section 5.7.1).

### **5.5.7 Presenting post-study survey questions**

***Requirement:** Participants should be presented with one or more pre-study survey questions.*

**Solution:** The Study-Run Manager presents the post-study survey questions after participants finish the study tasks. The post-study survey questions are presented to participants one after the other. The Study-Run Manager creates the appropriate GUI widget to collect responses of participants. After the post-study survey questions, participants are presented with a unique code which can be used by crowdsourced participants to proof they have completed the study.

### **5.5.8 Providing task translations**

Tasks can be presented to users as visualization- dependent or domain-dependent [83]. For example, the task "Are the highlighted nodes connected" can be translated into "Are the highlighted row and column connected?" in a matrix visualization, and can also be translated as "Are the highlighted proteins connected?" in a protein graph visualization.

**Requirement:** *Allow evaluators to translate tasks for different visualization conditions.*

**Solution:** The Study-Run Manager provides a functionality that allows evaluators to translate tasks at study run time for each of the visualization conditions being used for the study. If evaluators want to translate a task at run time for a given visualization, they will implement an interface method *translateTask* in the visualization. The translation method will be passed a task and it will return the translated task.

#### **5.5.9 Providing in-situ task translations**

Since tasks and interactions allowed in a visualization can be visualization dependent, VisUnit provides a functionality to help evaluators provide task hints and interactions for each of the visualizations to serve as a guide to participants during the study. For example, evaluators can provide hints on how participants can interact with the visualization to do the tasks (e.g. "Mouse-over nodes to see how many edges they are connected to"); and also provide hints on the possible interactions that participants can perform with the visualization (e.g. "click and drag to move objects around", "zoom-in and zoom-out with the mouse wheel", etc.).

**Requirement:** *Allow evaluators to provide in-situ task instructions for different visualization conditions.*

**Solution:** The Study-Run Manager provides a functionality that allows evaluators to provide hints to tasks and hints to interactions. If evaluators want to provide task hints for a given visualization, they will implement an interface method

*getTaskHints* in the visualization. The task hints translation interface will be passed a task and it will return hints for that task. Similarly, if evaluators want to provide interaction hints for a given visualization, they will implement an interface method *getInteractionHints*. The interaction hints method will be passed a task and it will return interaction hints for that visualization.

## **5.6 Results Interface to support analysis of study results**

The results interface is used by VisUnit to present the results of studies. It is used to present graphs of the accuracy and time results, and the appropriate statistical analyses of the study.

The design of the results interface took into consideration the following requirements of user study results analysis which is described in detail in the following sections.

1. Support filtering and cleaning of results data.
2. Provide graph summaries of the results
3. Provide statistical analyses of the results

### **5.6.1 Support filtering and cleaning of results data**

Evaluators clean and filter results of participants especially in web-based studies where participants may not take the study seriously.

***Requirement:*** *Evaluators should be supported to clean and filter the results of user studies to remove participants who did not take the study seriously.*

**Filtering and cleaning result data:** The results interface allows evaluators to filter and clean the results. To simplify filtering and cleaning of result data, VisUnit provides simplified results tables for the different stages of the study (i.e. pre-study survey questions, study tasks, and post-study survey questions) for each of the experimental conditions. Each row in the table has a checkbox, which evaluators can use to filter all the records that they want.

For example, for the study tasks, VisUnit shows the accuracy of users response for each of the study tasks that has ground truth. For each study task, the accuracy and completion time of users are placed in adjoining columns. This way evaluators will be able to filter results of users who performed badly or users who did not spent much time on tasks and performed badly.

To filter the results of a given user, the evaluator will have to select a checkbox next to the row of the user's result and all the data of the user across different tables will be filtered. Figure 5.11 shows an example of results table.



	arrow				tapered				circular			
	neighbor_one_step		neighbor_two_step		neighbor_one_step		neighbor_two_step		neighbor_one_step		neighbor_two_step	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
<input type="checkbox"/>	0.75	5.0	0.75	6.75	1.0	4.5	0.75	3.75	0.75	3.75	0.75	3.25
<input checked="" type="checkbox"/>	0.5	2.25	0.75	3.0	1.0	1.75	0.25	2.25	0.5	2.75	0.5	2.75
<input type="checkbox"/>	1.0	4.25	0.25	5.25	1.0	3.25	0.75	3.25	0.75	2.75	0.0	6.25
<input type="checkbox"/>	1.0	3.75	0.25	4.0	1.0	2.5	0.25	2.5	0.0	3.0	0.25	4.25
<input checked="" type="checkbox"/>	0.75	4.5	0.5	2.0	1.0	1.25	0.75	0.75	0.25	2.75	0.25	4.0
<input type="checkbox"/>	0.75	5.0	0.5	8.5	0.75	4.75	0.75	6.5	0.75	4.5	0.0	5.5
<input type="checkbox"/>	1.0	4.75	0.0	7.25	1.0	3.5	0.75	6.0	0.75	5.0	0.25	7.25
<input type="checkbox"/>	0.75	4.5	0.25	4.0	1.0	3.75	0.5	3.0	0.75	4.5	0.25	4.5
<input type="checkbox"/>	0.75	2.75	0.5	3.75	1.0	3.5	0.0	3.25	1.0	3.5	0.25	4.0
<input type="checkbox"/>	1.0	4.0	0.0	6.75	1.0	4.5	0.25	4.5	0.75	4.75	0.25	7.25
<input type="checkbox"/>	1.0	4.5	0.25	7.5	1.0	4.5	0.75	7.0	0.75	5.0	0.0	6.0
<input type="checkbox"/>	1.0	5.0	0.0	10.0	1.0	5.0	0.75	8.75	0.75	5.0	0.0	9.5
<input type="checkbox"/>	1.0	4.25	0.75	7.75	1.0	3.75	0.25	6.0	1.0	3.5	0.0	4.0

Figure 5.11: Results page (Part 1), showing a results table of the summarized quantitative results, with two rows filtered.

### 5.6.2 Provide graph summaries of the results

Evaluators generate graphs that show the summarized accuracy and completion times of study tasks that have ground truth.

**Requirement:** Automatically generate graphs that show accuracy and completion times of study tasks that have ground truth.

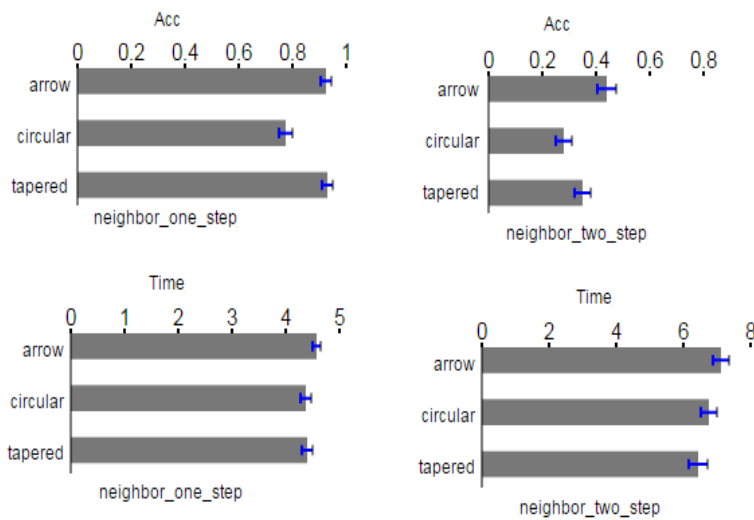
**Graphs summaries:** For each study task that has ground truth, bar-charts with standard error bars are presented for the accuracy and completion time (Figure 5.12).

### Number of completed studies (per experimental condition)

Condition-Name	Number Finished
arrow	62
tapered	62
circular	62

Select a data to show:

### Graphs of the Results



### Statistical Analysis of Results (R)

[shapiro-wilk-Analysis.txt](#)

[means-and-Standard-deviations.txt](#)

[anova-analysis.txt](#)

[friedman-analysis.txt](#)

[anova-PostHoc-Analysis.txt](#)

[friedman-PostHoc-Analysis.txt](#)

Figure 5.12: Results page (Part2). Showing the number of completed studies, bar charts of the accuracy and completion time of the quantitative tasks, and links to statistical analyses.

For each task that does not have ground truth and for pre-study and post-study questions, a graph of the summary of user responses are presented. For example, for nominal data such as numerical answers and multiple choice answers, a bar-chart that represents the distribution of user responses is presented, and for textual responses such as comments, a word-cloud that represents the summary of user responses is presented.

### 5.6.3 Provide statistical analysis of the study results

Evaluators perform means, standard deviations, and statistical analysis of the results of the study tasks.

**Requirement:** *Automatically generate means and standard deviations of the results of the study tasks.*

**Solution:** VisUnit automatically generates means, standard deviations, and statistical analysis of the results.

**Means and standard deviations:** For each study task that has ground truth, the mean and standard deviation for the accuracy and completion time is presented (Figure 5.12).

**Statistical analysis and effect Sizes:** VisUnit leverages the R [134] statistical software package for the statistical analysis of the accuracy and time results of the study tasks (Figure 5.12). For each result analysis, a Shapiro-Wilk normality test is first performed to determine the normality of the result. For each study task, the Shapiro-Wilk test checks the normality of the accuracy results and time results for each experimental condition. The accuracy of a given task is

categorized as normal if the Shapiro-Wilk test is normal across all the experimental conditions. Similarly, the time results of a given task is categorized as normal if the Shapiro-Wilk test is normal across all experimental conditions.

Depending on the number of experimental conditions and the experimental design of the study (i.e. between-subject or within-subjects), the appropriate statistical analyses and the appropriate effect size of the statistical analyses are computed. VisUnit performs parametric statistical analysis for normal distributions and perform non-parametric statistical analyses for non-normal distributions as shown in Table 5.1. Table 5.2 shows the type of effect size computation that is done for respective statistical analysis.

For a between-subjects study with exactly two conditions, an independent t-test is performed if the distribution is normal, and a Wilcoxon rank-sum test is performed if the distribution is non-normal. For a between-subjects study with more than two conditions, an independent ANOVA is performed if the result conforms to a normal distribution, and a Kruskal-Wallis is performed for non-normal distributions. For within-group studies with two conditions, a paired t-test is performed for normal distributions, and a Wilcoxon signed-rank test is performed for non-normal distributions. Finally, for within-group studies with more than two conditions, a repeated measure ANOVA is performed for normal distributions, and a Friedman test is performed for non-normal distributions.

Table 5.1: Statistical Analysis that are performed depending on the experimental design of the user study and the number of experimental conditions involved in the study.

Number of conditions	Between-subjects		Within-subjects	
	<i>Parametric</i>	<i>Non-Parametric</i>	<i>Parametric</i>	<i>Non-Parametric</i>
Exactly 2 experimental conditions	Independent T-Test	Wilcoxon rank-sum test	Paired t-test	Wilcoxon signed-rank test
More than 2 experimental conditions	independent ANOVA	Kruskal-Wallis	Repeated-measure ANOVA	Friedman test

Table 5.2: Types of effect sizes that are performed depending on the statistical analysis.

Statistical analysis	Effect size computation
Independent T-Test	<i>Cohen's d</i> [84]
Paired T-Test	<i>Cohen's dz</i> [84]
Wilcoxon rank-sum test	$r = Z/\sqrt{N}$ [85]
Wilcoxon signed-rank test	$r = Z/\sqrt{N}$ [85]
Independent ANOVA	Eta Squared ( $\eta^2$ ) [86]
Repeated measure ANOVA	Eta Squared ( $\eta^2$ ) [86]
Kruskal Wallis	$r = Z/\sqrt{N}$ [85] for pairs of experimental conditions.
Friedman test	$r = Z/\sqrt{N}$ [85] for pairs of experimental conditions.

**PostHoc analyses:** If a statistical result is significant and there are more than two conditions, a posthoc analysis is performed. For between-subjects studies, a TukeyHSD is performed for normal distributions, and for non-normal

distributions, a Wilcoxon rank-sum test is performed to compare pairs of the conditions followed by an adjustment of the resulting p-values with a Bonferroni correction. Also, for within-group studies with normally distributed results, paired t-test comparisons are performed on all condition pairs and the resulting p-values are adjusted with a Bonferroni correction. Finally, for within-group studies with a non-normal distribution of results, a Wilcoxon signed-rank test is performed to compare pairs of the conditions followed by an adjustment of the resulting p-values with Bonferroni correction.

## **5.7 Extending VisUnit with new tasks and datasets**

To support as many user studies as possible, VisUnit allows evaluators to create their own tasks, and use their own datasets.

### **5.7.1 Creating new tasks and survey questions**

To be able to support the creation of tasks, the design of VisUnit took into consideration the possible tasks that evaluators may use in a user study design.

***Requirement:** Evaluators should be able to create tasks that have ground truth, or does not have ground truth. They should be able to create tasks that can be answered with GUI widgets and tasks that can be answered by interacting with visualizations. Evaluators should also be able to create survey questions that can be answered with GUI widgets.*

**Solution:** VisUnit allows evaluators to create new tasks and survey questions using a user-friendly form (Figure 5.13). To create new tasks and survey

questions, evaluators will be required to provide the following details about a new task or survey question:

**Task shortname:** Evaluators will be required to provide a unique *shortname* (e.g. *neighborConnection*) for the task. This will allow easy identification of tasks in results.

**Task display question:** Evaluators will provide the display question for the task (e.g. "Are the two highlighted nodes connected?"). For tasks where the *display question* will be changing dynamically depending on the inputs of task instances, evaluators will place placeholders in the *display question* to specify the input that will be placed in the display question. For example, for the display question "What is the highest value for the attribute <attribute-name> ", evaluators will replace <attribute-name> with the placeholder "\$#", where # is a number that represents the number of the input to be used, e.g. \$2 means the second input). Display questions that have placeholders will be dynamically translated by the Study-Run Manager during the run of studies. For example, if the *display question* of a given task is "Are the highlighted nodes connected to \$2?", and the second input of the task instance is "commando" then the *display question* of the task instance will be translated to "Are the highlighted nodes connected to commando?".

**Task description:** A short description of the task that can shed light on what the task is about, for example, "In this task, two nodes will be highlighted and study participants will determine if the highlighted nodes are connected".

**Task category:** There are generally two categories of tasks. These categories are:

(a) Tasks that have correct answers or ground truth - these are tasks where the accuracy of a user's response can be determined. For such tasks, VisUnit can automatically check the accuracy of user responses by comparing them to the correct answers. An example of a task with correct answer is "How many clusters are in the visualization?"

(b) Tasks that do not have correct answers or ground truth - these are tasks where the accuracy of a user's response cannot be determined. For this category of tasks, the responses will be saved without determining the accuracy of user responses. An example of a task that does not have a correct answer is "How will you rate the helpfulness of the visualization between 1 and 5?".

**Response to tasks:** User responses to tasks and survey questions can be broadly categorized into two groups:

(a) Tasks that can be answered via GUI response widgets (e.g. radio buttons, drop down lists, textboxes). Tasks that require users to enter numerical answers, text-based answers, or multiple choice answers fall in this category. VisUnit supports the following types of answers that evaluators can select from: numbers, text, and options. VisUnit generates number boxes, text boxes, and radio buttons for numbers, texts and options answer types respectively.

There are two types of options that can be used. *Fixed options:* fixed options are options that will have the same values for all task instances. For example, for the task "Are the two highlighted nodes connected?", participants will



choose from the answer options "Yes/No" for all task instances. If the evaluator chooses "fixed options" then the evaluator will provide the options.

## Create a New Task

---

Provide a short name for this task

**Task Question**

Type the task question as will be shown to users

*(NB: If the question text will include dynamic values, use "\$1", "\$2", etc. as placeholders for the dynamic values, where the number 1 or 2 represents the position of the value among inputs provided for the task).*

Type a short description of the task

---

**Answer type**

Will this task have correct Answers?

Select the answer type

---

**Info about the inputs**

	Short name	How input will be provided	Short Description
1.	<input type="text"/>	<input type="text" value="Select One"/>	<input type="text"/>

*NB: If inputs are required by this task, you have to implement interfaces for each input type specified (i.e. accessor; and mutator methods). For example, if the shortname for an input type is "node", you have to implement the methods "getNode()" and "setNode()" in your visualization.*

---

Figure 5.13: An interface for creating new tasks.

***Dynamic options*** - dynamic options are options that will have different values for different options. For example, for the task "which of the following colored clusters have more members", each task instance may have different color options (e.g. red/blue, yellow/green) for participants to choose from. If the evaluator chooses "dynamic options", then the evaluator will include the unique options with the task instances.

For tasks that have correct answers, VisUnit will validate the response of participants to the correct answer. Alternatively, if evaluators have their own formula for computing accuracy, then they can specify the name of an interface in their visualization which can be called to compute the accuracy. For example, instead of a correct or wrong comparison, evaluators can choose to approximate how close the response is to the correct answer.

(b) Tasks that can be answered by interacting with the visualization - these are tasks where users provide responses by interacting with objects in the visualization. For example, for the task "select the node with the highest number of children", participants will be required to click on a node in the visualization. For such response types, the visualization should be able to inform the framework about the user responses through an interface implementation. VisUnit will expect the evaluator to provide a short name to represent the type of response (e.g. nodes). VisUnit will expect the evaluator to implement accessor and mutator methods for this response type, so that during the run of the study, the accessor method can be called to get the response and the mutator method can be called to reset the answer. For example, if the answer type is "nodes", then VisUnit will

expect the evaluator to implement in the visualization the accessor method "getNodes" and the mutator method "setNodes". Additionally, if the task is a task that has correct answer, VisUnit will expect the evaluator to implement an interface in the visualization which when called and passed a user's answer and the correct answer for that task, will return a value that represents the accuracy of the user's answer. For example, for the task "select the most connected node", an evaluator can have an interface in their visualization called "validateMostConnectedNode" which will be called to validate the responses of users.

**Inputs to tasks:** Some study tasks require inputs. For example, for the following tasks "What is the degree of the highlighted node?" and "What is the degree of node A?", the *highlighted node* and *node A* are variables that get changed for multiple task instances. Hence each of such task instances will have inputs for the changing variables.

If a task requires inputs, the evaluator will be required to specify the details of the inputs. For each input, the evaluator will provide a short name to represent the type of the input (e.g. node), a description of the input (e.g. a single node), and select from a combo-box how the input will be provided during task instance creation (i.e. either by typing in a text widget or by interacting with the visualization). Inputs can be broadly categorized inputs into two groups:

(a) Inputs that are highlighted in a visualization - For example, for the task "Approximate the average value of the highlighted items?", each task instance will have a set of inputs (i.e. "items") that will be highlighted. For these tasks, the evaluator will be required to implement a mutator method which when called and

passed the required inputs, will perform the highlighting of the inputs in the visualization. For example, if the type of the input is "node", the evaluator will be expected to implement the mutator method "setNode".

(b) Inputs that will be part of the *display question* of the task- for this type of inputs, there will be a placeholder ("\$#") in the *display question* of the task which VisUnit will replace with the value of the input. For example, if the input is "Cayenne", and the *display question* is "Does the connections of the highlighted ingredient include \$1", VisUnit will translate the display question to "Does the connections of the highlighted ingredient include Cayenne?"

The definition of the task is then saved in an XML file as shown in Figure 5.14

```

<?xml version="1.0"?>
<task_details>
  <taskname>twoNodesConnectionToANamedNode</taskname>
  <accuracyCheckingInterface></accuracyCheckingInterface>
  <outputtype></outputtype>
  <outputTypeDescription></outputTypeDescription>
  <answertype>options-fixed:::Yes::No</answertype>
  <taskquestion>
    Are the highlighted nodes connected to $2?
  </taskquestion>
  <taskDescription>
    Check if highlighted nodes are connected to a named node
  </taskDescription>
  <inputsize>2</inputsize>
  <input>
    <inputtype>nodes</inputtype>
    <inputdescription>
      This is a group of nodes at least 2.
    </inputdescription>
    <inputmedium>from-visualization</inputmedium>
  </input>
  <input>
    <inputtype>node</inputtype>
    <inputdescription>one node</inputdescription>
    <inputmedium>by-typing</inputmedium>
  </input>

  <hasCorrectAnswer>yes</hasCorrectAnswer>
</task_details>

```

Figure 5.14: An example of an XML file for tasks.

## 5.7.2 Creating new task instances

Task Instances are input and answer pairs which are based on a given dataset or are based on a given viewer (for studies that do not use datasets).

**Requirement:** *Evaluators should be able to create task instances for datasets and viewers*

**Solution:** VisUnit allows evaluators to either upload an xml file, or interactively create the task instance file step by step. To create task instances, the evaluator will specify the task, and the dataset or viewer that the task instances will be based on. Evaluators can then upload an xml file for the task instances or create the task instances step by step.

If evaluators choose to create task instances step **by** step, the task instance creation interface walks the evaluator through the process of providing input(s), and answers for each task instance. VisUnit will allow the evaluator to specify the inputs either by typing or by interacting with the visualization as specified in the task definition. Evaluators will provide responses to task instances by either using a GUI widget (for widget answer types), or by interacting with the visualization (for visualization interaction answer types).

The input and answer pairs provided by the evaluator are then saved in an XML file as shown in Figure 5.15.

```

<?xml version="1.0"?>
<taskInstancesFile>
  <question>
    <input>cabernet;;porcini</input>
    <input>cider</input>
    <answer>No</answer>
  </question>
  <question>
    <input>cognac;;rosemary</input>
    <input>porcini</input>
    <answer>Yes</answer>
  </question>
  <question>
    <input>peanut;;rose</input>
    <input>quince</input>
    <answer>No</answer>
  </question>
  <question>
    <input>chickpea;;peach</input>
    <input>orange</input>
    <answer>No</answer>
  </question>
</taskInstancesFile>

```

Figure 5.15: An example of an XML file for task instances.

### 5.7.3 Adding new datasets

**Requirement:** Allow evaluators to use their own datasets for user studies.

VisUnit allows evaluators to upload and use their own datasets for user studies.

VisUnit provides a user friendly interface which evaluators can use to upload their datasets. Evaluators can upload different file formats of the same dataset. For example, a given dataset can have one or more of the following file formats: JSON, tab-separated values (TSV), and comma-separated values (CSV).

## 5.8 VisUnit storage

This storage is used to store system data and user specific data. VisUnit uses two main storages: one for storing VisUnit's data (system data), and one for storing data for evaluators (user-specific data). VisUnit uses the system data storage for storing benchmark tasks ("system tasks"), benchmark datasets ("system datasets"), and task instances of benchmark datasets ("system task instances").

The user-specific data **storage** is dedicated to evaluators. VisUnit creates a directory for each evaluator within a directory called "users". Each "user" directory has the following directories: "tasks" - for storing tasks and questions created by the evaluator; "task instances" - for storing task instance files created by the evaluator; "Datasets" - for storing datasets uploaded by the evaluator; "Viewers" - for storing different directories of visualization files; "User Studies" - for storing different directories of user studies.

## 5.9 Summary

We have designed and implemented VisUnit a framework that semi-automates the process of designing, running, and analyzing results of a wide range of user studies that involve data visualizations. The design of VisUnit leveraged requirements gathered from the design and evaluation of GraphUnit, and requirements gathered from visualization literature. We describe the general architecture of VisUnit and describe how the architecture of VisUnit can be used to fulfill requirements that evaluators have when designing user studies, running user studies, and analyzing results of user studies.



## 6 VALIDATION: EVALUATION OF VISUNIT

In this chapter we present the evaluation of VisUnit. We demonstrate VisUnit's effectiveness by showing that it can be used to replicate a significant portion of existing user studies in the visualization literature, and we also describe samples of successful studies that we run with VisUnit.

We also evaluated the efficiency of VisUnit by showing that evaluators can use VisUnit to design complex user studies using their own datasets, tasks and task instances in less than an hour. Specifically, we conducted a user study involving 5 graduate students who are familiar with information visualization and who are unaffiliated with this research project. Participants downloaded freely available visualizations and datasets, created tasks and task instances, and designed a user study. We recorded the time it took these participants to set this user study up. To compare these time results to a baseline, we conducted a survey of researchers who have published papers involving user studies and asked them to estimate the time it would take graduate students to design a user study like the one we asked our participants to set up. We found the time it took our participants to design the study using VisUnit was significantly better than the time researchers expected graduate students to spend designing a similar study.

## 6.1 Evaluating the effectiveness of VisUnit

In this section, we validate the generality of the VisUnit framework by showing that it can be used to facilitate the design and deployment of a wide range of previous user studies.

Specifically, we surveyed the information visualization literature for papers involving controlled user studies that have appeared in IEEE Information Visualization (InfoVis) conference since 1995. We searched for the surveyed papers from a dataset organized by Stasko et al. [133] which contains all papers published in the InfoVis conference between 1995 and 2015. From this dataset of InfoVis papers, we searched for all papers that were tagged as “evaluation”, “experiment”, “usability, and “user study”. We found 101 of such papers. We read the procedure and description of the studies contained in the 101 papers and we determined if the features of the studies can be supported by VisUnit and if VisUnit can be used to replicate the studies. Of these 101 papers, we found the studies conducted in 85 of the papers (84%) to be replicable with VisUnit and the studies conducted in the remaining 16 papers (16%) to be non-replicable. We describe examples of the 85 studies that can be replicated with VisUnit in the following paragraphs, and in section 6.1.7, we discuss the reasons why VisUnit cannot be used to replicate 16% of the studies.

To understand qualitatively how VisUnit can support real-life studies, we identified the features of a subset of the 85 replicable studies. We organized this subset of replicable studies by selecting all studies that have averaged a minimum of 8 citations per year from their year of publication. There were 33 papers in total

and they cover a broad area of user studies conducted in InfoVis. Table 6.1 shows the features of the 33 studies that can be supported by VisUnit. Thirty-one (31) of the 33 studies can be replicated exactly as they were fielded with the current design of VisUnit. We describe in detail four (4) of these studies that are well cited in section 6.1.1 - section 6.1.4. For the remaining two (2) that cannot be replicated exactly as they were fielded, we describe why, and how VisUnit can be used to design a similar study that achieves the same objective in section 6.1.5 and section 6.1.6.

Table 6.1: The properties of a list of user studies published in InfoVis that can be supported by VisUnit.

<b>Properties of VisUnit</b>											
<p><b>A</b> (Introduction), <b>B</b> (Standardized tests), <b>C</b> - (Pre-study survey questions), <b>D</b> (Post-study questions),  <b>E</b> (training), <b>F</b> (one or more datasets), <b>G</b> (include within-subjects-fixed-order experiment design),  <b>H</b> (response use GUI widgets), <b>I</b> (response use interaction with visualization)</p>											
<b>No.</b>	<b>Study</b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>Can it be Replicated?</b>
1	Ghoniem et al. [28]	1		1		1	1		1		Yes
2	Laidlaw et al. [29]	1				1			1	1	Yes
3	Heer et al. [12]	1				1			1	1	Yes
4	Heer et al. [30]	1				1			1		Yes
5	Kosara et al. [13]	1		1	1	1	1		1		Yes
6	Borkin et al. [98]	1	1	1	1	1			1	1	Yes
7	Kobsa et al. [99]	1			1	1	1		1	1	Yes

8	Robertson et al. [38]	1		1	1	1	1	1		1	Yes
9	Sanyal et al. [37]	1				1	1		1	1	Yes
10	Haroz et al. [88]	1				1			1		Yes
11	Javed et al. [89]	1		1		1			1		Yes
12	Henry et al. [90]	1		1		1	1	1	1	1	Yes
13	Ziemkiewicz et al. [91]	1	1		1	1	1		1		Yes
14	Wen et al. [92]	1		1	1	1	1		1		Yes
15	Jianu et al. [15]	1				1	1		1	1	Yes
16	Saket et al. [93]	1				1	1		1	1	Yes
17	Willet et al. [100]	1			1	1	1				Yes
18	Lee et al. [101]	1			1	1	1	1			Yes
19	Plasaint et al. [102]	1			1	1	1		1		Yes
20	Van et al. [103]	1			1	1	1		1	1	Yes
21	Wong et al. [104]	1			1	1	1			1	Yes

22	Steinberger et al. [105]	1	1	1	1	1			1		Yes
23	Smith et al. [106]	1	1		1				1		Yes
24	Heer et al. [107]	1				1			1	1	Yes
25	Ziemkiewicz et al. [108]	1		1	1	1	1		1		Yes
26	Clarkson et al. [109]	1			1	1	1		1		Yes
27	Cao et al. [110]	1			1	1	1		1		Yes
28	Koh et al. [111]	1			1	1	1		1		Yes
29	Kong et al. [112]	1				1			1		Yes
30	Cao et al. [113]	1			1	1	1		1		Yes
31	Micallef et al. [114]	1			1	1			1		Yes
32	Rufiange et al. [115]	1				1	1		1		Yes
33	Harrison et al. [116]	1			1	1			1		Yes

### 6.1.1 Ziemkiewicz et al. [91]

The authors evaluated how users with different personality types react to varying visualization layout styles used in hierarchy visualizations. The study was a within-subjects study involving 4 hierarchy visualizations, four datasets, and two tasks (one instance per task). Before performing the study, participants were first given a personality test by asking them to provide their rating for each item on a 40-question personality scale. Participants were then trained, and then presented with the two tasks to perform on each of the visualizations. Each visualization was randomly associated with one of the four datasets and each participant saw all four datasets. After performing the tasks with each visualization, participants are presented with qualitative questions that asked them to rate their likeness of the visualization.

**How can this study be designed with VisUnit:** This study involves four visualizations conditions, and four dataset conditions. The experimental design of the visualizations is within-subjects and the experimental condition of the datasets is between-subjects. The pre-study questions will include a standardized test where the name of an html file that contains the required personality questions will be specified. The responses to tasks will be taken with widgets and the responses can either be validated by VisUnit or by calling an interface method specified by the evaluator. The name of the introduction file and the number of training trials will be specified.

### 6.1.2 Laidlaw et al. [29]

The authors evaluated six visualization methods for displaying 2D vector data (i.e. GRID, JIT, LIT, LIC, OSTR, GSTR), using 3 tasks. The visualizations used in this visualization were static visualizations (or images). It was a within-subjects study and users were expected to respond to some of the tasks by clicking in locations in the visualization. Each task instance was a different image.

**How can this study be designed with VisUnit:** This study involves six visualization conditions and no datasets. The experimental design of the visualizations is within-subjects. Task instances will be created for each of the visualization conditions, and each task instance will include the name of an image for that instance. The visualization conditions will be responsible for displaying the images for each given task instance. For the tasks that expect response through interaction, the specification of the task will include two interface methods (one for getting user responses, and one for validating user responses). The name of the introduction file and the number of training samples will also be specified.

### 6.1.3 Heer et al. [12]

The authors performed 4 experiments and VisUnit can be used to design all four studies. We describe two of these studies below.

**Experiment 1A** - The study involved 7 judgment types in 10 static charts for a total of 70 trials. Participants trained on sample charts before starting the actual test. For each of the 70 trials, participants responded to two tasks using widgets. The authors had a special way of validating answers using a log absolute measure.



**How can this study be designed with VisUnit:** Each of the 7 judgment types will be represented as a viewer condition. This study does not involve datasets so task instances will be based on the visualization conditions. Each task instance will include the name of the chart to display for that instance, and each visualization will be responsible for displaying the charts for each given task instance. The experimental design of the visualization will be within-subjects where participants performed the two tasks with each of the 7 judgment types. The specification of the tasks will include the name of an interface method that will be called to validate user responses. The name of the introduction file and the number of training samples will also be specified.

**Experiment 1B:** This study was also about judgments. They used a 2 display types (i.e. rectangles and tree maps) X 9 (aspect ratios) factorial design with 6 replications, a total of 108 trials. Participants performed the same two tasks they performed in Experiment1A.

**How can this study be designed with VisUnit:** In this study, there will be 18 (i.e. 2x9) visualization conditions. Similar to experiment 1A, this task does not involve datasets and as such, task instances will be based on the visualization conditions. The experimental design of the visualizations is within-subjects. Each task instance will include the name of the chart to display and the visualization conditions will be responsible for displaying the charts. The specification of the task will include the name of an interface method that will be called to validate user responses. The name of the introduction file and the number of training samples will also be specified.

#### 6.1.4 Haroz et al. [88]

The authors presented three experiments and each of the experiments was a within-subjects study involving 4 groups of static visual stimuli (i.e. color-grouped, color-random, motion-grouped, and motion-random). Participants performed one task in each experiment and 40 trials of the task was performed with each group of visual stimuli. The details of the experiments are as follows:

**Experiment 1** - Participants looked for a known target. Each trial began by displaying a square with the target cue followed by a blank screen, and finally the stimulus. Participants responded to the task using the keyboard.

**Experiment 2** - Participants looked for a unique target with an unknown appearance. For each trial, subjects were shown a visual stimuli and they determined if a unique target exists in the visualization.

**Experiment 3** - Participants compared the number of visual categories in a pair of visual stimuli and determine the visual stimuli that has more variety. For each trial, the first visual stimuli is presented for 500 ms followed by a one second blank gray screen and the second visual stimuli is displayed for 500 ms. Participants responded to the task using the keyboard.

**How can these studies be designed with VisUnit:** For each of these experiments, there are four visualization conditions (i.e. color-grouped, color-random, motion-grouped, motion-random). Since this user study does not involve datasets, task instances will be created for each of visualization conditions, and each task instance will include the name of the visual stimuli image of that task

instance. For Experiment 1 and 3 where participants are shown parts of the stimuli for a short number of seconds, the visualization condition will coordinate the timing of the display. For example, for the task performed in experiment 1, the visualization will be passed the target cue and the visual stimuli. The visualization will then display the target cue for the required number of seconds, display the blank screen for the required number of seconds, and then display the visual stimuli. Similarly, for the task performed in experiment 3, the visualization will be passed the two visual stimuli, and the visualization will display the first visual stimuli for the required number of seconds, display the blank screen for the required number of seconds, and then display the second visual stimuli. Participants responses to tasks will be received with widgets and the responses will be validated by VisUnit.

#### **6.1.5 Robertson et al. [38]**

The authors evaluated the effectiveness of animation in trend visualization. The study was designed as a 3 (visualization) x 2 (dataset size) within-subjects design. The order of the datasets was fixed with the smaller dataset used first for each visualization condition. The visualizations used were interactive, and participants provided answers by selecting objects in the visualization. There were 8 tasks per visualization condition, and participants performed the first four tasks with the smaller dataset and the other four with the large dataset. Participants were screened to ensure they were not color-blind before the test. After performing the task with each visualization, participants performed a survey question related to

that visualization. A set of general survey questions were also asked at the end of the study.

**How can this study be designed with VisUnit:** In this study there are 3 visualization conditions and 2 dataset conditions. The experimental design of the visualizations and the experimental design of the datasets are within-subjects (fixed-order). Evaluators will create a standardized test as a pre-study task. Though VisUnit cannot currently prevent participants that will fail this test from participating in the study, the evaluator can use this information to later filter the study results. The survey question that is asked after each visualization condition will be included as part of the actual study tasks. The survey question that is asked at the end of the general study will be included as part of the post-study tasks. Responses to tasks can be taken with widgets and through interaction with the visualization. The specification for tasks that require responses through interaction must include the name of an interface for getting the response, and the name of an interface for validating the response. Currently, VisUnit will not be able to present the tasks to participants exactly as it was done in the study, as noted in the following limitation.

**Minor limitation:** In this study, the first four questions were asked on the small dataset, and the next four questions were asked on the large dataset. VisUnit currently expects the study tasks to be performed on all the experimental conditions (i.e. visualizations and datasets), and hence cannot support asking users to perform subsets of the tasks with different experimental conditions. Therefore, if this study is to be designed with VisUnit, participants will be asked to

perform all the eight tasks with each experimental dataset condition. Alternatively, this study can be designed as two individual studies each involving one dataset.

#### **6.1.6 Jianu et al. [15]**

The authors evaluated techniques that are used in displaying cluster information in node-link diagrams. The study was a between-subjects study involving four visualizations (each having a different clustering technique), one dataset, and 10 tasks with each task having multiple task instances. The tasks were evaluated in four sessions with each session involving at least two of the tasks. Participants responded to tasks with widgets and also by interacting with the visualization.

**How can this study be designed with VisUnit:** This study involves 4 visualization conditions, one dataset and ten tasks. The experimental condition of the visualization is between subjects. Responses to tasks will be received with widgets and by interaction with the visualization. Responses received with widgets can be validated by VisUnit and responses received by interaction with the visualization can be validated by calling interface method specified by the evaluator.

**Minor limitation:** VisUnit currently does not support dividing the study into different sessions with each session containing a subset of the tasks of the study. However, there are two ways VisUnit can be used to design a study that achieves a similar goal: (1) the study can be designed with 10 tasks, and study participants

will perform all the ten tasks; or (2) this study can be designed as four similar studies with each study having the required number of tasks.

### **6.1.7 Discussion of the limitations of VisUnit**

We discovered that VisUnit cannot be used to replicate the controlled user studies reported by 16% of the 101 surveyed papers that have appeared in IEEE Information Visualization (InfoVis) between 1995 and 2015 such as these papers [117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132]. We found out that the following limitations of VisUnit makes it infeasible for such user studies.

**User studies involving complex visualization tools or non-web based visualizations:** User studies that ask users to perform activities with complex visualization tools are not supported by VisUnit [117, 118, 120]. Similarly, user studies that involve non-web based visualizations are not supported by VisUnit. VisUnit require some level of control over the visualization being evaluated in order to coordinate the user study process from beginning to end. As such, the evaluation of complex visualization tools or visualization systems that do not lend themselves easily to manipulation through web-interface calls are not supported by VisUnit.

**User studies that measure beyond time and error:** Studies that measure dependent variables beyond time and error are not supported by VisUnit. For example, studies that involve eye-tracking or studies that use think-aloud protocols cannot currently be supported by VisUnit [121, 129, 130, 131]. Similarly, user

studies that measure other variables apart from accuracy and completion time cannot be supported effectively by VisUnit.

**Insight-based studies:** Studies that require users to generate insight from visualizations cannot be supported efficiently by VisUnit [118,124, 131]. VisUnit supports questions that participants can provide answers to either through widgets or through interaction with the visualization. Insight-based studies that aim to learn something from user interactions cannot be efficiently supported by VisUnit because VisUnit does not currently generate user interactions logs.

**User studies involving collaborative systems:** VisUnit does not support user studies involving collaborative visualization systems or systems that expects participants to work in groups [119]. VisUnit assumes participants will be working alone on studies, as such studies that require participants to collaborate cannot be supported by VisUnit.

**User studies that require physical presence:** Studies that require the physical presence of the evaluator or the use of a specific physical environment cannot be efficiently supported by VisUnit [123, 125, 126, 128]. For example, studies that require participants to manually draw visual objects, studies that require the evaluator to manually change parameters of the visualizations for participants, and studies that require the use of specific devices. Although it is not feasible to automatically run such studies with VisUnit, VisUnit can be used to design and guide sections of such studies and to collect some data.

## **6.2 Evaluating the efficiency of VisUnit**

We conducted a user study with 5 graduate students who are familiar with web-based visualizations and are unaffiliated to this research project. In this study, participants were asked to compare two visualization techniques for representing tree hierarchies using a freely available tree dataset (i.e. “flare.json”) and freely available tree visualizations (i.e. a static radial tree (Figure 6.1) , and a collapsible indented tree (Figure 6.2)). Participants were asked to download the dataset and the visualizations from the D3 [11] website. We measured the time it took participants to download and prepare their visualizations, create new tasks, create new task instances, and design the user study. A description of the user study and its results is presented in section 6.2.1 and section 6.2.2 respectively.

To be able to compare the time results of this user study to a baseline and to put the support of VisUnit into perspective, we conducted a survey of researchers who have published papers involving controlled user studies. We asked the researchers to estimate the time it would take graduate students to design a user similar to the one designed by our study participants. A description of the survey and survey results is presented in section 6.2.3 and section 6.2.4 respectively. We provide a comparison of the user study results to the survey results in section 6.2.5.

### **6.2.1 User study setup**

Participants were first given spoken introductions to the functionalities of VisUnit including how to create tasks, how to create task instances, and how to design a user study.



Participants were then provided with the following written instructions to perform as duties of the study:

1. Download the two visualizations and the dataset from the D3 website.
  - (a) Collapsible tree: <http://bl.ocks.org/mbostock/1093025>
  - (b) Radial Reingold Tree: <http://bl.ocks.org/mbostock/4063550>
2. Modify the visualizations to have a *setDataset* function and ensure that the function can set the dataset for the visualization and display the visualization.
3. Create the following three study tasks that users can respond to using widgets: Does *nodeX* have more children than *nodeY*?, How many children does *nodeZ* have?, Does *nodeB* and *nodeX* belong to the same immediate parent?
4. Create ten task instances for each of the three tasks created in step 3.
5. Create the following survey questions: a question that asks for the age of participants; a question that asks participants to rate their experience on a scale of 1 to 5.
6. Design a user study involving the two downloaded visualizations and perform a demo of the designed study. For the design, at the pre-study stage, participants should be asked their age. At the actual study stage, participants should be asked to perform the 3 tasks created in step1, and at the post-study stage, participants should be asked to rate their experience on a scale of 1 to 5.



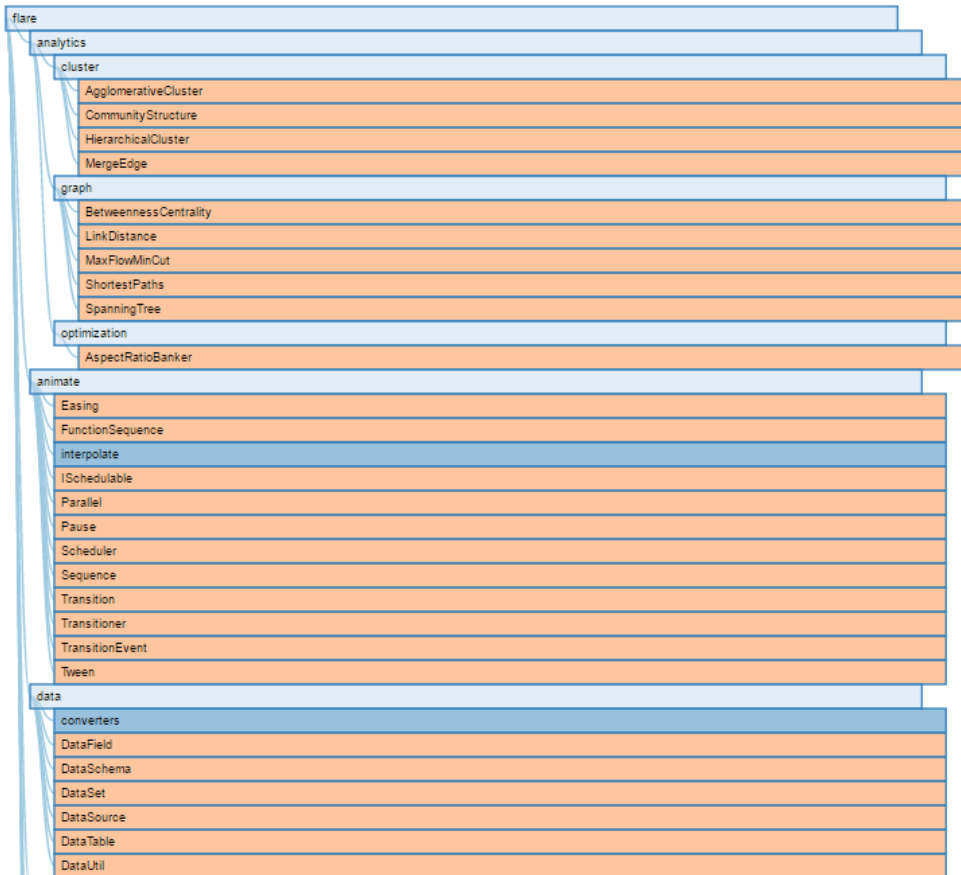


Figure 6.2 : Collapsible indented tree involved in the designed user study.

Table 6.2 : Results table showing the times in minutes that that it took study participants to create tasks, create task instances, and design user studies and see demos

Participant	Time (mins)					
	Step 1 & 2	Step 3	Step4	Step5	Step 6	Total
<i>User1</i>	6	6	12	3	8	35
<i>User2</i>	9	10	9	3	8	39
<i>User3</i>	11	9	12	3	10	45
<i>User4</i>	12	13	13	5	12	55
<i>User5</i>	9	8	10	4	9	40
<b>Average</b>						<b>42.8 mins</b>

### 6.2.2 User study results

The performance of participants is shown in Table 6.2. The mean time participants took to perform all the tasks in the study was 42.8 minutes, and the standard deviation was 7.7 minutes. Hence on average, it took study participants less than 43 minutes to prepare their visualizations, create tasks, create task instances, design a user study and see a demo. Specifically, it took graduate students on average, 9.4 minutes to download and prepare the visualizations, 9.2 minutes to create three study tasks, 11.2 minutes to create ten task instances, 3.6 minutes to create two survey questions, and 9.4 minutes to design the user study.

### 6.2.3 Survey involving researchers who conduct controlled user studies

We conducted a survey involving experienced researchers who have previously published a paper that involves a controlled user study. We designed

the survey to serve two main purposes: first, to have a baseline time to compare the results of our user studies to; and second, to gain an insight into the time it took the researchers to conduct the user studies in their papers. We contacted the authors of twenty (20) user study papers published in the information visualization community within the last 5 years (i.e. between 2011 and 2015). These papers were selected from the list of previous studies that can be replicated with VisUnit (Table 6.1). We contacted 43 authors in total by email.

In this survey, we showed the researchers one of the studies designed by our study participants and asked them to estimate the time it will take a graduate student to design such a user study (excluding the time used to implement the visualizations). Specifically, the researchers responded to the following two questions which served as a baseline to compare the results of our user study:

1. How long will it take to create the tasks (i.e. pick the data objects involved in each generic task)?
2. How long will it take to develop the web infrastructure that coordinates the whole user study (i.e. presents the tasks to users, enforces time limits, collects answers, etc.)

Additionally, we asked the researchers to tell us the time it took them to conduct the user study in their paper (excluding the time they used to implement the visualizations). Specifically, the researchers responded to the following three questions which provided an insight into the typical times researchers spend on user studies:

3. How long did it take you to design your user study and implement the infrastructure you used to conduct the user study (e.g. task instances, study presentation, answer logging)?
4. How long did it take you to actually run the user study (e.g., subject management)?
5. How long did it take you to analyze the results of the user studies?

#### **6.2.4 Survey results**

The lead authors of six (6) papers responded to our survey, and their responses to our baseline survey questions are shown in Table 6.3. On average, researchers expected graduate students to spend 20.5 days (492 minutes) to create the tasks, and 32 days (i.e. 768 minutes) to create the infrastructure for the study. Overall, researchers expected graduate students to spend 52.5 days (i.e. 1260 minutes) in designing the study, with a standard deviation of 46 days (i.e. 1114 minutes). The survey results of the time it took researchers to conduct the user studies in their own papers is shown in Table 6.4. On average, evaluators reported that they spent 26 weeks (~ 7 months) to design, run, and analyze the results of their user study. Specifically, researchers reported they spent on average 18 weeks to design their study (Q3), 5 weeks to run the study (Q4), and 4 weeks to analyze the results of the study (Q5).

These survey results confirm the fact that user studies are time consuming and it takes evaluators several months to conduct them successfully. As such, VisUnit can save evaluators a considerable amount of time.

Table 6.3 : Survey results table showing the times researchers expected graduate students to use in designing the user study.

<b>Researcher</b>	<b>Time to create task (Q1)</b>	<b>Time to develop web infrastructure (Q2)</b>	<b>Total (days)</b>
<b>Researcher1</b>	10 days	3 months	100
<b>Researcher2</b>	2 weeks	2 weeks	28
<b>Researcher3</b>	1 week	1 week	14
<b>Researcher4</b>	2 months	2 months	120
<b>Researcher5</b>	1 month	1 week	37
<b>Researcher6</b>	2 days	2 weeks	16
<b>Average</b>	<b>20.5 days</b>	<b>32 days</b>	<b>52.5 days</b>

Table 6.4 : Survey results table showing the time researchers reported to spend on designing user studies (Q3), running user studies (Q4), and analyzing the results of user studies (Q5).

<b>Researcher</b>	<b>Q3</b>	<b>Q4</b>	<b>Q5</b>	<b>Total (weeks)</b>
<b>Researcher1</b>	4 months	3 weeks	1 month	23
<b>Researcher2</b>	3 months	1 month	2 months	24
<b>Researcher3</b>	18 months	2 months	1 week	81
<b>Researcher4</b>	2 months	1 week	1 week	10

<b>Researcher5</b>	2 weeks	4 weeks	2 months	14
<b>Researcher6</b>	3 weeks	2 weeks	1 week	6
<b>Average</b>	<b>18 weeks</b>	<b>5 weeks</b>	<b>4 weeks</b>	<b>26 weeks</b>

### 6.2.5 Statistical comparison of user study results and baseline survey results

Figure 6.3 shows the comparison of the mean time in minutes between the user study results (mean=42.8, SD=7.7) and the baseline survey results (mean=1260, SD= 1098.1). A Shapiro-Wilk analysis of the results data showed it was normally distributed. An independent T-Test of the results showed a significance difference between the time graduate students used in designing the user study (with VisUnit) compared to the time researchers expected graduate students to spend on designing the user study (p-value = 0.04). The effect size of the results is also large ( $r = 2.4$ ).

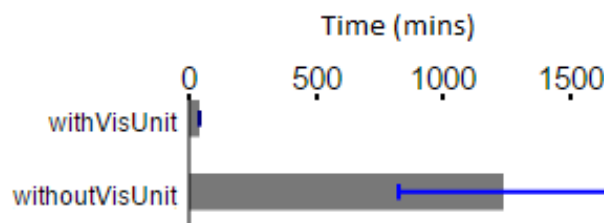


Figure 6.3 : A bar-chart showing the mean time of the user study results (withVisUnit) vs the mean time of the survey results (withoutVisUnit).



**Threats to Validity:** Although there is a significant difference between the time researchers expected the user study to be designed and the actual time graduate students designed the study with VisUnit, there are three main threats to the validity of our results.

First, we were comparing measured time performance of graduate students who performed our study to performance estimated by researchers who saw the designed study through a survey. It is possible that if we had given the researchers the instructions used by the graduate students in our study and told them to measure the time it will take them or their graduate students to design the study, the results could have been slightly different.

Second, in our lab-based studies, we were physically present to answer questions that graduate students might have on the functionalities of VisUnit. It is possible that if the graduate students were left to figure out some of the functionalities of VisUnit on their own, they could have taken longer to design the study.

Third, judging by the big difference between the time it took graduate students to create tasks and the time researchers expected them to take, it seems the researchers were factoring in the time it will take graduate students to deliberate and design the tasks correctly for the study. It is possible if we did not give users instructions on how to create the tasks, they could have taken a longer time to create the tasks. However, we think such a change in the study parameters would not have had a great impact on the current results. This is because, if we compare the results of our lab-based studies (*mean* = 42.8 min, *SD* = 7.7 min) only to the

time researchers estimated graduate students to develop the infrastructure (*mean* = 768 *min*, *SD* = 834.6*min*), the difference is still significant. Specifically, a Shapiro-Wilk analysis shows the result is not normally distributed, and a Wilcoxon-rank-sum test (*p-value* = 0.008, *effect-size* (*r*) = 2.7) shows researchers expected graduate students to take significantly longer time to design the study.

### **6.3 Using VisUnit to support research studies**

In this section, we describe an example of a research work that was significantly supported by VisUnit.

#### **6.3.1 Ecological validity in quantitative user studies – a case study in graph evaluation**

Quantitative user studies are too often judged by the magnitude of detected effects and basic soundness of their protocol, in detriment of their ecological validity. In this work, we show how considering ecological validity of a study's tasks, interactions, and data, can lead to important differences in evaluation outcomes and conclusions. Specifically, we revisit the highly cited study by Ghoniem et al. [28] which compared node-link diagrams (NLD) and adjacency matrices (AM), and found that for large graphs, AM performed better than the NLD in both accuracy and time for all of seven tasks. We discuss the ecological validity of the study within a formal framework, then show quantitatively that testing the

same fundamental ‘data-reading’ tasks but with slightly modified tasks and interactions can lead to different conclusions.

### 6.3.2 Original study

Ghoniem et al. [28] compared NLDs and AMs on seven graph tasks. Users had to: (1) estimate node count, (2) estimate edge count, (3) find the most connected node, and (4) find a node by its label. Given two selected nodes users had to: (5) find if they are connected, (6) find if they share a neighbor, and (7) find the path between them. Users could select multiple nodes and highlight another via mouse-over in both representations. Randomly generated graphs of three sizes (20, 50, 100) and densities (0.2, 0.4, and 0.6) were used. The AM was sorted lexicographically.

**The results of their study:** For small sparse graphs, NLD and AM were similar, but NLD was better in connectivity tasks (5,6,7). For large and dense graphs, AM outperformed NLD in all seven tasks.

### 6.3.3 A discussion of ecological validity

We formalize our discussion of ecological validity into a framework of five questions, which may be generalizable to evaluations beyond the current case study.

**Question 1:** Is the study using ecologically valid data? Real-life graph data rarely exhibits random topological structure. Moreover, an important benefit of graph visualizations is that they can reveal such structure. As such, random graphs may not be an ecologically valid choice of data.

**Question 2:** Is the presentation of the visualizations ecologically valid? Ghoniem et al. [28] used lexicographically ordered AMs. These support the tasks they evaluated well. For example, it is unsurprising that finding a node takes constant time in their AM, as this task reduces to scanning an ordered list of labels. However, lexicographic AMs do not reveal important topological properties, and may be used less often than those that reveal topological properties as illustrated in Figure 6.3

**Question 3:** Are the visualizations equivalent? We argue that NLDs are not equivalent to lexicographic AMs, since the first reveals structure while the second cannot. While it is true that the visualizations are equivalent for the subset of evaluated tasks, a complete answer needs to consider (1) how often are lexicographical AMs used, especially if topological ordering is also available, and (2) how often do users change AM ordering depending on their tasks. We believe the use of AM that expose topological structure (Figure 6.3) would have led to a more meaningful comparison.

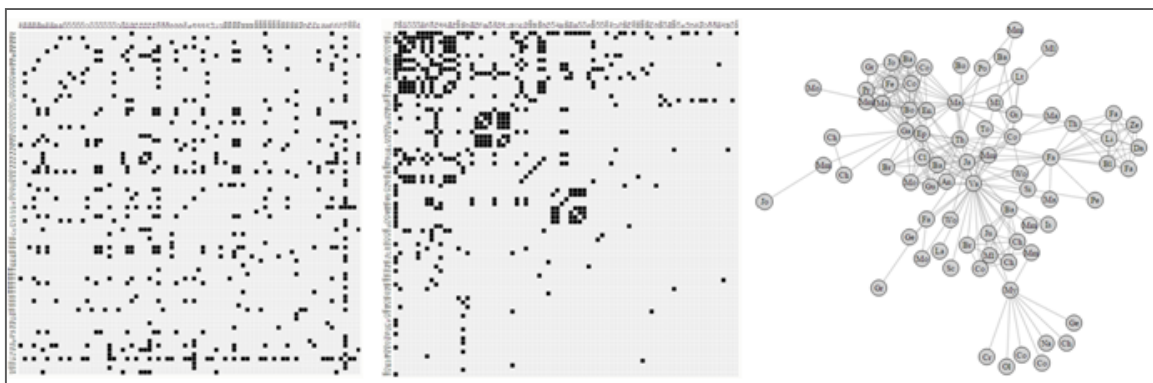


Figure 6.4 : Lexicographically ordered AMs (left) cannot reveal graph structure in the same way that a clustered AM (center) and an NLD (right) can

**Question 4:** Are the interactions equivalent? This is a difficult question because: (a) an interaction in one visualization may not have an equivalence in another; (b) the same interaction may aid each visualization in different ways and to different degrees.

For example, Ghoniem et al.'s [28] selection of nodes in the AM is not equivalent to the one in the NLD. As shown in Figure 6.3, the NLD allows users to easily read the neighbors of a selected node since they are exposed by their incident edges. This is more difficult in the matrix since a user has to trace from a dot vertically or horizontally through the matrix to reach its label, without any visual aid. As shown by the results of the study, this difference becomes important if the connectivity task is phrased differently.

Furthermore, an interaction that most NLDs implement is that of picking and moving nodes around. This interaction can often clarify where a selected node's edges end in a dense visualization. It does not however have an equivalence in the AM, and Ghoniem et al. [28] did not allow it in their NLD. As shown by the results of the study, the absence of this feature was the main reason of poor performances by Ghoniem et al.'s [28] users when using NLD on large graph connectivity tasks.

This raises an important question: *does adding this feature give an unfair advantage to NLD?* Introducing this feature should not give an unfair advantage to NLD because any interaction involves a cost in addition to a benefit. As long as

the interaction is useful and part of how the visualization is typically used, it is ecologically valid and should not be abstracted away.

**Question 5:** Are the chosen tasks, as presented to subjects, ecologically valid? As defined, this question has two components: (a) is the fundamental tasks valid, and (b) is the task phrasing valid?

For example, most people would agree that determining if two nodes are connected is a fundamental graph task. However, this task can be presented in many forms: users can be asked if a highlighted node is connected to another node; asked if two highlighted nodes are connected to each other; and asked if two unhighlighted nodes are connected to each other. These three scenarios are not equivalent because they involve different interactions and different overheads. Ghoniem et al. [28] evaluated the connectivity task by highlighting both nodes. We argue that their approach may be the least ecologically valid instantiation of this fundamental task because exploring a graph does not generally rely on pairwise node selections.

More generally, two questions can help quantify the ecological validity of a task: (a) how often do real users perform the task as phrased in the study?; (b) can a task be easily replaced by an equivalent, more efficient interaction or query? While the first question is somewhat evident, the second bears discussion. Ghoniem et al.'s [28] first three tasks could easily be implemented as graph queries, in the same way text editors offer functionality for counting words. Locating nodes by querying is also available in most visualization systems, and

finding a node should take constant time once the cost of visual search exceeds that of typing. More broadly, if a visual task can be replaced by a query that can be posed and computed faster, then the visual task may have limited ecological validity.

#### 6.3.4 User study

**Hypotheses:** We hypothesized that a user study following the aforementioned guidelines would yield different conclusions than those of Ghoniem et al. [28]. Specifically, we focused on two scenarios in which AM outperformed NLD: task 5 ('connectivity task') and task 6 ('common neighbor task'), both for large graphs. We made the following changes to Ghoniem et al.'s [28] study: (i) we used a real data set; (ii) we ordered the AM to reveal topological structure; (iii) we allowed users to drag nodes in NLDs; (iv) we created two versions of task 5: one using the original phrasing, in which both nodes are selected (5a), and a new task in which one node is selected while the other is named by its label (5b). Our hypotheses were that given these changes:

**H1:** NLD will outperform AM for both tasks 5a and 5b, even for large graphs.

Reason: moving nodes allows NLD users to better see where edges end.

**H2:** AM will perform worse on 5b than 5a. Reason: the AM selection, as implemented, is less powerful than the NLD one.

**H3:** NLD will outperform AM for task 6, even for large graphs. Reason: moving nodes allows NLD users to better see where edges end.

**Protocol and delivery:** We designed a 2 visualizations x 3 tasks between-groups study, and used VisUnit [94] to run it online via Amazon Mechanical Turk (MT). We drew the NLD using D3’s generic forced directed method, and we ordered the AM using public D3 code. The underlying data was a graph of 100 nodes with a link density of 0.2 and derived from a book recommendation dataset. We changed the book names to match the simplified nomenclature used by Ghoniem et al. [28] (i.e., A0..F9). We provided the same interactions as Ghoniem et al. [28], namely node selection and node highlighting (Figure 6.5), and added node dragging in NLD. Formally, we evaluated the following tasks:

**5a:** Given two highlighted nodes, determine if they are connected.

**5b:** Given one highlighted node and the label of a second, determine if they are connected.

**6:** Given two highlighted nodes, determine if they have a common neighbor.

Following an introduction, subjects trained on five instances of each task type (15 training tasks), then completed the study with another five instances of each type (15 actual tasks). To minimize boredom and learning effects between the three evaluated tasks, we alternated the order in which we presented them to users. We recruited a total of 90 Mechanical Turk (MT) users, 45 for each of the two visualizations.



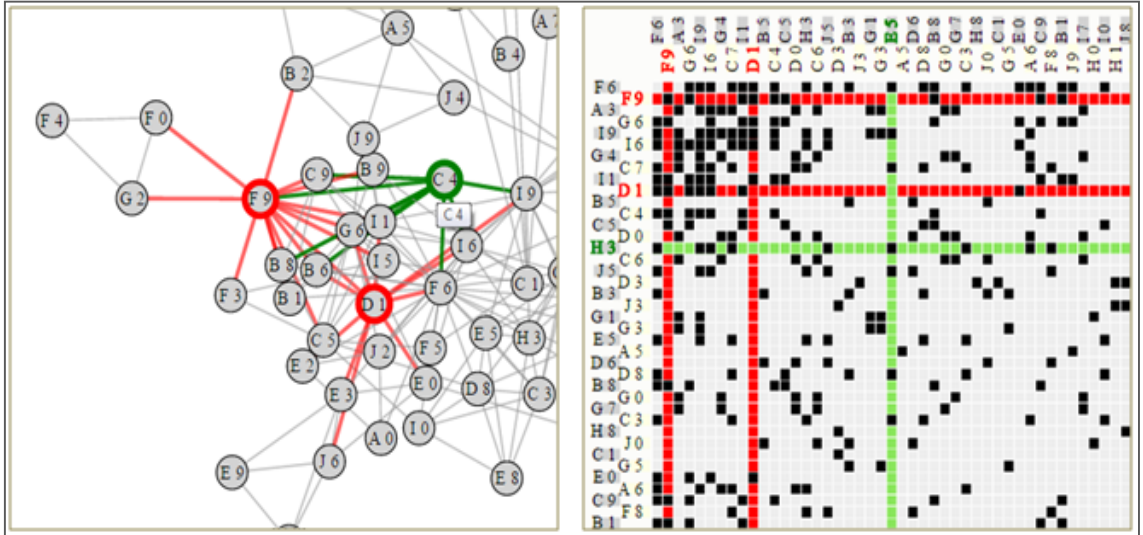


Figure 6.5 : Available interactions: hovering/selecting a node highlights it and its edges green/red; hovering a link highlights it and its end-points green. The images illustrate a task 5a instance

### 6.3.5 Results

A Shapiro-Wilk analysis of our users' time and accuracy showed it was not normally distributed. We thus used a Wilcoxon-rank-sum test to analyze both time and accuracy. Our results were different from those of Ghoniem et al. [28]. We found that participants that used NLD were more accurate than those that used AM for task 5a ( $p < 0.001$ ), more accurate than those that used AM for task 5b ( $p < 0.001$ ), and faster than those that used AM for task 5b ( $p = 0.002$ ). This confirms the hypotheses H1 and H2. Finally, participants that used NLD were significantly more accurate than those that used AM for task 6 ( $p < 0.001$ ), thus confirming hypothesis H3.

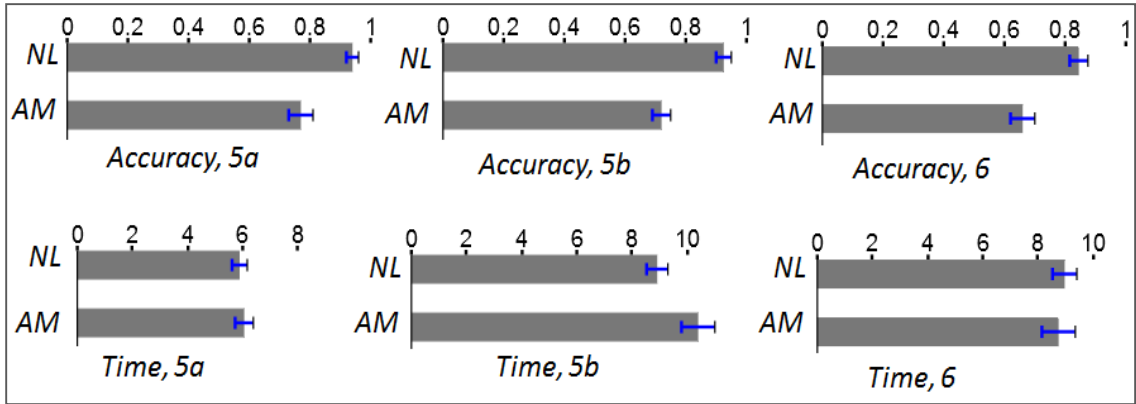


Figure 6.6 : Accuracy and time results for the three tasks.

Our contributions are in three-fold. First, we provide a framework for discussing the ecological validity of visualization user studies, and demonstrate its applicability in a case study. Second, we show how even small changes in study setup can lead to different outcomes. Third, we explain some of Ghoniem et al.'s [28] surprising results (e.g., inability to move nodes determined the lower NLD performance), and end up with a different recommendation: NLDs are significantly better for all evaluated topological tasks, regardless of graph size and density.

#### 6.4 Summary

In this chapter, we evaluated the effectiveness of VisUnit by showing that it can be used to replicate a wide range of user studies. This evaluation validates the effectiveness of VisUnit and achieves our research goal of designing a framework that supports a wide range of user studies.

We also evaluated the efficiency of VisUnit by performing a user study involving 5 graduate students who are familiar with web based visualization and are unaffiliated to this research project. Study participants were able to design user studies using publicly available code in less than an hour. This evaluation validates the efficiency of VisUnit and achieves our research goal of designing a framework that facilitates user studies and saves evaluators time. We also show the potential of VisUnit to support research by describing an example of a research work that was supported by VisUnit.

## 7 CONCLUSION AND FUTURE WORK

In this chapter, we provide a summary of the research presented in this dissertation, discuss our contributions to the field of visualization and discuss lessons learned as well as future research directions that can build upon and expand our work.

### 7.1 Summary

User studies play an important role in information visualization research to validate research findings. This is because they help choose appropriate visual encodings and techniques out of a wide array of competing options. However, conducting user studies is challenging, time consuming, and expensive. The research problem we investigated in this dissertation is *how to design an online framework that can reduce the overhead involved in conducting web-based controlled visualization user studies*.

We provide the design of a framework that reduces the overhead involved in conducting user studies and facilitates a wide range of user studies in information visualization. In the first phase of this dissertation, we investigated *how to design a framework to facilitate user studies involving graph networks*, a smaller problem that helped us test hypotheses on a smaller scale. To this end, we gathered requirements on graph user studies, designed, implemented and evaluated an open-source framework (GraphUnit) that facilitates graph user studies by semi-automating the design, run, and result analyses of graph user studies.

In the second phase of this dissertation, we investigated how to facilitate a wide range of visualization user studies. To this end, we gathered additional requirements on standard processes and fundamental properties of user studies involving visualization types such as multidimensional, temporal, tree, and 2D area. We then designed and implemented VisUnit, a framework that is flexible, user-friendly, and facilitates a wide range of user studies involving data visualizations. We evaluated VisUnit's effectiveness by demonstrating that it can be used to replicate a wide range of previous user studies and can be a supporting tool in new research as well. We demonstrated VisUnit's efficiency by showing that evaluators unaffiliated to this project were able to use VisUnit to design user studies involving publicly available visualizations and data in less than an hour.

## **7.2 Lessons learned in this research**

Over the course of working on this research, we have learnt that a lot of decision making goes into conducting user studies and a framework that facilitates the processes involved in running user studies helps save valuable time for evaluators. In the following paragraphs, we provide answers to the high-level questions we raised in Chapter 3 to validate the success of this dissertation.

*How can we design a framework that is user-friendly and flexible to semi-automate web-based user studies of many visualization types and task types?":* Although evaluators make diverse decisions during the design and organization of user studies, a considerable number of user studies in information visualization follow a standard protocol with processes that include: (1) providing introduction,

(2) getting demographic data and asking pre-study qualitative questions, (3) performing standard tests such as color-blindness tests, (4) training users with sample instances of study tasks, (5) presenting users with multiple instances of each task and collecting user responses, and (6) asking post-study qualitative questions (7) Using statistics to analyze study results. These processes can be automated.

To have a flexible framework that supports a wide range of user studies, we need a framework design that provides separation between the required processes, the visualizations being evaluated, the datasets used, the tasks, and other resources. The framework design can therefore act like a black-box, where evaluators provide their inputs (i.e. required processes, visualizations, datasets, tasks, task instances, etc.), and it presents a designed user study which can be managed semi-automatically from the first process to the last process in the user study.

*Can this framework be effective and efficient to facilitate a wide range of user studies?* This framework design is effective to facilitate a wide range of user studies involving data visualizations. We demonstrated that our framework can effectively be used to replicate a wide range of user studies involving data visualizations and demonstrate that evaluators were able to use our framework to design user studies in less than an hour.

### 7.3 Contributions

The following are the key contributions from the thesis.

1. *The design of a framework that can reduce the overhead involved in web-based user studies:* We explored requirements for our design from previous user studies in the information visualization literature, explored technologies that can be leveraged, and designed and implemented a prototype that facilitates graph user studies. We extended our prototype and designed and implemented VisUnit an open source framework and an online service that can provide real time support and time saving functionalities for user study evaluators.

2. *Evaluation of VisUnit effectiveness and efficiency:* We demonstrate the effectiveness of VisUnit by showing that it can be used to replicate a wide range of existing user studies. Specifically, we show that it can be used to replicate 84% of 101 controlled user studies published in IEEE InfoVis conference between 1995 and 2015. These user studies involve graphs, multidimensional visualizations, trees, 2D areas, and temporal visualizations. We also demonstrate the efficiency of our design by showing that evaluators can use VisUnit to design user studies in less than an hour; and run studies and analyze study results within a day. Specifically, we conducted a user study involving 5 computer science graduate students. Students were asked to design user studies to evaluate competing visualization techniques. On average, it took participants less than an hour to design a study.

## 7.4 Future directions

There are several research directions that can build and expand on this work. We discuss these ideas here.

*Measuring beyond time and error.* The design of VisUnit focused on measuring accuracy and time data. However, user studies can measure other variables beyond time and error. For example, in addition to accuracy and time, user studies can measure the number of clicks, and the number of answer changes that users had. Future studies should look at supporting measurements beyond time and error. The nature of web-based studies also poses *challenges* to measurements, for example, participants may be using different devices with different processing power (such as Smart phones, tablets, lap-tops, and desk-tops), or participants may have different screen size and screen resolutions. The differences between these devices can influence the results of studies. It will be interesting to investigate how web-based studies are impacted by the devices of participants, and how measurements can be adjusted based on devices of participants. One strategy that can be used to address some of these challenges will be to allow evaluators to specify a minimum or maximum requirement for hardware, screen size, and screen resolution, such that, participants who do not meet the requirements will be prevented from taking the study.

*Supporting Eye tracking and think aloud protocols:* Eye tracking studies are becoming common. The cameras of devices used by study participants can be employed for basic eye-tracking studies. Similarly, the microphones of devices used by participants can be used for think-aloud studies. Future studies should



investigate how to support web-based user studies that involve eye tracking and think-aloud protocols and how to measure such data remotely.

*Supporting reusable designs and results:* A lot of comparative user studies that evaluators perform can benefit from previous user studies. Future work should investigate how to standardize user studies, and allow evaluators to store their user study designs and results in publicly accessible repositories. This will provide opportunities for researchers to easily extend the work of other researchers using new tasks and additional variables. Moreover, this will allow researchers to leverage existing user studies for new comparative studies.

*Supporting participant profiles and qualifications:* It will be useful to keep track of participants who perform tasks, and remember qualifications that they have taken before. This way, evaluators can be able to use participants who meet certain qualifications, without the need for those participants to repeat those qualifications again. This can considerably reduce durations of studies. For example, evaluators should be able to easily perform user studies with participants that have normal vision, participants that are not color-blind or participants that meet a given criteria. Keeping profiles of study participants will also provide an opportunity for evaluators to perform longitudinal studies with specific groups of users. Future work should look at tracking participant profiles and qualifications.

*Detecting malicious behavior automatically:* Some web-based and crowdsourced workers do not take tasks seriously. The actions of such workers cost evaluators time and money. Evaluators spend additional time filtering results,

and lose money paid to such malicious users. Instead of waiting for such malicious users to finish the studies, it will be useful to prevent such users from completing the studies. Future work should investigate methods to automatically detect malicious users during the training stage or detect them early on in the study and prevent them from continuing the study.

*Supporting other categories of user studies:* This work focused on controlled user studies, however there are other categories of user studies that future work can support. For example, there is a growing interest in insight-based user studies [95, 96], as Shneiderman puts it aptly "The purpose of visualization is insight, not pictures" [97]. Future work should investigate the feasibility of running insight-based studies with web-based participants, and investigate the incentives such studies will require. Future work should also investigate how to provide infrastructure to support web-based usability studies, insight-based studies, and ethnographic studies.

## REFERENCES

- [1] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.
- [2] Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM, 2004.
- [3] Melanie Tory and Torsten Moller. Human factors in visualization research. *Visualization and Computer Graphics, IEEE Transactions on*, 10(1):72–84, 2004.
- [4] Chaomei Chen and Yue Yu. Empirical studies of information visualization: a meta-analysis. *International Journal of Human-Computer Studies*, 53(5):851–866, 2000.
- [5] Robert Kosara, Christopher G Healey, Victoria Interrante, David H Laidlaw, and Colin Ware. User studies: Why, how, and when? *IEEE Computer Graphics and Applications*, 23(4):20–25, 2003.
- [6] Geoffrey Ellis and Alan Dix. An explorative analysis of user evaluation studies in information visualisation. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–7. ACM, 2006.
- [7] Heidi Lam, Enrico Bertini, Petra Isenberg, Catherine Plaisant, and Sheelagh Carpendale. Empirical studies in information visualization: Seven scenarios. *Visualization and Computer Graphics, IEEE Transactions on*, 18(9):1520–1536, 2012.
- [8] Keith Andrews. Evaluating information visualisations. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–5. ACM, 2006.

- [9] Daniel A Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. *Mastering the information age-solving problems with visual analytics*. Florian Mansmann, 2010.
- [10] James J Thomas. *Illuminating the path:[the research and development agenda for visual analytics]*. IEEE Computer Society, 2005.
- [11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D<sup>3</sup> data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.
- [12] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–212. ACM, 2010.
- [13] Robert Kosara and Caroline Ziemkiewicz. Do mechanical turks dream of square pie charts? In *Proceedings of the 3rd BELIV'10 Workshop: BEyond time and errors: novel evaluation methods for Information Visualization*, pages 63–70. ACM, 2010.
- [14] Nadia Boukhelifa, Anastasia Bezerianos, Tobias Isenberg, and J Fekete. Evaluating sketchiness as a visual variable for the depiction of qualitative uncertainty. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2769–2778, 2012.
- [15] Radu Jianu, Adrian Rusu, Yifan Hu, and Douglas Taggart. How to display group information on node-link diagrams: an evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 20(11):1530–1541, 2014.
- [16] Tamara Munzner. A nested model for visualization design and validation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):921–928, 2009.

- [17] Sheelagh Carpendale. Evaluating information visualizations. In *Information Visualization*, pages 19–45. Springer, 2008.
- [18] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–5. ACM, 2006.
- [19] Bahador Saket, Paolo Simonetto, and Stephen Kobourov. Group-level graph visualization taxonomy. *arXiv preprint arXiv:1403.7421*, 2014.
- [20] Eliane RA Valiati, Marcelo S Pimenta, and Carla MDS Freitas. A taxonomy of tasks for guiding the evaluation of multidimensional visualizations. In *Proceedings of the 2006 AVI workshop on Beyond time and errors: novel evaluation methods for information visualization*, pages 1–6. ACM, 2006.
- [21] Jae wook Ahn, Catherine Plaisant, and Ben Shneiderman. A task taxonomy for network evolution analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):365–376, 2014.
- [22] Jean-Daniel Fekete and Catherine Plaisant. General tasks applicable to most trees. <http://www.cs.umd.edu/hcil/iv03contest/generaltasks.html>, 2003. Online; accessed 7-January-2015.
- [23] Catherine Plaisant, J Fekete, and Georges Grinstein. Promoting insight-based evaluation of visualizations: From contest to benchmark repository. *Visualization and Computer Graphics, IEEE Transactions on*, 14(1):120–134, 2008.
- [24] Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J van Wijk, J-D Fekete, and Dieter W Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. In *Computer graphics forum*, volume 30, pages 1719–1749. Wiley Online Library, 2011.

- [25] Adam Perer and Ben Shneiderman. Integrating statistics and visualization for exploratory power: From long-term case studies to design guidelines. *IEEE Computer Graphics and Applications*, 29(3):39–51, 2009.
- [26] I Scott MacKenzie. *Human-computer interaction: An empirical research perspective*. Newnes, 2012.
- [27] Danny Holten and Jarke J van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2299–2308. ACM, 2009.
- [28] Mohammad Ghoniem, J Fekete, and Philippe Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 17–24. IEEE, 2004.
- [29] David H Laidlaw, Robert M Kirby, Cullen D Jackson, J Scott Davidson, Timothy S Miller, Marco Da Silva, William H Warren, and Michael J Tarr. Comparing 2d vector field visualization methods: A user study. *Visualization and Computer Graphics, IEEE Transactions on*, 11(1):59–70, 2005.
- [30] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1303–1312. ACM, 2009.
- [31] Ray E Eberts. *User interface design*. Prentice-Hall, Inc., 1994.
- [32] Schuyler W Huck, William H Cormier, and William G Bounds. *Reading statistics and research*. Harper & Row New York, 1974.
- [33] Helen C Purchase. Performance of layout algorithms: Comprehension, not computation. *Journal of Visual Languages & Computing*, 9(6):647–657, 1998.

- [34] Helen C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interacting with computers*, 13(2):147–162, 2000.
- [35] Helen C Purchase, Eve Hoggan, and Carsten Görg. How important is the "mental map"?—an empirical investigation of a dynamic graph layout algorithm. In *Graph drawing*, pages 184–195. Springer, 2006.
- [36] Helen C. Purchase, Robert F. Cohen, and Murray I James. An experimental study of the basis for graph drawing algorithms. *Journal of Experimental Algorithmics (JEA)*, 2:4, 1997.
- [37] Jibonananda Sanyal, Song Zhang, Gargi Bhattacharya, Phil Amburn, and Robert J Moorhead. A user study to compare four uncertainty visualization methods for 1d and 2d datasets. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1209–1218, 2009.
- [38] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. Effectiveness of animation in trend visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1325–1332, 2008.
- [39] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 111–117. IEEE, 2005.
- [40] Stephen Wehrend and Clayton Lewis. A problem-oriented classification of visualization techniques. In *Proceedings of the 1st Conference on Visualization'90*, pages 139–143. IEEE Computer Society Press, 1990.
- [41] Michelle X Zhou and Steven K Feiner. Visual task characterization for automated visual discourse synthesis. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 392–399. ACM Press/Addison-Wesley Publishing Co., 1998.

- [42] Ron Kohavi, Randal M Henne, and Dan Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 959–967. ACM, 2007.
- [43] Kimberly Ling, Gerard Beenen, Pamela Ludford, Xiaoqing Wang, Klarissa Chang, Xin Li, Dan Cosley, Dan Frankowski, Loren Terveen, Al Mamunur Rashid, et al. Using social psychology to motivate contributions to online communities. *Journal of Computer-Mediated Communication*, 10(4):00–00, 2005.
- [44] Robert Kraut, Judith Olson, Mahzarin Banaji, Amy Bruckman, Jeffrey Cohen, and Mick Couper. Psychological research online: report of board of scientific affairs' advisory group on the conduct of research on the internet. *American psychologist*, 59(2):105, 2004.
- [45] Matthew J Salganik and Duncan J Watts. Web-based experiments for the study of collective social dynamics in cultural markets. *Topics in Cognitive Science*, 1(3):439–468, 2009.
- [46] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.
- [47] James Surowiecki. *The wisdom of crowds*. Anchor, 2005.
- [48] Yochai Benkler. Coase's penguin, or, linux and" the nature of the firm". *Yale Law Journal*, pages 369–446, 2002.
- [49] Jeff Barr and Luis Felipe Cabrera. Ai gets a brain. *Queue*, 4(4):24–29, 2006.
- [50] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.



- [51] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.
- [52] Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. 2008.
- [53] Omar Alonso, Daniel E Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, volume 42, pages 9–15. ACM, 2008.
- [54] Rion Snow, Brendan O’Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics, 2008.
- [55] Winter Mason and Siddharth Suri. Conducting behavioral research on amazon’s mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.
- [56] Steven Komarov, Katharina Reinecke, and Krzysztof Z Gajos. Crowdsourcing performance evaluations of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 207–216. ACM, 2013.
- [57] Casey Reas and Ben Fry. *Processing: a programming handbook for visual designers and artists*, volume 6812. Mit Press, 2007.
- [58] Marian Dork, Nathalie Henry Riche, Gonzalo Ramos, and Susan Dumais. Pivotpaths: Strolling through faceted information spaces. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2709–2718, 2012.
- [59] Fernanda B Viegas, Martin Wattenberg, Frank Van Ham, Jesse Kriss, and Matt McKeon. Manyeyes: a site for visualization at internet scale.

*Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1121–1128, 2007.

- [60] Trevor M O'Brien, Anna M Ritz, Benjamin J Raphael, and David H Laidlaw. Gremlin: an interactive visualization model for analyzing genomic rearrangements. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):918–926, 2010.
- [61] Wendy E Mackay, Caroline Appert, Michel Beaudouin-Lafon, Olivier Chapuis, Yangzhou Du, Jean-Daniel Fekete, and Yves Guiard. Touchstone: exploratory design of experiments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1425–1434. ACM, 2007.
- [62] Keith Andrews and Janka Kasanicka. A comparative study of four hierarchy browsers using the hierarchical visualisation testing environment (hvte). In *Information Visualization, 2007. IV'07. 11th International Conference*, pages 81–86. IEEE, 2007.
- [63] Wolfgang Aigner, Stephan Hoffmann, and Alexander Rind. Evalbench: a software library for visualization evaluation. In *Computer Graphics Forum*, volume 32, pages 41–50. Wiley Online Library, 2013.
- [64] Lane Harrison. experimentr. <https://github.com/codementum/experimentr>, 2014. [Online; accessed 15-December-2014].
- [65] C Papadopoulos, I Gutenko, and AE Kaufman. Veevvie: Visual explorer for empirical visualization, vr and interaction experiments. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):111–120, 2016.
- [66] Eytan Bakshy, Dean Eckles, and Michael S Bernstein. Designing and deploying online field experiments. In *Proceedings of the 23rd international conference on World wide web*, pages 283–292. ACM, 2014.

- [67] David C Parkes, Andrew Mao, Yiling Chen, Krzysztof Z Gajos, Ariel Procaccia, and Haoqi Zhang. Turkserver: Enabling synchronous and longitudinal online experiments. In *Proceedings of the Fourth Workshop on Human Computation (HCOMP'12)*. AAAI Press, 2012.
- [68] Michael Nebeling, Maximilian Speicher, and Moira C Norrie. Crowdstudy: general toolkit for crowdsourced evaluation of web interfaces. In *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 255–264. ACM, 2013.
- [69] Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 57–66. ACM, 2010.
- [70] Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 61–72. ACM, 2011.
- [71] Jacques Bertin. *Semiology of graphics: diagrams, networks, maps*. 1983.
- [72] Stuart K Card and Jock Mackinlay. The structure of the information visualization design space. In *Information Visualization, 1997. Proceedings., IEEE Symposium on*, pages 92–99. IEEE, 1997.
- [73] Andrew Walenstein. *Cognitive support in software engineering tools: A distributed cognition framework*. PhD thesis, Citeseer, 2002.
- [74] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996.

- [75] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.
- [76] Stephen P Borgatti, Ajay Mehra, Daniel J Brass, and Giuseppe Labianca. Network analysis in the social sciences. *science*, 323(5916):892–895, 2009.
- [77] George Chin, Daniel G Chavarria, Grant C Nakamura, and Heidi J Sofia. Biographe: high-performance bionetwork analysis using the biological graph environment. *BMC bioinformatics*, 9(Suppl 6):S6, 2008.
- [78] Emden R Gansner and Stephen C North. An open graph visualization system and its applications to software engineering. *Software Practice and Experience*, 30(11):1203–1233, 2000.
- [79] Michael Sedlmair, Miriah Meyer, and Tamara Munzner. Design study methodology: Reflections from the trenches and the stacks. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2431–2440, 2012.
- [80] Christopher Collins, Gerald Penn, and Sheelagh Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1009–1016, 2009.
- [81] Basak Alper, Nathalie Henry Riche, Gonzalo Ramos, and Mary Czerwinski. Design study of linesets, a novel set visualization technique. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2259–2267, 2011.
- [82] Maggie E Toplak, Richard F West, and Keith E Stanovich. The cognitive reflection test as a predictor of performance on heuristics-and-biases tasks. *Memory & Cognition*, 39(7):1275–1289, 2011.

- [83] Hans-Jorg Schulz, Thomas Nocke, Magnus Heitzler, and Heidrun Schumann. A design space of visualization tasks. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2366–2375, 2013.
- [84] Jacob Cohen. Statistical power analysis for the behavioral sciences. vol. 2. Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.
- [85] Robert Rosenthal. *Meta-analytic procedures for social research*, volume 6. Sage, 1991.
- [86] Bruce Thompson. *Foundations of behavioral statistics: An insight-based approach*. Guilford Press, 2006.
- [87] Helen Purchase. Which aesthetic has the greatest effect on human understanding? In *Graph Drawing*, pages 248–261. Springer, 1997.
- [88] Steve Haroz and David Whitney. How capacity limits of attention influence information visualization effectiveness. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2402–2410, 2012.
- [89] Waqas Javed, Bryan McDonnel, and Niklas Elmqvist. Graphical perception of multiple time series. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):927–934, 2010.
- [90] N Henry, Anastasia Bezerianos, and Jean-Daniel Fekete. Improving the readability of clustered social networks using node duplication. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1317–1324, 2008.
- [91] Caroline Ziemkiewicz, R Jordan Crouser, Ashley Rye Yauilla, Sara L Su, William Ribarsky, and Remco Chang. How locus of control influences compatibility with visualization style. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 81–90. IEEE, 2011.

- [92] Zhen Wen and Michelle X Zhou. Evaluating the use of data transformation for information visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1309–1316, 2008.
- [93] Bahador Saket, Paolo Simonetto, Stephen Kobourov, and Kai Borner. Node, node-link, and node-link-group diagrams: An evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2231–2240, 2014.
- [94] Mershack Okoe and Radu Jianu. Graphunit: Evaluating interactive graph visualizations using crowdsourcing. *Computer Graphics Forum (CGF)*, Vol. 34, No. 3, pp. 451-460, 2015.
- [95] Purvi Saraiya, Chris North, and Karen Duca. An evaluation of microarray visualization tools for biological insight. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 1–8. IEEE, 2004.
- [96] Chris North. Toward measuring visualization insight. *Computer Graphics and Applications, IEEE*, 26(3):6–9, 2006.
- [97] Ben Shneiderman. Research agenda: Visual overviews for exploratory search. *Information Seeking Support Systems*, 11:4, 2008.
- [98] Michelle A Borkin, Krzysztof Z Gajos, Amanda Peters, Dimitrios Mitsouras, Simone Melchionna, Frank J Rybicki, Charles L Feldman, and Hanspeter Pfister. Evaluation of artery visualizations for heart disease diagnosis. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2479–2488, 2011.
- [99] Alfred Kobsa. User experiments with tree visualization systems. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 9–16. IEEE, 2004.

- [100] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1129–1136, 2007.
- [101] Bongshin Lee, Nathalie Henry Riche, Amy K Karlson, and Sheelagh Carpendale. Sparkclouds: Visualizing trends in tag clouds. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1182–1189, 2010.
- [102] Catherine Plaisant, Jesse Grosjean, and Benjamin B Bederson. Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 57–64. IEEE, 2002.
- [103] Frank Van Ham and Jarke J van Wijk. Beamtrees: Compact visualization of large hierarchies. *Information Visualization*, 2(1):31–39, 2003.
- [104] Nelson Wong, Sheelagh Carpendale, and Saul Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 51–58. IEEE, 2003.
- [105] Markus Steinberger, Manuela Waldner, Marc Streit, Alexander Lex, and Dieter Schmalstieg. Context-preserving visual links. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2249–2258, 2011.
- [106] Greg Smith, Mary Czerwinski, Brian Meyers, Daniel Robbins, George Robertson, and Desney S Tan. Facetmap: A scalable search and browse visualization. *IEEE Transactions on visualization and computer graphics*, 12(5):797–804, 2006.
- [107] Jeffrey Heer and George Robertson. Animated transitions in statistical data graphics. *IEEE transactions on visualization and computer graphics*, 13(6):1240–1247, 2007.

- [108] Caroline Ziemkiewicz and Robert Kosara. The shaping of information by visual metaphors. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1269–1276, 2008.
- [109] Edward Clarkson, Krishna Desai, and James Foley. Resultmaps: Visualization for search interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1057–1064, 2009.
- [110] Nan Cao, Jimeng Sun, Yu-Ru Lin, David Gotz, Shixia Liu, and Huamin Qu. Facetatlas: Multifaceted visualization for rich text corpora. *IEEE transactions on visualization and computer graphics*, 16(6):1172–1181, 2010.
- [111] Kyle Koh, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. Maniwordle: Providing flexible control over wordle. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1190–1197, 2010.
- [112] Nicholas Kong, Jeffrey Heer, and Maneesh Agrawala. Perceptual guidelines for creating rectangular treemaps. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):990–998, 2010.
- [113] Nan Cao, David Gotz, Jimeng Sun, and Huamin Qu. Dicon: Interactive visual analysis of multidimensional clusters. *IEEE transactions on visualization and computer graphics*, 17(12):2581–2590, 2011.
- [114] Luana Micallef, Pierre Dragicevic, and J Fekete. Assessing the effect of visualizations on bayesian reasoning through crowdsourcing. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2536–2545, 2012.
- [115] Sébastien Rufiange and Michael J McGuffin. Diffani: Visualizing dynamic graphs with a hybrid of difference maps and animation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2556–2565, 2013.



- [116] Lane Harrison, Fumeng Yang, Steven Franconeri, and Remco Chang. Ranking visualizations of correlation using weber's law. *IEEE transactions on visualization and computer graphics*, 20(12):1943–1952, 2014.
- [117] Jeromy Carriere and Rick Kazman. Research report. interacting with huge hierarchies: beyond cone trees. In *Information Visualization, 1995. Proceedings.*, pages 74–81. IEEE, 1995.
- [118] Purvi Saraiya, Chris North, and Karen Duca. An evaluation of microarray visualization tools for biological insight. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 1–8. IEEE, 2004.
- [119] Sabrina Bresciani and Martin J Eppler. The benefits of synchronous collaborative information visualization: Evidence from an experimental evaluation. *IEEE transactions on visualization and computer graphics*, 15(6):1073–1080, 2009.
- [120] Trevor M O'Brien, Anna M Ritz, Benjamin J Raphael, and David H Laidlaw. Gremlin: an interactive visualization model for analyzing genomic rearrangements. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):918–926, 2010.
- [121] Michael Burch, Natalia Konevtsova, Julian Heinrich, Markus Hoeflerlin, and Daniel Weiskopf. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2440–2448, 2011.
- [122] Basak Alper, Tobias Hollerer, JoAnn Kuchera-Morin, and Angus Forbes. Stereoscopic highlighting: 2d graph visualization on stereo displays. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2325–2333, 2011.

- [123] Johnny Rodgers and Lyn Bartram. Exploring ambient and artistic visualization for residential energy use feedback. *IEEE transactions on visualization and computer graphics*, 17(12):2489–2497, 2011.
- [124] Andrew Vande Moere, Martin Tomitsch, Christoph Wimmer, Boesch Christoph, and Thomas Grechenig. Evaluating the effect of style in information visualization. *IEEE transactions on visualization and computer graphics*, 18(12):2739–2748, 2012.
- [125] Kim Marriott, Helen Purchase, Michael Wybrow, and Cagatay Goncu. Memorability of visual features in network diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2477–2485, 2012.
- [126] Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. Sketchstory: Telling more engaging stories with data through freeform sketching. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2416–2425, 2013.
- [127] Eirik Bakke, David R Karger, and Robert C Miller. Automatic layout of structured hierarchical reports. *IEEE transactions on visualization and computer graphics*, 19(12):2586–2595, 2013.
- [128] Mikkel R Jakobsen, Yonas Sahlemariam Haile, Søren Knudsen, and Kasper Hornbæk. Information visualization and proxemics: design opportunities and empirical findings. *IEEE transactions on visualization and computer graphics*, 19(12):2386–2395, 2013.
- [129] Rudolf Netzel, Michel Burch, and Daniel Weiskopf. Comparative eye tracking study on node-link visualizations of trajectories. *IEEE transactions on visualization and computer graphics*, 20(12):2221–2230, 2014.
- [130] Zhicheng Liu and Jeffrey Heer. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20(12):2122–2131, 2014.

- [131] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics*, 22(1):649–658, 2016.
- [132] Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. Hola: Human-like orthogonal network layout. *IEEE transactions on visualization and computer graphics*, 22(1):349–358, 2016.
- [133] John Stasko, Jaegul Choo, Yi Han, Mengdie Hu, Hannah Pileggi, Ramik Sadana, and Charles D Stolper. Citevis: Exploring conference paper citation data visually. *Posters of IEEE InfoVis*, 2, 2013.
- [134] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

## VITA

### MERSHACK BORTEY OKOE

- 2016            Doctoral Candidate, Computer Science  
Florida International University  
Miami, Florida
- Research: Supporting Web-based and Crowdsourced Evaluations of Data Visualizations.*
- 2014            M.S, Computer Science  
Florida International University  
Miami, Florida
- 2009            M.S, Advanced Computing  
University of Bristol  
Bristol, United Kingdom
- Research: Visualizing Structural Classifications of Protein (SCOP) domains in web-browsers without plug-ins*
- 2007            B.S, Computer Science  
Kwame Nkrumah University of Science and Technology  
Kumasi, Ghana

### PUBLICATIONS PRESENTATIONS AND ABSTRACTS

Mershack Okoe, and Radu Jianu. GraphUnit: Evaluating Interactive Graph Visualizations Using Crowdsourcing. *In Eurovis2015 and Computer Graphics Forum Journal (CGF)*, Vol. 34, No. 3, pp. 451-460, 2015.

Mershack Okoe and Radu Jianu. Ecological Validity in Quantitative User Studies - a Case Study in Graph Evaluation. *In IEEE VIS 2015 poster program*, 2015.

Mershack Okoe, Sayeed Safayet, and Radu Jianu. A Gaze-enabled Graph Visualization to Improve Graph Reading Tasks. *In Eurovis2014 and Computer Graphics Forum Journal (CGF)*, Vol. 33, No. 3, pp. 251-260, 2014.

Mershack Okoe and Radu Jianu. GraphUnit: Evaluating Interactive Graph Visualization Using CrowdSourcing. *In IEEE VIS 2014 poster program*, 2014.

Mershack Okoe, Sayeed Safayet, and Radu Jianu. Using Eye-Tracking as Interactive Input Enhances Graph Visualization. *In IEEE Visualization 2013 Poster Compendium*, 2013.