# Lowering the Barriers to Hypothesis-Driven Data Science

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Krishna Prasath Subramanian, M.Sc.

aus Coimbatore, India

Berichter: Prof. Dr. Jan Borchers
Prof. Dr. Paul Cairns

Tag der mündlichen Prüfung: 17. Juni 2022

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

*Aachen, January 15, 2023*
*Krishna Subramanian*

# Contents

# List of Figures

# List of Tables

# Abstract

Data science is a frequent task in academia and industry. One common use of data science is to validate hypotheses, in which the analyst uses significance-based hypothesis testing to draw insights about a population distribution based on experimental data. Apart from data scientists, who are professionally trained in data science and have high skills levels, many non-professional analysts also carry out data analysis. These non-professionals, who we refer to as *data workers,* are domain experts who lack expertise in data science, such as academic researchers, project managers, and sales managers.

Through interviews, observations, online surveys, and content analyses, we aim to understand data workers' workflows across important tasks in hypothesis testing: learning theoretical and practical statistics, selecting statistical procedures, using data science programming IDEs to experiment with ideas in source code, refine and refactor source code, and disseminating findings from an analysis. We present our findings grouped into two steps when performing data science tasks:

1. **Preparing to perform data science tasks:** We discuss our findings about the impact of formal training on real-world statistical practice; trade-offs among information sources used for selecting statistical procedures; perceived complexity and uncertainty about statistical procedure selection; and reluctance among data workers to adopt alternative methods of analysis.

   Based on the above findings, we present design recommendations and one artifact to improve data workers' workflows. Our artifact *StatPlayground* is an interactive simulation tool that can be used to self-learn or teach statistical concepts and statistical procedure selection.

2. **Performing data science tasks:** Our findings include an overview of data workers' workflows when performing hypothesis testing using programming IDEs, which follows an *exploratory programming* workflow; and a comparison of existing interfaces for data science programming, namely computational notebooks, scripts, and consoles, and a discussion of how well they support various steps in

hypothesis testing.

To improve data workers' workflows when performing data science tasks, we contribute design recommendations and two artifacts. Our artifacts include *StatWire,* an experimental hybrid-programming interface that encourages data workers to write high-quality source code; and *Tractus,* an interactive visualization that can lower the cost of working with experimental source code.

Based on our work, we present four takeaways that can be used by researchers, software developers, and educators to lower the barriers to hypothesis testing.

# Überblick

Data Science wird in der Wissenschaft und Industrie häufig benötigt. Ein wichtiger Nutzen ist die Prüfung von Hypothesen, wofür der Analyst signifikanz-basierte Hypothesentests verwendet, um auf Basis von Daten aus Experimenten Erkenntnisse über eine Populationsverteilung zu gewinnen. Neben Data Scientists, die professionelles Training in Data Science und hohe Kompetenz haben, üben viele nicht-professionelle Analysten Data Science aus. Diese bezeichnen wir als *Data Workers* und sie sind Experten in einem Gebiet, denen jedoch Expertise in Data Science fehlt. Data Workers umfassen akademische Wissenschaftler, Projektmanager und Vertriebsleiter.

Mithilfe von Interviews, Beobachtungen, Online-Umfragen und Inhaltsanalyse haben wir versucht, den Arbeitsablauf von Data Workern in wichtigen Aufgabenbereichen von Hypothesentests zu verstehen: Theoretische und praktische Statistik erlernen; statistische Verfahren auswählen; Data Science Programmierumgebungen nutzen, um mit Ideen in Quellcode zu experimentieren; den Quellcode verfeinern und überarbeiten; sowie die Analysergebnisse verbreiten. Wir gruppieren unsere Ergebnisse in zwei Teilschritte von Data Science:

1. **Die Vorbereitung zur Durchführung von Data Science-Aufgaben:** Wir diskutieren unsere Ergebnisse zum Einfluss von formaler Ausbildung auf die tatsächliche Praxis; Abwägungen zwischen Informationsquellen, welche die Auswahl von statistischen Verfahren herangezogen werden; empfundene Komplexität und Unsicherheit über die getroffene Wahl statistischer Verfahren; und den Widerwillen von Data Workern, alternative Analysemethoden anzuwenden.

   Aufgrund dieser Ergebnisse präsentieren wir Designempfehlungen sowie ein Artefakt, welche die Arbeit von Data Workern verbessern sollen. Unsere Artefakt ist *StatPlayground,* ein interaktives Simulations-Tool, das zum selbstständigen Erlernen statistischer Konzepte und Verfahren genutzt werden kann.

2. **Die Durchführung von Data Science-Aufgaben:** Unsere Ergebnisse beinhalten eine Übersicht über den Arbeitsablauf eines Hypothesentests unter Nutzung einer Programmierumgebung, der die Form von *explorativer Programmierung* nimmt;

und ein Vergleich existierender Oberflächen für Data Science-Programmierung, nämlich Notebooks, Skripte und Konsolen, sowie eine Diskussion darüber, wie gut sie die verschiedenen Schritte der Hypothesentests unterstützen.

Um den Arbeitsablauf von Data Workern während solcher Data Science-Aufgaben zu verbessern, präsentieren wir Designempfehlungen und zwei Artefakte. Die Artefakte sind *StatWire,* eine experimentelle Oberfläche zur hybriden Programmierung, die Data Worker dazu ermutigen kann, hochwertigen Quellcode zu schreiben; und *Tractus,* eine interaktive Visualisierung, die den Aufwand der Arbeit mit experimentellem Code verringert.

Basierend auf dieser Arbeit stellen wir vier Erkenntnisse vor, die von Wissenschaftlern, Software-Entwicklern und Ausbildern genutzt werden können, um die Hürden zum Erlernen des Umgangs mit Hypothesentests zu senken.

# Acknowledgements

This work would not have been possible without the help and support of many.

Many thanks to Prof. Chat Wacharamanotham, who instilled in me a passion for HCI, research, and statistics. Chat, I am grateful to you for your words of advise and encouragement, and for making me believe in myself.

I would like to express my sincere thanks to my first examiner, Prof. Jan Borchers, for his guidance, patience, and for giving me the freedom to organize my research projects.

I would like to thank my second examiner, Prof. Paul Cairns, for his encouragement and support. His research has motivated a substantial part of this thesis.

I am thankful to all study participants for their time and trust. I hope this work paves way for improving their lives.

I am grateful to all the wonderful people at i10. Working with and learning from talented colleagues and students has profoundly impacted this work and me. Special thanks to Nur Hamdan and Christian Corsten for their words of encouragement and advise at crucial times during my time at i10.

Thanks to Johannes Maas and Kalaivani Rajendran for their help in preparing this thesis.

Finally, I would like to thank my family and friends for their care and support during this work. Special thanks to my parents for their unparalleled love, and for giving me the freedom to pursue my ambitions.

# Conventions

We use the following conventions throughout this thesis.

Definitions of terms, author's contributions, and download links are set off in colored boxes.

**DEFINITION**
Definitions of key concepts and terms are set off in orange-colored boxes.

Definition:
*Definition*

https://hci.rwth-aachen.de/project [a]

[a] Links to supplementary materials are set off in green-colored boxes.

**PUBLICATIONS AND AUTHOR'S CONTRIBUTIONS**
Author's publications that are discussed in a chapter and his contributions are set off in blue-colored boxes. Such boxes are added at the beginning of each chapter that discusses our research contributions.

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

```
myClass
```

We follow the APA manual of style for using title vs. sentence cases [DeCleene and Fogo, 2012]. The titles of references in the thesis text, headings at Levels 1 and 2, and named sections within the thesis are written in title case. The titles of references in the reference list entries and headings at Levels 3–5 are written in sentence case.

The author of this thesis wishes to use the first person plural, “we”, to acknowledge the contributions of his collaborators. Plural first person pronouns, such as “we” and “us”, are used to refer to the readers in order to help with narration, e.g., “Let us assume that ...”. We use neutral third person pronouns, such as “they” and “their” to refer to individual study participants to maintain their anonymity.

The author’s individual contributions are clarified at the beginning of each chapter.

The whole thesis is written in American English.

# Chapter 1

# Introduction

*"We are drowning in information,*
*while starving for wisdom."*

*—Edward O. Wilson*

In the past, Greeks used information about spatial positions and trajectories of celestial bodies to predict the future. To do this, Greek thinkers like Apollonius, Hipparchus, and Ptolemy used *epicycles*[1]—geometric models of the sun, moon, and planets [Evans and Cannan, 2014]. Epicycles are fairly complex models capable of accounting for variations in speed and trajectories of these celestial bodies. Strikingly, an epicycle is not a model borne out of physics, but of *data*.

Data helped ancient Greeks predict the future.

Although we now have more sophisticated methods to track celestial bodies, many of our scientific and technological advancements are still driven by data. Audiovisual recordings of animals in the wild help us understand their behavior [Heithaus et al., 2002], measurements of surface radiative temperature of earth help us understand the impact of greenhouse gases on global warming [Jin and Dickinson, 2002], and tweets produced before elections help us forecast the outcome of the election [Rathi, 2017; Bovet et al., 2018].

Data drive advancements in science and technology.

To extract such insights from data, researchers and data ana-

[1]means "circle moving on a circle"

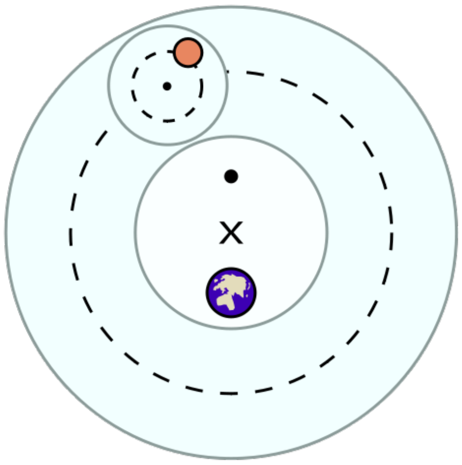


**Figure 1.1:** Epicycles are geometric models of celestial bodies used in Greece. The image shows a planet on an *epicycle* (smaller dashed circle) rotating around an axis referred to as a *deferent* (larger dashed circle). This model is borne out of data, and can account for variations in speed and trajectories of various planets. This is an example of a situation in which data drive scientific advancements. (Source: Wikimedia Commons.)

lysts use various tools and techniques. These tools and techniques, as well as the governing principles, constitute *data science*.

Definition: *Data Science*

**DATA SCIENCE**
Data science is a set of tools, such as significance testing and machine learning, and fundamental techniques that "support and guide the principled extraction of information and knowledge from data" [Provost and Fawcett, 2013].

Data science is no longer a skilled practiced just by professionals.

As a skill, data science has never been more necessary, widespread, or accessible. Modern technologies, such as the internet, mobile devices, and wearables, produce data at frequent rates. In order to gain an advantage over their peers [Provost and Fawcett, 2013], industries often utilize such data to drive their strategic decisions. For example, research by Brynjolfsson et al. shows that companies that are data-driven increase their productivity by 5–6% [Brynjolfsson et al., 2011]. As a result of this need, there is a demand for data analysts who can work with data to produce insights [Zita, 2021; Glassdoor Inc., 2021]. Further, modern computers are capable enough of handling vast amounts of data; analysts no longer need powerful supercomputers for this. This has lowered the barriers to performing data science tasks, and data science is now more widespread and accessible than ever before [Provost and Fawcett, 2013].

**Data scientist**
"A high-ranking professional with the training and curiosity to make discoveries in the world of big data" [Davenport and Patil, 2012].

Definition:
*Data scientist*

**Data worker**
Anyone who regularly works with data without necessarily having had extensive or any formal training [Boukhelifa et al., 2017]. Data workers do not typically self-identify as data scientists.

Definition:
*Data worker*

All of this has given rise to a new group of individuals who work with data called *data workers*. With this term, borrowed from research by Boukhelifa et al., we refer to domain experts who

Data workers are domain experts who work with data without necessarily having had extensive or even any formal training.

- work with data without necessarily having had extensive or even any formal training, and
- do not self-identify as data scientists [Liu et al., 2020].

As a result, data workers' workflows constitute varying levels of formalism [Harris et al., 2013b] and deviate from normative practice. Data scientists, in contrast, are extensively trained and typically operate with high skill levels [Davenport and Patil, 2012]. Data workers work in various domains, such as social sciences, archaeology, human-computer interaction (HCI), and medicine. They work under different designations, such as academic researchers, project managers, and sales managers.

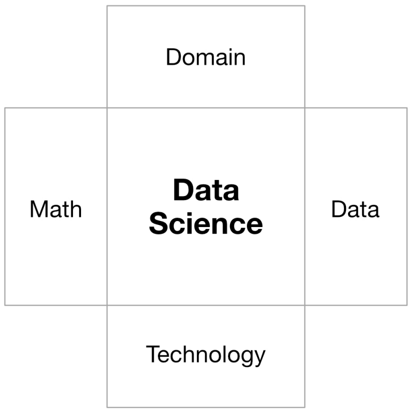


**Figure 1.2:** There are four dimensions to data science: *Domain,* which indicates the field in which data science is applied, such as research and business analytics; *data,* which drives many of our scientific and technological advancements; *technology,* which includes the programming IDEs and software tools that are used to perform data science tasks; and *math,* which includes the techniques, such as neural networks, statistical hypothesis testing, and regression analysis. We use these dimensions to specify the scope of our work. The image has been adapted from www.dezyre.com.

## 1.1 Hypothesis Testing

Statistical inference helps predict characteristics of a population from sample data points.

Data science has many applications, one of which is prediction [Davenport and Patil, 2012; Hernán et al., 2019]. A common method for prediction is *statistical inference,* in which the analyst predicts the characteristics of the population based on sample data. Several paradigms of analysis can be used to achieve such inference, such as the frequentist inference, Bayesian inference, and AIC statistics. Along with Bayesian inference, frequentist inference is the most common paradigm.

Hypothesis testing is prevalent across many research fields.

Frequentist inference includes methods such as statistical hypothesis testing (or significance testing), 95% confidence intervals, and effect sizes. Although this research work is primarily focused on hypothesis testing, we aim to generalize our work to other data science techniques as discussed later in Section 7.2. Hypothesis testing is the most common technique for inference in many disciplines, such as HCI [Cairns, 2007; Kaptein and

Robertson, 2012], psychology [Hubbard and Ryan, 2000], and medicine [Rutledge and Loh, 2004]. In HCI, hypothesis testing is routinely used to validate new interactions, devices, and systems [Kaptein and Robertson, 2012; Besançon and Dragicevic, 2019], e.g., via A/B testing. To quantify the prevalence of hypothesis testing in HCI, we surveyed publications at CHI, a premier HCI conference [Guide2Research, 2020]. 351 out of 703 papers ($\sim$50%) use quantitative methods. Excluding 50 papers that reported only descriptive statistics, we find that 276 out of 301 papers (91.6%) use hypothesis testing as the method of inference.

Hypothesis testing has problems of methodology and practice, and is considered hard to teach.

Despite such prevalence, hypothesis testing has garnered criticism over questionable use of and over-reliance on $p$-values, which leads misinterpretation of results, e.g., [Kline, 2004; Dragicevic, 2016]. Over-reliance of $p$-values leads to *dichotomous thinking,* under which results of research studies are incorrectly classified into "significant" and "non-significant", e.g., [Dragicevic, 2016]. Beyond criticisms of the method, hypothesis testing is also considered hard to learn and teach [Garfield and Ben-Zvi, 2007, p. 375].

## 1.2 Motivation and Approach

This thesis aims to lower barriers to performing hypothesis testing.

This thesis aims to understand and improve data workers' workflows with hypothesis testing. We focus on data workers, since their workflows differ from normative practice and very little is known about their workflows [Boukhelifa et al., 2017]. We chose hypothesis testing, since it is the de facto standard for statistical inference in several fields.

In our work, we employed an inductive approach to understand data workers' current practices. We do this through qualitative research methods, such as observations and interviews, and substantiate the findings with data from surveys. Based on the insights we gathered from such data collection and analyses, we identified opportunities for design, some of which we contribute as research artifacts in this thesis. In our work, we aim to answer the following questions:

- How do data workers *prepare* to perform hypothesis testing? What information sources do they use? What information do they require? How do they select the appropriate statistical procedure?
- How do data workers *perform* hypothesis testing with various interfaces? How well do these interfaces support the various tasks in hypothesis testing?

## 1.3 Contributions and Structure

This thesis has contributions across two key tasks in data science as shown in Figure 1.3: *preparing to perform data science tasks* and *performing data science tasks*. Our contributions are as follows:

1. Preparing to perform data science tasks
   (a) Chapter 3: Two interview studies and a content analysis that show how data workers prepare to perform data science tasks: what knowledge they seek, which information sources they use to seek this, and how such information sources are used.
   (b) Chapter 4: Based on our findings, we contribute design recommendations and one artifact: *StatPlayground,* a simulation-based prototype to self-learn or teach practical statistics, and *Statsplorer,* a tool that helps educate data workers about common issues in hypothesis testing.
2. Performing data science tasks
   (a) Chapter 5 and Chapter 6: A content analysis that shows common problems in data science programming. Our findings led to an artifact, *StatWire,* an exploratory prototype for hybrid data science programming.
   (b) Chapter 5: Two observation studies that help us understand two aspects of data science programming: How the analyst's source code evolves across hypothesis testing, and how existing programming interfaces support various steps in hypothesis testing.

(c) Chapter 6: Our findings from these observation studies led to *Tractus,* a tool that visualizes the source code by grouping them into hypotheses.

Before presenting our contributions, we provide some background information that helps better situate our contributions in Chapter 2.

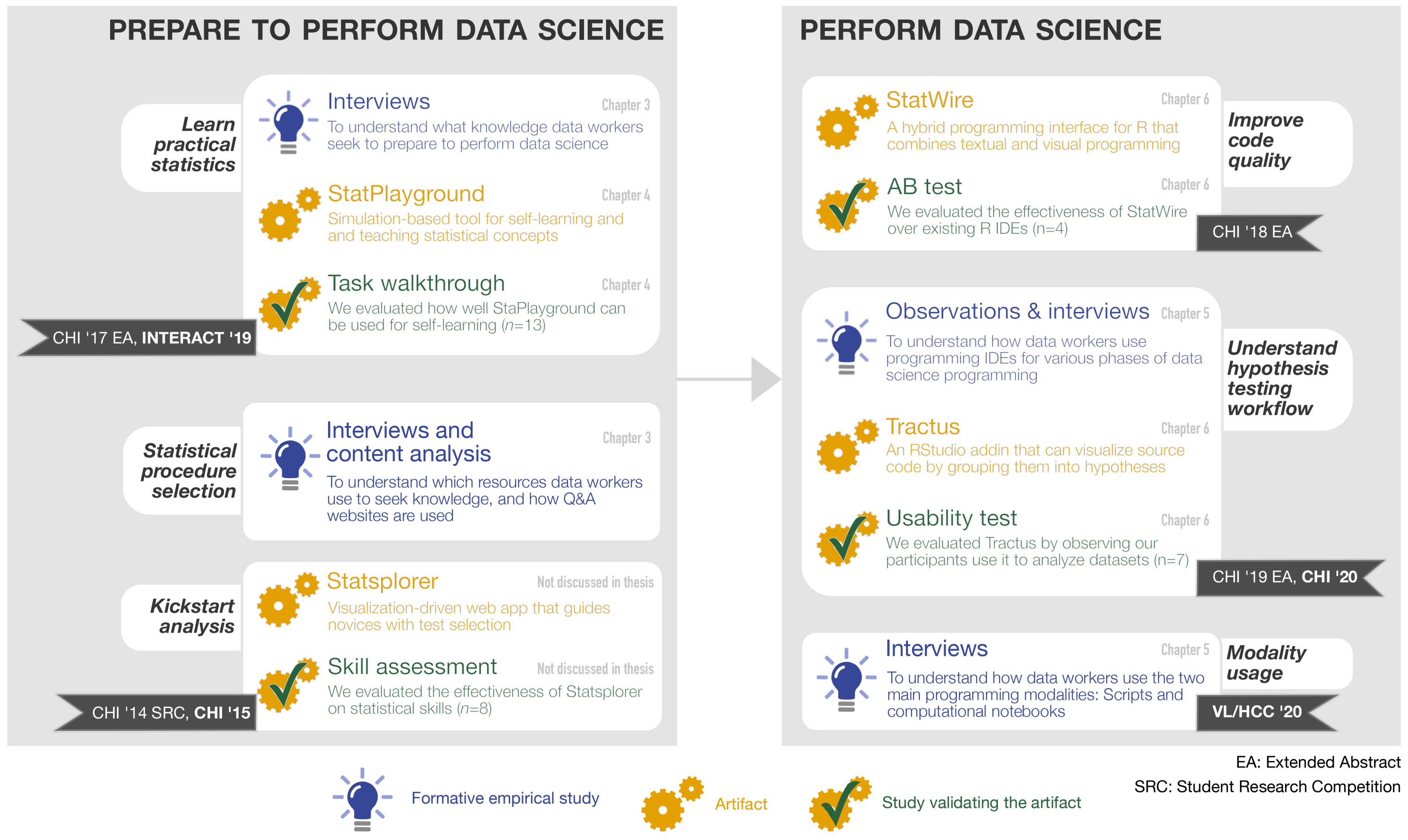

PREPARE TO PERFORM DATA SCIENCE
Learn practical statistics
Interviews
Chapter 3
To understand what knowledge data workers seek to prepare to perform data science
StatPlayground
Chapter 4
Simulation-based tool for self-learning and and teaching statistical concepts
Task walkthrough
Chapter 4
We evaluated how well StaPlayground can be used for self-learning (n=13)
CHI '17 EA, INTERACT '19
Statistical procedure selection
Interviews and content analysis
Chapter 3
To understand which resources data workers use to seek knowledge, and how Q&A websites are used
Kickstart analysis
Statsplorer
Not discussed in thesis
Visualization-driven web app that guides novices with test selection
Skill assessment
Not discussed in thesis
We evaluated the effectiveness of Statsplorer on statistical skills (n=8)
CHI '14 SRC, CHI '15
PERFORM DATA SCIENCE
StatWire
Chapter 6
A hybrid programming interface for R that combines textual and visual programming
AB test
Chapter 6
We evaluated the effectiveness of StatWire over existing R IDEs (n=4)
Improve code quality
CHI '18 EA
Observations & interviews
Chapter 5
To understand how data workers use programming IDEs for various phases of data science programming
Tractus
Chapter 6
An RStudio addin that can visualize source code by grouping them into hypotheses
Usability test
Chapter 6
We evaluated Tractus by observing our participants use it to analyze datasets (n=7)
Understand hypothesis testing workflow
CHI '19 EA, CHI '20
Interviews
Chapter 5
To understand how data workers use the two main programming modalities: Scripts and computational notebooks
Modality usage
VL/HCC '20
EA: Extended Abstract
SRC: Student Research Competition
Formative empirical study
Artifact
Study validating the artifact

**Figure 1.3:** An overview of this thesis, and our contributions. We used an inductive approach to understand data workers' workflows, and then built artifacts to improve these workflows.

# Chapter 2

# Background and Motivation

Data science is a broad term that is commonly used to refer to a wide range of tasks, techniques, and principles used to obtain insights from data. Over the years, there has been an increasing uncertainty about what data science is, and, as a result, data science is often mistaken for related disciplines, such as data mining, big data, and data analytics [Provost and Fawcett, 2013; Dhar, 2013]. To situate our research more clearly within the broad discipline of data science and to provide background information that is required for understanding the contributions outlined in this thesis, we aim to answer the following questions in this chapter:

Data science is an umbrella term used to refer to a range of tasks, tools, and techniques to extract knowledge from data.

- Section 2.1: What is hypothesis testing? What are the steps in hypothesis testing?
- Section 2.2: What is data science? What 'part' of data science is hypothesis testing?
- Section 2.3: What are some prominent issues in hypothesis testing?
- Section 2.4: What is the role of hypothesis testing in data science?
- Section 2.7: Who are *data workers*? What are their characteristics? Why is it interesting to study them?

Additionally, we discuss some notable research that motivated our work and some background literature on topics related to our work, such as characteristics of data science (Section 2.5) and interfaces used to perform analysis (Section 2.6).

## 2.1 Steps in Hypothesis Testing

Hypothesis testing involves numerous steps. At each step, the analyst needs to make decisions that impact the remainder of their analysis.

A *scientific hypothesis* is a concrete statement that gives one potential explanation of a phenomenon [Toledo et al., 2011]. Hypotheses need to be tested for it to become scientific truth (or a *theory*), and hypothesis testing provides a numerical approach to do this. Hypothesis testing is a complex task that involves several steps as discussed below. These steps are adapted from existing research [Fife, 2020; Field, 2013], and are grounded in author's own experience of working with data workers during the course of this research work.

1. **Formulation:** Determine one or more hypotheses based on the research questions. The analyst typically identifies such hypotheses from observations, surveys, experiments, literature survey, or other such methods. For example, in HCI, many researchers initially perform exploratory studies, and analyze the resulting data to generate hypothesis [Cockburn et al., 2018].

2. **Design the experiment:** In this step, the analyst designs the experiment by picking the *measurements* (also known as dependent variables) and *factors* (also known as independent variables); sets up the experiment to limit extraneous variables; designs the stimuli and task to ensure validity and reproducibility of the research.

3. **Determine the sample:** The analyst then recruits participants who fit the inclusion criteria, e.g., age, experience, and handedness. At this step, the analyst would also determine an appropriate sample size based on a statistical power estimation.

4. **Run the experiment:** The analyst runs the experiment to collect the experimental data.

5. **Data cleaning:** The analyst cleans data to remove outliers, convert data to the correct format, and normalize values.

6. **Exploratory analysis:** The analyst understands their data by performing exploratory data analysis, a process in which the analyst uses visualizations and summary statistics to detect patterns and anomalies. Such exploratory analysis could include significance tests, but such tests should not be considered as statistical proof for or against the hypothesis. The results should instead be considered new hypotheses that are then evaluated in another experiment and analysis, one targeted at statistically accepting or rejecting a hypothesis.

7. **Check data assumptions:** Most statistical significance tests assume certain characteristics about the data. The most common assumptions are: Type of distribution, whether the distributions have similar variances, and whether the factor was within-subjects or between-subjects. The analyst checks these assumptions before carrying out the significance tests.

8. **Perform significance tests:** Based on data characteristics and sample size, the analyst selects and performs a statistical test. In situations where a factor has more than two levels, the analyst performs an *omnibus test* like ANOVA. Omnibus tests are used to test for a significant effect across all distributions corresponding to these levels. If such a test is successful in showing an effect, the analyst performs pairwise tests to reveal effect between each distribution pair.

9. **Interpret the results:** After performing significance tests, the analyst interprets the results. This might involve calculating the point estimates, e.g., effect sizes, and interval estimates, e.g., 95% confidence intervals. These estimates can be used to interpret the practical significance of the findings from analysis.

10. **Report the findings:** Once the analyst interprets the results, they reports their analysis process, methods, and findings in publications and presentations, e.g., by using the APA style for reporting [American Psychological Association, 1994].

## 2.2 Situating Hypothesis Testing in Existing Classifications of Data Science

Data science involves the use of principles and analysis techniques to extract insights from data.

Data science refers to the tools, techniques, and principles used to extract information and knowledge from data [Provost and Fawcett, 2013]. While tools and techniques are well-known, principles are equally important and govern an analyst's approach towards a data science problem. For example, an organization might request that analysts ensure that their analysis source code conforms to *clean code* principles [Martin, 2008].

Analysis techniques typically come from a certain data science paradigm.

*Techniques* (also known as *methods*) facilitate the actual extraction of knowledge. Techniques often conform to a school of thought, referred to as a *paradigm*. For example, unpaired $t$-test, unsupervised learning, neural networks, and regression models are some analysis methods, whereas frequentist inference [Field, 2013], Bayesian inference [Kruschke, 2015], and AIC-based inference [Akaike, 1973] are paradigms of analysis [Granville, 2016].

We now review some of the existing classifications of data science, in order to situate hypothesis testing in these classifications. These classifications are not mutually exclusive, and instead act as conceptual lenses with which to view the vast field of data science.

### 2.2.1 Hernán's Classification of Data Science by Task Purpose

Hypothesis testing is used across all tasks in Hernán's classification of data science tasks.

Hernán et al. classified data science tasks into three types on the basis of the purpose or intent: *description, prediction,* and *counterfactual prediction* [Hernán et al., 2019]:

- *Description:* Using data to provide a quantitative summary of certain features in the world. Example techniques include summary or descriptive statistics, cluster analysis, and visualizations.
- *Prediction:* Using data to map some features of the world to other features of the world. Example techniques in-

clude correlation coefficients, random forests, and neural networks.

- *Counterfactual prediction:* Using data to predict certain features of the world as if the world had been different. g-methods are commonly used to make counterfactual predictions.

Hernán argues that hypothesis tests are used across all three data science tasks, usually to generalize findings from sample to population.

### 2.2.2 Descriptive vs. Inferential Statistics

Descriptive statistics are used to describe samples.

During hypothesis testing, it is often impossible to collect measurements from the entire population. Therefore, analysts collect measurements about a sample data, and then use statistics to *estimate* information about population. Descriptive statistics refers to summary statistics that are used to describe a *sample*. These include measures of central tendency, such as mean, median, and mode, and measures of variability, such as standard deviation and variance. Although it is important to understand the sample, the main goal of hypothesis testing is to draw conclusions about the population based on the sample.

Inferential statistics are used to extrapolate information from sample to the underlying population.

Inferential statistics can provide this information. These include point estimates, e.g., effect size, and interval estimates, e.g., 95% confidence intervals. These also include hypothesis tests, such as t-tests and ANOVA, which results in a $p$-value that acts as evidence for or against the null hypothesis and a test statistic that measures how well the data points fit a predefined model. A thorough discussion of inferential statistics, including the underlying concepts of Central Limit theorem and Gaussian distribution, is beyond the scope of this thesis. See, e.g., [Field, 2013] for a detailed discussion of the basics.

### 2.2.3 Tukey's Classification of Data Analysis: Exploration and Confirmation

Tukey proposed that data analysis consists of two distinct, but related stages: Exploratory Data Analysis (EDA) and Confirmatory Data Analysis (CDA) [Tukey, 1980]. EDA is the initial stage of getting acquainted with the data; it involves computing descriptive statistics and plotting visualizations. The objective of analysts at this stage is to understand the data, detect patterns and anomalies, and generate hypotheses. EDA could even involve hypothesis tests or Bayesian analysis, but the results of such tests should only be considered as tentative and as a first step to confirmatory analysis.

Analysis involves exploratory analysis to understand data patterns followed by confirmatory analysis to validate hypotheses.

CDA follows EDA, and involves validating one or more hypotheses through hypothesis tests or Bayesian analysis. Descriptive statistics is used predominantly during EDA, whereas inferential statistics is used during CDA. Common statistics used in CDA are $p$-values, Bayes factors, effect sizes, and 95% confidence intervals. Using another conceptual lens [Kell and Oliver, 2003], EDA can be thought of as the process of *induction*, i.e., generalizing data to generate ideas. Conversely, hypothesis-driven data science is based on *deduction*, which validates ideas through data. Both inductive and deductive forms of data science complement each other.

EDA generates ideas which are validated in CDA.

### 2.2.4 Paradigms of Statistical Inference

Although frequentist inference is the most prevalent, there are many other established paradigms or schools of statistical inference: a) classical statistics or frequentist inference, b) Bayesian statistics, c) the Akaikean-Information Criterion-based (AIC) statistics, and d) likelihoodist statistics [Bandyopadhyay and Forster, 2011].

There are four paradigms of statistical inference.

*Frequentist inference* is based on the assumption that data distributions, when repeatedly sampled over time, have properties similar to the population. Based on certain factors like sample size and type of distribution, frequentist inference allows us to estimate population distribution's characteristics with a certain threshold of error. There are different approaches to frequentist inference, such as the processes introduced by Neyman-Pearson, Fisher, and Lindquist [Perezgonzalez, 2015].

Frequentist inference allows us to estimate characteristics of population.

Of these approaches, significance testing is one of the most prevalent methods for validating research hypotheses [Cairns, 2007; Cockburn et al., 2018]. It involves computing $p$-values and using them as thresholds to validate hypotheses [Nickerson, 2000]. It is often employed in dichotomous testing, where the researcher would accept or reject a hypothesis on the basis of statistical significance [Dragicevic, 2016]. Over the past decade, significance testing has garnered a lot of criticism in HCI [Dragicevic, 2016; Cockburn et al., 2018]; we discuss a few prominent issues in Section 2.3.

Significance testing involves the use of $p$-values to accept or reject a hypothesis.

*Bayesian inference* is based on Bayes' theorem, under which the probability for a hypothesis to be true can become updated as more data becomes available. Bayesian inference can be computationally intensive and might require the analyst to specify information that is rarely available, such as the prior probabilities. As a result, analysts often select priors in a subjective fashion, which is a subject of contention among statisticians, e.g., as discussed in [Casella and Moreno, 2006]. Nevertheless, as computers have become more powerful, many researchers have recommended the use of Bayesian inference over hypothesis testing [Gelman, 2006; Dienes, 2011; Kay et al., 2016].

Bayesian inference, which is based on Bayes' theorem, is an alternative to significance testing.

*AIC statistics* uses a metric called Akaike[1] Information Criterion (AIC) to estimate prediction error of a model. This metric is used to compare models, typically the same models constructed in hypothesis tests. *Likelihoodist statistics* uses the likelihood function, which measures the goodness of fit of a statistical model as a factor of sampling density. Likelihoodist statistics can be used alongside frequentist and Bayesian inference.

AIC statistics use a metric to compare models. Likelihood statistics use the likelihood function to compare models.

### 2.2.5 Model Selection vs. Significance-Based Testing

In addition to significance-based hypothesis testing, data analysts can use model selection to validate hypotheses. In this approach, the analyst determines several alternative hypotheses, and models each hypothesis using a model [Burnham and Anderson, 2002]. Then, metrics like AIC [Akaike, 1973; Bozdogan, 1987] and Bayes factor [Kruschke, 2015] are used to *select* the model that has the best fit (or another relevant cri-

Model selection allows comparison of models to determine the model that best fits the data.

[1]named after Hirotugu Akaike, the Japanese statistician who developed the metric

teria). Thus model selection can be seen as an extension of significance-based testing, in which the analyst obtains evidence to prefer one hypothesis over another.

Significance testing is used to obtain evidence against null hypothesis.

In contrast to model selection, hypothesis testing involves comparing two or more data distributions against each other to obtain evidence against the null hypothesis[2]. Based on the experimental data, a statistic ($p$-value) is calculated, based on the assumption that the data distributions were originally sampled from the same population distribution. If this statistic falls below a certain threshold, the null hypothesis is rejected, and, consequently, the alternative hypothesis is accepted.

## 2.3 Prominent Issues in Hypothesis Testing

We now have a better understanding of what hypothesis testing is and where it fits into data science. At this point, it is important to briefly review prominent issues in hypothesis testing. Over the years, there has been a lot of criticism against the use of hypothesis testing across several research fields. In this section, we discuss some of the prominent criticism against hypothesis testing. We group the existing issues into two:

- issues that arise due to misinterpretation of how significance tests work, and
- issues that arise due to incorrect application of significance tests.

Hypothesis testing issues are methodological or issues prevalent in practice.

The two groups are not meant to be mutually exclusive. Indeed, incorrect application of tests can follow misinterpretations of significance tests. However, the issues in the first group arise mainly due to the inherently complex nature of hypothesis testing, and might be harder to address without addressing statistical education. In contrast, issues in the second group have more involvement of sociological factors, e.g., over-reliance on $p$-values is a consequence of how the research community has traditionally functioned. For more details on

[2]A null hypothesis indicates that there is no effect of the factor on the measurement.

the issues discussed below, please refer to prominent works in HCI, psychology, and statistical practice, such as [McCloskey and Ziliak, 1996; Hubbard and Bayarri, 2003; Cairns, 2007; Armstrong, 2007; Kaptein and Robertson, 2012; Dragicevic, 2016; Kay et al., 2016].

### 2.3.1 Misconceptions in Significance Testing

#### Misinterpreting $p$-values

The first group of issues arise from the inherently complex nature of significance tests, and a major contributor to complexity is $p$-value. Several studies in the past have shown that even researchers, even if they are trained, have difficulty interpreting $p$-values [Oaks, 1986; Kline, 2004; Beyth-Marom et al., 2008].

$p$-values are hard to interpret, even for trained researchers. This leads to several issues when interpreting results of hypothesis tests.

When conducting empirical research, the researcher comes up with a null hypothesis and an alternative hypothesis. The null hypothesis is a statement that postulates that there is no effect view, i.e., a view of the world as it currently stands. The alternative hypothesis states a new theory, i.e., a view of the world that the research is hoping to confirm. The analyst ideally wants to know the probability of their alternative hypothesis being true. The $p$-value, however, is a measure of the evidence *against* the null hypothesis, not *for* the alternative hypothesis. If there is sufficient evidence against the null hypothesis, then the analyst de facto 'rejects' the null hypothesis, and, as a consequence, 'accepts' the alternative hypothesis. Known more commonly as "fallacy of the transposed conditional", many analysts tend to misinterpret $p$-value as the probability of the null hypothesis being true [Cohen, 1994; Haller and Krauss, 2002]. Other common misinterpretations include interpreting high $p$-values to mean that there is no effect [Dienes, 2014] and assigning too much reliability to $p$-value [Dragicevic, 2016].

#### Statistical vs. practical significance

A significantly lower $p$-value, usually less than 0.05, only tells us that the effect is not caused by a sampling error, that is, there

Analysts tend to incorrectly associate practical significance to $p$-values.

is a difference at the population level. $p$-value does not, and cannot, provide information about whether the effect, that is, the difference between the sample distributions, has any *practical implications*. This fallacy, of associating practical significance to $p$-values, can be particularly problematic in studies involving large sample sizes: Since $p$-values are sensitive to large sample sizes, it is possible to obtain $p < 0.05$ even when the difference between the distributions is trivial! Conversely, it is possible to obtain a practically significant effect even when the $p$-value is higher than 0.05.

#### Dichotomous thinking and publication bias

Data workers and researchers tend to binarily classify results based on $p$-values, which leads to publication bias.

Due to the standard cutoff of 0.05 for $p$-values, many analysts classify the results of their analysis into a binary "significant" or "non-significant" result. Such classifications nullify the error and uncertainty associated with $p$-value and significance testing, and can be particularly problematic in studies with small sample sizes. Such *dichotomous thinking* is also prevalent among reviewers and results in *publication bias*: The tendency for papers that show a statistical significant effect (i.e., $p < 0.05$) to have a higher chance of being accepted for publications than papers that do not [Dickersin et al., 1987]. Such practice can hamper scientific progress[3].

### 2.3.2 Practical Issues in Statistical Testing

#### Overlooking assumptions and using incorrect tests

Select the appropriate statistical procedure can be complex, and data workers may overlook checking assumptions.

When conducting analysis, analysts need to make a number of critical decisions that determine the correct statistical procedure to use. To make these decisions, they keep track of key information such as the assumptions of the statistical test e.g., normality of distributions and sphericity; dataset characteristics, e.g., nature of the variables and how measurements are made; and experimental design, that is, within- or between-groups or factorial designs [Field, 2013]. Obtaining these de-

[3]https://www.economist.com/briefing/2013/10/18/trouble-at-the-lab

tails can be tricky and may require the researcher to do data wrangling operations and make more decisions. Existing research has shown that overlooking assumptions and using incorrect tests is quite common [Cairns, 2007].

### Overlooking statistical power

*Statistical power* is the ability of the statistical test to find a statistically significant effect in case such an effect is present. *Statistical power estimation* or *power analysis* can help calculate the likely minimum sample size required to detect an effect size of given magnitude. Since many researchers do not employ power analysis [Cohen, 1992], many published studies lack adequate statistical power, which calls into question the validity of published findings [Maxwell, 2004].

Data workers might overlook statistical power, ability of a statistical test to find an effect, if such an effect is present.

### Over-testing and $p$-hacking

In hypothesis testing, type I error (or false-positive) occurs when we reject the null hypothesis when it is actually true in the hypothesis. Type I error rate increases with more statistical tests the analyst runs. This issue is called *over-testing*. There are several ways to address over-testing, e.g., the Bonferroni correction of $p$-values is a common technique.

Over-testing can bleed into data-dredging, which is also known as $p$-hacking, fishing, and HARKing[4]. In this questionable statistical practice, the researcher conducts several analyses, e.g., checking if there is an effect of a factor with each measurement made in the experiment, finds a statistically significant result, and reports this as a result of confirmatory analysis [Kerr, 1998; Cockburn et al., 2018]. However, since the type I error rate is inflated and data is not collected from an experiment designed to test a particular hypothesis, the results are misleading. There are several research works that discuss this issue in detail [Cockburn et al., 2018; Pu and Kay, 2018], and offer alternatives, e.g., [Dragicevic et al., 2019], and guidelines for good practice, e.g., [Cairns, 2019].

Data workers might inadvertently perform multiple tests in search of significant results, and then selectively report the findings.

[4]An acronym for Hypothesizing After the Results are Known

### Issues in statistical reporting

Statistical reports in HCI research papers were found to not conform to APA standards of reporting.

According to Cairns, problems with reporting was one of the dominant issues in HCI publications he analyzed [Cairns, 2007]. Several papers did not meet the APA standards. Papers omitted information about the tests that were done, key statistics, and even descriptive statistics, such as mean and standard deviations. Cairns also speculates that authors of these papers might have performed several tests, without reporting them all. This relates to the HARKing issue we discussed earlier.

## 2.3.3 Problems in Statistical Education

Theoretical and practical hypothesis testing can be difficult to teach.

Since hypothesis testing concepts can be hard to grasp, there has been a lot of research aimed at improving statistical education. Hypothesis testing can be even harder to teach than apply in practice because, unlike real-world practitioners who could sometimes view hypothesis testing as a “black box”, students need to understand the underlying concepts. Statistical educators concur that students do find it hard to understand the fundamental concepts, and that there is a mismatch between expected learning outcomes what students are actually tested for [Garfield, 1995, pg. 27].

Students’ misconceptions about hypothesis testing are resilient to change.

Garfield discussed some common misconceptions in statistics and probability [Garfield, 1995], some of which we discussed in Section 2.3.1. Researchers have also warned that misconceptions about statistical concepts are resilient to change [Garfield, 1995; Schmidt and Hunter, 1997]. Ignoring or disapproving students’ misconceptions will leave the misconceptions unchanged. Past studies have reiterated that even fundamental concepts of probability are difficult for students to grasp [Garfield and Ahlgren, 1988; Shaughnessy, 1992]. Educators tend to overestimate how well their students understand statistical concepts.

## 2.4 Role of Hypothesis Testing in Data Science

Based on criticisms in the last section, one might wonder if hypothesis testing has a role in data science. Despite questions over its validity and concerns of incorrect use, hypothesis testing is commonplace in data science work. A vast majority of HCI researchers continue to use NHST despite prevalent criticisms. Recall that in our survey of CHI publications, we find that 91.6% of papers that perform quantitative analysis use hypothesis testing.

Despite criticisms against it, hypothesis is still very common.

We take the stance that hypothesis testing has its place in data science and academic research, when the analyst conforms to normative practice. Analysts need to differentiate analyses carried out on data collected from exploratory studies and studies intended to gather evidence for or against a hypothesis. Hypothesis testing can still be done on data collected in exploratory studies, as long as the results are not interpreted as *causation* but instead a *correlation*. Hypothesis testing in the confirmatory-style of analysis should be done on data collected from an experiment that is designed to validate the hypothesis [Abelson, 2012], as discussed in the previous section.

In our work, we aim to support hypothesis testing as a normative practice.

## 2.5 Characteristics of Data Science

In this section, we discuss some foundational concepts of data science workflows and practices. This discussion help us understand current data science practice and reflect upon the analysts' pain points.

### 2.5.1 Exploratory Programming Workflow

*Exploratory programming* is a term originally introduced by Beau Shiel in 1983 [Sheil, 1983]. In exploratory programming, the programmer needs to prototype their ideas in source code in order to make progress. In conventional programming, the programmer has specifications for source code and writes code

In exploratory programming practice, programmers begin with open goals and explore ideas through code to make goals concrete.

to match these specifications. In exploratory programming, however, the programmer identifies the programming goals through exploration, and these open-ended goals continue to change over explorations. Examples of exploratory programming tasks include interface design [Hartmann et al., 2008a], data science [Kery et al., 2017], digital art [Montfort, 2016], and many software engineering tasks, e.g., developing algorithms [Yoon and Myers, 2014] and understanding how to use a programming API [Robillard, 2009]. Exploratory programming is an umbrella term that includes programming practices like bricolage/tinkering, sketching, live coding, and code bending [Bergstrom and Blackwell, 2016].

Exploration extends beyond programming. Exploratory programming deals with workflows specific to programming.

Exploration is not unique to programming. For example, designers who use GUI interface builders follow iterative prototyping techniques to progressively develop good interfaces. Research about exploratory programming, however, focusses on the consequences of using text-based programming to perform the source code explorations, and studies how programmers manage these explorations in code. Here are some key consequences of exploratory programming practice:

Exploratory programmers do not invest in writing high-quality code during experimentation.

- Exploration often results in low-quality source code, e.g., less or no modules and sub-optimal code [Kery and Myers, 2017; Kery et al., 2017]. Since it is unclear which explorations will eventually fruitful, programmers focus on making progress with their overall goal while compromising code quality.

Exploratory programmers frequently backtrack to earlier code.

- In a study with 21 programmers, Yoon and Myers found that, during exploratory programming, programmers often backtracked to a piece of code, often repeatedly, during exploration [Yoon and Myers, 2014]. Since code explorations can span across a long source code file, or even across multiple source code files, there is a need for programmers to navigate source code effectively. In addition to helping with explorations, history of explorations also act as a record of programmers' exploration practice [Davidson and Freire, 2008].

For more details about the characteristics of exploratory programming, and how it differs from related concepts like opportunistic programming and debugging into existence, see [Kery and Myers, 2017].

Does hypothesis testing conform to an exploratory programming workflow? During EDA, data workers explore various combinations of measurements and factors, and visualize them using a variety of visualization techniques to detect patters and generate hypotheses. During CDA, data workers experiment with different data transformations and models to determine which approach works best. Thus significant portions of analyses adopt an exploratory programming workflow.

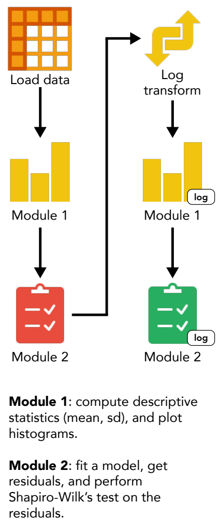

Load data
Log transform
log
Module 1
Module 1
log
Module 2
Module 2
**Module 1**: compute descriptive statistics (mean, sd), and plot histograms.
**Module 2**: fit a model, get residuals, and perform Shapiro-Wilk's test on the residuals.

**Figure 2.1:** Iterative and non-linear workflow. Code acts on different data in each iteration.

### 2.5.2 Iterative, Non-Linear Workflow

Hypothesis testing follows an iterative, non-linear workflow [Tukey, 1977; Lubinsky and Pregibon, 1988]. Consider Figure 2.1. In typical analysis, analysts make changes to data. These changes result in the analyst iterating back to previous steps and modifying them to suit the new data characteristics. Such non-linear workflow is a characteristic of all exploratory programming workflows.

Using an example, we illustrate the iterative, non-linear nature of hypothesis testing.

To understand this, consider an example analysis as shown in Figure 2.1. The analyst loads their data, performs data preprocessing to deal with missing values and conversions to long or wide formats, views the descriptive statistics (mean, standard deviation), and visualizes each distribution using a histogram. Since none of the distributions are normally distributed, they fits a model to get residuals, and perform a Shapiro-Wilk's test for the residuals. The results confirm that the distributions are not normally distributed. The analyst then views the histograms again and notices that the distributions could be log normal. To confirm this, they apply a log transformation to the dataset, view the descriptive statistics again, plot histograms, and perform tests for normality on the residuals of transformed distributions. With each iteration, the same chunk of analysis code is applied to different data distributions.

## 2.6 Interfaces to Perform Analysis

Data analysts have a variety of tools at their disposal that can help them analyze data. The most common options are GUIs, visual programming tools, and text-based programming tools.

### 2.6.1 GUIs for Data Analysis

GUIs are popular among data analysts, but might lack powerful statistical methods and can be expensive.

GUI-based software, such as JMP[5] [Goos and Meintrup, 2016], Stata[6] [LP StataCorp, 2007], SAS[7] [O'Rourke et al., 2005] and SPSS[8] [Verma, 2012], are popular among data analysts. GUIs offer a WIMP-interface that can be easy to use, and provide quick access to several statistical functions and utilities. However, as the source code is typically not accessible, GUIs might not have a generative value, and might not allow novice data workers to graduate to professional data scientists [Myers, 1990]. Certain statistical software support scripting through programming languages, e.g., with JSL[9] (JMP Scripting Language). This allows power users to extend the statistical functions of the analysis software. Further, powerful GUI software, such as JMP and Stata, can be quite expensive to use for students and other non-professional analysts.

Certain web-based apps like StatWing[10] are also GUI-based, but lack the power and functionalities of desktop software.

### 2.6.2 Text-Based Programming

Text-based programming can be appealing to data workers who know programming and is considered more productive than GUI.

In conventional text-based programming, the programmer submits a set of statements as a unit to "direct the behavior" of the computing system [Illingworth, 1997]. Data workers, especially from a computer science background, might use pro-

[5]https://www.jmp.com
[6]https://www.stata.com/
[7]http://sas.com
[8]https://www.ibm.com/de-de/analytics/spss-statistics-software
[9]https://www.jmp.com/support/help/en/15.2/jmp/introduction.shtml
[10]https://www.statwing.com

gramming to perform analysis. Indeed, the statistics community considers using programming languages like R to perform analysis to be more productive than using a GUI [Valero-Mora et al., 2012].

The most popular languages for data analysis are R [Ihaka and Gentleman, 1996] and Python [Sanner, 1999]. There are two main interfaces that analysts can use to write and execute source code in these languages: *script files* and *computational notebooks*. Script files allow storage and execution in a linear fashion. In R and Python, when source code is executed, the output is shown on the console window (for text output) or in a separate window (for plots and other graphics). In computational notebooks, analysts store and execute source code in units called 'cells'. A notebook is made up of several cells arranged in a top-down manner. In addition to source code, analysts can add rich text and graphics via Markdown to create a *narrative* of the analysis. Notebook environments like Jupyter[11] [Kluyver et al., 2016] and Google Colab[12] [Bisong, 2019] have become popular among all analysts.

Popular data science programming languages, R and Python, support programming via scripts and computational notebooks.

### 2.6.3 Visual Programming

Visual Programming Languages (VPL) refer to a broad collection of interfaces that allow users to program "in a two (or more) dimensional fashion" [Myers, 1990]. In VPLs, users specify the input and output arguments or even the entire program using visual notations. Visual programming languages can also act as program visualizations, in which the graphics are used to illustrate some aspect of the textual program [Myers, 1990]. Visualizations of data flow in source code are particularly well-suited to data analysis, and has been shown to improve understanding and debugging [Wongsuphasawat et al., 2017].

Visual programming allows the use of visual notations for 'programming' and can be better suited to data analysis.

VPLs offer several advantages for programming in general: they lead to better programmer performance [Pandey and Burnett, 1993; Baroth and Hartsough, 1995] than text-based programming. They can help improve code comprehension [Cun-

Visual programming can improve code comprehension and navigation.

[11]https://jupyter.org/
[12]https://colab.research.google.com/

niff and Taylor, 1987; Scanlan, 1989; Bragdon et al., 2010a] and code navigation [Coblenz et al., 2006; DeLine et al., 2006], two tasks programmers spend 60–90% of their time on during programming [Erlikh, 2000]. However, VPLs are not always better in the absolute sense—even well-designed visual notations will not facilitate certain tasks [Green and Petre, 1992; Green et al., 1991; Wright and Reid, 1973], making them ineffective for such tasks. For a more complete discussion of the characteristics of VPLs, see [Whitley, 1997].

There are two main styles of visual programming: mainly visual or mainly textual.

VPLs can be inherently visual or textual [Chang, 1987; Zhang, 2013]. Inherently visual VPLs like Scratch [Resnick et al., 2009] let the user directly 'specify' the program through visual representations. Inherently textual VPLs instead provide a graphical abstraction of the underlying textual representation. ViSta [Young and Bann, 1996], Orange [Demšar et al., 2004], RapidMiner [Hofmann and Klinkenberg, 2013], and KNIME [Berthold et al., 2009] are examples of statistical tools that are inherently textual VPLs.

## 2.7 Data Workers

It is not just professional data scientists who work with data.

In this section, we discuss data workers, the target users of this research. In both academia and industry practice, there are many who take on the role of analyst and work regularly with data. The most well-known and well-researched of them are *data scientists*, who work with data as their primary job function. Data scientists undergo extensive training and usually have established workflows.

Data workers are less studied and their workflows can be significantly different from normative practice.

There are also those who work with data but not as their primary job function. Such analysts, who we refer to as *data workers*, are experts in their own field of interest but work with data on a regular basis. Data workers do not self-identify as data scientists [Liu et al., 2020], possibly because they work with data without much or even any formal training and learn tool usage themselves [Boukhelifa et al., 2017; Lowndes et al., 2017]. As a result, data workers have varying skill levels[13], perform anal-

[13]The more experienced data workers might have skills that are on par with data scientists, but such data workers might still not self-identify as

ysis tasks with “varying levels of formalism”, and have workflows that differ significantly from normative practice [Harris et al., 2013a; Kandel et al., 2012]. According to research by Wolff, data workers are the fastest-growing group of employees in the U.S. between 1950 and 2000 [Wolff, 2005].

### 2.7.1 Existing research about data workers

Prior research has focussed mainly on understanding and supporting data scientists. Existing research on data workers is sparse. We borrow the term data workers from Boukhelifa et al., but references to such domain experts have been made previously, e.g., as knowledge workers [Davenport, 2005] or information workers [Wolff, 2005].

There is sparse work aimed at understanding and supporting data workers’ workflows.

In the recent years, some research works investigated data workers’ practices and built tools to support them in their work. For example, Boukhelifa et al. and Skeels et al. studied how data workers cope with uncertainty in data [Boukhelifa et al., 2017; Skeels et al., 2008], and Liu et al. studied how data workers explore and manage alternatives in analysis, e.g., with data sources and analysis methods [Liu et al., 2020]. Researchers have also looked at how data workers visually explore model simulations in biological applications [Boukhelifa et al., 2019], finding that data workers often have to manage multiple competing objectives, and require expertise across multiple topics to finish their analysis task. Kandel et al. studied enterprise data workers to find, among other things, that there is a disparity in programming proficiency among data workers [Kandel et al., 2012].

### 2.7.2 End-User Programming

Since text-based programming is a common way data workers perform analysis, we look to existing literature on end-user programming to get more insights in data workers’ practices.

“End-users” is a term used to refer to non-professional pro-

data scientists.

grammers. Indeed, not all data workers are non-professional programmers: A computer scientist might be proficient in programming, but might not be well-versed in data science techniques and principles. Back in 2006, in U.S., it was estimated that there are about 12 million end-users, far more than the number of professionals, who are estimated at 3 million [Myers et al., 2006]. Although many end-users might use GUIs and visual programming interfaces for their ease of use, end-users in certain fields use conventional text-based programming for the power and flexibility it offers.

End-users are non-professional programmers, who can benefit from tool support to make programming easier.

Prior research by has found that end-users face a lot of barriers in learning programming. Taking advantage of the human visual system, through visual programming [Myers, 1990] and direct manipulation interfaces for programming [Cypher and Halbert, 1993], can help end-users. Developers and researchers have also built programming IDEs to better support programming, e.g., by helping novices write source code in the correct syntax [Miller et al., 1994; Teitelbaum and Reps, 1981] and handle errors in code [Ko and Myers, 2004]. Researchers have also shown that end-users write source code that is error-prone [Rode and Rosson, 2003; Rosson et al., 2004], and tend to rely a lot on code reuse [Blackwell, 2002]. Such findings might also apply to data workers, who are end-user programmers using programming to work with data.

End-users write error-prone code and have an over-reliance on code reuse.

## 2.8 Motivation: Biehler's Vision for Statistical Tools

In 1997, Biehler wrote about his vision for a computer-supported statistical tool that can be used to learn and perform hypothesis testing [Biehler, 1997]. Among other things, he suggested the use of *microworlds,* interactive environments for simulation that can be used by students to understanding statistical concepts. This active learning approach could help students get a solid understanding of statistical concepts, which can help them make better decisions when performing hypothesis testing. He suggested that software could have a more prominent role in statistical education, taking over some of the roles and responsibilities of teachers and tutors. But perhaps

Biehler envisioned a statistical tool to help learn and perform hypothesis testing.

the biggest idea proposed by him is to merge the tools for learning and doing statistics; beginners need to be able to do statistics when they learn them and the other way around. As Biehler puts it, *"a single adaptable tool may become all we need."*

Biehler called for changes to statistical tools, such as more cohesion among tools and better adaption to user's expertise.

Biehler also discussed three problems with the then available statistical tools: *complexity of tool problem,* the *closed microworld problem,* and the *variety problem.* Tools developed for professionals might be too hard for beginners, e.g., students, to use. The co-evolution of the user and the tool's functionalities is, according to him, an important factor to consider. The closed microworld problem describes the state of many microworlds in statistical tools that are not flexible. These microworlds only work for certain situations, and thus do not have a generative value. The variety problem is illustrated with the various tools out there, which all seem to support analysis in their own way with no cohesion among them.

McNamara built upon Biehler's vision by describing attributes for a tool that unifies learning and performing analysis.

**Have we achieved Biehler's vision?** In early 2000s, TinkerPlot [Fitzallen, 2012] and Fathom[14] were developed by following the guidelines proposed by Biehler. TinkerPlot and Fathom are visualization-driven tools that can be used to teach basic concepts of statistics and probability, such as descriptive statistics, distributions, and clustering. McNamara built upon Biehler's vision by proposing attributes for a statistical tool that can be used to learn and perform statistics [McNamara, 2015]. McNamara calls for no separation between the tools at any level, and her attributes cover a wide range of aspects, such as focus on novices, support for a cycle of exploratory and confirmatory analysis, support for randomization, an emphasis on interactivity, and support for narration, reproducibility, and extensibility.

[14]`https://fathom.concord.org/`

# Chapter 3

# Understanding How Data Workers Gather Knowledge to Perform Data Science

**Publications and Author's Contributions**
The work in this chapter has contributions from Yuliya Sergiyenko and Yue Hu. Part of the research in this chapter has been published as an extended abstract [Subramanian and Borchers, 2017] and a paper [Subramanian et al., 2019a]. The author of this thesis developed the research questions and planned all studies. The author is responsible for the survey and part of the content analysis. Sergiyenko conducted the initial interviews used to understand information needed to perform analysis, which is discussed in Section 3.5 [Sergiyenko, 2018]. Hu conducted the interviews used to understand about use of information sources, discussed in Section 3.6, as well as the content analysis of queries on Q&A platforms, discussed in Section 3.8 [Hu, 2019].

Data workers prepare to perform data analysis by gathering information about theoretical concepts, tool usage, and practical statistics.

There are several tasks that preclude an analysis task. In addition to designing and carrying out the experiment, collecting the data, and preparing it for analysis, analysts need to gather information about tool usage, statistical concepts, and domain-specific information about statistical procedures, e.g., knowing that a 95% cut-off threshold is the standard in HCI research. Understanding what information is needed to perform analysis, and how data workers gathering this information can help us gain more insights into the problems data workers face.

Formal education might not prepare data workers for real-world analyses.

Previous research in statistical education [Garfield, 1995] has shown that there is a disparity between what students are taught in school or college, and what information they need to use to solve a real-world problem. By understanding the information gathering process leading up to data analysis, we may be able to gain more insights into this.

This chapter presents empirical research about how data workers gather information to perform analyses.

In this chapter, we discuss how data workers gather information about statistical procedures, the nature of this information, how they decide upon the statistical procedures for their experimental data, and so on. We also compare the various information sources data workers use, such as books, research publications, and interpersonal communication, highlighting a tension between formal and informal sources. Based on this discussion, we show existing gaps in the workflows, which we aim to tackle with our artifact contribution in the next chapter.

We first provide motivation for our work by discussing the difficulties data workers face in learning statistical concepts and selecting statistical procedures in Section 3.1 and then discuss related work about information behavior of researchers and students' learning practice in Section 3.2. To situate our work, we name the considerations of our work in Section 3.3, and discuss our guiding research questions and contributions in Section 3.4. We present our own empirical studies and a survey in which we investigated how data workers gather information to prepare for an analysis in Sections 3.5–3.8. Finally, we address the limitations of our work in Section 3.9.

## 3.1 Background: Statistical Procedure Selection

Selecting statistical procedures can be difficult.

We provide two arguments for why statistical procedure selection could be difficult, particularly for data workers: 1) inherent complexity due to the wide-and-deep decision structure involved and 2) difficulty in developing a clear mental model due to the abstract nature of information in statistical procedure selection. We now expand upon these arguments below.

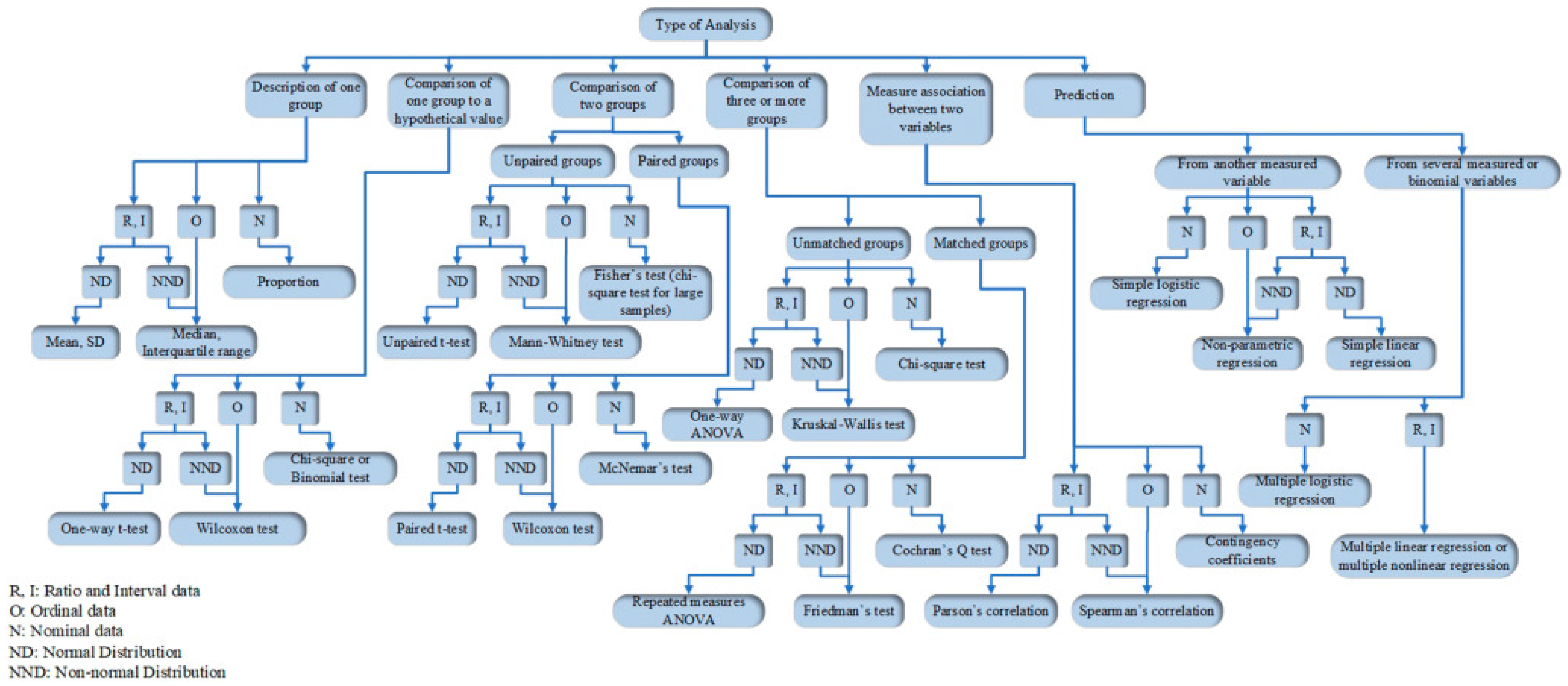

Type of Analysis
Description of one group
Comparison of one group to a hypothetical value
Comparison of two groups
Comparison of three or more groups
Measure association between two variables
Prediction
Unpaired groups
Paired groups
Unmatched groups
Matched groups
From another measured variable
From several measured or binomial variables
R, I
O
N
ND
NND
Mean, SD
Median, Interquartile range
Proportion
Unpaired t-test
Mann-Whitney test
Fisher's test (chi-square test for large samples)
Chi-square test
One-way ANOVA
Kruskal-Wallis test
Simple logistic regression
Non-parametric regression
Simple linear regression
Chi-square or Binomial test
One-way t-test
Wilcoxon test
Paired t-test
McNemar's test
Multiple logistic regression
Repeated measures ANOVA
Friedman's test
Cochran's Q test
Parson's correlation
Spearman's correlation
Contingency coefficients
Multiple linear regression or multiple nonlinear regression
R, I: Ratio and Interval data
O: Ordinal data
N: Nominal data
ND: Normal Distribution
NND: Non-normal Distribution

**Figure 3.1:** A decision tree for statistical procedure selection. Selecting statistical procedures requires making multiple decisions, with each decision a choice among multiple options. Such wide-and-deep decision structure indicates a complex task. The image is taken from [Okinda et al., 2020], CC 4.0.

### 3.1.1 Procedure Selection is Inherently Complex

Data workers need to make several decisions to select statistical procedures.

To illustrate how complex it can be to select statistical procedures, we briefly discuss a scenario. Assume that a data worker has collected data via an experiment to determine the performance of different keyboards. This experimental data involves several independent and dependent variables. For example, independent variables would be the keyboard layout, gender, etc., and dependent variables are standard measures like typing speed and number of typing errors. After the data has been loaded into an analysis software, the data worker views the data and visualizes the data points using various plots to get an overview of her data. She notices that there are some outliers, i.e., extreme data points that are non-representative of the normative user performance. She removes these data points using an outlier removal technique [Ben-Gal, 2005]. After removing outliers, the data worker decides to test if an independent variable influences a measurement, i.e., has an effect. To determine the appropriate statistical procedure, the data worker needs to check the assumptions of the procedure. One of these assumptions is that of normality, i.e., determining if the distributions are Gaussian or normally distributed. However, checking this assumption might not be straightforward.

Deciding statistical procedures requires information of existing methods as well as statistical expertise.

There are numerous tests for normality, such as the Kolmogorov-Smirnov test, Shapiro-Wilk test, Jarque-Bera test, Pearson's chi-squared test, and Anderson-Darling test [Yazici and Yolacan, 2007]. The data worker needs to select the test is that best suits her data and research hypotheses. For example, for highly skewed and log-normal distributions, Jarque-Bera test of normality works well; for symmetric distributions of small sample size, Kolmogorov-Smirnov and Anderson-Darling tests work well; and, although commonly found in analysis software, Shapiro-Wilk test is not recommended for large sample sizes [Yazici and Yolacan, 2007]. Additionally, tests of normality might also require certain assumptions to be held true. In addition to such statistical tests, data workers could also use graphical methods, e.g., QQ plots, to determine if one or more distributions are normal. Despite the plethora of options available for checking the normality of distributions, it is possible that the data worker does not receive a binary result. For example, it is agreed that for large sample sizes, graphical methods are more reliable, and minor deviations from normality would not be an issue [Pallant, 2020].

The discussion above shows that a lot of experience and expertise is required just to perform one step in hypothesis testing, that of checking whether the distributions are normally distributed. Other steps, such as checking for other assumptions, selecting the significance test, and $p$-value corrections (for post-hoc significance tests) can be equally, if not more, complex. Figure 3.1 provides a good visual indication of the complexity involved in selecting statistical procedures.

### 3.1.2 Data Workers May Lack a Clear Mental Model

Since many data workers are novices, they may not completely understand the underlying statistical concepts required for selecting statistical procedures. For example, several studies in the past have shown that even trained researchers have difficulty interpreting $p$-values [Oaks, 1986; Kline, 2004; Beyth-Marom et al., 2008]. Due to this reason, data workers might treat steps in statistical procedure selection as 'black boxes', i.e., view it as an abstract step that yields results. Such an approach might work in standard scenarios. However, since data workers lack a clear mental model, they might struggle to *transfer* knowledge to novel scenarios [Halasz and Moran, 1983]. Therefore, statistical procedure selection might be a complex task for novice data workers who lack a clear mental model of procedure selection, especially when they inevitably face a novel scenario.

Data workers who lack statistical expertise may treat steps in procedure selection as abstractions, and might consequently find it hard to deal with novel situations.

## 3.2 Related Work

In this section, we describe existing work about information seeking behavior, and take a deeper look at research on teaching statistical concepts. Finally, we conclude with some existing research and commercial tools that can be used to self-learn and/or teach hypothesis testing.

### 3.2.1 Statistical Procedure Selection

Even trained data workers might find procedure selection a difficult task.

Issues about statistical procedure selection has been well documented in statistical education research. For undergraduate Psychology students, one of the five learning goals is to be able to select the appropriate statistical procedure [Society for the Teaching of Psychology Statistical Literacy Taskforce, 2014]. Despite this, selecting statistical procedure is a difficult task even for the high performing students in Psychology [Gardner and Hudson, 1999]. Problems with statistical procedure selection extend beyond classroom, and has been well documented in scientific publications [Cairns, 2007; Bakker and Wicherts, 2011].

### 3.2.2 Information Seeking Behavior

Information seeking behavior is a well studied area of research. Of particular interest to our work are research about information behavior of scientists and programmers.

#### Scientists' behavior

Scientists' information seeking behavior is well explored in research.

Researchers have studied scientists' citation behavior [Borgman and Furner, 2002; Bornmann and Daniel, 2008; Leydesdorff and Milojevic, 2012], how they handle scientific literature [Hemminger et al., 2007], and how they formulate and discuss scientific problems in scientific communities [Moore, 2005; Tuominen et al., 2005].

Variations in people, context, and information sources make it hard to generalize findings about information seeking behavior.

Palmer came up with a classification of scientists based on their information gathering process. This includes *nonseekers; lone, wide rangers,* who worked alone to gather a lot of information; *unsettled, self-conscious seekers,* who work in groups and do not have a focused topic; *confident collectors,* who are experts in their field; and *hunters,* who sought specific research findings [Palmer, 1991]. This serves to illustrate why studying information seeking behavior is difficult: there are immense variations among people, situations, and information sources that make it hard to generalize findings.

For this reason, many researchers have attempted to model information seeking behavior of users in a generic manner. Prominent examples are the Ellis [Ellis, 1989], Kuhlthau [Kuhlthau, 2004], and Savolainen [Savolainen, 1995] models. Although these models give us an overview of various steps in information behavior tasks, commonly known as the *search process* [Whitmire, 2000], they do not apply well to specific tasks and users.

There are many generic models of information seeking behavior.

There are still open questions and challenges about scientists' information seeking behavior that motivated our studies. First, there is a paucity of research aimed at scientists' use of information sources [Case, 2016]. Second, information about appropriate statistical procedure is available in several information sources, many of which are informal sources like web forums. Such sources often have conflicting, outdated, and even incorrect information. Therefore, it might be particularly difficult for data workers to navigate existing information sources to determine the appropriate statistical procedure.

Our research is motivated by many open questions and challenges in understanding information seeking behavior of data workers.

**Programmers' behavior and use of Q&A platforms**

In addition to studies about scientists' behavior, existing research works about programmers behavior, especially with Q&A platforms, are also relevant to our work. Programmers spend a majority of their time seeking information [Singer et al., 1997], which makes them an natural choice for researchers to study. O'Brien and Buckley proposed a five-stage, non-linear model for programmers' information seeking behavior during software maintenance [O'Brien and Buckley, 2005]. One key finding is that since programmers use a variety of information sources to obtain information, a lot of time goes into browsing and differentiating these information sources based on programmers' requirements for using the information. Seaman found that programmers evaluate the quality of information sources based on the quality of the source code presented in them, and highly value information from a informationable person [Seaman, 2002]. This increased reliance on interpersonal sources of information is well established in research on information seeking, e.g., [Hertzum and Pejtersen, 2000; Yitzhaki and Hammershlag, 2004]. Nevertheless, so-

Programmers tend to carefully curate information they collect from various information sources.

cial risk (e.g., embarrassment from asking questions) and poor availability of interpersonal sources might prompt users to seek other information sources.

StackOverflow is a popular Q&A platform that relies upon the expertise of community members to moderate quality of questions and answers.

StackOverflow, a popular Q&A platform, receives 100 million visitors every month [Stack Overflow, 2021]. Visitors can ask questions, post answers to existing questions, and browse existing questions and answers. Questions and answers can be voted on by community members; this helps moderate quality of content on the platform. Prior research about Q&A platform use has investigated the dynamics of user behavior, e.g., [Bachschi et al., 2020]; the influence of culture, gender, and domain expertise on such platforms, e.g., [Oliveira et al., 2016; Calefato et al., 2018; Nivala et al., 2020]; and quality of source code answers provided on these platforms, e.g., [Treude and Robillard, 2017].

## 3.3 Research Considerations

Based on the above discussion, we summarize the key considerations of empirical studies carried out in this chapter:

1. Data workers might work with limited experience and expertise on statistical concepts.
2. Although statistical education occurs over a long period of time, statistical procedure selection typically occurs in a limited period of time leading up to the analysis. One evidence of this *just-in-time learning* approach is the proliferation of questions about statistical procedure on Q&A websites like StackExchange and CrossValidated.
3. Data workers often have to balance several criteria to select a statistical procedure, such as statistical power, feasibility of the implementation, familiarity, and acceptance in the community.
4. Data workers may have to deal with conflicting opinions about the appropriateness of statistical procedures. Because of the proliferation of informal sources like Q&A websites, weeding out unreliable information could be a challenge.

5. Unlike many other information seeking tasks, selecting the appropriate statistical procedure can be highly crucial as it can lead to the rejection of a publication. For this reason, data workers also need to place a high value on the credibility of the information source when making the choice.

6. In general, information seekers are prone to *satisficing,* i.e., choosing the first acceptable solution to a problem [Berryman, 2008; Warwick et al., 2009].

## 3.4 Research Questions and Contributions

Through our research about how data workers prepare to perform data science tasks, we seek to answer the following questions:

- What are the different types of information data workers seek immediately before performing data science tasks, i.e., during just-in-time learning?
- What information sources do data workers use to gather information about statistical procedure. How do these sources compare against each other? What criteria do data workers use to determine which resource to use?
- What strategies and coping mechanisms do data workers use in order to select a statistical procedure?
- How do data workers use Q&A websites, which are one of the most common information sources used by our study participants? What information do data workers present in the questions, what is the quality of answers like, and what are some of problems respondents face?

We attempt to answer the above questions via three contributions:

1. The first contribution involves findings from twelve interviews and three observations. We employed semi-structured interviews with contextual walkthroughs to

gain an understanding of the various types of information data workers seek in preparation for their data science task. Additionally, we also investigated what sources are used for gathering these different types of information, and how they compare against each other.

2. The second contribution involves findings from twelve interviews. Unlike the first contribution, these interviews were conducted with participants from a wide range of fields, such as business statistics, HCI, psychology, and automotive engineering. The focus of these interviews were about information seeking behavior for the selection of statistical procedures. We investigated information sources, our participants' criteria for selection of statistical procedures, and their coping strategies.

3. The third contribution is a content analysis of 76 questions and answers from Q&A websites. This was done as a follow-up to our second contribution, and deals with how information is presented in the Q&A websites, how respondents handle questions with incomplete/missing information, and so on.

## 3.5 Study: Understanding Knowledge Needed to Perform Analysis

We first describe our data collection and analysis method, and then present the key findings from our analysis.

https://osf.io/dwy43/?view_only=1df362f3d15843ad965bdfa140ba259c[a]

[a]Contains our interview/walkthrough protocol, questionnaire used for interviews, an example analysis with work modeling tools (redacted personal information), and the final affinity diagram.

| No. of participants | When the analysis was carried out |
|---|---|
| 4 | 1–2 weeks ago |
| 1 | 1 month ago |
| 2 | 2 months ago |
| 1 | 6 months ago |

**Table 3.1:** Our participants' experience with analysis

### 3.5.1 Data Collection and Method

To investigate how data workers prepare for a real-world analysis task, we interviewed twelve data workers (three female) from a local HCI lab. We initially used convenience sampling [Saunders et al., 2019] to recruit participants, but eventually aimed to recruit participants with more experience in conducting analysis. We recruited participants via email, in which we a) clarified the purpose of the user study, b) mentioned our key requirement that the participant must be planning to perform an analysis within the next three months, and c) requested the receiver to forward the email to other potential participants, in order to facilitate *snowball sampling* [Goodman, 1961].

We interviewed twelve data workers to understand how they prepare for analysis.

Our participants include two undergraduate, eight graduate students enrolled in Master's programs, and two doctoral students. All participants were from an engineering background. All but the two doctoral students considered themselves novices, whereas the doctoral students self-reported to be experts. All participants but one had performed at least one analysis involving hypothesis testing in the past two months (median = 10 days). The interviews took place over a period of four months.

Study participants include self-reported novices and experts, and have experience conducting analysis.

We initially aimed to observe all participants as they prepare for a real-world hypothesis testing task. However, only three participants had on-going analysis during our study. For the remaining participants, we invited them to walk us through their recent analysis in interviews, and used this as prompts for our interview and observation sessions. On average, interviews lasted 45 minutes and observation sessions, which included discussions between the observer and participant, lasted 60 minutes.

Observations were not always possible since some participants did not have on-going analysis for us observe.

We used a protocol and questionnaire[1] to guide our interview and observation sessions. The questionnaire also acted a checklist of topics to cover, and questions were interwoven non-verbatim during the discourse with the interviewee. For later interviews, we redesigned the questionnaire to get more details on specific topics, such as the resources used by our participants. In our study sessions, we first collected demographic information from our participants and briefed participants about the study purpose and procedure. We then sought to understand their background in HCI and hypothesis testing. Subsequent questions were aimed at gaining an understanding of their previous analyses, particularly how they prepared for their analyses. In addition to the interviews and observations, we also carried out member checks four weeks after the study session. Both interviews and observations were conducted in the lab at which all participants worked. During the study session, participants were rewarded with drinks and snacks.

We collected demographics, background of participants, and our participants' workflows with previous analyses through interviews and observations.

To prepare for our analysis, we prepared full transcripts of both the interview and observation sessions. One researcher analyzed the transcripts by using the recordings from observations as supplements. This researcher and the author of this work did work modeling, by using an affinity diagram and flow diagram [Beyer and Holtzblatt, 1997] to organize and understand the key insights from analysis. A total of 390 insights were structured and organized into an affinity diagram [Holtzblatt et al., 2004].

We use an affinity diagramming technique to organize study insights.

### 3.5.2 Findings

We grouped the main insights in our affinity diagram, which resulted in the following findings:

#### Significance of formal education

We first wanted to understand the role of formal education in performing hypothesis testing. We find that although all participants had taken a statistical course earlier, most (11 out of 12

Although all participants were formally trained, they needed to re-learn statistics to perform analyses.

[1]Both protocol and questionnaire are available in supplements.

participants; 92%) had to learn statistics again to prepare for real-world analyses. Participants provided the following reasons (ordered from most to least frequent) for having to re-learn statistics:

- *The courses were taken a long time ago* (10 out of 12 participants; 83%).
- *The courses had no practical examples* (9 out of 12 participants; 85%).
- *The courses were not adequate to help perform real-world analysis* (7 out of 12; 58%).

The second and third reasons indicate that courses on hypothesis testing focus on theoretical concepts, and that the knowledge transfer to real-world tasks is difficult.

**Perceived incompetence in hypothesis testing**

Some participants associated negative connotations with hypothesis testing.

In addition to the reasons discussed above, half of the participants (6 out of 12 participants; 50%) mentioned that they had originally lacked the motivation to learn hypothesis testing, which is why they had to re-learn it. These participants used the following terms when conversing about hypothesis testing: “very hard”, “necessary evil”, “not enjoyable”, and “least favourite”. It is possible that these participants’ own perception of competence or incompetence with hypothesis testing might have affected their attitudes and perceptions about it [Frey and Ruble, 1987; Stodolsky et al., 1991]. Few participants had positive associations with hypothesis testing: “fun”, “interesting”, and “useful”.

**Information required to perform real-world analysis**

We identified three groups of information that our participants required in order to perform real-world analysis: *theoretical foundations* of statistics and probability, *practical statistics,* and *tool usage.*

A limited amount of theoretical information is enough to perform hypothesis testing.

Theoretical foundations include information of fundamental statistical concepts, such as Gaussian distributions, measures of central tendency and spread, and the central limit theorem. These concepts are part of introductory courses in statistics and probability, which, as we discussed earlier, most of our participants had taken. In general, participants agreed that a limited information of theoretical concepts is adequate for performing hypothesis testing, and that performing statistics required skills that were not taught in statistical courses.

Tool usage informs participants how to use a statistical software.

Tool usage involves expertise in using a statistical tool, be it a GUI or programming. Participants used a wide range of tools. We discuss how participants decide upon tools to use in Section 3.5.2. Four participants (33%) mentioned that their statistical courses used simpler statistical tools, if at all, than what was needed for their real-world analysis.

Practical statistics is a bridge that connects information of theoretical concepts to tool usage.

Practical statistics relates to the practical information or know-how required to perform analysis. *"Which test should I apply here?", "How do outliers affect my distribution?",* and *"What happens if my data are left skewed?"* are some example questions our participants had that seek to illustrate practical statistics. Practical statistics requires information of procedure selection but also the relationship between the various statistical concepts. Our more experienced participants mentioned that this is a skill they could develop only over time, by facing new scenarios that required them to deviate from the statistical procedures typically used. Practical statistics is essential to perform a real-world analysis, but our participants could gain them only through experience.

### Just-in-time learning practice

Participants reported to work under a time constraint, which affected their choice of information sources.

Our next finding concerns the time constraint under which our participants reported to prepare for hypothesis testing. Participants reported to resort to just-in-time learning to gather information required for performing statistics. As a result, traditional learning methods like books, lecture slides, and online courses, while considered useful, were reported to be minimally used during preparation. Participants also reported that using these information sources required them to spend a lot of time in identifying specific information relevant to their analysis. Participants mentioned that tutorials like Practical Statistics for HCI[2] [Wobbrock, 2011] and hcistats[3] helped them determine statistical procedures as well as tool usage with hypothetical datasets. Online searches and web articles helped students answer specific questions about practical statistics, but often had inaccurate or unreliable information. Even when students found reliable sources, they had to contextualize the information in those sources to their analysis, which is not always straightforward. We compare these information sources in more detail in Section 3.6.

### Allocating time for analyses

Most participants underestimated the time it would take for them to perform hypothesis testing.

Since participants gathered information needed for their analysis only shortly before their analysis began, we were curious if they were able to finish their analysis as planned. 7 out of 12 participants (58%) reported that their analysis tasks took more time than they had originally expected. The delay was attributed to difficulties in learning statistics (6 out of 7 participants; 86%), learning the analysis tool (6 out of 7 participants; 86%), and problems in experimental data collection (3 out of 7 participants; 43%). Therefore, as we discussed earlier, e.g., in Section 3.1, selecting the appropriate statistical procedure and performing analysis are complex tasks.

### Deciding upon the analysis software

Participants decided upon statistical tool based on interpersonal recommendations, expense, and familiarity.

Our findings also shed light on how our participants select tools for analysis. Participants reported to use the following criteria (ordered from most to least frequent) to decide upon the statistical tool for their work:

- The tool was used and/or recommended by colleagues and supervisors (8 out of 12 participants; 67%).

[2]`https://depts.washington.edu/acelab/proj/ps4hci/`

[3]`https://yatani.jp/teaching/doku.php?id=hcistats:start`

- The tool was available for free (8 out of 12 participants; 67%).
- The tool was familiar (4 out of 12 participants; 33%) and came up on a Google search (3 out of 12; 25%).

### 3.5.3 Summary of Key Findings

From our analysis, we find that

- Formal education does not prepare data workers adequately for real-world analyses.
- Since hypothesis testing may be highly complex for certain novice data workers, they develop negative connotations about hypothesis testing.
- Three types of information are required for analysis: Theoretical foundations, tool usage, and practical statistics, which acts as a bridge between theoretical foundations and tool usage.
- There is more evidence that statistical procedure selection and carrying out an analysis are complex tasks.

## 3.6 Study: Understanding Resource Use

https://osf.io/rm9ba/?view_only=dba6b1b49d72416bba1c4d26f528e569 [a]

[a]Contains the final coding scheme from our analysis discussed in this section as well as the first few pages of a transcript with codes, to illustrate our analysis process.

To understand resource use, we conducted interviews and a survey.

To expand on our previous study, and to gain a comprehensive understanding of how data workers select statistical procedures in their work, we conducted detailed interviews with data workers. We chose to use qualitative methods, such as interviews and grounded theory analysis, as they are well suited to help understand a complex process [Charmaz, 2006]. We describe our data collection and method, and then discuss the key findings from our interviews. To triangulate our interview findings, we conducted an online survey, the results of which are discussed in Section 3.7.

| *ID* | *Discipline* | *Status* | *Methods used* |
|---|---|---|---|
| IP01 | Business statistics | MSc | Decision trees, GARCH models |
| IP02 | Applied psychology | PhD | ANOVA, chi-squared tests |
| IP03 | Business statistics | MSc | Regression, PCA, clustering |
| IP04 | HCI | PhD | ANOVA, frequency tests |
| IP05 | Cognitive neuroscience | PhD | ANOVA, regression |
| IP06 | Computer science | PhD | Neural networks, random forests |
| IP07 | Risk and model analysis | Ind | n-gamma model, regression |
| IP08 | Statistics | Ind | Regression |
| IP09 | HCI | PhD | $t$-test, chi-squared tests |
| IP10 | Automotive engineering | MSc | Correlation, neural networks, regression |
| IP11 | Psychology | MSc | Structural equation model, ANOVA |
| IP12 | Labor economics | MSc | DID, regression |

MSc: Master's student; PhD: PhD student; Ind: Industry practitioner.
GARCH: Generalized AutoRegressive Conditional Heteroskedasticity; PCA: Principal Component Analysis; DID: Difference in Differences analysis.

**Table 3.2:** Details of participants we interviewed about resource use.

### 3.6.1 Data collection

We used purposive sampling [Etikan and Bala, 2017] to recruit 12 participants from our personal and professional networks (six female, median age = 26.5, will be referred to as IP01–12) via university mailing lists and word-of-mouth. Purposive sampling gave us the best chance of identifying information-rich participants. At various stages of the sampling process, we looked for data workers with specific experience level and whether they worked in industry or academia. To improve external validity, we sampled participants from various domains, such as HCI, psychology, and economics. Since procedure selection is also a common issue in machine learning, we included three participants (IP06, IP07, and IP10) who used sta-

Our participants came from various disciplines, worked on various data science tasks, and included two industry practitioners.

tistical machine learning techniques, such as neural networks and classification. Most of our participants are graduate students enrolled in a graduate program. We sampled two industry practitioners (IP07 and IP08) as we were curious to learn about their workflows.

Participants performed various analysis tasks.

Except for IP06, all participants used statistical inference methods, such as regression and significance tests. For example, IP01 analyzed house renting situation based on data from renting websites, IP02 investigated how the position and layout of images in an advertisement could affect users' responses, and IP07 used natural language processing techniques to classify whether comments on a website were useful or spam. Interviewees' experiences also varied substantially. Graduate students pursuing their Master's degree usually had only 1–3 years of experience with analysis techniques, whereas our most experienced interviewee, a doctoral student, had over seven years of experience. For participant details, see Table 3.2.

### 3.6.2 Method

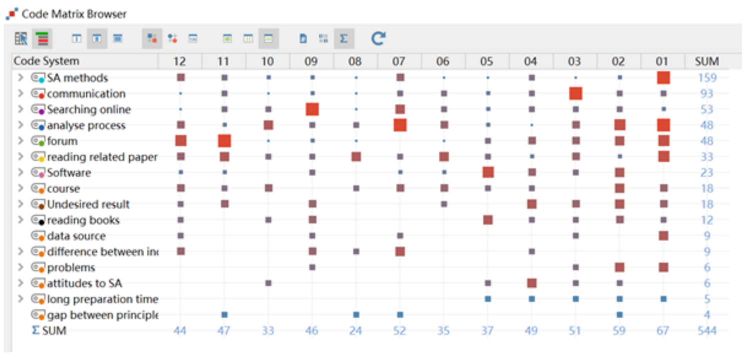
#### Semi-structured interviews

In our interviews, we focused on information source usage aimed at helping select statistical procedures.

Before interviews, each participant gave their consent to be recorded and have their data collected. We conducted semi-structured interviews to understand how our participants seek information about statistical procedure. In semi-structured interviews, participants are asked several key, predetermined questions about topics of interest, but not in a specific order or with a particular phrasing [Adams, 2015]. The semi-structured nature of the interviews allowed us the flexibility to focus on particular topics, depending on how the interview transpires. However, since the semi-structured interviews have some structure, the data coding process is more effective, than when employing unstructured interviews [Minichiello et al., 2008]. We developed and used a questionnaire for the interview, which continued to evolve over the course of the interviews.

Our interviews consisted of three phases.

Our interviews consisted of three phases. During the first phase, we collected background information about our participants. This included questions such as *"How long have you*



Code Matrix Browser

| Code System | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | SUM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SA methods | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | 159 |
| communication | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | 93 |
| Searching online | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | 53 |
| analyse process | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | 48 |
| forum | ■ | ■ | ■ | ■ | ■ |  | ■ | ■ | ■ | ■ | ■ | ■ | 48 |
| reading related paper | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | 33 |
| Software | ■ | ■ |  | ■ |  | ■ | ■ | ■ | ■ | ■ | ■ |  | 23 |
| course | ■ | ■ | ■ |  | ■ | ■ | ■ | ■ | ■ |  | ■ | ■ | 18 |
| Undesired result | ■ | ■ |  | ■ |  |  | ■ |  | ■ | ■ | ■ | ■ | 18 |
| reading books | ■ |  | ■ | ■ |  |  |  | ■ | ■ | ■ | ■ | ■ | 12 |
| data source | ■ |  |  | ■ |  | ■ |  |  |  | ■ |  | ■ | 9 |
| difference between in | ■ |  |  | ■ | ■ | ■ |  |  | ■ |  |  |  | 9 |
| problems |  |  |  | ■ |  |  |  |  |  | ■ | ■ | ■ | 6 |
| attitudes to SA |  |  | ■ |  |  |  |  | ■ | ■ | ■ | ■ |  | 6 |
| long preparation time |  |  |  |  |  |  |  | ■ | ■ | ■ | ■ | ■ | 5 |
| gap between principle |  | ■ |  |  | ■ | ■ |  |  |  |  | ■ |  | 4 |
| Σ SUM | 44 | 47 | 33 | 46 | 24 | 52 | 35 | 37 | 49 | 51 | 59 | 67 | 544 |

**Figure 3.2:** The resulting categories after qualitatively coding the interview transcripts. We performed two cycles of coding in MAXQDA software.

*been doing hypothesis testing?"* and *"Have you taken any hypothesis testing course?"*. During the second phase, participants were asked to walk the interviewer through a recent analysis task. Through this walkthrough, we aimed to understand how participants chose the statistical procedure, how they selected the statistical tool, and how they dealt with the problems that came up during these steps. Our questions in the third phase were aimed at understanding our participants' attitude towards hypothesis testing as well as how they use various information sources. When possible, we encouraged participants to present their artifacts, e.g., their analysis or browser history, to get a more deeper insight into our participants' workflows.

Interviews lasted 45 minutes on average. The shortest interview was 25 minutes long, and the longest interview was over one hour. The length of the interviews depended on the stage of theoretical saturation achieved and on the amount of data gathered. Interviews were conducted face-to-face where possible, but some interviews were conducted via Skype.

Interviews varied in length, depending on the stage of theoretical saturation.

### Analysis method

We recorded the audio of the interview sessions, and then prepared full transcripts. We used a grounded theory approach, a well-known qualitative method used to explore relationships and complex phenomenon. This approach particularly suited

We used qualitative methods to analyze interview transcripts.

our goal of understanding a process or interaction [Creswell, 2013].

We performed two cycles of coding, and eventually grouped codes into 14 categories.

We performed two cycles of qualitative coding on these transcripts and field notes: *Open coding* [Corbin and Strauss, 2014] followed by *focused coding* [Charmaz, 2006]. During open coding, we added sentence-level codes to our transcripts. During focused coding, we grouped the codes we developed (and iterated on) in the first phase, and allowed theoretical categories to emerge. We followed the methodology described in most qualitative research manuals, e.g., [Saldaña, 2013]: We wrote memos, discussed alternate explanations, and used a constant-comparative method to gather data for validating our explanations. Overall we identified 544 codes, which were subsequently grouped into 14 categories as shown in Fig. 3.2.

### 3.6.3 Findings

We present the main findings from our analysis, grouped into three topics: 1) Uncertainty in statistical procedure selection, 2) coping strategies for procedure selection, and 3) information sources used for selecting statistical procedures.

#### Uncertainty in statistical procedure selection

Prior research has shown that even highly trained students may find it difficult to select the appropriate statistical procedure [Gardner and Hudson, 1999]. Our participants also considered statistical procedure selection to be complex, and gave specific reasons.

Uncertainty about procedure selection arises mainly from methodological concerns over the use of frequentist hypothesis testing.

Several participants (P03, P04, and P09–12) involved in academic research reporting feeling uncertain about statistical procedure selection. Even though hypothesis testing has been the standard technique for statistical inference, researchers have recently raised concerns over its use, e.g., the over-reliance on $p$-values. As a result, there is a proliferation of research works that propose alternative approaches (e.g., [Kay et al., 2016; Dragicevic et al., 2019]), as well as practical changes and guidelines to the current approach (e.g., [Cockburn et al., 2018; Transparent Statistics in Human–Computer Interaction Working Group, 2019]). Such changes require academic researchers to make changes to their existing analysis workflows, sometimes drastically by having to adopt new workflows. Our participants reported that since “things were always changing” (IP4), they had to be cautious about statistical procedure selection:

> *“There is a lot of uncertainty with test selection and there are no standards [about which procedure is best]. For me, it is hard to know which test to use.”* - IP4

Some participants reported to feel apprehensive about procedure selection, which sometimes manifested itself as a form of learned helplessness (*“I never start by selecting the significance test to analyze my study data, because I am not sure I can do it properly!”* – IP01). Such apprehension is warranted for some participants, who had used an incorrect method themselves, or recommended it to a colleague. For example, IP03 had used an incorrect method earlier in their work that rendered their findings invalid (*“[...] we analyzed factors that affect the winning percentage of each team in the NBA. I chose the wrong method for it.”* – IP03), and IP04 had suggested incorrect methods to others (*“I once recommended an analysis to a colleague, and it turned out to be wrong. I am now reluctant to answer questions about statistics [...]”* – IP04).

Some participants reported to delay procedure selection as they were apprehensive of the task.

**Coping strategies**

Since reporting the results of an incorrect procedure can result in a publication getting rejected, our participants reported to employ various strategies to minimize chances of incorrect selection. As one can expect, most participants (83%) reported to use the statistical procedure that is most accepted and common in their community (*“Regression is the most common method for influence factor analysis in my research field.”* – IP12). The benefits of selecting a well-accepted, prevalent method seemed

Participants prefer to use well accepted, familiar procedures.

to outweigh any potential advantages an alternative method could offer (*"There are always different approaches to do the same thing. Several methods could be correct, and each has [its] advantages. In those situations, I just use the familiar one."* – IP04). Interestingly, we also observed this strategy among our more experienced participants, IP05 and IP09, who had the knowledge and expertise required to adopt an alternative method (*"Generally, there are fixed routines when you are doing projects. Normally we will use ANOVA because it's easier and better to use, and it's a common method in psychology experiments."* – IP05). This indicates that there is a general reluctance to adopting new statistical procedures.

Participants find it hard to convince stakeholders of alternative procedures, and want to avoid risking rejection of their publication.

When asked about this reluctance, our participants reported that even when they were willing to adopt better methods, getting the stakeholders, e.g., supervisors, reviewers, and clients, on-board was difficult. For example, IP07 conducted a risk analysis by applying an unconventional but valid method, but had to revert to using regression models and decision trees later upon the insistence of their superior (*"My boss told me that in the fields of p2p (peer-to-peer) and banking, the most frequently used method is regression as the results are easier to interpret."* – IP07). Indeed, participants who conducted academic research were also cautious about using new methods as this might put the acceptance of their paper in jeopardy (*"[...] if I use this method in my paper, will the reviewer of the paper know this approach? If they won't, will they reject my paper? So, normally we will just stick to the methodology already out there [...]"* – IP04).

Participants are hesitant to use new statistical procedures, and satisfice even when they are aware of better procedures.

Satisficing behavior is prevalent in information behavior research [Berryman, 2008]. Nevertheless it is concerning when data workers satisfice with an inferior method with the knowledge of better methods (P07, P08, and P10): (*"I have discussed this with a credit data analysis company [...] I asked if they use deep learning models for analysis, they said 'no, we don't use such a complex method, we just use simple logistic regression, even though the former could get a better result.'"* – IP08).

### Information sources to help with statistical procedure selection

In our interviews, we also investigated the various information sources our participants used to help select statistical procedures. We identified five types of information sources: books, research publications, interpersonal sources, Q&A platforms, and other websites.

Books have depth of information but finding information specific to a situation might take time.

**Books:** Books are a traditional medium to gain information. Many participants (P01–05, P07, P09, and P12) reported reading books to gain a deeper understanding of the statistical methods (*"I read some books about the methodology I'm going to use. I know the overall methodology, but some specifics I sort of identify on-the-fly."* – IP04). However, as we discussed in our previous study in Section 3.5.2, most participants did not report using books to help decide on the statistical procedure for specific situations, as finding the relevant, specific information can take up significant time. Overall, books were used to gain in-depth information, but were not preferred to decide upon statistical procedures.

Participants reported to use existing publications as a template for their research and analysis.

**Research publications:** All participants who conducted research in academia reported to use research publications to help decide on statistical procedures. However, the prominent reason was not that publications had high credibility or introduced novel, state-of-the-art procedures. Instead, participants sought publications that reported an analysis of a study similar to theirs, in order to gain a template for their own analysis. This gave them an assurance that the statistical procedure they use in their analysis will be accepted by the reviewers (*"When I get my topic, I will read lots of reference papers, determine what methods they use [...] I will probably use the same one."* – IP05). Some participants also reported to use these publications to identify possible measurements and factors (*"I will search for some paper related to this field, find the factor mentioned in the paper, and for each factor determine the independent variables that were used."* – IP03). Some participants who conducted research in analysis-heavy disciplines, such as risk analysis and business statistics, reported to rely upon publications to be informed about the state-of-the-art statistical procedures (P01, P03, and P12).

Participants considered interpersonal sources to be credible, but most use this source only to validate their choice of statistical procedure.

**Interpersonal sources:** Almost all participants (IP01–05 and IP07–12) agreed that seeking the validation of experts would be the most credible, time-saving approach (*"... asking others is really important in the work. Normally if you find something by yourself, it is time-consuming. [...] you should combine searching by yourself and asking people [experts]."* – IP07). It is important to note that seeking validation from experts is different from using experts as an information source for determining statistical procedures; with validation, participants want to confirm that their choice is correct (*"I have these-and-these tests, and he will check it, and we agree on which test to use."* – IP09). Having an expert validate their procedure seems to reduce the uncertainty around procedure selection (*"(Referring to books) ... even if I read all these [books,] I would still not be sure... especially if it is just [myself] making the decision."* – IP09). One issue with this approach is that experts may not always be available—five participants reported facing difficulties finding an expert in a timely manner.

Q&A platforms are popular among our participants, despite concerns over credibility of responses.

**Q&A platforms:** Almost all participants (IP01–07 and IP09–12) reported using Q&A platforms like StackExchange[4], ResearchGate[5], and Zhihu[6]. While some posted questions about statistical procedure selection on these platforms, most reported to discussions of existing queries that had described a scenario similar to theirs. Participants' opinions about Q&A platforms were mixed. What primarily attracted our participants was the possibility to receive answers from many able experts in a timely manner (*"I asked the question I mentioned before, related to principal component analysis. There [were] 4 or 5 replies within two days."* – IP12). However, most participants mentioned that browsing existing discussions to find the relevant one can be difficult, and raised concerns over the credibility of responses. We discuss the use of Q&A websites in more detail in Section 3.8.

Educational websites, blogs, and wikis can help inform statistical procedure selection.

**Blogs and wikis:** Many participants (P04, P05, P09, P10, and P11) reported to use certain websites that were not Q&A platforms, such as educational pages, blogs, and wikis that contained decision trees and tutorials. Participants' opinions about these websites were comparable to that of Q&A platforms, es-

[4]`https://stats.stackexchange.com`
[5]`https://researchgate.net`
[6]`http://www.zhihu.com/`; used in China

pecially for reputable websites, e.g., a blog maintained by a well respected academic researcher.

### Disciplinary variations

Participants' concerns about procedure selection varied according to the discipline they worked in.

We found that there were certain differences in statistical procedure selection across disciplines. Participants who performed machine learning tasks were concerned less with whether a chosen statistical procedure was valid, and instead focused on the utility of statistical procedures. For example, IP6, who built neural networks sought neural network architectures that could yield a better model fit and prediction estimates. Furthermore, for participants who did machine learning tasks, it was not uncommon to use several procedures and compare them before deciding upon one or more procedures. Such an approach was somewhat less common with participants who performed statistical hypothesis testing. Finally, the chosen statistical procedure in machine learning tasks, e.g., neural networks, is seen as a more valuable contribution towards the publication than hypothesis testing. Therefore, there seems to be less satisficing behavior among participants who performed machine learning tasks than hypothesis testing. Despite such differences, we found no differences across disciplines in the information sources used.

## 3.6.4 Discussion

Below we discuss some of the issues that resulted from our analysis.

### Low adoption of alternative statistical procedures

As data workers are hesitant to adopt new alternative procedures, existing procedures continue to be prevalent over time.

We find that since data workers, especially novices, are reluctant to adopt new procedures, procedures that are prevalent in publications and industry practice continues to stagnate. For example, from our analysis of CHI '19 publications, only 8% of papers (25 out of 301 papers that report statistical inference) use procedures that are not significance testing, such as estimation statistics and Bayesian statistics. We expect that as researchers continue to adopt alternative methods, albeit in small numbers, older methods will become less prevalent. Here are some solutions that could help data workers adopt alternative procedures faster:

Our recommendations range from standardization to pedagogical changes.

- Encourage changes and establish standards at a conference level, e.g., CHI attempted to address the replication crisis by encouraging authors to include research materials along with publications[7].
- Provide incentives to papers that adopt alternative methods successfully, such as badges and awards, similar to the attempts to incentivize transparency in research publications [Robson et al., 2021].
- Teach alternative procedures in high school and universities by showing how they are better, e.g., easier to perform and provides better insights, than existing procedures. This can be achieved with the aid of simple computer simulations.

**Learned helplessness about procedure selection**

The helplessness novice data workers develop due to the uncertainty in procedure selection can be addressed by offering more hands-on training at a high-school or university level. Additionally, working with an experienced data worker or a mentor can have a high value, since experienced data workers can offer assurance and guidance in adopting a new procedure.

### 3.6.5 Summary: Key Findings

Based on our analysis of initial interviews, we have identified the following:

[7]https://chi2021.acm.org/for-authors/presenting/papers/guide-to-a-successful-submission

- Data workers prefer commonly accepted statistical procedures due to the insistence of stakeholders or to not jeopardize their publication's acceptance, even when they know that better methods are available.
- Academic data workers use publications as a template, and use them to inform their analysis procedure.
- Data workers prefer validation from experts, either through interpersonal communication or Q&A forums, since it is the path of least effort [Zipf, 2016] when compared to formal sources like books and research publications.
- In situations where finding the right expert is difficult, data workers resort to informal sources like Q&A forums and other websites.

## 3.7 Survey : Understanding Usage of Information Sources

https://osf.io/tjq59/?view_only=cabf188de919402ea45c0184c712c867 [a]

[a]Contains the online survey and results from our survey.

To triangulate our interview findings discussed in the previous section, and to gain more insights about data workers' statistical practice, we conducted an online survey. We now discuss the procedure of our survey, and the key findings from our analysis.

Our survey aims to triangulate key findings from our interviews.

### 3.7.1 Data Collection

We designed a brief 5-minute survey that aimed to collect respondents' background and details about the various information sources they use to help determine statistical procedures. We recruited our respondents by (a) contacting authors of relevant CHI 2019 papers via email, (b) posting on one social

We sent the survey to academic data workers from various disciplines.

media group that includes researchers, and (c) sending the survey to a local HCI lab's mailing list. We identified the relevant papers using an existing survey of the CHI 2019 proceedings [Reinhard, 2020]. We emailed the first authors of the papers that reported to use quantitative methods. To extract the authors' email addresses from proceedings, we used an existing script used for a similar survey of CHI '19 papers [Wacharamanotham et al., 2020]. Respondents had the chance to enter a raffle worth \$30. The survey was closed after five days.

### 3.7.2 Details of survey respondents

Survey respondents were from various backgrounds, most were formally trained in statistics, and most reported to have a lot of statistical expertise.

51 respondents from 15 countries took the survey, with most respondents from Germany (31%) and the US (27%). 49% were academic researchers, i.e., post-docs and professors; 31% were PhD students; 10% were Master's students; 6% were industry practitioners; and the rest (4%) were Bachelor's students. While most respondents conducted their research or worked in HCI, we also had a few respondents from twelve related disciplines, such as psychology and education. 77% reported to be formally trained in hypothesis testing, but over a third of those (44%) reported that the training did not prepare them well enough for real-world analyses. Most respondents had plenty of experience with hypothesis testing, with 22% reporting to have performed 5–10 analyses and 33% performing over 10 analyses in the last year. Only 20% reported performing just one or two analyses in the last year.

We now present the key findings from our survey.

### 3.7.3 Certainty About Statistical Procedure Selection

Many of our interview participants reported to be uncertain about their choice of statistical procedure. To validate this finding, we asked our survey respondents to rate how confident they were in selecting statistical procedures (*"In general, how confident are you that the statistical procedure you have chosen is correct?"*).

Most respondents reported to be confident about their choice of statistical procedure.

Most respondents (77%) rated a 5 (*'Very confident'*) or 4 (*'Con-*

*fident'*). Only 23% provided a rating of a 3 (*'Neutral'*) or 2 (*'Not confident'*). This contrasts our interview finding, in which we found that most interview participants were not confident in their choice of statistical procedure. We have two potential explanations for this. First, since many of our survey respondents are quite experienced, they might have higher levels of confidence in statistical procedure selection. Second, this increased confidence might be the result of interpersonal discussions with experts and/or the publication getting accepted, and might not be indicative of the respondents' initial uncertainties about their choice of statistical procedure.

### 3.7.4 Perceived Complexity of Procedure Selection

More than half of our survey respondents consider statistical procedure selection a complex task.

Since many interview participants reported to find the selection of statistical procedures complex, we wanted to validate this in our survey. We asked respondents to rate their perceived complexity of statistical procedure selection (*"Based on your experience, how complex do you think finding the correct statistical procedure is?"*). The results validated our interview finding, with over half of our survey respondents (53%) providing a rating of a 5 (*'Very complex'*) or 4 (*'Complex'*). Only 10% rated a 2 (*'Straightforward'*) and the rest (37%) rated a 3 (*'Neutral'*). Most respondents (63%) who rated a 2 or 3 had conducted five or more analyses, leading us to believe that inexperienced researchers consider procedure selection a complex task.

### 3.7.5 Timeline of Research Process

Most participants decide upon statistical procedures after collecting some or most data from their experiment.

In our first interview study, we had found that many participants did not allow themselves enough time to decide upon the statistical procedure and perform analysis. Therefore, we were curious about the stage in a research process at which data workers decide upon statistical procedures. To determine this, we asked respondents to select from these stages in research process: before collecting any data, after collecting some data, or after collecting all data. The results show that more than a third of our respondents decided upon a statistical procedure before collecting any data (37%) or after collecting some data (33%), with the rest (29%) making the decision after collecting all data. This shows the prevalence of preregistration practices among our survey respondents, most of whom were experienced academic researchers.

### 3.7.6 Information Sources

Survey respondents were asked to choose which information sources they used.

From our interviews, we had found that our participants used a variety of information sources, with most relying upon expert validation, either through interpersonal discussions or Q&A platforms. To validate this, we asked our survey respondents to select all the information sources they use to help decide on statistical procedures: books, research publications, discussion with experts, Q&A platforms, other websites, and lecture notes or materials. The results confirmed our suspicions, with discussion with experts being the most popular choice (chosen by 84% of our respondents), followed by Q&A platforms (65%), and formal information sources (books: 63%, publications: 61%). Lecture notes or materials were the least popular (45%), a reaffirmation that formal training tends to focus less on practical statistics as we discussed in Section 3.5.2.

We asked respondents to rate information sources on various criteria.

Next, to understand how our respondents valued these various information sources, we asked them to rate each source on the basis of trustworthiness (*"How reliable is the information?"*), speed (*"How quickly can you get the information?"*), and availability (*"How readily available is the information?"*). The available ratings ranged from 'Very bad' to 'Very good'. These factors were based on existing research on information behavior [Pinelli et al., 1991] and the insights we gathered from interviews. Not surprisingly, books and research publications were considered the most trustworthy. Discussion with experts were also considered very trustworthy, but Q&A platforms and other websites were considered less trustworthy. Formal sources like books and publications were considered to take more time to obtain information than informal sources like Q&A platforms and other websites. Experts were the least available, followed by books and publications. Therefore informal sources are more accessible but less trustworthy, whereas experts are more trustworthy and act as a quick source of information but are less available.

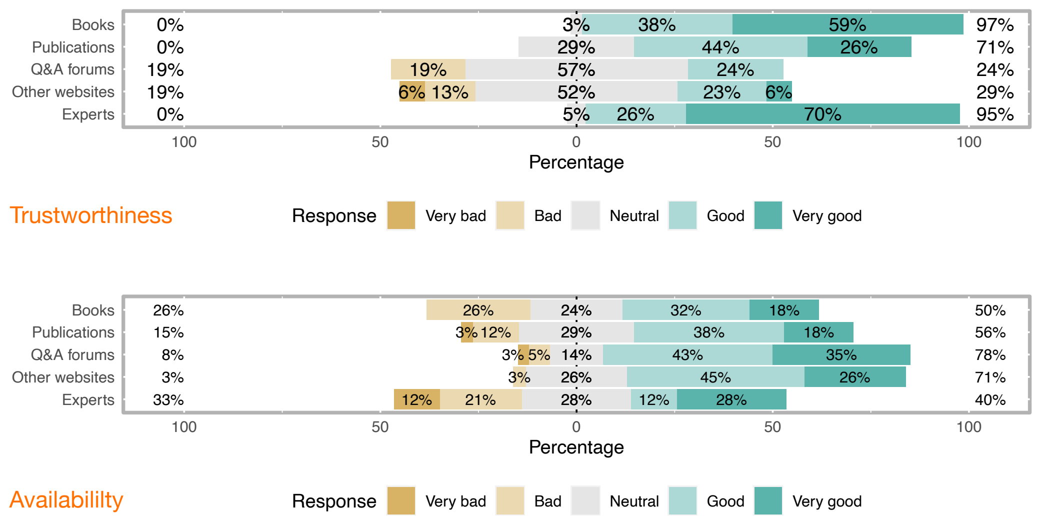


**Figure 3.3:** Results of our Likert scale questions, in which our respondents rated the five different information sources—books, publications, Q&A platforms, other websites, and discussion with experts—on the basis of trustworthiness (above) and availability (below). Our results show that discussion with experts is considered trustworthy but less available, and that informal sources like Q&A platforms and websites are more available but less trustworthy. This could explain the widespread use of Q&A platforms, despite the general perception that they are less trustworthy.

### 3.7.7 Summary: Key Findings

In summary, the key findings we can takeaway from our survey are:

- Our survey findings confirm the prevalent belief among data workers that selecting the appropriate statistical procedure is complex.
- Experts are the most common information source for data workers; data workers prefer interpersonal communication, but many data workers also use Q&A forums.
- Discussions with experts were considered trustworthy and a quick source of information, but were not always readily available. This could explain the widespread use of Q&A forums, despite its lower levels of trustworthiness.

## 3.8 Content Analysis: Understanding Queries About Analysis Procedures

https://osf.io/3rd4n/?view_only=a77946842b7649758bdcf1b36d0ecbcc[a]

[a]Contains the URLs of the questions we analyzed and attributes of questions we manually coded.

We analyzed questions and answers about statistical procedure selection from Q&A websites to understand how data workers seek expert validation.

Inspired by our findings from interviews, we wanted to understand how Q&A websites are used by users to post questions about statistical procedure selection and how such questions are handled. The findings from this content analysis helps us understand how data workers seek validation from experts, which is one of the preferred information sources that help decide upon statistical procedures. In this section, we first describe our data corpus (Section 3.8.1), our analysis method (Section 3.8.2), and then present the key findings from our analysis (Section 3.8.3).

### 3.8.1 Data Collection

We carefully sampled 76 questions from Q&A websites.

We selected a random sample of 76 questions[8] about hypothesis testing (Q01–76) from eminent Q&A websites, CrossValidated[9] ($n$=36) and ResearchGate[10] ($n$=40). CrossValidated and ResearchGate were mentioned by several participants in our previous interview studies, and are well known in data science community. CrossValidated is run by the StackExchange network, a popular Q&A platform [Begel et al., 2013], and, with over 15 million academic users as of 2019, ResearchGate is one of the most well-known academic social networks.

We sampled our data from CrossValidated and ResearchGate after surveying other platforms used for data science discussions.

Independently, we also informally surveyed other platforms that one might use to ask questions about statistical procedures. To do this, we did a Google search for hypothetical ques-

[8]https://docs.google.com/spreadsheets/d/1xSt12wIbXNa3kFIqjE55_gNxO4T-ri15GS49hgTwhYE/edit?usp=sharing
[9]https://www.crossvalidated.com
[10]https://www.researchgate.com

tions about statistical procedure selection and surveyed the results. In addition to CrossValidated and ResearchGate, other results either a) included discussions based on a particular statistical tool, e.g., Statalist[11], or b) had no dialogue between questioner and respondents, e.g., discussions in ISIXSIGMA[12].

We used the following criteria to sample questions from CrossValidated and ResearchGate:

1. question should have a relevant tag: At least one of *'statistical test'*, *'significant test'*, and *'hypothesis testing'*;
2. question should have at least one answer written by a respondent; and
3. question should be about *selection* of the appropriate statistical procedure, and not about usage details, i.e., how to use a certain programming language or software to perform a given statistical procedure.

## 3.8.2 Method

We manually analyzed various characteristics of questions and responses.

We analyzed our sample of 76 questions[13] manually by identifying several attributes of the question such as *characteristics* of the dataset (sample size, experimental design, measurements, factors, etc.), *intention* of the question (was it used to confirm a given procedure, fact-check a specific detail, or ask an open-ended question?), and the *time-point* during research when the question was asked (before data collection, after data collection but before analysis, or after an initial analysis). We also investigated the answers to these questions by identifying *acceptance rates* of the answers, whether the respondents *assumed* information about the question, and *agreement* among the respondents' answers.

A solo coder analyzed the textual content of the posts.

A solo-analyst iteratively developed the attributes after an ini-

[11]`https://www.statalist.org/forums/forum/general-stata-discussion/general/`

[12]`https://www.isixsigma.com/tools-templates/normality/dealing-non-normal-data-strategies-and-tools/`

[13]We analyzed 101 questions in total, but after analyzing the first 25 questions, we realized that these questions included requests for implementation or programming details, so we decided to exclude these.

tial analysis. We used qualitative analysis only to analyze the text content of the posts. This helped us understand the characteristics of the questions, as well as the conversation between questioner and respondent. The coding was done by a single analyst, who came up with a coding scheme after a phase of open coding and through regular discussions with the author of this work. A formal inter-coder agreement score was not computed.

### 3.8.3 Findings

#### Prevalence of validation questions

Most questions we analyzed already included a possible solution, and sought to confirm it.

In about half of the questions (36 out of 76; 47%), questioners already knew which statistical procedure they should be using, but were uncertain and hesitant to apply them to their data. These questions sought to validate their choice with experts. Prevalence of such questions is further validation of the uncertainty in statistical procedure selection. In such questions, questioners knew the statistical procedure, but had specific issues, such as smaller sample size or unclear distribution type which made them doubt their choice:

> *"...Based on previously published data, an one way ANOVA would be best for what I am doing... When I include steroid A on the graph[,] nothing is significantly different from my basal reading... This seems incorrect to me as this means there is dependence between each response... Is this because of the incorrect test and which test should I then use? Or if this is correct, why should the bar for steroid A be effecting the hypothesis testing of my remaining steroids?"* (Q46)

#### Information specified in questions

Questions contained three groups of information as shown in Table 3.3: dataset, experimental design, and questioner's intentions. We discuss these groups in detail below.

| | | Information | No. of questions |
|---|---|---|---|
| Dataset (32/76) | | Complete raw data | 13 |
| | | Partial raw data | 7 |
| | | Fabricated data | 3 |
| | | Summary data | 11 |
| | | Assumptions | 14 |
| Experimental design (76/76) | Factor | Name | 66 |
| | | Other | 66 |
| | Measure | Fabricated | 6 |
| | | Name | 64 |
| | | Other | 19 |
| | | Sample size | 32 |
| | | Procedure | 17 |
| | | Factor design | 19 |
| Intentions (67/76) | | Hypothesis | 26 |
| | | Analysis goal | 67 |
| | | Potential solution | 36 |

**Table 3.3:** Information questioners included in questions about procedure selection

**Dataset:** Information about dataset is critical for questions about statistical procedure and includes descriptive summary statistics, i.e., measures of central tendency such as mean, median, and mode, and measures of spread such as standard deviation and variance; and other data characteristics, such as homogeneity, linearity, and distribution type.

Although such information is important, only 42% (32 out of 76) of questions included it. Of these, in three questions the information was provided only upon respondents' request. Of the remaining 44 questions, 34 questions (77%) had adequate information to elicit answers. Of this, 27 of these questions had incomplete information about dataset, but the given information allowed respondents to *assume* certain details about the dataset. For example, in the following question, the respondents could assume the distribution type and certain other characteristics based on the measurements and factors:

Not all questions include information about dataset.

> *"...I am trying to study the association between blood groups (categorical variable) and cholesterol levels (continuous variable) adjusted confounders? I was wondering which statistical test would be more appropriate?"* (Q05)

It is possible that unless the respondents' assumptions are explicitly stated in the answers as shown below, the resulting answer could be incorrect:

> *"...I suspect the distribution of hospital costs is positively skewed, is it? If so, I would strongly consider using quantile regression rather than OLS linear regression."* (Q42)

There were ten questions (23%) had not provided enough information for the respondents (*"...Can't respond responsibly without more information. Please show more data, or speculations of how data may look."* in Q15).

**Experimental design:** All questions provided information about experimental design, even though in some cases the information was minimal. This includes information about factors such as the levels and whether they are between- or within-subjects and possible interaction between factors.

As information about experiment might include confidential data, some questioners use hypothetical names and/or characteristics.

While most questioners provided actual variable names from their experiment, some provided fabricated variables. This was done in three ways:

1. Questions included descriptions of *hypothetical variables with similar properties.* This approach was done mainly to conceal sensitive information (*"Suppose I am doing a case control study. Lets say... For Example: If "satisfaction with life" and "quality of life" are research variables in two groups... Please explain as my research question is different, and I have just given an example here."* in Q47).
2. Questions included *real variable names, but the characteristics were hypothetical.* We found an example, in which

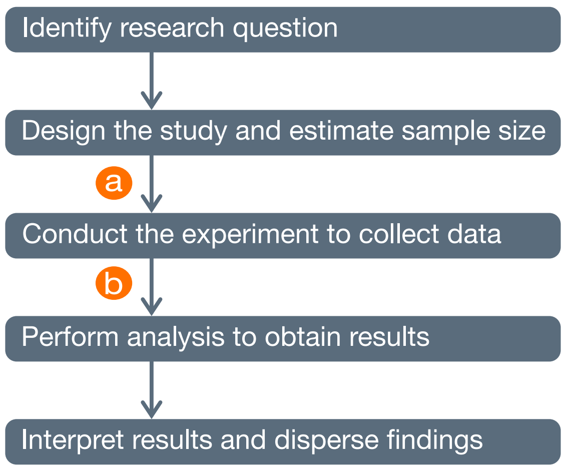

Identify research question
Design the study and estimate sample size
a
Conduct the experiment to collect data
b
Perform analysis to obtain results
Interpret results and disperse findings

**Figure 3.4:** Experimental research consists of five steps that are shown above. For most experiments, it is preferable to decide the statistical procedure before conducting the experiment (a). But from our analysis, we find that most questions about statistical procedure selection are posted after the data have already been collected (b). (Adapted from [Jones, 2004].)

the questioner had specified different number of levels in the question, but later clarified it (*"I should say that my data has two more score values than in my example (i.e., question) above, it was designed exemplary."* in Q01).

3. Questions included *vague descriptions of the variables and hypothetical variable names:* (*"I am running Kruskal-Wallis tests in SPSS (a hypothesis testing software) to compare answers to Likert [-scale] items among 3 (variable 1) and 4 (variable 2) groups."* in Q29).

In addition to experimental design, nearly a half of the questions also included sample size.

**Timeline of procedure selection and data collection**

A typical analysis has several steps as shown in Figure 3.4: Identify research question; design the study and estimate sam-

There are various stages during the research process at which data workers can decide upon the statistical procedure.

ple size; conduct the experiment to collect data; and perform analysis to obtain results (adapted from [Jones, 2004]). Depending on the type of analysis, researchers should ideally decide the analysis procedure before conducting the experiment. For typical controlled experiments and hypothesis testing, it is recommended that researchers pre-register their analysis procedure [Cockburn et al., 2018], i.e., decide the procedure before conducting the experiment (Figure 3.4a). For more exploratory research works, which are still somewhat prevalent in HCI, it would be OK to determine the analysis procedure post-data collection (Figure 3.4b).

Most queries about statistical procedure selection were made after all experimental data had been collected.

By analyzing the phrasing in the question and whether raw data were provided, we were able to determine the research stage at which the questions were posted for all but nine questions. Over half the inquiries (36 out of 67; 54%) were made after all the experimental data have been collected. A substantial amount of inquiries (27 out of 67; 40%) were made *after* an *initial* data collection. Only four questions were posted *before* any data were collected.

### 3.8.4 Problems in Formulating Questions

Questions with missing and unclear information can lead to difficulties in answering them.

Based on our analysis, we identified two key problems that questioners face in formulating questions. A good question posted in Q&A websites is comprehensible and contains enough information so that the respondents can provide answers with certainty. When respondents encounter questions that do not fulfill these requirements, it often leads to an increase in the back-and-forth communication between the questioner and respondents. More importantly, it can result in inaccurate, and sometimes incorrect, responses.

#### Questions with missing information

We analyzed the discussion threads in Q&A platforms to determine which questions had missing information.

While the information presented in Table 3.3 are important to be included in questions, not all information is *required* for accurate answers. To determine whether a question has missing information, we analyzed the discussion thread for each question as well as the answers. We specifically tried to answer the following:

1. Do the respondents require additional information to provide an answer?
2. Do the respondents provide multiple answers or assume information? For example: (*"First, specify whether the data produced by questionnaire is continuous or categorical. Chi-square is a better option for categorical data. If continuous, then you need to specify whether your data is parametric or non-parametric ... "* in Q56).

Around a third of the questions (29 out of 76; 38%) we analyzed had missing information. The missing information varied from fundamental details of the analysis, such as the goal of the analysis and research hypothesis, to information specific to the scenario described in the question (*"What is the research question? Is there any statistical hypothesis related to it?"* in Q08).

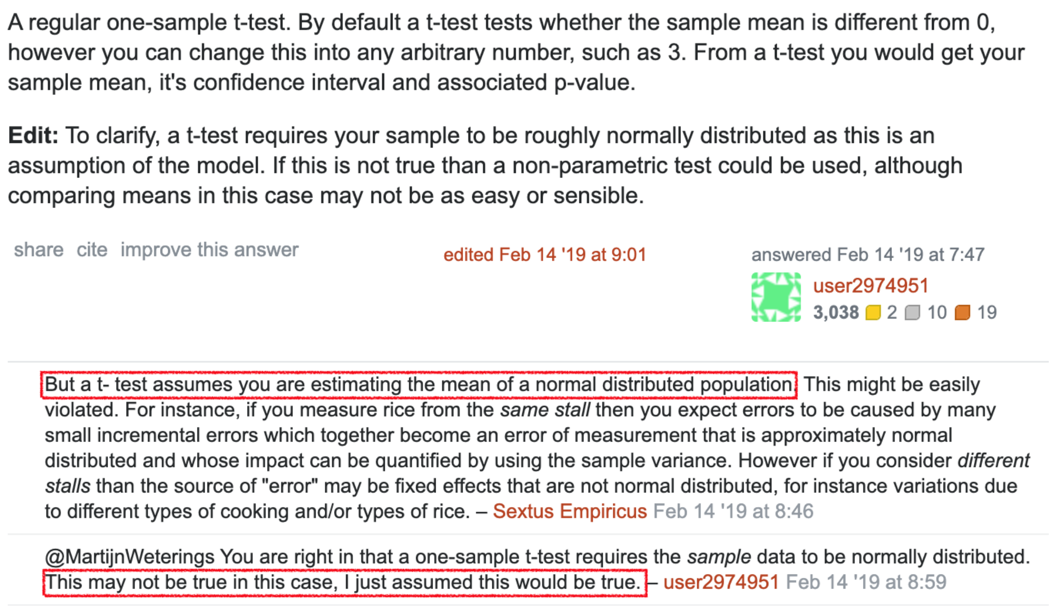
The information that was found missing varied from one question to another.

Many questioners were seemingly novices, and were unsure which information they should provide (*"Please let me know if you would like more detail. I'm still something of a statistics beginner, so I'm not sure how much depth is required to be able to answer this question."* in Q41).

Novices might have difficulty knowing which information should be included in questions.

To understand the consequences of missing information in questions, we inspected the discussion threads, e.g., as shown in Figure 3.5. Almost always, respondents seek the missing information by requesting for it. Ideally, this information is provided by the questioner and the respondent is able to answer the question. However, in most situations (22 out of 29; 76%), respondents offer multiple answers (10 out of 22), assume information about the question (7 out of 22), do not answer questions (3 out of 22), or use self-fabricated data to answer questions (2 out of 22). While offering multiple answers can be a better choice over incorrectly assuming information, it bloats the answers and requires the questioner to carefully process the answers.

Analysis of discussion threads informed us how respondents dealt with missing information.



A regular one-sample t-test. By default a t-test tests whether the sample mean is different from 0, however you can change this into any arbitrary number, such as 3. From a t-test you would get your sample mean, it's confidence interval and associated p-value.

**Edit:** To clarify, a t-test requires your sample to be roughly normally distributed as this is an assumption of the model. If this is not true than a non-parametric test could be used, although comparing means in this case may not be as easy or sensible.

share cite improve this answer
edited Feb 14 '19 at 9:01
answered Feb 14 '19 at 7:47
user2974951
3,038 🟨 2 ⬜ 10 🟧 19

But a t- test assumes you are estimating the mean of a normal distributed population. This might be easily violated. For instance, if you measure rice from the *same stall* then you expect errors to be caused by many small incremental errors which together become an error of measurement that is approximately normal distributed and whose impact can be quantified by using the sample variance. However if you consider *different stalls* than the source of "error" may be fixed effects that are not normal distributed, for instance variations due to different types of cooking and/or types of rice. – Sextus Empiricus Feb 14 '19 at 8:46

@MartijnWeterings You are right in that a one-sample t-test requires the *sample* data to be normally distributed. This may not be true in this case, I just assumed this would be true. – user2974951 Feb 14 '19 at 8:59

**Figure 3.5:** One of the answers posted to a question about statistical procedure selection (Q06). Since the question had missing data, the respondent, `user2974951`, assumed certain characteristics about the dataset. In our analysis, we found that over a third of the questions (38%) did not include enough information for the respondents to answer the question with certainty.

### Questions with unclear information

The lack of clarity in questions stems from issues of language and statistical terminology.

Apart from questions that had missing information, 13 out of 76 questions (17%) provided information that we consider unclear. We determined a question to have unclear information in a manner similar to that described above for determining missing information. The lack of clarity mostly originated from language and terminology issues, e.g., ambiguous sentences and inaccurate phrasing of text, as well as a lack of adequate information of statistical procedures on the questioners' part, e.g., misrepresenting analysis procedures and using incorrect charts.

Questions found it difficult to use the appropriate terms.

Questioners had particular difficulty in representing information about their analysis in a clear and concise manner ("*. . . I've got 3 columns one for pre manipulation of language which will be my control. I only have positive and negative columns. In addition, I will also have a column for age(3 levels) and gender*

*(2 levels) and want analysis how these demographics affect the effect that positive and negative language has on investor preferences."* in Q55). Notice the use of the word 'column', which is used to refer to both the variables in the dataset as well as the levels of the factor. This led to the following comment by one of the respondents: (*"3 columns:1– language classified into +ve -ve. 2– age classified into 3 levels. 3– gender classified into male female. And I understood that: investor preferences is the outcome."* in Q55).

Questioners also had difficulty representing the experimental design in questions. Take the following question: (*"Each subject is assigned to one of three different Times (morning, noon or afternoon) and receives one Treatment (A,B or C)."* in Q01). One respondent assumed that it was designed to be a within-groups design and gave a corresponding solution, only for this to be clarified by the questioner later. Respondents dealt with such questions in a manner similar to handling questions with missing information, i.e., by requesting details and then resorting to providing multiple answers or making assumptions if the requested information was not satisfied.

Questioners found it difficult to specify experimental design.

### 3.8.5 Discussion

Through interviews, survey, and content analysis, we identified three key problems about how data workers search for and decide on statistical procedure in their work: a) Data workers feel uncertain about their choice of statistical procedure, b) they have difficulty presenting adequate information to experts, and c) they decide upon statistical procedure after collecting the data, thereby violating the recommended practice.

#### Uncertainty in procedure selection

There is no better evidence for the first problem than the prevalence of questions that seek to simply confirm the questioner's choice of a statistical procedure. Many data workers consider statistical procedure selection a complex task. While discussions with experts, being informed by reading research papers, and research artifacts that help with procedure selection

As data workers gain more experience in perform analysis, they become more certain in statistical procedure selection.

[Wacharamanotham et al., 2015; Jun et al., 2019] can help, the underlying problem seems to be a lack of practical training in and deep information of statistics. While most data workers undergo formal training at universities, this seems to emphasize theoretical aspects. Data workers who underwent a more practical training reported being more capable of handling the complex nature of procedure selection. For example, IP02 is a psychology PhD student and credits their confidence to practical training in a Bachelor's course: *"I'm confident of choosing the right method because in my bachelor's, my teacher asked us to calculate a lot of the statistics manually [...]"*. The resulting increase in confidence would also data workers to tackle the second key problem, by making convincing arguments for their choice of statistical procedures.

### Presenting information about experiment and dataset to experts

Our second problem deals with mostly novice data workers who face issues seeking help from experts. They do not know whether the information they present is adequate and how to meaningfully present them. This leads to an increase in interactions between the data worker and experts, not to mention incorrect recommendations from experts. Herein lies opportunities for researchers and developers to target. We discuss some ideas in the next chapter, in Section 4.1.6.

### When should we seek statistical procedures?

Queries about statistical procedures, when asked after collecting all data prevent questioners to address fundamental issues with data collection.

As discussed in Section 3.8.3, most questions were posted after collecting the entire experimental data. There are advantages and disadvantages with this approach. During this stage, the entire dataset can be included in the question, even though most questioners did not. Therefore the respondents can answer questions with more certainty. In fact, when the entire dataset was provided, respondents were able to identify the missing information themselves (*"Not only do you have heteroskedasticity [sic], with proportions (as in your example) very close to 1 (or to 0) they'll also be highly skewed ... "* in Q10). However, as the data have already been collected, any issues with experimental design or sampling cannot be fixed without running the experiment again (*"[. . . ] try to generate other similar questions that you can combine into a scale, because that [. . . ] would allow you to use a more flexible set of statistics."* in Q43).

Queries about statistical procedures, when asked before collecting any data limits the certainty with which respondents can provide answers.

In questions posted before collecting any data, questioners sought validation for a predetermined approach. As no data have been collected, respondents had to assume information based on their own understanding, which might lead to less certain answers (*"[. . . ] can you state your objective in terms of the two tasks? Can you show the data for at least a few of the subjects?"* in Q15).

Queries about statistical procedures, when asked after collecting some data allow for more certain answers while still allowing for changes to experiment.

In questions posted after collecting some but not all experimental data, questioners were able to provide enough information to improve respondents' certainty, but could still revise experimental design if necessary. Therefore, it might serve the questioner best to post questions about statistical procedure after collecting some but not all experimental data, e.g., data from a pilot. While this is the recommended practice in most research, our analysis shows that existing practice is quite different.

Preregistration can help avoid some of the issues we have discussed.

Although our survey results show that preregistration is not that uncommon, we need to validate this finding with data workers from other domains and with lower experience. The benefits of preregistration are clear: it can improve confidence in statistical findings, and deciding on a statistical procedure before collecting data can also help avoid fundamental issues with experimental design and research methods, as shown in our content analysis. Cockburn et al.'s work includes a good discussion of the relevance and benefits of preregistration within HCI [Cockburn et al., 2018].

## 3.9 Limitations and Future Work

We had data workers from various backgrounds and with various experiences take part in our studies.

A significant limitation of our work is that our study participants were not sampled from a consistent population. For example, our survey participants turned out to have more experience than our interview participants in general. Although this allows us to gain insights about experienced data workers, e.g., how they are more assured about their choice of statistical procedures, such insights are no more than speculations as we did not get an opportunity to interview them at length. Further work is needed to understand the specific factors that help novice data workers mature into experienced, more assured data workers.

Even though we attempted to unravel the impact of disciplines and data science tasks, e.g., machine learning and hypothesis testing, on data workers' workflows, we do not have enough data to generalize our findings to all disciplines and data science tasks. Nevertheless, the findings in this chapter help us understand workflows when preparing to perform hypothesis testing in HCI and related disciplines, and offers a starting point for further research into understanding other data science tasks and disciplines.

Our content analysis results are only indicative of how data workers might seek validation from experts.

Finally, the questions we analyzed in our content analysis might not be a completely representative sample: we do not always know the experience and expertise levels, discipline of the questioners and the respondents. However, due to the large sample size, the findings are largely indicative of the workflow involved in asking and answering queries about statistical procedure selection. Future work about interpersonal sources of information can help confirm our findings.

## 3.10 Summary

To understand how data workers prepare to perform hypothesis testing, we conducted two interview studies, a content analysis, and a survey. Our findings focused mainly on procedure selection and information sources used to select procedures. We find uncertainty about procedure selection leads to prevalence of expert validation practices, and a general reluctance, or even apprehension, to adopting new statistical procedures. Although the on-going criticisms about hypothesis testing contributes to this uncertainty, a deeper reason seems to be lack of adequate practical training in statistics. A lack of training also leads to incomplete mental models about hypothesis testing, which leads to high levels of perceived complexity in procedure selection and proliferation of questions about statistical procedures with missing and unclear information. Motivated by our findings from this chapter, we next present research artifacts and recommendations to improve data workers' workflows when preparing to perform hypothesis testing.

Chapter 4

# Supporting Data Workers in Preparing to Perform Data Science

**Publications and Author's Contributions**
The work in this chapter has contributions from Jeanine Bonot, Radu Coanda, and Malte Meng. Part of the research in this chapter has been published as an extended abstract [Subramanian and Borchers, 2017], and one paper [Subramanian et al., 2019a]. Bonot extended StatPlayground, discussed in Section 4.1, by adding fine-level control and foreshadowing features; and evaluated StatPlayground with users [Bonot, 2019]. Coandă developed Cheno, discussed in Section 4.1.3, which helps with statistical computations [Coandă, 2019]. Meng developed an exploratory prototype to help data workers formulate questions for posting on Q&A platforms [Meng, 2020]. Although this prototype is not part of this thesis, the prototype helped us identify several design recommendations discussed in Section 4.1.6.

In the previous chapter, we discussed how data workers prepare to perform real-world analysis tasks. In this chapter, we discuss solutions to address two prominent problems that came up in this discussion: 1) practical statistics is an important skill needed to perform analysis, and this skill is hard to develop through formal training; and 2) difficulties in presenting information about dataset and experiment in when asking questions about statistical procedure selection Q&A platforms. Before discussing these solutions, we address a prominent problem in statistical practice.

## 4.1 StatPlayground: Helping Data Workers Learn Practical Statistics

https://osf.io/me2dc/?view_only=efd437103561482286c4e230b75c1288 [a]

[a]Contains pictures of design alternatives, Cheno Python package used for backend computations, a demonstration video (of a previous iteration of the prototype), a flipbook prototype used in previous design iterations, and an example transcript of the feedback session for two participants.

We propose a tool to help data workers learn statistical procedure selection and the relationship between statistical concepts.

From our discussions in the previous chapter, we find that formal education does not train data workers adequately with practical statistics, which is essential for selecting statistical procedures and performing analysis. By reviewing existing research on improving statistical education, which will be discussed in Section 4.1.1, we developed a tool to help data workers learn *practical statistics,* i.e., statistical procedures and relationship between statistical concepts. We first review existing tools to help teach and self-learn statistics in Section 4.1.1. Then discuss our prototype, *StatPlayground,* an exploratory artifact that can help learn statistical concepts, and discuss how we validated it.

### 4.1.1 Related Work: Guidelines for Improving Statistical Education

We now discuss existing guidelines to improve statistical education and software tools used to teach statistics; these works motivated the design of StatPlayground.

Students change their statistical misconceptions only when challenged.

One prominent issue in teaching statistics is that students' misconceptions about statistics are resistant to change. Since students learn by constructing their own knowledge [Resnick, 1987; Von Glasersfeld, 1987], they accept new ideas only when they are challenged to confront their misconceptions [Garfield, 1995, pg. 31].

Active learning with hands-on activities have been shown to help improve learning outcomes in statistics.

To address this issue, we investigated existing research into improving statistical education. Prior research has shown that students learn statistical concepts better when they are actively engaged. Active engagement can be achieved in several ways, such as by involving students in hands-on activities [Lunsford et al., 2006] and having students work in small groups with others [Gnanadesikan et al., 1997]. Hands-on activities are effective as they allow students to apply theoretical knowledge to novel situations, a task that might evaluate their understanding of statistical concepts. Johnson et al., Goodsell et al., and Fink provide guidelines for setting up collaborative, hands-on learning environments [Johnson et al., 1991; Goodsell et al., 1992; Fink, 2013].

The use of software aids in teaching environments has also garnered the attention of researchers. Software simulations have been successfully used to teach electrical circuits [Carlsen and Andre, 1992] and mathematics [Reed, 1985]. In statistical education, Rice Virtual Lab in Statistics[1], Seeing Theory[2], and Wise Interface for Statistical Education[3] are some prominent simulations. While research on the effective of simulations on statistical education is sparse, there are some promising results. Simulations have been shown to improve learning outcomes in comparison to traditional teaching methods [Wackerly and Lang, 1996; Hodgson, 1996; Schwartz et al., 1998]. However, not all simulations are equally effective, and their effectiveness depends on several factors, such as how they are used by the instructor, stimuli presented to the students, e.g., whether they are given a question to solve, and the type of feedback offered. For example, Pfaff and Weinberg found that their simulations had a detrimental effect on improving learning outcomes [Pfaff and Weinberg, 2009]. The authors outline some reasons behind this in their paper, namely problems with implementation of the simulation and the method used for assessing learning outcomes [Pfaff and Weinberg, 2009, p. 9–10].

Simulations can help teach statistics, but require careful use to be effective.

Overall, there are no methods of teaching that would work well with all students, and solutions need to be evaluated with the particular usage scenario in mind.

We now discuss StatPlayground, an exploratory prototype for self-learning and teaching practical statistics. StatPlayground can help users understand how several statistical concepts are

[1] `http://onlinestatbook.com/rvls.html`
[2] `http://seeingtheory.io`
[3] `http://wise.cgu.edu`

related to each other and understand statistical procedure selection. It is based on active learning, and allows data workers to explore statistical procedure selection and relationship between various statistical concepts. We now discuss the design of StatPlayground.

### 4.1.2 Design

StatPlayground allows users to explore statistical concepts and procedure selection through discovery learning.

StatPlayground is a self-learning or teaching tool, meant to be used alongside a course on statistical hypothesis testing. StatPlayground allows users to generate hypothetical data. These data and relevant characteristics, such as the shape and skewness of the distribution, variance, and sample size, are visualized in StatPlayground. Based on data and characteristics, StatPlayground selects and performs an appropriate statistical procedure, such as a $t$-test or ANOVA. To allow for active learning, users can *modify* data and characteristics, e.g., by directly manipulating the box plot visualization, to see how changes to one characteristic affects other data characteristics, as well as the resulting statistical procedure and results. We compare this to students asking what-if questions about statistics in a classroom. This intuitive manner of interacting with visualizations separates StatPlayground from other statistical simulation tools like TinkerPlots [Fitzallen, 2012].

#### Design process

StatPlayground has been developed in a user-centered, iterative manner.

We designed StatPlayground in an iterative manner. An initial software prototype was first informally evaluated with five experts in statistical hypothesis testing. All experts had at least two years of experience in with inferential statistics; two of them reported to regularly teach statistics as a part of statistics course aimed at undergraduate and graduate students.

Expert evaluation allowed us to improve StatPlayground by adding two interaction features.

Through this evaluation with experts, we identified design changes to our prototype, such as a) *fine-level control* over data characteristics, by allowing users to set the range of values for data characteristics, lock/unlock the value from changing, and b) feedforward mechanism to show how the statistics would

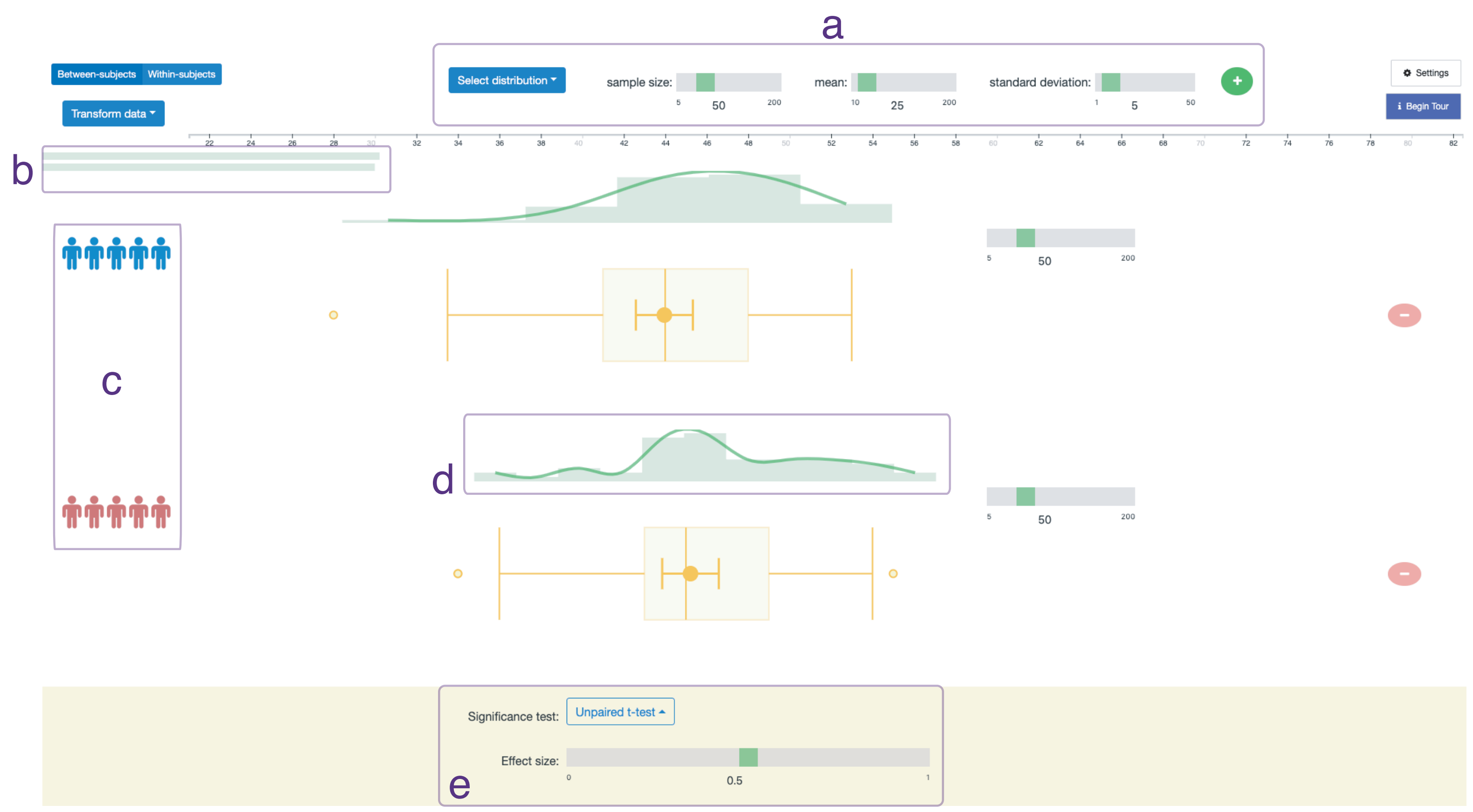


**Figure 4.1:** StatPlayground contains three main panels: distribution generator (a), which can be used to generate common distribution by specifying parameters; results (e), which display the results of the current data configuration; plot, which uses box plot to visualize data distributions. Additionally, StatPlayground visualizes other data characteristics: shape of the distribution (d), homogeneity of the variances (b), and experimental design (c).

change due to the current interaction, a concept we term *foreshadowing,* as shown in Figure 4.4. Fine-level control allows the user to ask more fine-grained what-if questions like *"Would the effect size of this comparison of two distributions increase even if the means of both distributions remain the same?"*. After designing these new features, we developed the current prototype of StatPlayground.

### Layout

StatPlayground has three main panels: data generation (Figure 4.1a), plot (Figure 4.1 center), and results (Figure 4.1c). Users can generate data by using the data generation panel at the top. The panel allows the user to create common distributions such as Gaussian, uniform, log normal, and bimodal distributions by setting the sample size and other relevant parameters. When a distribution is created, it is visualized as a box plot in the plot panel. The box plot shows the mean, median, whiskers, and outliers of the distribution, as well as the 95% confidence interval of each mean. The plot panel also displays important data characteristics that affect the resulting statistics: shape of the distribution as a histogram (Figure 4.1c), variance of each distribution as an animated bar chart at the top-left corner (Figure 4.1d), and experimental design of the factor to the left of the box plot (Figure 4.1e).

### Interaction

StatPlayground allows direct manipulation of data points. Changes to one data characteristic affect other characteristics, which StatPlayground can visualize in real-time.

Users interact with data in StatPlayground through *direct manipulation*; we employ direct manipulation to invoke active engagement and a sense of play. Users can click-and-drag data elements to *modify* them e.g., click-dragging a mean will modify the mean of the distribution.

We now describe an example interaction with outliers. To create a new outlier, users can click on the empty space either side of the whiskers as shown in Figure 4.2. Similarly, to delete an outlier, users can click on an existing outlier point in the box plot. When data are changed, the effect is visualized in real-time. For example, Figure 4.2 shows the interface after

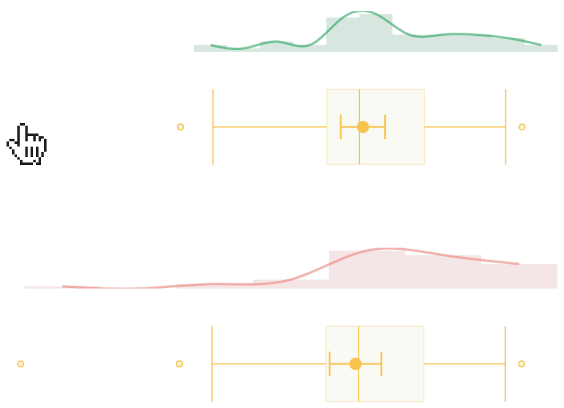


**Figure 4.2:** Users can create, edit, or delete data by directly interacting with the box plot. In this image, the user clicks next to a box plot to create an outlier. StatPlayground visualizes the effect: the mean shifts and the distribution is no longer normal.

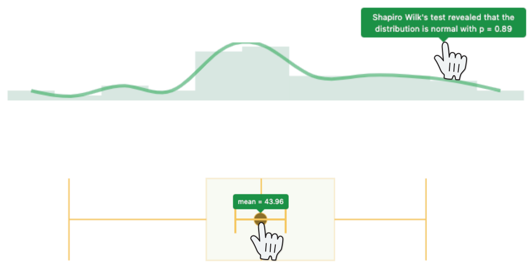

Shapiro Wilk's test revealed that the distribution is normal with p = 0.89
mean = 43.96

**Figure 4.3:** StatPlayground uses progressive disclosure to reveal detailed information. User hovers over interface elements to view tooltips.

the user adds an outlier to a distribution; the histogram curve changes to indicate that the *test for normality* has failed and a *non-parametric test* has been chosen as a result.

Detailed statistics and details are revealed on demand.

StatPlayground utilizes progressive disclosure [Springer and Whittaker, 2019] to reveal detailed information upon user request. For example, hovering over any data point shows the parameter and value (Figure 4.3), and hovering over the histogram curve reveals the test that was performed

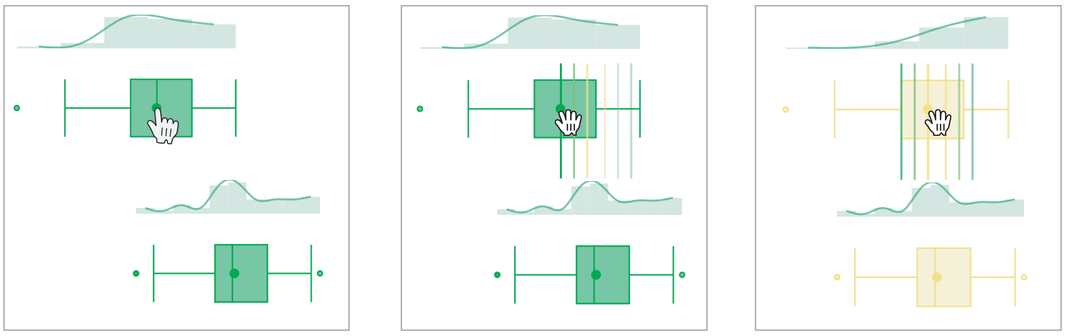


**Figure 4.4:** StatPlayground provides feedforward during direct manipulation to help the user predict the statistical result. Here, the user is manipulating the mean by click-dragging it closer to the mean of the other distribution. When they do this, they notice that a feedforward overlay shows up, which indicates the various effect sizes they will reach upon continued dragging. Dark green indicates a large effect size and yellow represents a small effect size.

(e.g., Shapiro-Wilk test) along with the resulting *p*-value (Figure 4.3).

### Fine-level control and foreshadowing

Fine-level control allows users to ask more detailed what-if questions.

Hovering over data points for a longer time period (2 seconds) shows the *fine-level controls* that can be used to lock/unlock and set the range of values for a data characteristic. This can be used by the users to ask more fine-grained, what-if questions. E.g., in Figure , the user queries, *"What happens to the effect size if . . . ?"*. If a value is locked, then changes to other characteristics are permitted only when the change does not modify the characteristic whose value is locked.

With foreshadowing, StatPlayground can show the resulting statistics before the user interaction is complete.

In addition to fine-level control, StatPlayground supports a feature we call foreshadowing. During direct manipulation, StatPlayground can visualize the possible results the user can expect to reach along the path of her manipulation via *foreshadowing*. For example, in Figure 4.4, when the user is click-dragging the mean of a distribution further away from the other distribution, they can see that moving them further apart would result in incrementally *smaller* effect sizes. We chose to use effect size over *p*-value to encourage the use of estimation statistics, an alternative to significance testing [Cumming, 2014].

### 4.1.3 Implementation Details and Challenges

We built the front-end of StatPlayground's using modern web technologies: JavaScript (Angular v7) and HTML/CSS. Some statistical computations are done natively in JavaScript using libraries like Jerzy[4] and jStat[5]. However, inverse-computation of data configurations from a given statistic ($p$-value, effect size), which is needed for the *foreshadowing* feature, proved to be a challenge. Since many data configurations can produce the same statistical results, we had to find and generate more than one data configuration. While existing research works, e.g., [Matejka and Fitzmaurice, 2017], could generate data configurations from statistics, they are a) limited to descriptive statistics like mean and standard deviation, and b) have a different objective, that of generating visually distinct datasets.

Inverse computations proved to be challenging due to the need to compute a multitude of data configurations.

To perform such inverse computations, we built a Python package called *Cheno*[6] that computes datasets that result in the same given statistics. Unlike existing approaches, Cheno does not generate visually different data configurations. Instead, it generates the possible range of each data parameter, such as mean and standard deviation, which can in turn be used to produce visually different data configurations. We integrated Cheno into StatPlayground via Flask[7] to support statistical computations, but it can also be integrated into other simulation-based learning tools. Despite the addition of Cheno, inverse computations proved to be computationally intensive, and infeasible for our user evaluation. For this reason, in our evaluation, described in the next section, we decoupled Cheno from StatPlayground, and instead used the Wizard of Oz prototyping technique.

We built Cheno, a Python package to generate data configurations that would result in a given set of statistics.

[4]`https://github.com/pieterprovoost/jerzy`

[5]`https://jstat.github.io`

[6]Cheno is an anagram of (Jacob) Cohen, an American statistician known for his work on statistical power and effect size.

[7]`http://flask.pocoo.org`

### 4.1.4 Evaluation

Since StatPlayground is built around a novel concept, we wanted to evaluate it in a holistic manner. The purpose of our evaluation is to gain more insights into how StatPlayground can be used to learn practical statistics.

StatPlayground was evaluated in two phases.

The evaluation had two phases. We collected data from six participants[8] in the first phase, used feedback from participants to improve StatPlayground's design and evaluation procedure, and then evaluated StatPlayground with eight more participants. Below, we briefly discuss the data collection, procedure, and limitations of the first phase of our study, and then discuss the second phase of our study. We then discuss the key findings that resulted from the second phase of the study.

#### Study phase I

We recruited eight participants from the local lab to evaluate StatPlayground.

**Data collection:** We collected data from eight participants (two female). We recruited the participants via emails sent to a HCI research methods course mailing list. Participants were not offered any incentives, and participation was voluntary. We used convenience sampling [Etikan and Bala, 2017] to recruit participants from our local lab because a) these participants fit our target user group well and b) due to technical limitations, it was not possible to evaluate StatPlayground remotely. All participants had taken an introductory statistics course before.

We allowed participants to refer to material about statistical concepts and gave a tutorial on StatPlayground.

**Procedure:** To ensure that all participants had the same base-level understanding of statistics, we first provided a brief overview of basic theoretical statistical concepts, such as effect size, *p*-value, and statistical significance. Participants were also allowed reference material about these concepts. Once participants were acquainted with the statistical concepts, they were presented with a hypothetical dataset, and given questions they have to answer by using StatPlayground. The experimenter also verbally described the interface, going over the layout of the interface and basic components. The experimenter also

[8]We collected data from eight participants, but audio recording failed for two participants.

mentioned how StatPlayground can be used, e.g., the interface can be clicked and dragged upon. Although this might prime the participant to perform such actions, our goal is not to evaluate how discover-able such actions are, but instead evaluate if and how StatPlayground can help participants learn statistical concepts.

Participants' tasks were story-driven and guided.

Questions were story-driven, e.g., "assume that out of all participants in your text-entry experiment, one of them has practiced touch-typing; how does she (i.e., the outlier) affect your analysis?". These questions guided the user's interaction with StatPlayground. Some questions were used to test if the participant was able to understand a concept that came up in an earlier question, e.g., "currently, the results are not significant. How can you modify the distribution so that the result is slightly significant but without modifying the mean/median?".

Through our observations, we could evaluate whether the participant was able to perform the following tasks via our story-driven prompts:

1. Change the data distribution by click-dragging the mean.
2. Modify the distribution properties so that there is not enough evidence for the null hypothesis to be rejected.
3. Modify the distribution properties so that there is enough evidence for the null hypothesis to be rejected.
4. Determine whether the statistical test's assumptions are satisfied.
5. Add and remove outliers.
6. Prevent outliers from being modified.

We encountered several issues in study phase I, including buggy implementation and inconsistent stimuli.

**Issues with the study:** We had both technical and methodological issues with the first phase of the study. Audio capture failed for two participants, and, although we had the screen recording for these participants, it was near impossible to analyze them. More importantly, the artifact had several bugs and missing implementation. Participants were severely hampered in their tasks, e.g., creating an outlier would cause the box plot to become non-interactive. Methodologically, study phase I had two critical issues: a) There was no script to guide the experimenter during the study, which caused inconsistencies in participant stimuli and b) Since the tasks were communicated to participants verbally, some participants needed to be reminded about their tasks, which caused a distraction that could have been avoided.

### Study phase II

To address the issues with study phase I, we made the following key changes[9]:

- Information about scenario-based tasks that the participants had to do as a part of the study was presented to the user on a printed sheet of paper in addition to verbal instructions. The initial tutorial about statistical terms was also printed on a sheet of paper, instead of being conveyed verbally.
- A tutorial was added to StatPlayground that illustrated the different features of StatPlayground. This replaced the need for verbal instructions, and the resulting inconsistencies.

**Participants:** Seven participants (two female) volunteered to take part in the study. All participants are university students, our primary target user group, who have taken an introductory statistics course and/or performed hypothesis testing earlier. None of the participants had taken part in study phase I. As with study phase I, we recruited the participants via emails sent to a HCI research methods course mailing list. Participants were not offered any incentives, and participation was voluntary.

Participants were tasked with answering story-driven questions using StatPlayground.

**Procedure and analysis:** Similar to study phase I, we first established base knowledge among our participants. Unlike phase I, however, we gave our participants a handout with descriptions of relevant statistical concepts. Participants were

[9]For a complete list of differences between the two phases of the study, see [Bonot, 2019, Appendix B.2.2].

then given another handout that described the dataset used in the subsequent tasks. Participants then went over StatPlayground tutorial to become familiar with the interface. Similar to study phase I, participants were then given tasks that they had to solve by using StatPlayground, e.g., "How would you manipulate this outlier so that the heart rate will increase by a minimum of 55 bpm, regardless of how stressful the user interface is?".

We used a guided approach instead of open-ended exploration a) in order to better structure our participants' tasks and to keep our participants focused, and b) because it was infeasible with our implementation to handle open-ended interaction inputs in real-time. Sessions lasted 40 minutes on average. We encouraged the participants to think aloud, and concluded the session by getting feedback from participants. We used an unstructured interview approach here for the feedback session[10], and used our field notes to determine the feedback questions. The experimenter kept the prompts to a minimum and aimed to triangulate findings from observations, per recommendations of Boren and Ramey [Boren and Ramey, 2000]. We recorded the audio, screen, and the webcam video footage.

We gave questions to our participants to guide them through the study session.

We first used an online tool to automatically transcribe the audio of the feedback session. The analyst carried out an informal, reflexive thematic analysis [Braun and Clarke, 2021] to group participants' utterances from the feedback session into themes, the results of which are discussed below.

### Key findings from evaluations

StatPlayground shows potential in helping data workers learn practical statistics.

After analyzing our transcripts, we found several instances in which StatPlayground helped participants better understand statistical concepts, e.g., the influence of measures of central tendency (mean, median) and spread (variance) on the shape of the distribution, how effect sizes work in comparison to *p*-values, and how confidence intervals work. P13 mentioned that if they were to continue using this tool, they would be able to predict the statistics just by looking at the graphs, which is a skill one can expect experts to have. Participants commented

[10]Sample transcripts from feedback session can be found in supplements.

that the interface was, in general, *easy* and *fun* to use, indicating that it can be immersive, and that they liked the general idea of being able to learn in an *interactive* manner. P3 compared StatPlayground against their statistics course in her Bachelor's degree, in which the concepts were taught in a theoretical and abstract manner: *"[StatPlayground] is an intuitive way of grasping the concepts and understanding how they relate to each other."*

Participants had difficulty using and understanding certain components of StatPlayground's interface.

Despite these positives, there were several instances in which our participants encountered difficulties when using StatPlayground. Participants had difficulty using and interpreting fine-level controls, and interpreting the graph that compared the variances of distributions. Some participants were concerned that the interface was too cluttered, and commented that it was hard to see all the causal changes. Participants were also concerned whether such a tool could be misused, e.g., to help cherry-pick data points so that a significant $p$-value is obtained. We elaborate on this concern in Section 4.1.5.

## 4.1.5 Discussion

### Modes of interaction

StatPlayground can be tailored towards the end-user's needs.

In our study, we observed that our participants had two clearly discernible ways of using StatPlayground. Some participants were adventurous and used StatPlayground in an exploratory manner, interacting with the visualization beyond what was required to answer the questions. Most other participants were cautious in using the interface and needed gentle nudges. P7 mentioned, *"I could just trial and error, but I would not be sure whether what I'm currently observing is interesting."* This indicates two modes of usage: *guided* and *unguided*. Cautious users could use StatPlayground in a guided mode, in which the interaction is driven by questions, which could be prescribed by an educator. More adventurous users and experts could use StatPlayground in an unguided manner to explore concepts based on their own what-if questions.

#### Possible misuse

StatPlayground is intended to be used as a learning tool to understand how the different statistical concepts and data characteristics are interrelated. Interestingly, a few participants wondered whether StatPlayground could be used to analyze their own data, at least as a quick way to pilot test their analysis. We are concerned that since StatPlayground allows users to modify data characteristics, users could tamper with their experimental data, even if it is done unintentionally, in order to obtain statistically significant results. To avoid this, StatPlayground does not allow users to upload and work with their data, and instead requires users to generate hypothetical data.

StatPlayground can only be used with hypothetical data to prevent potential misuse.

#### Deployment considerations

Before using StatPlayground in a classroom or for self-learning, instructors need to design the questions for the guided mode. Future work could look into an interface for instructors to design these questions. For best results, we recommend instructors to follow best practices from research on statistical education, e.g., [Garfield, 1995; Zieffler et al., 2008], in combination with StatPlayground.

StatPlayground needs to be integrated into teaching after considering recommendations from statistical education research.

### 4.1.6 Limitations

StatPlayground is not a production-ready tool, but only a prototype, due to the challenges we discussed in Section 4.1.3. We have contributed the novel design of StatPlayground with our prototype, and evaluated it with users. Our evaluation with users shows promise, but further studies are needed in classroom or self-learning settings to validate how effective StatPlayground can be for learning practical statistics.

StatPlayground is a prototype, more work is needed before our vision is attained.

## 4.2 Helping Data Workers Ask Questions About Statistical Procedure Selection

The second problem deals with how data workers post queries about selecting statistical procedures. In the previous chapter, we identified several issues with the use of Q&A platforms for statistical inquiries. An important issue deals with missing information about experiment and research questions, both of which are required for respondents to answer their queries. A related but different issue is that questioners specify information in an unclear manner, which also leads to difficulties in answering their queries. In this section, we discuss some approaches to tackle these issues.

Our vision is a tool that collects information about experimental data and design in a structured manner.

Borrowing techniques from Q&A platforms used for questions on programming, we recommend to provide a structured way to specifying information about dataset and experiment. Tools like Plunkr[11] and CodePen[12] allow programmers to share reproducible source code so that it is easier for respondents to debug them. We envision a tool that allows data workers to specify information about their dataset and experimental design in a structured manner, and use this structured format, e.g., as a visualization, in the questions posted on Q&A platforms. For example, Touchstone2 is a research artifact that provides a structured way to provide details about dataset and experiment [Eiselmayer et al., 2019].

Q&A platforms can integrate features for obfuscating sensitive information about dataset.

In our content analysis, we found that there was a need for questioners to obfuscate sensitive information about their dataset. There are several technologies that allow such an obfuscation while retaining the underlying characteristics of the data, such as the IBM library for differential privacy[13]. Q&A platforms can integrate such functionalities to allow data workers to hide sensitive information when needed.

Q&A platforms should allow respondents to collaboratively edit questions to include missing and unclear information.

Finally, we identified several examples where respondents were able to assume information about the dataset and experiment, when such information was either missing, incomplete, or unclear. While this can potentially lead to incorrect

[11]https://plnkr.co/
[12]https://codepen.io/
[13]https://github.com/IBM/differential-privacy-library

responses, allowing experts to collaboratively agree upon such missing or unclear information can get all respondents on the same page, helping them provide responses with more certainty. CrossValidated, one of the websites we sourced questions from, allows respondents who have adequate reputation to edit the information in questions.

# Chapter 5

# Understanding How Data Workers Perform Data Science

*"I suppose it is tempting,*
*if the only tool you have is a hammer,*
*to treat everything as if it were a nail."*

*—Abraham Maslow*

**Publications and Author's Contributions**
Part of the research in this chapter has been published as two extended abstracts, [Subramanian et al., 2018] and [Subramanian et al., 2019b], and two papers, [Subramanian et al., 2020b] and [Subramanian et al., 2020a]. The author of this thesis developed the research questions, conducted interviews and observations, and analyzed data for all studies. Johannes Maas helped compute the inter-rater agreement for our study investigating hypothesis testing workflow, discussed in Section 5.1. Nur Hamdan contributed to the analysis of our study investigating programming modality use discussed in Section 5.2. Prof. Chat Wacharamanotham contributed to our code analysis in Section 5.3.

We discuss data workers' workflows with programming languages for data science.

The artifacts and recommendations we introduced in the previous chapter help data workers learn practical statistics, improve their workflow for selecting statistical procedures, and help them avoid common errors in statistical practice. In this chapter, we present data workers' workflows when performing data science tasks. We discussed earlier that there are three interfaces that support hypothesis testing: GUIs, visual programming interfaces, and programming. Of these, programming languages like R and python are popular among both professional data workers and data workers [Analytics, 2017; Piatetsky, 2018]. R, for example, has the highest adoption rate over the last decade, mainly due to its package infrastructure, community support, and open-source nature. Our research in this chapter deals with data workers' workflows with programming languages, particularly R and python.

Our first observation study motivated a content analysis and another observation study.

We report findings from two observation studies and a content analysis. In our first observation study, discussed in Section 5.1, we aim to understand data workers' overall workflow when performing hypothesis testing. The findings from this study motivated our second study and a content analysis. The second observation study, discussed in Section 5.2, helps us understand how the various programming interfaces support data science tasks. Finally, our content analysis, discussed in Section 5.3, aims to quantify the quality of source code in data analysis.

## 5.1 Study: Understanding the Hypothesis Testing Workflow: Exploration and Confirmation

https://osf.io/45jkf/?view_only=15a5624c310e4e60a78b46700284412c[a]

[a]Contains the coding scheme and details about our observation study participants discussed in this section.

Data science programming involves exploratory programming workflows, which can lead to issues when writing production code.

When using programming to perform data science tasks, data workers follow an exploratory programming workflow. After experimentation, data workers need to store and/or present their code. To do this, data workers may rewrite their code to improve code quality [Sandberg, 1988], document the insights and rationale behind their analysis decisions [Sandve et al., 2013; Abdul et al., 2018]. This can be laborious, as experimentation results in poor quality source code [Kery et al., 2017].

Figure 5.1 illustrates this workflow.

We observed ten data workers experiment and confirm their source code.

To understand how data workers write exploratory code and work with this code afterwards when storing and presenting, we observed ten data workers perform hypothesis testing tasks. In the study, our participants experimented with ideas in source code, and then revisited their experimental code to refactor them, as well as to document the insights and rationale. This gave us an insight into both their exploratory and *'confirmatory'* workflows. We use the term exploratory phase to refer to the initial analysis phase that conforms to an exploratory programming practice. The term confirmatory phase (not to be confused with confirmatory data *analysis* [Tukey, 1980]) refers to writing production code and reports for the sake of storage and presentation. In the remainder of this section, we first discuss existing work about understanding workflows in hypothesis testing that motivated our work, and then present our methods, findings, and a discussion of our findings.

### 5.1.1 Related Work: Overview of Hypothesis Testing

Exploratory and confirmatory analyses constitute hypothesis testing.

We discussed background information about hypothesis testing, such as the exploratory programming practice and phases in data science, in Section 2.5. In Section 2.2.3, we discussed two phases of hypothesis-driven data analysis: (a) exploratory data analysis (EDA) and (b) confirmatory data analysis (CDA). In HCI, EDA is often used in exploratory studies [Cockburn et al., 2018] and in certain domains like data visualization [Fekete et al., 2019]. CDA involves methods like null hypothesis significance testing (NHST), estimation using 95% confidence intervals, and regression analysis.

An important issue in hypothesis testing is selective reporting of analysis.

One key critique of hypothesis testing is *HARKing,* i.e., Hypothesizing After the Results are Known [Kerr, 1998; Cockburn et al., 2018], also known as "$p$-hacking" [Gelman and Loken, 2013], "fishing" [Humphreys et al., 2013], or "wandering down the garden of forking paths" [Gelman and Loken, 2013]. It refers to a situation where the researcher tries many analyses, but reports only the successful analysis. Like Gelman [Gelman and Loken, 2013] and Pu et al. [Pu and Kay, 2018], we believe that HARKing is unintentional, and that it is a prob-

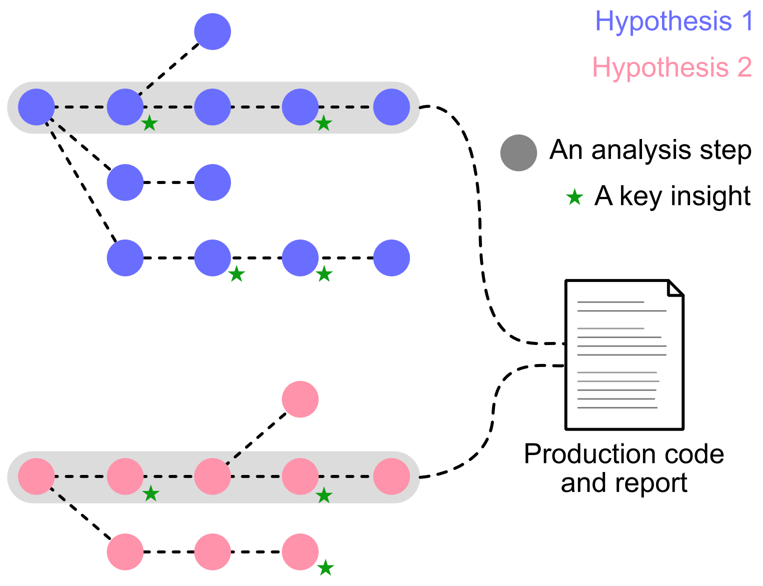


**Figure 5.1:** During data analysis, data workers validate several hypotheses. This involves several steps like loading data, viewing descriptive statistics, and confirmatory analysis. Analysis generates insights, which may lead to further analyses. After experimentation, data workers write production code and/or reports.

lem of analysis tool design. Omitting parts of the analyses from reports also leads to a lack of transparency [Simmons et al., 2011; Dragicevic et al., 2019].

### 5.1.2 Observational Study: Understanding the Tension Between Exploration and Confirmation

We first discuss how we collected and analyzed the data, and then present the key findings in Section 5.1.3.

#### Data collection

We collected videos of ten data workers performing data analysis tasks. All participants reported to have prior experience (median = 2 years, range = 0.5 to 10 years) using RStudio, a commonly used IDE for data analysis [Vikraman, 2018]. Participants were recruited via emails sent to local university and to R discussion group[1] in a nearby city. We used convenience sampling initially, but eventually aimed to sample participants with specific criteria. For example, after learning about potential differences in workflows between HCI and psychology students, we sent emails who were taking an advanced statistics for psychology course at the local university.

Our participants are RStudio users from various backgrounds and with varied experiences.

We deliberately recruited participants who mainly use RStudio to minimize the effect of programming IDE on analysis workflow. Some participants used multiple programming languages and IDEs, but our observations and interviews are only based on their RStudio usage. We aimed to improve the external validity of our experimental data by sampling participants with varied experiences and from different disciplines, such as like numerical analysis, applied psychology, and HCI. In the following discussion, we will refer to our participants as P1–P10.

Most observations (8 out of 10) occurred at the participant's place of work, which included various research labs at the university. For the other observations, the participants were invited to the local research lab. Seven participants used their own computers to analyze data during the session. Others used a machine at the local lab setup with RStudio and datasets.

We observed our participants analyze their own dataset or a fabricated dataset.

Six participants analyzed fabricated data comparable to real-world analysis, while others conducted analysis that was part of their work. Fabricated datasets were developed by the author of this work, and mimicked data from a typing experiment with different keyboard layouts. Participants who analyzed fabricated data did so because either a) they did not have any on-going analysis at the time of our study or b) they worked on data-sensitive projects. Tasks involved both exploratory and confirmatory phases of the analysis, as we were interested in learning about both. After the experimenter discussed the data set and analysis intent with the participant, the participant was asked to come up with initial hypothesis that they wanted to validate that they then explored in RStudio.

Several participants analyzed fabricated data.

During the session, participants were encouraged to think

[1]`https://www.meetup.com/de-DE/koelnrug/`

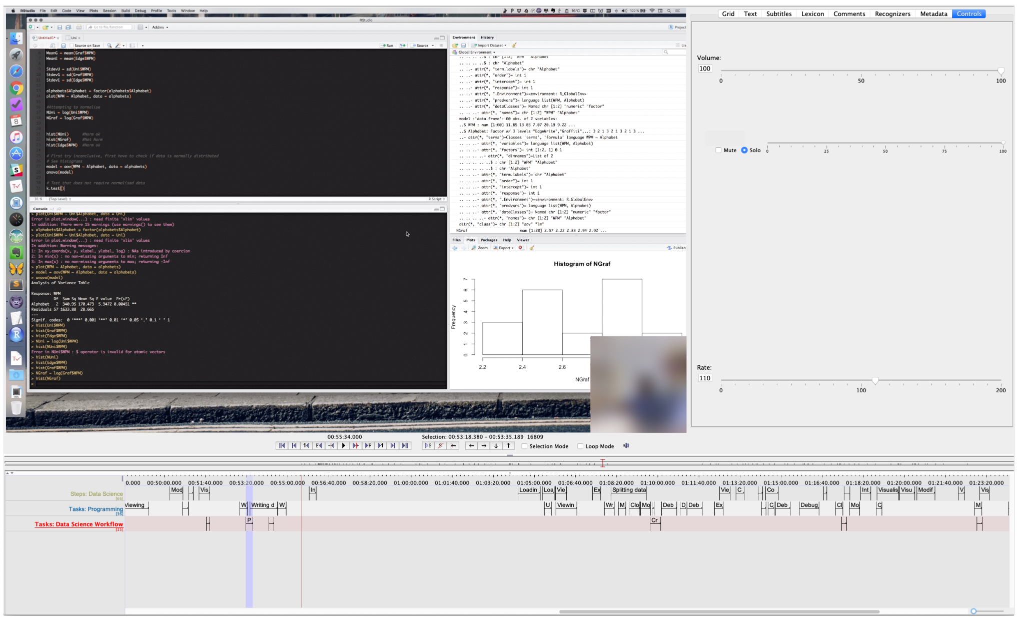


**Figure 5.2:** Our analysis setup and coding process in ELAN (`https://archive.mpi.nl/tla/elan`) annotation software. To analyze videos, after getting acquainted with the content, we marked snippets corresponding to three levels of codes: domain-agnostic programming tasks, tasks in analysis, and steps in exploratory programming workflow.

We employed a think-aloud protocol with minimal prompts from the experimenter.

aloud. The experimenter took notes of what participants said and did, and noted points of interest, e.g., where participants encountered difficulty during the analysis. Since the participants' utterances were level-3 verbalizations [Ericsson and Simon, 1980], they are not reliable and have limited validity. For this reason, using proposal from Boren and Ramey [Boren and Ramey, 2000], we use the observational videos as the main source of qualitative analysis, and use these verbalizations to only provide a potential explanation of events. After the session, the experimenter clarified any questions that came up during the observation. We collected both the video and audio logs of our study sessions. In total, we collected approximately 8 hours of audiovisual recordings (median = 54 minutes).

### Method

We began our analysis by extracting clips from recordings that met one or more of the following criteria: (a) participant interacts with RStudio, (b) participant interacts with another app to conduct analysis, e.g., does a web search on analysis procedure, and (c) participant utters something relevant about their analysis. After a round of initial coding, we generated three tiers of *process codes* [Bogdan and Biklen, 2007] as shown in Figure 5.2. These codes were used to categorize

We extracted clips from recordings that were relevant, and used qualitative coding to analyze them.

1. domain-agnostic *programming tasks*, e.g., writing comments, creating a new file, or cloning code;

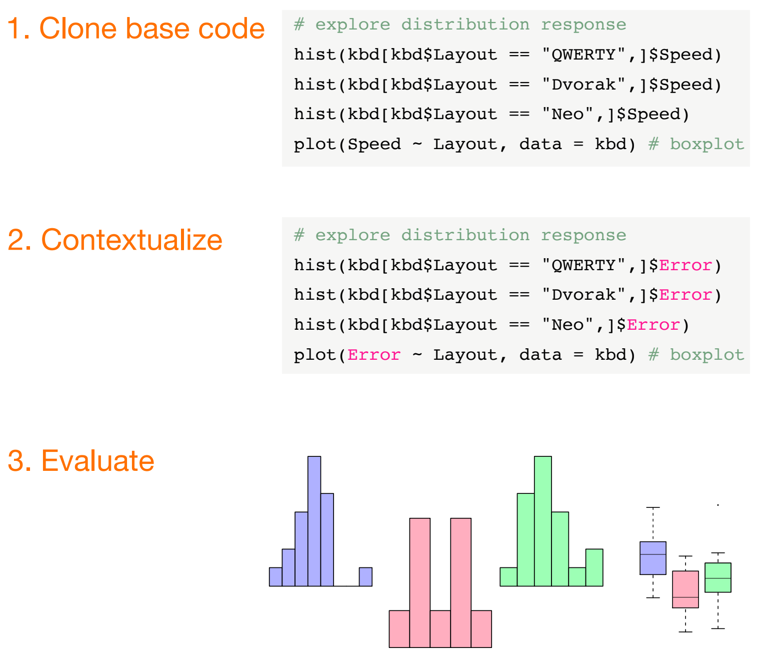
2. tasks in *analysis*, e.g., computing descriptive statistics, visualizing data, or building models; and

3. steps in *exploratory programming workflow*, e.g., creating alternatives, writing production code, or searching for code.

All analysis was done by the author of this work, with regular discussions with one researcher and regular memoing to facilitate reflection. An inter-coder agreement score was not computed. The resulting coding scheme can be found in supplements. We did not use any system logs as our goal was not to provide a statistical breakdown of data workers' workflows, but instead gain qualitative insights about it.

## 5.1.3 Findings

In this section, we describe our participants' workflows to get from exploratory to confirmatory phase. Sections 5.1.3–5.1.3 discuss exploratory phase, and the remaining sections focus on confirmatory phase, specifically how information in exploratory code is utilized by our participants during confirmatory phase.



```
# explore distribution response
hist(kbd[kbd$Layout == "QWERTY",]$Speed)
hist(kbd[kbd$Layout == "Dvorak",]$Speed)
hist(kbd[kbd$Layout == "Neo",]$Speed)
plot(Speed ~ Layout, data = kbd) # boxplot
```

```
# explore distribution response
hist(kbd[kbd$Layout == "QWERTY",]$Error)
hist(kbd[kbd$Layout == "Dvorak",]$Error)
hist(kbd[kbd$Layout == "Neo",]$Error)
plot(Error ~ Layout, data = kbd) # boxplot
```

**Figure 5.3:** Data workers follow an exploration routine: Clone base code, view the context of current dataset to modify the arguments of cloned code, and execute code to determine its state.

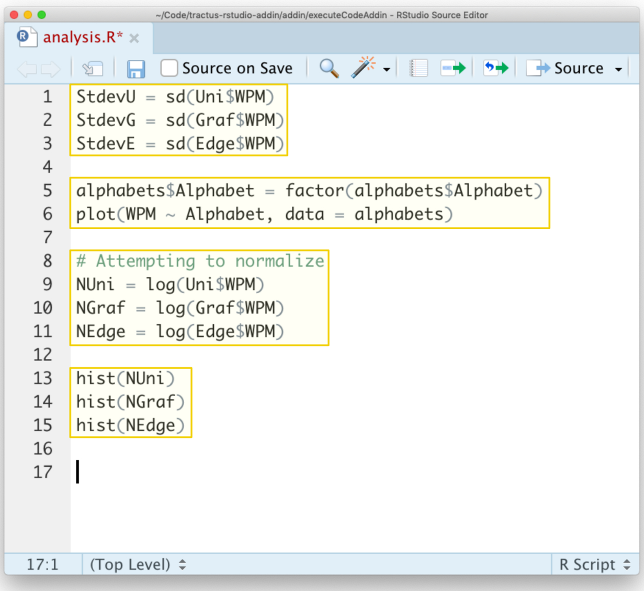
### How do data workers experiment through code?

Consoles were used to kickstart analyses.

Most participants (P1, P3, and P7–10) used consoles to begin their analysis and then eventually documented source code in scripts. Consoles were used to test programming syntax and libraries, and participants reported to prefer consoles for such tasks as they are more interactive than using source code files. We discuss consoles in more detail in Section 5.2.4.

Experimentation involved a standard routine of cloning and modifying source code.

After using consoles to kickstart their analysis, participants wrote source code in script files. Experimentation was a recurring task for all participants. All participants used a standard routine (Figure 5.3) to experiment with code by exploring alternatives: (1) *Find* and *clone* base code, (2) *contextualize* code, and (3) *evaluate* state. Base code, such as code from previous or current analyses or samples from the web, is there-


~/Code/tractus-rstudio-addin/addin/executeCodeAddin - RStudio Source Editor
analysis.R*
Source on Save
Source
17:1
(Top Level)
R Script

```
StdevU = sd(Uni$WPM)
StdevG = sd(Graf$WPM)
StdevE = sd(Edge$WPM)

alphabets$Alphabet = factor(alphabets$Alphabet)
plot(WPM ~ Alphabet, data = alphabets)

# Attempting to normalize
NUni = log(Uni$WPM)
NGraf = log(Graf$WPM)
NEdge = log(Edge$WPM)

hist(NUni)
hist(NGraf)
hist(NEdge)
```

**Figure 5.4:** A reconstruction of the analysis code written by P08. Data workers use code blocks, sometimes prefixed with a descriptive comment, to group meaningful steps in the analysis.

fore a crucial component in source code experiments. As one can expect, such code experiments were mostly conducted in a non-reusable, non-modular fashion (P1, P4, and P6–10). This validates a finding from prior work by Kery et al. [Kery et al., 2017].

After cloning, participants used the variable names and variable values from their session to update the arguments in the clone to suit their new exploration. As the final step in this routine, participants evaluated the source code by executing it. This allowed them to compare source code alternatives, gain insights, and determine next steps.

Participants need to compare outputs of their alternatives to progress in their analyses.

### How do data workers organize source code?

Participants organized their source code into blocks, and used them to navigate source code later.

Most participants (P2–4 and P6–10) organized their source code into *blocks*. Each block represents one meaningful step in the data science task, e.g., loading data or checking assumptions for a statistical test. Participants usually prefixed these blocks with a short comment describing the high-level task. Only two participants, P1 and P5, reported to use blocks minimally in their work. While documentary structures like white spaces and source code comments are used to improve code readability [Vanter, 2002], blocks were reported to help navigate source code later (P1, P4, and P8) (*"[...] blobs [i.e., blocks] are useful when I go through the source code [...] help me parse code easier."* – P4).

### How are hypotheses validated and what prompts alternative analyses?

Several tasks in hypothesis testing used R's formula notation.

As stated earlier, a hypothesis is a binary statement that expresses the relationship between two or more variables. All participants performed statistical significance tests during our observation. In addition to these tests, other steps that constitute hypothesis validation are visualizing data, computing descriptive statistics, performing tests for statistical assumptions, and performing post-hoc tests. Our participants expressed their hypotheses in source code, required for performing significance tests, by using R's formula notation[2].

Formula notations have clear syntactic signals that allow us to identify the hypothesis tested using the source code.

The simplest formula notation is of the form `measure ~ factor`, which refers to a hypothesis that investigates whether the factor has a significant effect on the measure. While source code corresponding to other steps, such as visualization and tests of statistical assumptions, also used this formula notation, column selection and dataset manipulation operations were more prevalent for such tasks. For example, P1 performed a test for normality using `shapiro.test(data[data$method == "Unistrokes",].speed)`, where `Unistrokes` is a level of the factor, `method`, and `speed` is the mea-

[2] `http://tinyurl.com/y5of72lr`

surement. (The participant analyzed a dataset that compared text entry techniques in mobile phones.) The statement would therefore be part of the source code that validates the hypothesis `WPM ~ method`, i.e., a hypothesis that investigates whether typing methods have an effect on the typing speed.

Alternative explorations were usually preceded by modifications to data.

We encountered situations in which our participants performed multiple analyses to validate the same hypothesis. We found that several participants (P1, P2, P4, and P10) conducted these alternative analyses after the data have been modified, e.g., by transformations and removal of outlier data points. This is not surprising, since the analysis method is almost entirely dependent on the data characteristics [Gelman and Loken, 2013]. Changes to data mostly resulted from obtaining insights from another step in the analysis, e.g., learning that data are log-normally distributed prompted a change to data through data transformation. There were only a few situations in which data were modified impulsively by participants (*"I will [... see] what happens to distribution when these [data] points are removed."* – P2). Participants used variable names like `logData` and `data_new` to track the changes to data.

### How do data workers rationalize their analysis?

Participants used prior statistical knowledge, execution output, and external information sources to rationalize the choices they made in analysis.

As discussed earlier, data workers have the responsibility of reporting the rationale behind the decisions they made in their analysis, such as transformations and using a non-parametric statistical test, alongside reporting the key insights they obtained in their analysis. Participants used the following information from source code to rationalize analysis decisions:

- Most participants (P1, P5–10) had *predetermined* one or more analysis steps, often based on their prior experience and expertise in statistics. For example, before performing analysis, P7 knew that one of the factors in his data had three levels and decided to use an appropriate statistical test (*"I will probably be doing an ANOVA test here, followed by pairwise comparisons."* – P7). Participants do not capture such rationale *explicitly* during exploration, but later include them in production code or analysis reports.

- All participants used *results* of previous executions as rationale, since results often lead to new insights about data. For example, P8 used a quantile-quantile plot to rationalize the use of a non-parametric test.
- In addition to above information, some participants (P2, P3, P6, and P9) made references to external resources, e.g., web articles[3], including them as rationale. These were usually documented in the production code using comments and reported to be included in the reports.

**How do data workers track data insights?**

Insight were often too verbose to be captured as comments.

While insights are typically gathered from source code executions, they are often more than the results themselves and include the analyst's interpretation. Therefore insights in our study were often detailed and sometimes too verbose to be captured as source code comments, e.g., as P8 mentioned:

> *"I would recommend [users to] use EdgeWrite [a text-input technique] here because the variance [of typing speed] is low, but one can also use Graffiti which has a higher average."* - P8

Most participants relied upon their memory to recall such insights.

Such insights are hard to capture as comments, and most participants simply attempted to retain this insight in memory. Only two participants, P4 and P5, used RMarkdown notebooks to track insights. P6 and P9 used comments with some references to capture insights, e.g., `Test is inconclusive, see model o/p`. However, most participants (P1–3, P7, P8, and P10) did not use comments to document insights and relied upon their memory instead (*"The information [about insights from exploration] is something I still have in my head and it's usually [just] a few key insights."* – P7).

Participants had to re-execute their exploratory code often several times.

Except for P4 and P5 who used RMarkdown notebooks, all participants had to execute their source code over and over again

[3]E.g., `https://stats.idre.ucla.edu/other/mult-pkg/whatstat/`

when writing production code to recall rationale and insights. This shows that data workers overlook the need for capturing information on regular intervals during exploration. As further evidence of this fallacy, we found that some participants found it difficult to keep track of the source code that produced a data insight even during the exploration stage. For example, P7 could not find a code snippet they was looking for during exploration and soon became frustrated (*"One of these three distributions is not normal... where is the line [of code] where [sic] I computed [i.e., plotted] the histograms?"* – P7).

### What do data workers use comments for?

Data workers used comments to capture insights and rationale; for navigation; and to manage source code alternatives. While comments were commonplace in production code, some participants (P1, P7, P8, and P10) were reluctant to use comments during exploration (*"I write comments [only] when I have found something interesting [i.e., an insight]."* – P8).

Comments were used to capture a variety of information, but were not commonly used during exploration.

This is an implication of the exploratory programming practice, in which the focus is on getting to the results faster and evaluating them than writing high-quality source code. Comments in production code were used to provide a high-level task description, e.g., `apply ANOVA` (P6). Some participants P2–4 used comments to describe what they did programmatically, e.g., `loop through each data segment...` (P2). Additionally, several participants (P2, P3, P5, P7, and P8) used comments to capture rationale and insights in production code, e.g., `Preconditions for wilcox test are met` (P7).

Comments in production code captured analysis details, programming details, and rationale.

P2 used stylized comments to distinguish comments about insights from other comments. P3 used section comments[4] for task descriptions, to help them navigate code more easily. There were some individual differences in frequency and style of comments, such as variations in verbosity and preference of inline vs. tail comments. Some participants (P3, P4, and P7) used comments to temporarily disable code snippets.

There were variations among our participants in their use of comments.

[4]https://support.rstudio.com/hc/en-us/articles/200484568-Code-Folding-and-Sections

**How do data workers rewrite source code?**

Participants rewrote source code by cleaning up experimental code or writing code from scratch.

After exploring alternatives to obtain data insights, participants rewrote exploratory code to be able to reuse it in analysis reports or for production. Such source code would be disseminated and/or stored for later. Participants rewrote code in two ways: (1) clean up current code (P1, P2, P4–6, and P9) and (2) rewrite code from scratch (P3, P7, P8, and P10). To help prune their exploratory code, participants used code blocks, comments, and variable names to evaluate source code statements.

Execution output from consoles and comments were used to identify experimental code snippets.

To help identify code snippets they were looking, participants made use of executions in the interactive console and source code comments. However, since most participants did not use detailed comments during the initial exploration phase, most participants had to re-execute source code several times, a behavior we also observed during the initial exploration phase. Once located, source code is often cloned into a new source code file or re-organized in the current file.

Writing production involved a number of steps that relied upon experimentation.

Participants changed variable names in situations where temporary names were used, added or modified comments, and rearranged code snippets within or across source code files, although the latter occurred less frequently. Rewriting source code required participants to retrace their steps; they did this by looking at the current exploratory code, previous executions that were displayed in the interactive console, and the panel that listed a history of statements they have executed.

Finding relevant source code that worked as expected proved to be difficult for our participants.

We identified the following issues our participants faced from our observations: (1) participants found it difficult to find the correct version of a source code snippet, and (2) had difficulty making sure that the execution dependencies of the code were correctly cloned. For example, after validating several hypotheses, P7 wanted to move their source code that was used to validate one of the hypothesis to a new file. They looked through their code to find this code, and then pasted it into the new file, only to find upon executing it that an earlier statement (that was used to set a variable as a factor) had not been copied. Failing to copy all execution dependencies can result in faulty executions or incorrect results, which can be even more problematic.

### 5.1.4 Limitations and Future Work

Although our participants came from diverse disciplines, most were from academia and do not represent the population of data workers accurately. We recommend the reader therefore to refrain from generalizing our findings to all data workers. For example, industry practitioners may follow rigorous coding guidelines and write more modular code, reducing the issues in finding prior code and execution results. We observed each participant for about an hour and nudged them to show us their workflow of getting from exploration to confirmation. This might have hastened them, and led to workflows not representative of their usual work. Also, real-world data science projects tend to last for weeks or even longer, and the analysis code could span multiple files. Future studies could investigate data workers' workflows through a longitudinal study.

Our participants were mostly from academia, which limits the external validity of our findings. Further longitudinal studies are needed to validate data workers' workflows.

### 5.1.5 Summary: Key Findings

In summary, we can take away the following from our observational study on understanding data workers' hypothesis testing workflows:

1. During exploration and while rewriting code, data workers have difficulty keeping track of the source code that had produced key insights and the states of their source code experiments.

2. Exploration involves a standard routine of *finding* base code, *cloning, contextualizing,* and *evaluating* it.

3. Hypotheses are the building blocks of an analysis. Source code written to validate hypotheses have syntactic signals that make them detectable. Data manipulations, such as transformations and outlier removal, lead to alternative analyses, and data workers have to remember variable names to keep track of these data changes.

4. Data workers organize their code into blocks when writing code; these blocks are used as to navigate the source code later during confirmation.
5. Data workers use (a) prior knowledge of statistical procedure, (b) text and graphic output of source code, and (c) external resources like web pages to rationalize their analysis.
6. Data workers do not capture data insights initially, but instead rely on their ability to recall from memory and sparse source code documentation.
7. It is hard for data workers to track the execution dependencies in their source code. This leads to faulty executions or incorrect results when such code is cloned into a new file, a step that is not too uncommon when writing production code.
8. Data workers *rerun* code frequently in order to *recall* rationale, insights, and the states of their source code explorations.

## 5.2 Study: Understanding How Various Interfaces Support Data Science Programming

https://osf.io/n6hsv/?view_only=4c036ab7a1324935bdfad681b6107fa3[a]

[a]Contains details of our coding scheme.

We conducted a study to investigate how various interfaces for data science programming support data science tasks.

In the previous section, we discussed the overall workflow of data workers when performing hypothesis testing. During our study, we became curious about the use of consoles, which some of our participants used for experimentation as discussed in Section 5.1.3. We were also interested in learning how well computational notebooks, which uses a cell-based programming approach, could support data science programming in comparison to scripts. We therefore conducted an observation study to understand how data workers use these three programming modalities: *scripts*, *computational notebooks*, and *interactive consoles*. We provide a brief overview of the three modalities below, discuss our data and method in Section 5.2.2, and then present and discuss our findings in Section 5.2.4.

### 5.2.1 Brief Overview of Programming Modalities

*Scripts* (Figure 5.5, left) support conventional storage and execution of source code. All existing IDEs used for data science programming support execution of entire scripts, with most also allowing selective execution of snippets. When scripts are executed, text output and error messages are shown in the interactive console (Figure 5.5, center), while graphic output is usually shown in a dedicated window.

Scripts support linear execution and storage of source code.

*Consoles* (Figure 5.5 center) follow the Read-Eval-Print Loop style of execution [Iverson, 1962]. Source code can be directly written onto the console and executed line-by-line or can be executed from the script file. Lines of code are executed sequentially with the output displayed immediately below each executed line of code. The console usually has a display limit of a few thousand lines, and the session's state beyond the current limit is lost. To view the previously executed commands, the analyst needs to press the ↑ arrow key or scroll using her mouse.

Consoles allow quick and easy execution, but does not explicitly store source code and output.

*Computational notebooks* (Figure 5.5, right) allow users to organize their source code into *cells.* A cell typically represents one computational chunk of the data science task, and can be executed in a non-sequential order. Unlike scripts, execution output is juxtaposed with the cell. Additionally, computational notebooks support narration via Markdown. These are textual descriptions of, e.g., the analysis rationale, that can be interwoven with the code.

Notebooks offers cells to organize source code. Notebooks also support Markdown for adding verbose text.

In order to evaluate how common these modalities are in current programming IDEs, we surveyed existing IDEs of the main scripting languages. We found that most modern IDEs support all three modalities, although there are some differences in how well they support them. For example, Jupyter and R Markdown allow the programmer to run all code cells above or below the selected cell; MATLAB only allows the program-

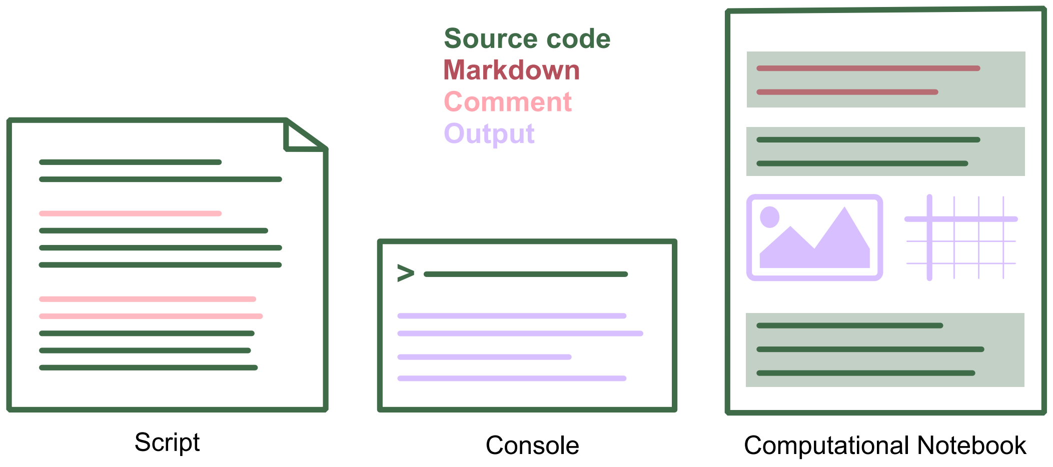


**Figure 5.5:** Current scripting language IDEs support writing and executing code via three programming *modalities*: *scripts* (left), *consoles* (middle), and *computational notebooks* (right). In this paper, we investigate how these modalities are used in data science programming.

mer to run all code cells below, but not above. Results of our survey are in supplements.

We support our findings in this section using the data collected from observations and interviews with 21 data workers; results of an online survey, which we use to extrapolate our key findings about modality usage to a larger sample; and by applying Green's cognitive framework for programming [Green and Petre, 1996] to help reason why a modality suits data science tasks better than others. Based on our insights, we present design recommendations for future programming IDEs, discussed in the next chapter, in Section 6.3.

### 5.2.2 Data Collection and Method

Our study sessions with 21 data workers included a walkthrough and observations. Our participants analyzed real-world or fabricated dataset.

We collected data from 21 data workers (nine female, median age of 27). We aimed to account for differences in data workers' background, domain, and scripting languages used. We used purposive and snowball sampling methods to invite participants through emails to university mailing lists and direct emails to contacts from our professional network. We encouraged the receivers to forward the email to others who might

| **Participant** | **Experience** | **Domain (Scripting language)** | **Modalities** |
|---|---|---|---|
| IP01 | 1 | Significance testing (*R*) | scripts |
| IP02 | 2 | Significance testing (*R*) | scripts |
| IP03 | 1 | Significance testing (*R*) | scripts |
| IP04 | 1 | Significance testing (*R*) | scripts |
| IP05 | 2 | Machine learning (*Python*) | notebooks |
| IP06 | 5 | 3D data processing (*Python*) | scripts |
| IP07 | 3 | Significance testing (*R*) | scripts |
| IP08 | 2 | Machine learning (*Python*) | both |
| IP09 | 0.5 | Financial analysis (*R*) | both |
| IP10 | 3 | Machine learning (*Python*) | both |
| | | Significance testing (*R*) | |
| IP11 | 1 | 3D data processing (*Python*) | scripts |
| | | Numerical analysis (*MATLAB*) | |
| IP12 | 2 | Equation modeling (*R*) | scripts |
| | | Significance testing (*R*) | |
| IP13 | 1 | Machine learning (*Python*) | both |
| | | Significance testing (*Python*, *R*) | |
| IP14 | 5 | Machine learning (*Python*, *MATLAB*) | both |
| IP15 | 3 | Machine learning (*Python*) | both |
| IP16 | 10 | Machine learning (*Python*, R) | both |
| IP17 | 3 | Machine learning (*Python*, *MATLAB*) | both |
| IP18 | 7 | Numerical analysis (*MATLAB*) | scripts |
| IP19 | 8 | Numerical analysis (*MATLAB*, *Python*) | scripts |
| IP20 | 5 | Machine learning (*Python*) | scripts |
| IP21 | 8 | Significance testing (*R*) | scripts |
| | | Machine learning (*Python*) | |

**Table 5.1:** Participant details from our study on programming modalities

fit the criteria. Participants include 13 academic data workers who self-identified as researchers from different fields such as psychology, HCI, and electrical engineering, as well as 8 graduate students. All participants reported to use at least one programming modality regularly in their work, which is a criteria to take part in the study.

Our study involved collecting background data and information about participants' workflows via interviews, and then a walkthrough.

We first conducted an interview of about 20 minutes with each participant to gather background information: programming experience, scripting language experience, current and previous projects with scripting languages, and IDEs they were regularly use or were familiar with. After obtaining information about our participants' background through interviews, we asked them to walk us through their recent data science projects. This phase, which lasted 30 minutes on average, helped us understand our participants' workflow with the programming IDE. We requested the analysis files our participants showed us, but participants were hesitant to share these files with us as these either contained sensitive information or participants were not sure if these could be shared with us.

After walkthroughs, we wanted to observe participants, either remotely or in-person, perform real-world tasks. Most participants felt this was intrusive, and provided a screen recording of their work instead. Also, since some participants (P1, P2, P4, P7, and P10) worked on projects with sensitive data, they analyzed a fabricated dataset to mimic their work. We collected audio and video logs of the interview and walkthroughs, and screen capture of observations. We later viewed the screen recordings, and followed through with participants for clarifications. For details of our participants and the data we collected, see Table 5.1.

We used grounded theory methodology to analyze our data. We substantiated our interviews and observations with an online survey.

To analyze our data, we followed the constant comparative method [Boeije, 2002; Glaser and Strauss, 2017] of the grounded theory methodology [Strauss and Corbin, 1994; Smith et al., 1995]. We first created full transcripts of the interviews. Then, following the guidelines in [Saldaña, 2013], the author of this work did initial coding to come up with a coding scheme (see supplements). This coding scheme was validated by an independent coder to achieve a good agreement of *Cohen's* $\kappa = 0.84$ [Landis and Koch, 1977].

We performed two cycles of coding to analyze the data we collected in the study.

We used *process codes* [Bogdan and Biklen, 2007] to code our participants' workflows with all modalities, and *descriptive codes* [Wolcott, 1994] to code the problems faced by our participants and information about their background. Process codes allowed us to categorize the actions in the observational video, e.g., "exploring alternative" and "executing code". Descriptive codes, in which the analyst assigns a topic to each coded entity, allowed us to categorize a) the demographic information and b) the analysis technique and tools used. After the first coding cycle, we developed *axial codes* by using the following models: *action/interaction*, *causal conditions*, and *phenomenon* [Strauss and Corbin, 1998]. Each model gives us a perspective in which to look at the codes from the first coding cycle in order to find grouping patterns. In addition to interviews and observations, we also conducted an online survey with 62 data workers to substantiate our findings about modality usage for data science programming, discussed in Section 5.2.4.

We now discuss the main findings resulting from the analysis.

### 5.2.3 Phases in Data Science

Based on our analysis, and grounded in earlier work by Guo [Guo, 2012], we identified four phases in data science programming as shown in Figure 5.6:

1. **Data collection and cleaning:** In this first phase, participants reported to collect and clean data. Data are mostly collected by another person (P2, P3, P6, P8, P9, and P12–19), sometimes by the participants themselves (P1, P4, P7, P10, P16, P20, and P21); public data were used otherwise (P5, P11, and P16). After collecting data, participants prepare the dataset for analysis, e.g., by converting it to the right format and removing outliers. Data cleaning takes a lot of time, and is a recurring task done throughout analysis.

2. **Experimentation:** Participants often experiment with different approaches to obtain insights from their data. These approaches are implemented as "quick and dirty" prototypes in the source code, often in a less modular, reusable manner. In this phase, the data science workflow is highly iterative and unpredictable—experiments lead to comparisons, comparisons generate even more ideas to explore, and so on. For comparisons, our participants employed various criteria, e.g., code metrics like execution time and memory, but also domain-dependent

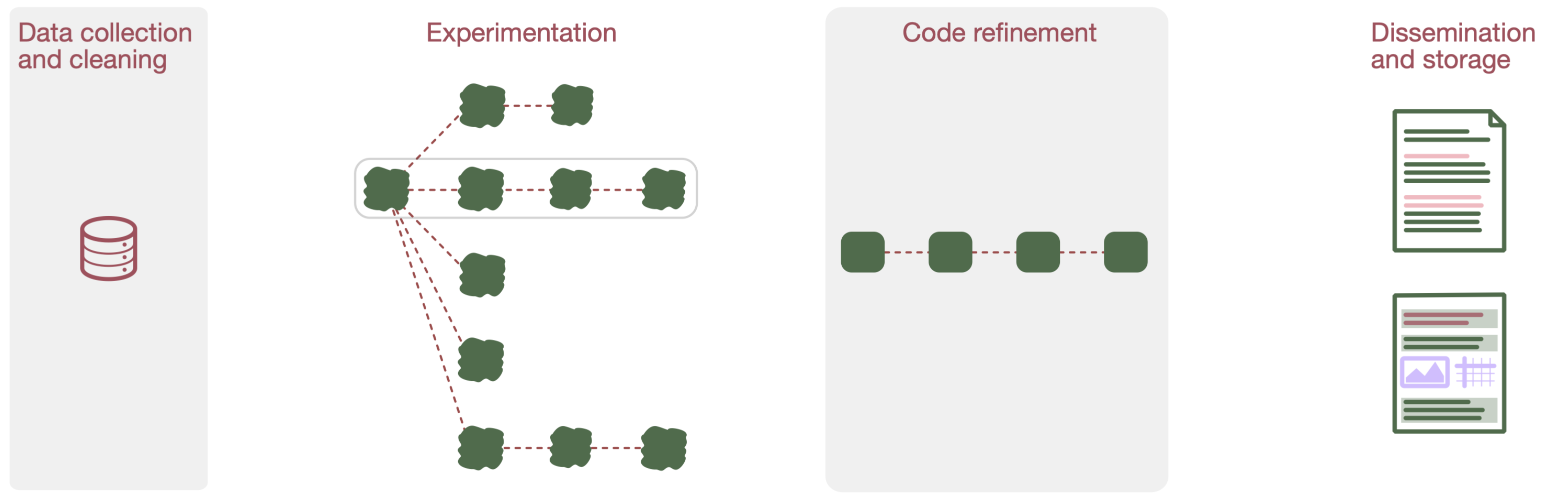


**Figure 5.6:** Data science programming involves four phases. After collecting and cleaning their data, data workers experiment with various approaches to solve the analysis task, before refining code for dissemination and storage.

criteria like statistical power (P4) and results of a fitting function (P10).

3. **Code refinement:** To prepare the source code of the analysis for dissemination, participants (a) improve the readability of source code by adding further documentation and pruning scratchpad code (P2–P4, P7, P9, P10, and P13), (b) refactor source code either in-place or into a new script file or computation notebook to improve code quality and reusability (P2–04, P8, P10, P13, and P15), and (c) extend source code so that it works with a wider range of input (P8 and P13). Extending source code to work with more input was more common in machine learning.

4. **Dissemination and storage:** The final phase in data science programming is to disseminate the results of the analysis to the outside world, e.g., as a research publication, or store it for later reuse. All participants reported disseminating the insights from their analysis, as well as storing their source code for later reuse. Many reported to disseminate source code (P5, P8, P9, P11, and P13–19), although in some situations only as snippets or pseudo-code (P8, P9, and P13–17).

### 5.2.4 Findings: How Data Workers Use Programming Modalities

Before looking at how data workers use programming modalities during the phases discussed above, we aim to determine how capable these programming modalities are in supporting the various tasks that occur these phases. Since data cleaning occurs throughout analysis, we consider it to be a task similar to experimentation in our discussion below. We use Green's cognitive dimensions [Green and Petre, 1996] to achieve this.

We describe phases in data science programming, and how interfaces should be designed to support them.

- **Experiment:** A key task of data workers during experimentation is to create and test alternative approaches to obtain insights from data. In comparison to other languages, scripting languages already lower the cost of experimentation by allowing programmers to program at

Data workers begin analysis by using source code to try out various approaches to perform data science.

a higher level of abstraction [Sandberg, 1988]. Since new source code uses existing code as a starting point [Kery et al., 2017], the programming interface should help users find existing code. Using Green's cognitive dimensions [Green and Petre, 1996], this would require the interface to have a high-level of *role-expressiveness*, i.e., the capacity of an interface to express the functionality of a piece of code to the programmer.

Data workers need to compare output of the alternative explorations to determine which exploration to continue with.

- **Compare execution results of alternatives:** During experimentation, data workers compare their alternative explorations, and select an exploration to further work on. To do this, data workers need to compare the execution results of these explorations. Data workers therefore need to be able to (1) locate the source code that belongs to an exploration, for which the interface should have high levels of role-expressiveness; (2) map the execution output to the source code that generated it; and (3) view execution outputs and source code for multiple explorations simultaneously. To support (2), the proximity of output to source code is important, a concept we term *'code-result distance'*. A high-distance means that the execution output is located farther away from the source code that generated it, and vice versa. (3) can be achieved with a high-level of *juxtaposability,* the capacity of an interface to display "any two portions of the program on screen side-by-side at the same time" [Green and Petre, 1996].

When writing code for storage or production, data workers have to curate their experimental code to remove scratchpad code and refactor source code.

- **Curate source code:** When refining code, data workers need to understand their code so that they can remove the scratchpad code. This would require the programming interface to have high levels role-expressiveness and low code-result distance. Some programming interfaces could alternatively support secondary notations like tags and programmer's comments to support this task. Further, since scratchpad code is removed, it is vital that any *hidden dependencies* are retained in source code [Green and Petre, 1996].

- **Present analysis and help reuse:** To present the analysis and results to stakeholders or in publications, the ability to add narrative and reproduce expected results from source code is desired. Code reuse can be achieved by allowing programmers to locate relevant code, as well as by copying-over or minimizing hidden dependencies.

Using this discussion as motivation, we wanted to reason how scripts, computational notebooks, and consoles support various data science phases. What are their roles? Do we need all modalities to co-exist? We attempt to answer these questions below.

### Role of consoles in data science programming

We would first like to address the role of interactive consoles in data science programming. Consoles can be considered as single-celled, stateless equivalents of notebooks. Most of our participants (P1–4, P7–10, P12, P14–17, P20, and P21) use consoles only for secondary tasks, e.g., to test the syntax of a function from a new library API, check if a file has been parsed correctly, and view the variable value (*"The console is for what I consider to be fire-and-forget tasks."* – IP4).

Consoles were mainly used for secondary tasks.

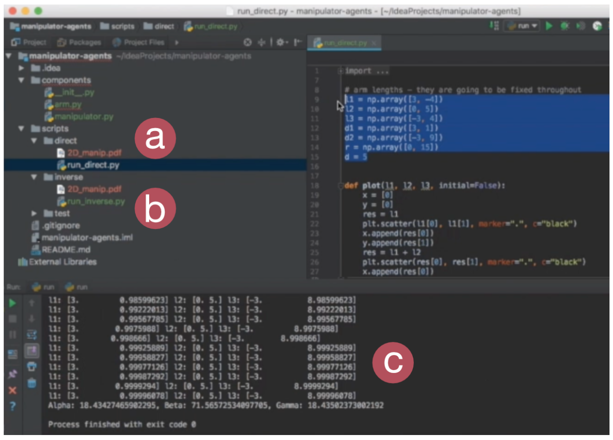
However, some participants (P2, P3, P4, and P9) reported to use consoles for core data science tasks. Consoles do not store the source code and execution output in a manner similar to scripts or notebooks. It instead implicitly stores them in the console window, requiring the programmer to incessantly scroll this window to find source code and execution output. This often led to issues as IP9 describes:

The interactivity and ease offered by consoles can be appealing.

*"I have had so many issues with losing commands [in consoles] that I try to put everything into scripts, but sometimes the console lures me in."* - IP9

The 'lure' referred to by IP9 above arises from the interactivity offered by consoles. Small-scale experimentation can be performed more conveniently by pressing the ↑ key to bring up the history of commands to modify and re-execute them, a concept known as *debugging into existence* [Rosson and Carroll, 1993]. Most participants who overused consoles had limited experience conducting analysis, indicating that the appeal of

Inexperienced participants had a tendency to overuse consoles.



```
import ...

# arm lengths - they are going to be fixed throughout
l1 = np.array([3, -4])
l2 = np.array([0, 5])
l3 = np.array([-3, 4])
d1 = np.array([3, 1])
d2 = np.array([-3, 9])
r = np.array([0, 15])
d = 5


def plot(l1, l2, l3, initial=False):
    x = [0]
    y = [0]
    res = l1
    plt.scatter(l1[0], l1[1], marker=".", c="black")
    x.append(res[0])
    y.append(res[1])
    res = l1 + l2
    plt.scatter(res[0], res[1], marker=".", c="black")
    x.append(res[0])
```

```
l1: [3.         0.98599623] l2: [0. 5.] l3: [-3.         8.98599623]
l1: [3.         0.99222013] l2: [0. 5.] l3: [-3.         8.99222013]
l1: [3.         0.99567785] l2: [0. 5.] l3: [-3.         8.99567785]
l1: [3.        0.9975988] l2: [0. 5.] l3: [-3.        8.9975988]
l1: [3.       0.998666] l2: [0. 5.] l3: [-3.       8.998666]
l1: [3.         0.99925889] l2: [0. 5.] l3: [-3.         8.99925889]
l1: [3.         0.99958827] l2: [0. 5.] l3: [-3.         8.99958827]
l1: [3.         0.99977126] l2: [0. 5.] l3: [-3.         8.99977126]
l1: [3.         0.99987292] l2: [0. 5.] l3: [-3.         8.99987292]
l1: [3.        0.9999294] l2: [0. 5.] l3: [-3.        8.9999294]
l1: [3.         0.99996078] l2: [0. 5.] l3: [-3.         8.99996078]
Alpha: 18.43427465902295, Beta: 71.56572534097705, Gamma: 18.43502373002192

Process finished with exit code 0
```

**Figure 5.7:** An example that illustrates the low juxtaposability of script files. IP7 stored his code experiments in two different python script files (a) and (b). To compare the execution outputs of these experiments, he had to execute a script file individually, remember or note down the result from the console (c), and then repeat this action with the other script file.

consoles for core data science tasks reduces as one gains more experience.

### Scripts and notebooks for experimentation and comparing experiments

Computational notebooks were preferred over scripts for experimentation.

Among our participants, many (P1–4, P6, P7, P9, P11, P12, and P18–21) do not use computational notebooks on a regular basis in their work. Of the remaining participants who use both computational notebooks and scripts, most (P5, P8, P13, and P15–17) reported to prefer computational notebooks over scripts for experimentation. The consensus is that scripts are not well suited to exploratory work. Scripts are low in code-result distance and juxtaposability. When a code snippet is executed, the execution output is displayed on the console or stored in the file system, away from the source code. As we discussed earlier, console windows can get cluttered over a period of time, further aggravating this problem.

As an example, consider Figure 5.7. IP07 has programmed their experiments, in this case, various statistical models for training data, into two Python scripts as shown in Figure 5.7a and b. To compare these models, they execute each script file individually and evaluates the output on the console (Figure 5.7c). Since they cannot see the execution outputs of both the scripts simultaneously, they have to remember insights in memory or note them down.

Script files make it difficult to compare execution output of programmer's experiments.

Notebooks, on the other hand, are better suited to experimentation. Our participants provided two main reasons for this. First notebooks allow programmers to organize their source code into manageable chunks in cells, allowing for an easier experimentation without the need to use messy program structures like comments (*"(On notebooks) I don't have to worry about the other code for now. I can focus on this [chunk of] code."* – IP15). Second, notebooks provide a low code-result distance, allowing data workers to easily map the execution output to the code cells that generated them. For smaller code cells, the juxtaposability is also improved, allowing data workers to view multiple pairs of cell and execution output at the same time.

Notebooks are suited to experimentation because of cell-based programming.

Beyond these two reasons, two participants (IP13 and IP16) reported that their preference for notebooks came from Markdown, which helped them be more oriented in their analysis (*"I like to write down what I want to do in notebooks before I start programming. This helps me organize the code better and be focused."* – IP13). However, we discovered that this is not common: most participants do not use much Markdown or programming comments during the experimentation phase.

Some participants reported to use Markdown to think about and organize their analysis.

### Scripts and notebooks for code refinement

Our participants performed two broad tasks when refining source code: (a) refactor and migrate code, which might require the programmer to prune the scratchpad code; and (b) add documentation and narratives. Scripts were the preferred modality for (a). Scripts are supported in IDEs like PyCharm and Visual Studio, which offer powerful code refactoring functionalities, e.g., [Filippov, 2015]. Computational notebooks,

Scripts were preferred over notebooks for refactoring code.

on the other hand, are mostly used in web browsers, as this facilitates collaboration and sharing, and offer less powerful features for code refactoring.

For larger data science projects, script files can offer a more manageable structure.

Larger notebooks have been known to get unstructured and unmanageable [Kery et al., 2018]. Using several scripts instead, where each script acts as a "black box" code package could be a simpler alternative in production. Despite these limitations, we will next discuss how notebooks can be indispensable for data workers to present their work.

### Scripts and notebooks for dissemination and storage

Scripts are predominantly used for storage.

Most participants (P1-4, P7-10, P12, P20, and P21) use scripts for dissemination and storage, since scripts have better support for use in the production pipeline. For example, IP8 and IP12 work on projects with a large code base that is already organized as multiple scripts; it is easier for them to add new source code as scripts. Other participants (P8, P15, P16, and P17) reported to use scripts for dissemination and storage due to several features, such as execution from the command shell, better support for file stream access, and possibility of automation (*"Compute cluster accepts only scripts. IPythons can be run as well, but it is frowned upon [by those who maintain the compute cluster] because it blocks computing resources."* – IP17).

Notebooks offer a convenient interface for communicating results to others.

Some participants (P5, P11, P13, P16, and P17) reported to use notebooks mainly for dissemination, e.g., to stakeholders, colleagues, or blog readers (*"I like to make my analysis into a report that one [readers, colleagues, etc.] can just read through and execute cell-by-cell to understand."* – IP16). However, none of our participants reported using computational notebooks exclusively.

Scratchpad notebooks, which contain data workers' initial experiments, can support further analyses.

Many data workers kickstart their analysis by basing it on existing code [Guo, 2012]. Some participants (P8, P13, P14, P15, and P17) reported using scratchpads notebooks, i.e., notebooks used for typically unstructured, exploratory work, for this purpose. These notebooks were reported to help get reacquainted with API usage and programming language syntax. Some participants (P8, P14, and P15) considered these scratchpad note-

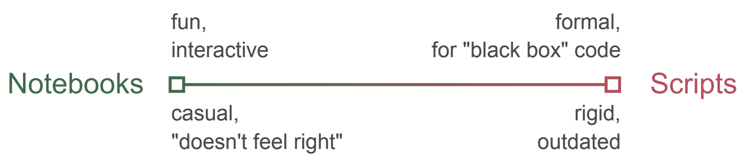


**Figure 5.8:** Terms our participants used to refer to computational notebooks and scripts. Notebooks were considered to be interactive and fun, but too casual, whereas scripts were considered formal but too rigid.

books to be even more important than the source code files that contain the production code, as it helps them reason about the execution output better.

### Data workers' perceptions of notebooks and scripts

Participants had both negative and positive connotations about scripts and notebooks.

Throughout our study sessions, participants associated scripts with terms like *"black box," "formal,"* and *"reliable,"*, but also *"rigid"* and *"outdated"* as shown in Figure 5.8. Notebooks, on the other hand, were associated with *"interactive"* and *"fun,"* but also *"too casual"* and *"doesn't feel right"*. These connotations about scripts and notebooks came from discussions about how these modalities were used during different phases of data science programming.

The ability to decouple results from code can be beneficial after experimentation.

Scripts, unlike computational notebooks, allow data workers to view source code output without having to step-through the source code cell-by-cell. This high code-result distance or the ability to decouple the results away from the code can be desirable if the data worker just wants to obtain the results without having to step through the code, one chunk at a time. Conversely, the higher-level of interactivity offered by notebooks via low code-result distance and cell-by-cell execution acts as an inconvenience when reusing code:

> *"I have tried writing the final version of my code blocks in Python notebooks, but it just doesn't feel right. . . Once something is a black box, it should be-*

*long in scripts. I don't want to run it in a notebook anymore because it runs through line-by-line and I don't want that. It doesn't feel clean."* - IP17

There is *tension* between the use of scripts and notebooks.

Also, support for computational notebooks in production pipeline is still a work in progress. E.g., for IP17, a data worker who builds neural networks, a standard practice in his field is to outsource computationally-intensive executions to external GPUs. These GPUs support scripts in the standard *.py file format, and not Python notebooks, making it difficult for IP17 to adopt computational notebooks. There is thus a *tension* between the two main modalities, computational notebooks and scripts (Figure 5.8). The interactivity of computational notebooks is great for experimentation, but scripts' are a more reliable medium for reuse and storage.

**Prevalence of computational notebooks and scripts**

We conducted an online survey to understand usage statistics of scripts and notebooks.

Since a significant number of our participants did not report using notebooks, we wanted to extrapolate the usage statistics of notebooks and scripts to a larger sample. To do this, we conducted an online survey with 62 data workers from various domains like machine learning and significance testing. Respondents were recruited via word of mouth and social media. Respondents self-reported an average expertise of 3.51 (1 = novice, 5 = expert). Combining the survey results with data from our interview participants ($n = 21$), we were able to gather modality usage statistics for 83 data workers as shown in Figure 5.9.

Although scripts are most common, notebooks are rising in popularity. Most notebook users use scripts alongside notebooks.

Scripts are the most commonly used programming modality across R, Python, and MATLAB. Only five respondents use notebooks exclusively, all for Python programming. Conversely, 40 respondents use scripts exclusively. Most respondents ($n = 41$) reported to use *both* computational notebooks and scripts. Computational notebooks are more popular among Python programmers (36 out of 53 respondents; 67.9%) than MATLAB (3 out of 11 respondents; 27.3%) and R programmers (7 out of 25 respondents; 28%). This indicates that Python notebooks are more popular than the MATLAB Live Editor and

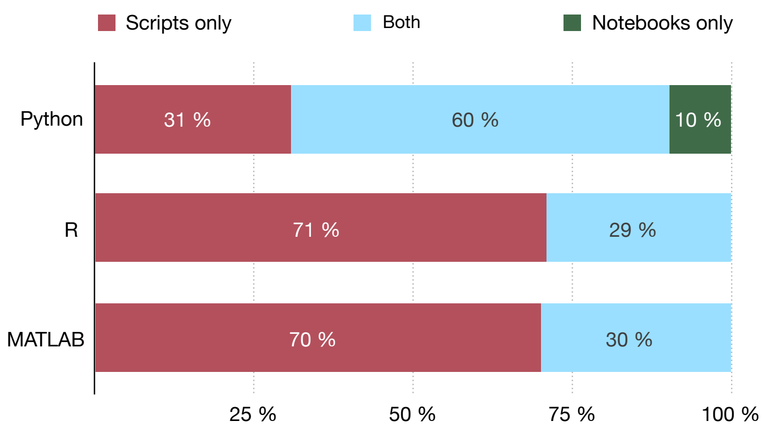


**Figure 5.9:** Results of our online survey where respondents chose the programming modalities they use for data science programming. Scripts are the most commonly used modality across all programming languages, with many respondents using a combination of scripts and computational notebooks.

RMarkdown notebooks, but there may be other explanations. For example, notebooks could be more popular in the machine learning community, in which Python is frequently used.

To investigate why scripts are more common than computational notebooks, we turned to our interview transcripts. Twelve participants (P1–4, P6, P7, P9, P11, P12, and P18–21) who do not use computational notebooks regularly had reasons that varied from not being aware of notebooks to finding notebooks unnecessary for their work. Some reported that they had tried to use notebooks earlier, but did not gain a substantial benefit. There is thus no clear monopoly of modalities; although scripts are still the most common modality in data science, many data workers use notebooks often in combination with scripts.

Some participants find notebooks unnecessary for their work.

### 5.2.5 Limitations and Future Work

We based our findings mainly on the interview and walkthrough sessions. While we methodically coded and analyzed the observation videos, we did not do a detailed analysis of the

Among other limitations, we used the observations only as a supplement to interview transcripts.

| **Data science task** | **Notebooks** | **Scripts** |
|---|---|---|
| Experimentation | ✓ | – |
| Refactor code | – | ✓ |
| Large data science project | – | ✓ |
| Present code | ✓ | – |
| Share code | ✓ | – |
| Execute from command line or GPU | – | ✓ |
| Store code | – | ✓ |
| Re-run past code | – | ✓ |

**Table 5.2:** Trade-offs between notebooks and scripts

videos. The videos were instead used to confirm and extend the insights we had developed from analyzing interview transcripts and walkthrough sessions. In some cases, videos helped develop open questions that were then clarified by analyzing the interview transcripts. To ascertain data workers' long term behavior and to understand how data workers' workflows change across various project requirements, longitudinal studies need to be done. We discuss the implications of our findings and provide design recommendations in Section 6.3.

### 5.2.6 Summary: Key Findings

In summary, we conclude the following from our study understanding how data workers use programming modalities:

- Scripts and computational notebooks are both part of the data worker's toolbox. Although scripts are still the most common interface, many data workers, especially in fields like machine learning, use notebooks alongside scripts.

- Scripts are preferred during production as the ability to use them as 'black box' with code refactoring features makes them desirable.

- Notebooks are preferred during experimentation as they allow for easier comparison of alternative explorations and notebooks' cell-based programming approach better suits the non-linear, iterative nature of hypothesis testing.
- Although consoles are used mainly for secondary tasks, such as testing a library API, their ability to juxtapose results with source code and quick prototyping by pulling up previous commands make them appealing to novice data workers.
- We summarize our participants' preferences for modalities in Table 5.2.4. ✓ indicates the preference of most of our participants.

## 5.3 Source Code Analysis: Quantifying the Quality of Data Science Source Code

https://osf.io/85nmv/?view_only=e13feadffa4b49c0a6543b084838db78 [a]

[a]Contains the results of our code corpus analysis along with URLs to the OSF projects we sampled from.

From the discussion above, and from previous research, we now know that source code in data science tasks is of a poor quality. While the quality of source code tends to increase after the confirmation phase, we were curious how high this can get, especially because academic data workers in certain fields like HCI typically do not share their analysis code in publications [Wacharamanotham et al., 2020]. To quantify the quality of source code after confirmation phase, we examined analysis scripts written in R by academic data workers. Since these were analysis scripts after the confirmation phase, we consider the results to provide an upper threshold of analysis code quality; we believe that the quality of experimental code would be lower.

We examined analysis scripts written by data workers to quantify the quality of data science source code.

We collected 40 R analysis scripts, used in HCI and psychology publications at top-tier conferences such as CHI '13 and ISWC

Our code corpus of 40 scripts was sampled from reputed sources.

’16, from two sources. We collected 32 scripts from 23 projects on the Open Science Framework[5] (OSF) platform We sorted the projects by their date of creation, and looked for R scripts provided in the project. Beyond selecting scripts written in R, we did not filter our sample in any other manner. In a similar manner, we collected eight scripts from three researchers at our local university. We randomly selected no more than three scripts from the same project or researcher to avoid biasing our analysis to a particular data worker. We analyzed a total of 20,303 source lines of code (SLOC), with an average of 508 SLOC per file. We identified two key issues, which we discuss below.

### 5.3.1 Excessive Code Cloning

Excessive code cloning can lead to several issues.

From our previous discussions, we know that code cloning is a prevalent practice. Excessive code cloning can make analysis scripts difficult to maintain, understand and modify [Koschke, 2008; Mayrand et al., 1996; Monden et al., 2002], limiting replication and code reuse for writing production code and future analyses. We were interested in identifying instances of parameterized code cloning, in which the parameters of one or more lines of code are cloned as shown in Figure 5.10.

Using a standard procedure, we found that 70% of our code corpus are clones.

We followed a standard procedure [Rattan et al., 2013] to identify instances of code cloning. First, we did a preprocessing of our analysis scripts to remove empty lines and comments; this helped identify the Lines of Interest (LOI). Then we used code fingerprints, short representative strings used to uniquely identify code [Johnson, 1993], as both the intermediate representation of the code and the match detection technique to compute instances of code cloning. We identified that out of 4098 LOI from all scripts, 2861 LOI, or 70%, were parametric clones [Rattan et al., 2013].

[5] `https://osf.io/`

```
# 2. perform bootstrap
wis_case <- bootnet(wis_network,nBoots=2500,type='case')
tbs_case <- bootnet(tbs_network,nBoots=2500,type='case')
nimh_case <- bootnet(nimh_network,nBoots=2500,type='case')

# 3. calculate the CS coefficient
wis_cs <- corStability(wis_case)
tbs_cs <- corStability(tbs_case)
nimh_cs <- corStability(nimh_case)

# 3. if the CS coefficient is high enough, run
# another bootnet and interpret its bootnet plot
wis_boot <- bootnet(wis_network,nBoots=2500)
tbs_boot <- bootnet(tbs_network,nBoots=2500)
nimh_boot <- bootnet(nimh_network,nBoots=2500)

plot(wis_boot)
plot(tbs_boot)
plot(nimh_boot)
```

**Figure 5.10:** Example of parameterized code clones from our corpus. The analyst clones a chunk of code and modifies the parameters to suit the current task in analysis.

### 5.3.2 Prevalence of Non-Modular Code and Functions

Modular code, characterized through the use of functions, is generally considered to lead to a more productive programming [Hughes, 1989]. Non-modular code, on the other hand, is inefficient and does not match the hypothesis testing workflow, since a typical analysis would involve iterative executions of a module with different input arguments. This can be particularly problematic when analyzing datasets with several variables, since modules would be reused several times.

Modular code matches the iterative nature of hypothesis testing.

To analyze how modular source code is, we first classified code snippets in a script into the three major steps of hypothesis testing: *preprocessing*, *exploratory analysis*, and *confirmatory analysis* [Tukey, 1977]. We manually created a mapping between the R function calls in our script files and the steps these function calls belonged to. We ignored functions outside of these categories such as `source()` and `print()`. Then we identified how many of these steps were part of each script in our code corpus. We found that 97.5% of all scripts contained two or more steps, and 60% contained all three steps. This issue was also found to be prevalent in longer analysis scripts.

We manually identified how many statistical steps were included in each script.

We found that about two in three scripts did not use functions.

Additionally, we found that 27 out of our corpus of 40 files (67.5%) did not include any user defined function, which further validates the prevalence of non-reusable code. This behavior has previously been noticed by R experts, who recommend use more, smaller functions [FitzJohn and Falster, 2013; Mächler, 2014].

### 5.3.3 Discussion: Implications for Data Science Programming

Excessive cloning can hinder code reuse, an important part of analysis.

While code cloning can be beneficial in certain use cases, e.g., when the programming language does not support inheritance [Kapser and Godfrey, 2008], excessive code cloning has been associated with several issues. It results in analysis scripts that are difficult to understand and modify [Mayrand et al., 1996; Koschke, 2008], as well as maintain [Mayrand et al., 1996; Monden et al., 2002]. Difficulty in understanding and modifying source code could hinder code reuse.

Current IDEs might not support hypothesis testing workflow.

We believe that current IDEs for R themselves do not support the repetitive and non-sequential workflow of hypothesis testing. An R script stores and executes the code in a sequential manner. While the analyst can manually select and execute a linear subset of the script, such manual execution of R code snippets can lead to misinterpretations when the data are outdated or execution dependencies are not selected. We continue this discussion in the next chapter.

We therefore conclude that programming using R results in code that has excessive code cloning, which makes the analysis process hard to understand, maintain, and replicate, and contains non-modular code, and does not support the repetitive and non-sequential workflow of hypothesis testing. We posit that this is a design problem, and that there is a need for environments that better support the non-linear, iterative execution of data science source code.

# Chapter 6

# Supporting Data Workers in Performing Data Science

**PUBLICATIONS AND AUTHOR'S CONTRIBUTIONS**
The work in this chapter has contributions from Johannes Maas and Michael Ellers. Part of the research in this chapter has been published as an extended abstract [Subramanian et al., 2018] and two papers, [Subramanian et al., 2020b] and [Subramanian et al., 2020a]. The author of this thesis helped design both artifacts, StatWire and Tractus. The author also evaluated both artifacts. Michael Ellers developed the first version of StatWire [Ellers, 2017], which was improved by Johannes Maas [Maas, 2017], discussed in Section 6.1. Johannes Maas developed Tractus, discussed in Section 6.2. Prof. Wacharamanotham helped evaluate StatWire.

In the previous chapter, we discussed data workers' workflows during data science programming. We identified three specific problems:

We identified three problems in the previous chapter, which we aim to address in this chapter.

1. In Section 5.3, we found that data workers write poor quality source code that is hard to maintain: 70% of source code in data analysis are clones, and source code files are typically large and non-modular.
2. A prominent reason why data workers write poor quality source code is exploratory programming practice, during which data workers experiment with ideas in source

code. As we discussed in Section 5.1, this makes it difficult for data workers to recall their experiments when writing production code.

3. In Section 5.2, we found how scripts and notebooks play critical roles across various phases of data science programming. By having to use them both, data workers have to switch contexts and often copy-paste code, resulting in inefficient workflows.

The problems we solve are not limited to data science.

The problem of excessive cloning is not specific to data science programming. Clones are very common in software projects [Baker, 1995; Kapser and Godfrey, 2003], and lead to increased maintenance costs [Fowler, 1997; Geiger et al., 2006]. The two other problems originate due to the exploratory programming practice, which is also prevalent beyond data science programming. We now discuss some approaches to address these issues.

## 6.1 Encouraging Data Workers To Write Modular, Maintainable Source Code

https://osf.io/79rux/?view_only=c9b6ae0421034bffb43b19cce92cbbc9 [a]

[a]Contains a demonstration video of StatWire and datasets used in evaluation.

https://github.com/i10/StatWire [a]

[a]Contains the source code for StatWire.

We identified three reasons why data science source code contains excessive clones and is non-modular.

From our content analysis in the previous chapter, we determined that source code written for data analysis contains numerous clones and is non-modular. We believe that these problems arise due to a combination of a) iterative, non-linear workflow of data analysis; b) exploratory programming practice, in which the programmer does not invest resources in writing high-quality code when experimenting; and c) lack of incentives for sharing code as a part of publications. We have discussed b) at length in Sections 2.5.1 and 5.1.3. We elaborate on a) and c) below.

The design of data science programming interfaces does not match the data science programming workflow.

Programming interfaces for data analysis predominantly support linear storage and execution of source code. Most programming interfaces support selective execution, but even such execution is linear. This, however, does not match the non-linear and iterative nature of data analysis, and can result in inefficient usage scenarios, such as multiple executions and using incorrect variable values, which we discussed in Section 5.2. As we discussed earlier, computational notebooks and interactive consoles can better support non-linear, iterative workflows due to the high level of interactivity they provide. Despite this, such interfaces have limited support for writing and maintaining production code. There is a potential for redesigning interfaces used for data science tasks so that they may better support non-linear, iterative workflows.

There is little incentive to share source code used to perform data analysis.

Several fields of research such as psychology [Open Science Collaboration, 2015], HCI [Wacharamanotham et al., 2020], and medicine [Begley and Ioannidis, 2015] have reported that it is uncommon for researchers to share research artifacts, which includes the source code used for analyzing experimental data, in order to facilitate replications. A key reason seems to be lack of incentives for sharing such artifacts [Feger et al., 2019; Wacharamanotham et al., 2020]. Since data workers do not have strong incentives for sharing their source code, they do not have to spend time and effort modifying their source code to refactor and add documentation.

We introduce *StatWire,* a hybrid programming interface to encourage data workers to write modular source code.

How can we encourage data workers to write better source code? As an exploratory solution, we introduce *StatWire,* with which we explored a data-driven, hybrid programming approach to encourage data workers to write modular source code. StatWire integrates a traditional text editor with a visual data-flow editor. After discussing the idea behind StatWire in Section 6.1.1, we describe how StatWire works using a short interaction walkthrough in Section 6.1.3, and discuss its intended benefits through a small-scale study with data workers in Sections 6.1.5–6.1.6.

### 6.1.1 Idea: Hybrid Programming to Encourage Modular Programming

Text-based programming allows users to express logic concretely, and visual programming allows users to reuse code and get an overview of analysis.

We aimed to create an interface that encourages programmers to write source code in a modular fashion. Text-based programming is powerful, and allows programmers to express business logic in a concrete manner. It does not, however, allow easy reuse of code, as functions are less common in data science programming due to the iterative workflow. As it stands, reuse is possible only via copy-paste programming. On the other hand, visual programming environments organizes programmers' source code into visual blocks, which can be readily reused. Further, such environments abstract the details of the underlying business logic, allowing programmers to focus on higher-level tasks. We were interested in exploring a combination of text and visual programming environments to get the best of both worlds. We designed and developed StatWire with this idea in mind.

### 6.1.2 Related Work: Hybrid Statistical Tools

In many tools, users can use visual editors to generate textual code.

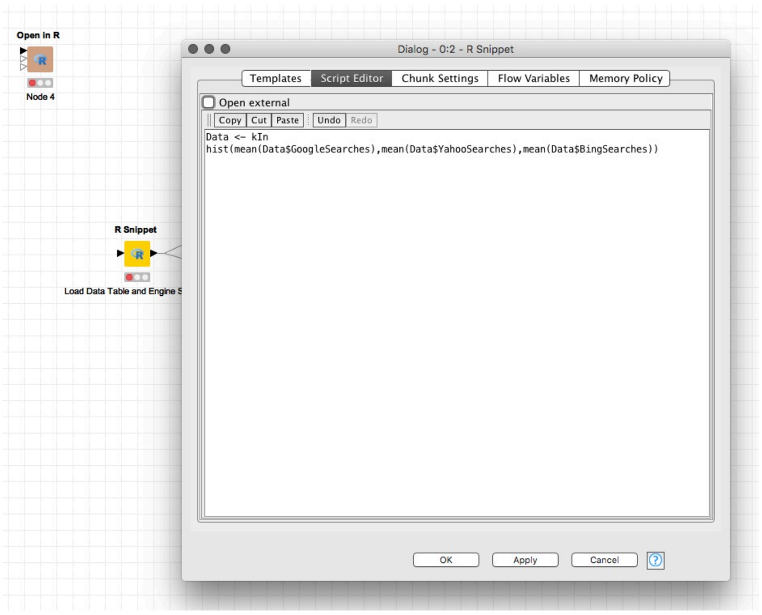
Several tools have tried to combine text-based programming with visual representations. Rehearsal World [Finzer and Gould, 1993], Aura 2 [Dannenberg, 2004], and more recently BrickLayer [Cheung et al., 2009] let the user work with a visual editor in order to generate the underlying source code. Heterogenous languages allow programming by letting users intermingle text and visual notations [Erwig and Meyer, 1995].

ViSta has both visual and text editors, but the text editor cannot be used to author new functionalities.

For hypothesis testing, ViSta [Young and Bann, 1996] is a visual statistical system that allows users to work with both a visual interface and a text-based interface simultaneously. However, the text-based interface is used only to use predefined modules, not to author new modules from scratch. The text-based interface only acts as a textual interface for invoking the predefined statistical modules, not defining them.

RapidMiner and KNIME are primarily visual interfaces, with limited support for text integration.

Both RapidMiner [Hofmann and Klinkenberg, 2013] and KNIME [Berthold et al., 2009] offer visual interfaces to construct workflows for predictive modeling, data mining, hypothesis testing, etc. Users can drag-and-drop existing widgets into a



```
Data <- kIn
hist(mean(Data$GoogleSearches),mean(Data$YahooSearches),mean(Data$BingSearches))
```

**Figure 6.1:** Tools such as RapidMiner and KNIME (shown above) separate the text-based editor from the visual data flow editor.

blank canvas, specify the data flow via click-and-drag, and run the workflow to view the results. Extensions to RapidMiner and KNIME allow integration of R code snippets and plots. The user can add an R code snippet module and can view and edit the underlying source code using a pop-up dialog box as shown in Figure 6.1. However, the interfaces of both these tools separate the text-based editor from the visual data flow editor. Editing the textual code has no effect on the visual data flow editor, and to add an argument to the textual code, the user needs to use the visual editor. Such tools are more suited to functions that are already written, tested, and debugged in a different environment e.g., in a text-based IDE. Writing analysis code from scratch using such interfaces is difficult because of a lack of support for the debugging process.

### 6.1.3 Interaction Walkthrough

StatWire consists of two panels as shown in Figure 6.2: visual data flow editor (Figure 6.2A) and a textual programming environment (Figure 6.2B). Each node in the visual editor can be either a *statlet,* which is a step in analysis (Figure 6.2F), or a *viewlet,* which shows plots (Figure 6.2E1) or data (Figure 6.2E2). The edges (Figure 6.2H) represent the flow of data across the nodes.

The two interfaces seamlessly combine to help the user perform data science programming.

The visual editor is initially empty, and the user can create a statlet by right-clicking on the empty canvas. When a statlet is created, the text editor (Figure 6.2C) is shown with a default function template. This encourages the user to think in a modular fashion and use functions. When authoring code, users can use the output pane next to the code (Figure 6.2D) for debugging the source code. The two editors allow the user to operate at two levels of abstraction: the visual editor provides an overview of the entire analysis process including the flow of data, whereas the text editor allows the user to focus on the business logic of each step. In addition to receiving input from other statlets, statlets can also take a file as an input via a file browser.

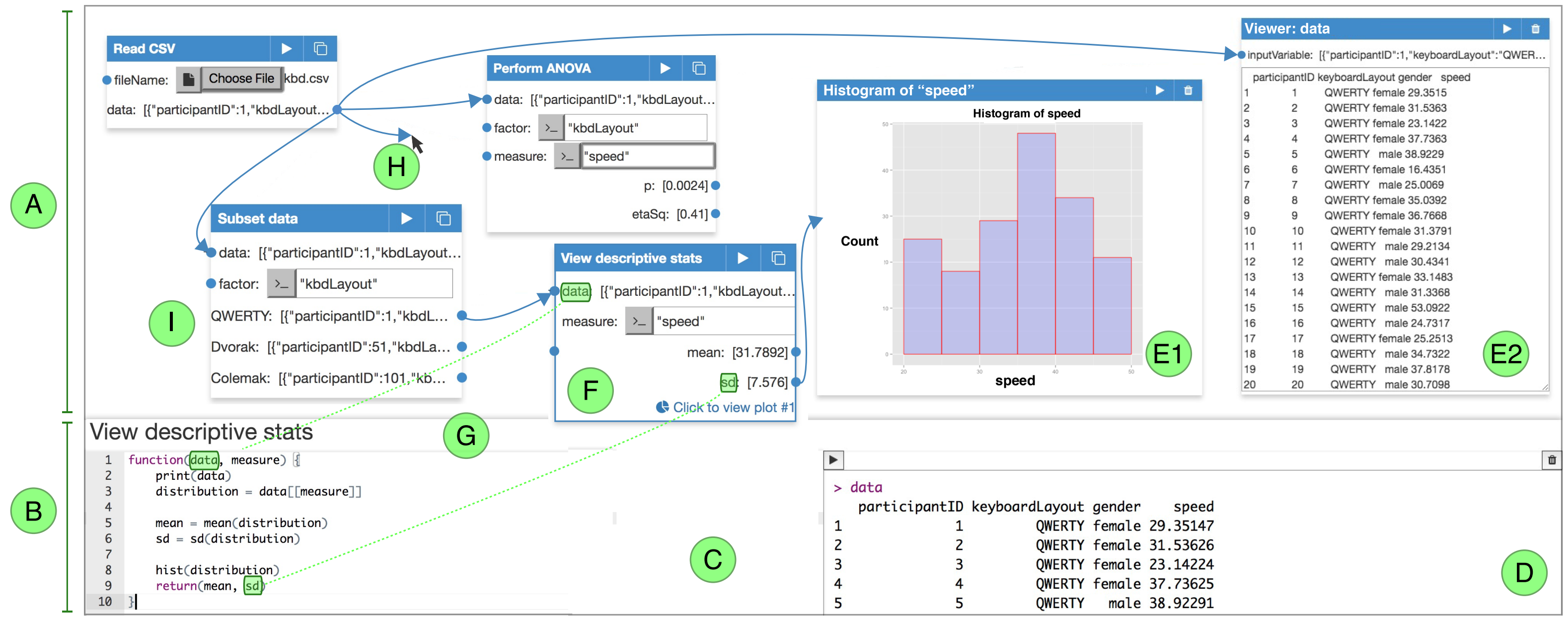
The two interfaces are tightly integrated. When one interface is altered, the other interface changes to reflect this.

StatWire tightly couples the two editors. When input and output arguments are added to the function header in the text editor, the corresponding node in the visual editor is automatically updated (Figure 6.2G). The statlet displays the output values in the visual editor. If the user wants to view an entire data table or a plot, they can utilize a viewlet, which uses a dedicated window. Viewlets take takes only one input and no output.

### 6.1.4 Design Process

We used a user-centered, iterative process to design and develop StatWire.

We developed StatWire by following a user-centered, iterative design process, during which we explored several design alternatives to support a tight integration between the two programming environments. In earlier prototypes, e.g., users could add arguments via the visual programming environment. However, evaluations with existing users of R and one statis-



Read CSV
fileName: Choose File kbd.csv
data: [{"participantID":1,"kbdLayout…

Perform ANOVA
data: [{"participantID":1,"kbdLayout…
factor: "kbdLayout"
measure: "speed"
p: [0.0024]
etaSq: [0.41]

Subset data
data: [{"participantID":1,"kbdLayout…
factor: "kbdLayout"
QWERTY: [{"participantID":1,"kbdL…
Dvorak: [{"participantID":51,"kbdLa…
Colemak: [{"participantID":101,"kb…

View descriptive stats
data [{"participantID":1,"kbdLayout…
measure: "speed"
mean: [31.7892]
sd [7.576]

Viewer: data
inputVariable: [{"participantID":1,"keyboardLayout":"QWER…

```
   participantID keyboardLayout gender  speed
1        1     QWERTY female 29.3515
2        2     QWERTY female 31.5363
3        3     QWERTY female 23.1422
4        4     QWERTY female 37.7363
5        5     QWERTY   male 38.9229
6        6     QWERTY female 16.4351
7        7     QWERTY   male 25.0069
8        8     QWERTY female 35.0392
9        9     QWERTY female 36.7668
10      10     QWERTY female 31.3791
11      11     QWERTY   male 29.2134
12      12     QWERTY   male 30.4341
13      13     QWERTY female 33.1483
14      14     QWERTY   male 31.3368
15      15     QWERTY   male 53.0922
16      16     QWERTY   male 24.7317
17      17     QWERTY female 25.2513
18      18     QWERTY   male 34.7322
19      19     QWERTY   male 37.8178
20      20     QWERTY   male 30.7098
```

View descriptive stats

```
function(data, measure) {
    print(data)
    distribution = data[[measure]]

    mean = mean(distribution)
    sd = sd(distribution)

    hist(distribution)
    return(mean, sd)
}
```

```
> data
  participantID keyboardLayout gender    speed
1             1         QWERTY female 29.35147
2             2         QWERTY female 31.53626
3             3         QWERTY female 23.14224
4             4         QWERTY female 37.73625
5             5         QWERTY   male 38.92291
```

**Figure 6.2:** StatWire is an IDE for R, the most common statistical programming language. It seamlessly integrates a visual data flow editor (A) with a text-based editor (B). A qualitative study shows that StatWire helps users structure their code better and reduces disorientation during programming.

tics expert who teaches introductory courses in R revealed that users preferred to update the arguments in the textual programming environment. Users wanted the visual editor to be used exclusively for the higher-level task of managing the overall statistical workflow. To address this issue, we incorporated live updates to the visual editor upon changes to the textual editor.

### 6.1.5 Evaluation

We expect StatWire to improve code quality and code comprehension.

The primary intended benefit of StatWire is to improve the quality of source code written for data science tasks. Specifically, we expected StatWire to increase the use of functions and reduce cloning. Additionally, since the visual editor provides an overview of the analysis process, we wanted to understand if using StatWire led to increase in comprehension of the analysis workflow.

We compared StatWire to RStudio and two visual programming tools in a user study with four data workers.

To evaluate whether StatWire can provide these benefits, we conducted a study to identify how well StatWire supports statistical programming with R. In this study, we compared StatWire to RStudio[1], a widely used IDE for R, as well as RapidMiner[2] and KNIME[3], two visual programming environments with R integration. We identified these tools based on an informal survey of R-based IDEs and programming environments. We did not consider environments that expose a frontend for R programming language, since our aim is to understand changes in programming behavior.

We recruited four data workers (one female) who use RStudio from our research lab. Three participants had taken a graduate-level introductory analysis course in R. All participants had used RStudio before, but none had used KNIME or RapidMiner. All participants were experienced in performing hypothesis testing—all reported to have at least three years of experience.

Participants used the programming tools we provided to analyze hypothetical datasets.

We first collected some background information from each

[1]https://www.rstudio.com/
[2]https://rapidminer.com
[3]https://knime.com

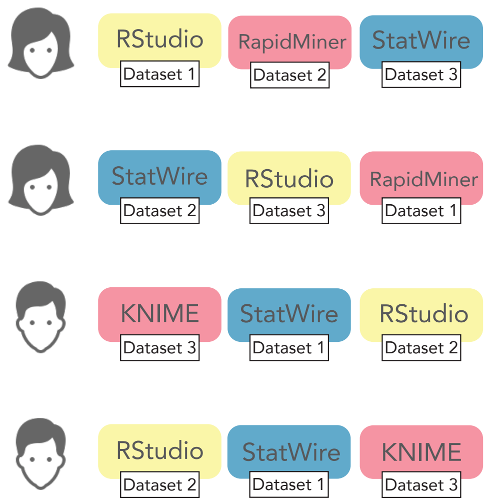


**Figure 6.3:** The experimental design of our study to evaluate StatWire. We balanced the order of conditions and randomized data-tool pairing.

participant: Their experience with statistical analysis, experience with R programming, which other analysis tools they use, and experience with functional programming. Each participant performed analysis with each analysis tool for a minimum of 30 minutes.Participants analyzed a random representative dataset. Datasets were designed by the author of this work, and included hypothetical data from HCI experiments. For example, the first participant evaluated data from a mobile text entry study that compared the typing speed for three different handwriting-recognition input techniques, namely Unistroke [Goldberg and Richardson, 1993], EdgeWrite [Wobbrock et al., 2003], and Graffiti [MacKenzie and Zhang, 1997]. They were given a list of predetermined research questions to focus their analysis on. Additional datasets with research questions were provided when participants managed to finish the analysis before the stipulated minimum duration of 30 minutes. The second participant began analyzing an additional dataset, but could not complete the analysis.

Participants analyzed data from hypothetical HCI experiments.

We balanced the order of conditions and randomized both the dataset-tool pairing and the characteristics of datasets to minimize learning effects (Figure 6.3). Beyond standard hypothesis tests, the analysis required the use of typical but non-trivial techniques such as factor encoding, data transformation, and post-hoc tests.

We employed thematic analysis to analyze experimental data.

We collected the screen, audio, and webcam feed of the sessions. To analyze the experimental data, we employed an informal, reflexive thematic analysis [Braun and Clarke, 2021], in which we sought to identify patterns of user behavior. After partially transcribing the audio recording, a solo analyst assigned codes to them, and then generated initial themes. These candidate themes were then evaluated over time by analyzing additional data. We used the expected benefits of StatWire to guide our analysis. Below we present some of the key findings from our analysis.

### 6.1.6 Significant Findings

#### Prevalence of bad quality code

When using RStudio, our participants wrote non-modular code with excessive clones.

We first wanted to establish the prevalence of inefficient programming practice with existing analysis tools. When using RStudio, none of the participants followed a modular programming approach. There were several instances of code cloning, and participants used comments to structure their analysis source code. Despite this, because users were familiar with RStudio, they reported to feel at ease when using it.

#### Existing visual programming environments lack tight integration with text-based programming

Our participants encountered several problems with using text-based editor when using RapidMiner and KNIME.

RapidMiner and KNIME are two popular tools that primarily expose a visual programming environment, but also has support for editing the underlying source code. With both RapidMiner and KNIME, participants found the lack of integration between visual data flow editor and textual environment frustrating. For example, while authoring textual code, the visual data flow editor did not update automatically. It was also not possible to view both environments simultaneously. Over time, participants were able to work around these shortcomings.

#### Evidence of StatWire's benefits

We identified evidence for benefits of StatWire.

With StatWire, users benefited from the tight integration between the two programming environments. Two users reported to understand the analysis better as a result of the visual data flow editor. When conducting analysis with StatWire, over half of the statlets (18 out of 35; 51%) were reused, whereas no reuse was observed with other tools.

StatWire is open-source, and available as a local web application from the StatWire project home page[4].

### 6.1.7 Limitations and Future Work

Our study shows that hybrid programming approaches can be promising, but more work is needed.

While our preliminary study is promising, it is limited by sample size and guiding hypotheses, and further longitudinal studies are required to understand StatWire's effects on code understanding, structuring, and navigation, as well as on the productivity of the analysis. Further, the artifact can be extended to allow transformation of existing analysis scripts into a more structured and reusable format by, e.g., automatically highlighting similar chunks of code and semi-automatically converting them to a reusable module.

## 6.2 Supporting Exploratory Programming Workflow

https://osf.io/pe9a2/?view_only=d201aeb945b747e49328ec86bc58e68b[a]

[a]Contains the results of parser evaluation, datasets used in evaluation, and source code for a previous iteration of Tractus.

https://github.com/i10/tractus[a]

[a]Contains the source code for the latest version of Tractus.

[4]https://hci.rwth-aachen.de/statwire

Our observational study in Section 5.1 helped us understand data workers' workflows when performing hypothesis testing. We found that there are two phases in analysis: an *experimentation phase,* during which the data worker experiments in source code with their ideas, and then a *confirmation phase,* during which the data worker rewrites code and writes statistical report. A key finding is that data workers have difficulty in recalling the key insights and rationale for their analysis from the experimentation phase.

*Tractus* helps data workers track source code and insights through source code visualization.

We now introduce a potential solution to this issue: *Tractus.* Tractus is an addin for the RStudio IDE. It helps data workers track source code that had yielded insights during exploration, and understand their source code explorations better when writing production code and reports later. We envision Tractus to reduce code re-runs, help data workers manage explorations, and rationalize their analysis choices in reports. We first discuss related work in Section 6.2.1, discuss how Tractus works and how we implemented it in Sections 6.2.2–6.2.2, and describe how we evaluated Tractus in Section 6.2.3.

## 6.2.1 Related Work

Tractus is not the first tool to help users understand and find analysis code. We discuss some prominent tools that do this, and discuss prominent works about source code visualization that informed the design of Tractus.

### Tool support for understanding and finding analysis code

Previous tools help by visualizing data science activities and execution output; maintaining alternatives in code; and helping find source code.

Previous research has produced several artifacts to help data workers understand and find prior code. Burrito [Guo and Seltzer, 2012] captures and displays source code outputs, timeline of activities, and notes from a data science project to help data workers capture their data science workflow. Variolite [Kery et al., 2017] is a lightweight version-controlling system that helps data workers maintain code alternatives and track outputs. Code Gathering Tools (CGT) [Head et al., 2019] is an interactive extension to Python notebooks that can help data workers find, clean, and manage code, by providing mechanisms to locate source code. Verdant [Kery et al., 2019] is also a notebook plugin that visualizes code history to help programmers find prior code.

Unlike these tools, Tractus tracks the data worker's experiments by visually grouping them into the corresponding *hypotheses,* presents this structured visualization to help the user stay oriented, and, during confirmation phase, helps locate the source code that yielded particular insights.

### Visualizations of source code and history

Source code visualizations has several use cases and benefits.

We consulted several existing works on source code visualization to inform the design of Tractus. Systems like Code Bubbles [Bragdon et al., 2010b], Code Thumbnails [DeLine et al., 2006], and Stacksplorer [Krämer et al., 2010] visualize code to help improve comprehension and navigation. Programming IDEs employ other forms of visualization like icons and graphical overlays next to the code to encode information like syntax highlighting, code conventions, and version control information [Sulír et al., 2018].

Past works help track data sources and source code history.

In data science, an important task is tracking the sources of data, i.e., data provenance. Provenance Explorer is a tool that supports this task by visualizing data and events associated with it as a graph [Cheung and Hunter, 2006]. Prior research artifacts, e.g., [Kery et al., 2019; Wittenhagen et al., 2016; Yoon et al., 2013], visualize source code *history* to improve code comprehension and foraging.

Unlike Verdant [Kery et al., 2019], which is the closest to our work, Tractus works at a more abstract level by grouping source code into hypothesis, captures execution dependencies, and contextual information from source code that data workers use to recall rationale and insights.

## 6.2.2 Interaction Design

Tractus consists of three components as shown in Figure 6.4:

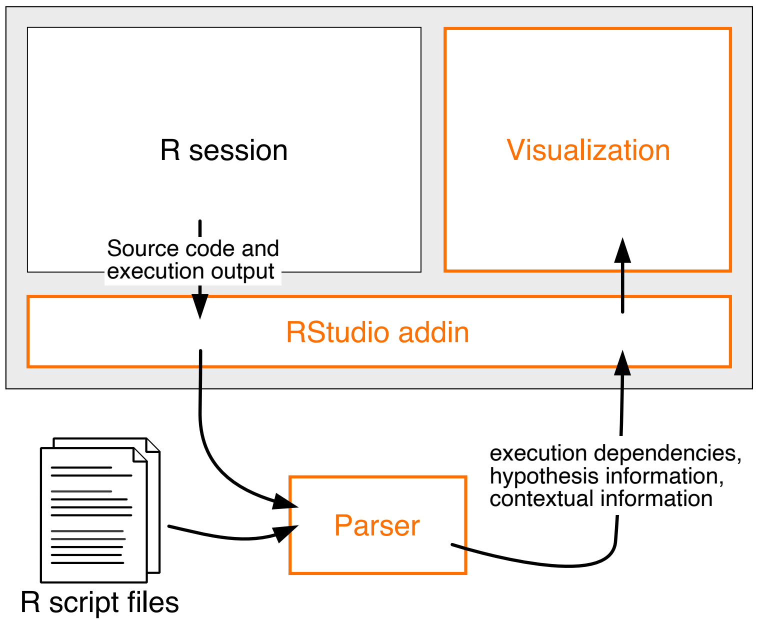

R session
Visualization
Source code and execution output
RStudio addin
execution dependencies, hypothesis information, contextual information
Parser
R script files

**Figure 6.4:** Tractus consists of three components: RStudio addin, parser, and the visualization. The *RStudio addin* feeds the R code and execution output from the R session to the parser. The *parser* breaks down the code, detects the hypothesis that the code belongs to, and finds execution dependencies in code. This information is then visualized.

Tractus' parser is responsible for identifying the information that is visualized in Tractus.

1. The *parser* is the back-end of Tractus. It breaks down R source code and obtains (a) the *hypothesis* investigated by each statement in the analysis, (b) the execution *dependencies* among variables in source code, and (c) *contextual information* in source code such as the block and tail comments. Our RStudio addin is responsible for capturing the comments, execution output, and the order of execution of statements, and sending these to the parser. Hypotheses are the atomic building blocks of analysis and reflects the data worker's mental model. We believe that leveraging this information can help data workers find source code more easily. Execution dependencies are captured to help minimize incorrect and faulty execution of production code. Finally, contextual information helps data workers rationalize their analysis and rewrite source code after exploration.
2. The *visualization,* which acts as the front-end of Tractus

as shown in Figure 6.5. The visualization receives information about how the source code should be grouped according to the hypothesis they belong to, execution dependencies, and contextual information from the parser, and visualizes them in real-time. The visualization monitors the parser output for changes, and updates the visualization on change. In the visualization, the source code is organized into blocks to improve navigation, and variables used in the analysis are emphasized to help track provenance of data. Furthermore, since a common task during analysis is to copy-paste code and modify arguments, the visualization also supports *data injection*. Data injection allows data workers to select a block of code, and modify the dependent and independent variables in it. The visualization can be shown in the RStudio viewer pane or in a native web browser. Since RStudio's viewer pane does not support certain features like autocomplete or copy-to-clipboard, data workers might prefer using web browsers.

Tractus uses a tree-based visualization to group source code into hypothesis, along with contextual information.

3. The *RStudio addin* integrates the parser and web app into the R session. The addin watches the R session for new source code executions, captures them, and feeds them to the parser along with execution results. Only statements that successfully execute are sent to the parser. The addin is also responsible for displaying the web app, i.e., the visualization front-end of Tractus, in the viewer pane of RStudio.

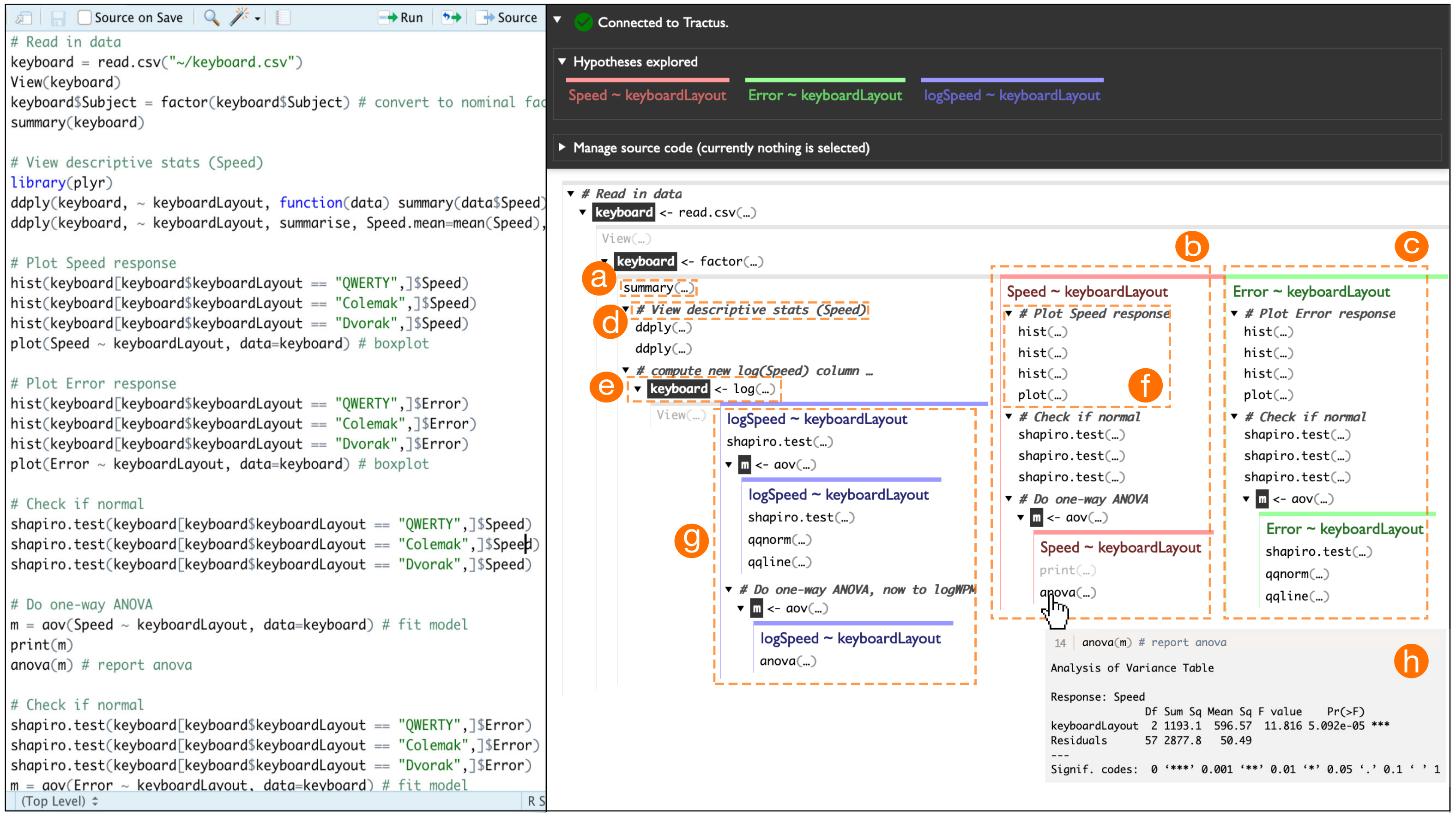
The addin integrates the parser and visualization.

We designed Tractus in an iterative manner, at every stage gathering feedback from R users and data workers. After using low-fidelity sketches to test the design of Tractus' visualization, we built two high-fidelity implementations as shown in Figure 6.6 and Figure 6.5 to evaluate the user interaction with the visualization. We now describe the parser and visualization.

### Parser

The parser is the back-end of Tractus. It is responsible for detecting information from the source code that is visualized by Tractus' front-end. The parser is agnostic to where the R code is written by the data worker—it can read R source code from

The parser is the back-end of Tractus and is agnostic to the source of R source code.



```
# Read in data
keyboard = read.csv("~/keyboard.csv")
View(keyboard)
keyboard$Subject = factor(keyboard$Subject) # convert to nominal fac
summary(keyboard)

# View descriptive stats (Speed)
library(plyr)
ddply(keyboard, ~ keyboardLayout, function(data) summary(data$Speed)
ddply(keyboard, ~ keyboardLayout, summarise, Speed.mean=mean(Speed),

# Plot Speed response
hist(keyboard[keyboard$keyboardLayout == "QWERTY",]$Speed)
hist(keyboard[keyboard$keyboardLayout == "Colemak",]$Speed)
hist(keyboard[keyboard$keyboardLayout == "Dvorak",]$Speed)
plot(Speed ~ keyboardLayout, data=keyboard) # boxplot

# Plot Error response
hist(keyboard[keyboard$keyboardLayout == "QWERTY",]$Error)
hist(keyboard[keyboard$keyboardLayout == "Colemak",]$Error)
hist(keyboard[keyboard$keyboardLayout == "Dvorak",]$Error)
plot(Error ~ keyboardLayout, data=keyboard) # boxplot

# Check if normal
shapiro.test(keyboard[keyboard$keyboardLayout == "QWERTY",]$Speed)
shapiro.test(keyboard[keyboard$keyboardLayout == "Colemak",]$Speed)
shapiro.test(keyboard[keyboard$keyboardLayout == "Dvorak",]$Speed)

# Do one-way ANOVA
m = aov(Speed ~ keyboardLayout, data=keyboard) # fit model
print(m)
anova(m) # report anova

# Check if normal
shapiro.test(keyboard[keyboard$keyboardLayout == "QWERTY",]$Error)
shapiro.test(keyboard[keyboard$keyboardLayout == "Colemak",]$Error)
shapiro.test(keyboard[keyboard$keyboardLayout == "Dvorak",]$Error)
m = aov(Error ~ keyboardLayout, data=keyboard) # fit model
```

```
▼ # Read in data
  ▼ keyboard <- read.csv(…)
      View(…)
    ▼ keyboard <- factor(…)
        summary(…)
      ▼ # View descriptive stats (Speed)
          ddply(…)
          ddply(…)
      ▼ # compute new log(Speed) column …
        ▼ keyboard <- log(…)
            View(…)
            logSpeed ~ keyboardLayout
            shapiro.test(…)
            ▼ m <- aov(…)
                logSpeed ~ keyboardLayout
                shapiro.test(…)
                qqnorm(…)
                qqline(…)
            ▼ # Do one-way ANOVA, now to logWPM
              ▼ m <- aov(…)
                  logSpeed ~ keyboardLayout
                  anova(…)

Speed ~ keyboardLayout
▼ # Plot Speed response
    hist(…)
    hist(…)
    hist(…)
    plot(…)
▼ # Check if normal
    shapiro.test(…)
    shapiro.test(…)
    shapiro.test(…)
▼ # Do one-way ANOVA
  ▼ m <- aov(…)
      Speed ~ keyboardLayout
      print(…)
      anova(…)

Error ~ keyboardLayout
▼ # Plot Error response
    hist(…)
    hist(…)
    hist(…)
    plot(…)
▼ # Check if normal
    shapiro.test(…)
    shapiro.test(…)
    shapiro.test(…)
  ▼ m <- aov(…)
      Error ~ keyboardLayout
      shapiro.test(…)
      qqnorm(…)
      qqline(…)
```

```
14 | anova(m) # report anova
Analysis of Variance Table

Response: Speed
               Df Sum Sq Mean Sq F value    Pr(>F)
keyboardLayout  2 1193.1  596.57  11.816 5.092e-05 ***
Residuals      57 2877.8   50.49
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

**Figure 6.5:** Tractus is an algorithmic and visualization extension to RStudio that can support data science workflows. Tractus detects, captures, and visualizes: (1) source code experiments grouped as hypotheses, e.g., Figure 6.5c, (2) dependencies across source code, which are visualized as an indented branch in the tree, e.g., Figure 6.5g shows code that is dependent on the log-transformed dataset `keyboard` (Figure 6.5e), and (3) based on our formative study, information that data workers use to recall rationale and insights such as *block* comments (Figure 6.5d) and execution output (Figure 6.5h). Code sections corresponding to hypotheses that have the same execution dependency are placed next to each other to facilitate comparison, e.g., Figure 6.5b and Figure 6.5c.

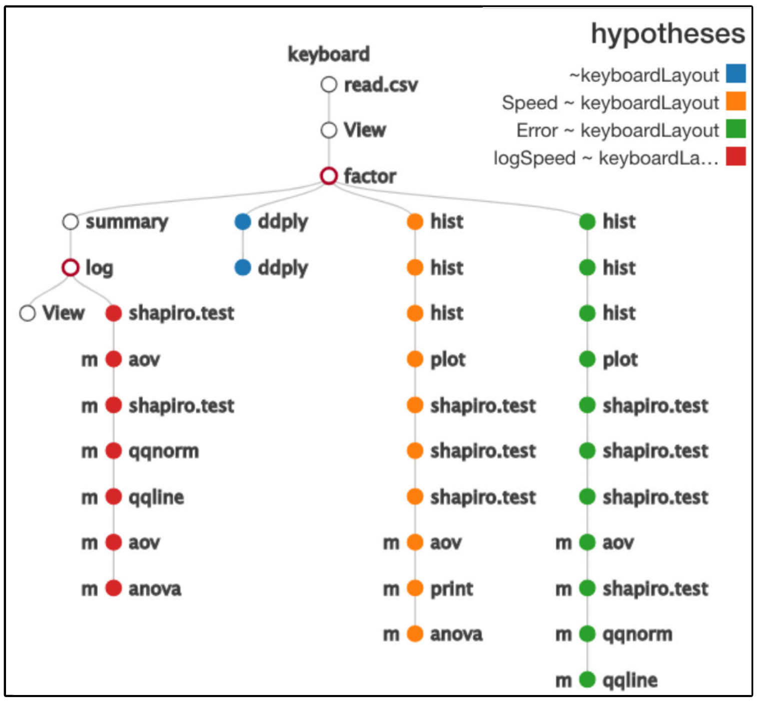


**Figure 6.6:** The first version of Tractus. After evaluating this version with users, we made several design improvements, e.g., symmetrical tree required horizontal scrolling for large files, and improved the underlying architecture in the current version.

an R script file, an R session's history database file, or even through the RStudio addin. Tracking the R session's history allows Tractus to capture code experiments performed in interactive consoles, a common practice among novice data workers. We validated the parser's ability to process R source code by using it to parse a test dataset of existing R scripts; we discuss the validation results later in Section 6.2.2.

**Detecting components of a statement:** The parser deconstructs the given R source code into an Abstract Syntax Tree (AST) representation [Baxter et al., 1998]. The AST representation breaks down each statement into variable, expression, function name, and arguments (name and value). The parser filters out statements that do not have to be visualized, such as package installations, statements that do not execute successfully, and control structures like loops and conditions. Unlike

The parser breaks down each statement into an AST representation to obtain the individual components, such as variable, function name, and arguments.

existing parsers, our custom parser captures comments, both inline and block, and line feeds so that it can detect code blocks.

**Detecting execution dependencies:** Earlier works have already successfully detected execution dependencies in source code, e.g., [Weiser, 1981; Higo and Kusumoto, 2009]. Our parser detects execution dependencies by keeping track of variables and statements. A statement that uses a variable depends on the statement that had defined or modified that variable. Statements that use multiple variables depend on multiple statements; conversely, a variable can be depended upon by multiple statements. Our parser ignores dependencies in control structures, e.g., dependencies from statements that are inside an if-block to those outside the if-block. Our parser validation revealed only a few instances of this, since hypothesis testing typically has a linear, but branching, control flow. Therefore, we concluded that capturing and displaying control structures into the dependency graph would not have yielded significant benefits.

Tractus tracks all variables and statements to determine execution dependencies for each statement.

Alternative explorations are easy to spot in Tractus.

In addition to helping users understand their explorations better, revealing the execution dependencies also helps capture the alternative explorations that result from data modifications. Alternate explorations use different data and are tracked in our visualization more easily (Figure 6.5g).

Tractus uses a DAG to capture execution dependencies.

The parser first keeps track of the variables resulting from the AST representation and then uses this information to cumulatively detect execution dependencies in the code. These dependencies are captured by the parser as a labeled Directed Acyclic Graph (DAG), in which each node is a statement and each directed edge is labeled with the variable name that establishes the dependency between the connected statements. Traversing this graph results in all statements required to execute a statement with correct values.

Tractus uses the R formula notation to detect which hypothesis is being tested in a statement.

**Detecting hypotheses:** Data workers use one or more hypotheses to investigate relationships between variables. To validate a hypothesis, there are typically one or more alternative explorations. Our parser groups source code based on the hypothesis it validates by exploiting R's formula notation, as well as data selection and data manipulation operations. As mentioned earlier, this notation is used to encode the relationship between variables, and is commonly used across several analysis steps, such as significance tests, assumption tests, and plots or graph creation.

Our parser implementation had some challenges.

In R, there are certain significance tests that allow data workers to specify hypothesis without this formula notation, e.g., the ezANOVA function in the ezANOVA package[5]. However, we found very few instances of this in our test dataset of R scripts that we used for our parser evaluation. We encountered false positives in cases where the formula notation was used in a plotting function rather than for specifying relationships between variables. An example is the `ddply` function, in which the user uses the formula notation to specify how to split the data frame, e. g. `ddply(kbd, ~ Layout, function(data) summary(data$Speed))`. In general, however, we did not discover significant mismatches between our intentions and the parser results in our parser validation. To summarize how our parser works, hypothesis detection occurs by looking at the following information:

- R's formula notations like `measurement ~ factor` and `measurement ~ factor1*factor2*factor3`. R's formula notations can be used to specify advanced factor designs.
- Dataset manipulation operations like subdivisions: `subset(data, factor == "level")$measurement`
- Dataset column selections, e.g., `data[data$factor == "level",]$measurement`

We captured the code blocks by looking for lines of code after a block comment was detected.

**Capturing code blocks:** Data workers organize their source code into code blocks and prefix the block with a comment. We call such comments *block comments*. We wanted to capture the entire block, including the block comment when present. (A block therefore includes all statements in the block as well as the leading comment.) To achieve this, whenever the parser

[5] https://www.rdocumentation.org/packages/ez/versions/3.0-1/topics/ezANOVA

encounters a line of code that is a comment, we assume that a new block is present. All comments following the first line of comment are considered to be the block's comment until the first line containing an expression is encountered. This and all subsequent expressions are linked to the same block until an empty line is encountered, upon which the block is considered to have been completely captured.

Tractus uses a tree representation to group source code based on the hypothesis being tested.

**Parser's output - hypothesis tree:** The parser uses a tree data structure to capture the hypothesis information of source code. We refer to this as the *hypothesis tree.* The parser constructs this tree by parsing the source code one statement at a time, and extracting the source code components (variable, function name, arguments, etc.) and dependencies. To represent execution dependencies in the tree, the parser adds dependent statements as a child to the statements they depend on. In situations where there are multiple parents, we pick the most recent parent in the source code to retain a tree structure. Even though a DAG would reflect this one-to-many dependency more accurately, our tree representation is simpler and resembles the source code more closely. If the statement belongs to a hypothesis, it is added under the corresponding branch in the tree. Each branch represents a hypothesis; a branch is created when a new hypothesis is first encountered in a statement. Metadata associated with the statement, such as its execution output and tail comment, are also added to the hypothesis tree.

Our parser achieves a good coverage of a test corpus of 38 R scripts.

**Parser evaluation:** We evaluated our parser using a test dataset of 38 R scripts. We randomly sampled these scripts from the Open Science Framework (OSF)[6], a platform that allows researchers to share research material, We also solicited scripts from researchers at our local university. By testing these scripts with our parser, we were eventually able to achieve a 82.4% successful coverage with these files. Four files had syntax errors and failed to execute, so we removed them from our test dataset. Of the remaining 34 files, Tractus successfully parsed and visualized 28 files. The parser failed to parse the six remaining files due to several reasons, e.g., deeply nested statements. The parser was also capable of successfully parsing large files (> 7500 LOC).

[6] `http://osf.io`

### Visualization

The RStudio addin runs a web view alongside the R source code that visualizes the hypothesis tree. Below, we describe the design of this visualization and how users can interact with it. We start by describing the layout of the app, how information is presented and organized at a higher level of abstraction, and then discuss the concrete details.

**Layout:** The visualization is located next to the user's code. At the top of the visualization, a panel provides an overview of all the hypotheses explored in source code. Clicking on a hypothesis highlights the corresponding nodes in the visualization below. The panel at the top also includes options to perform new explorations and generate source code that can reproduce results. We discuss how Tractus supports new explorations in Section 6.2.2 and generate code for reproducing results in Section 6.2.2. Below the top panel, Tractus visualizes the user's code, including execution dependencies, execution output, and contextual information, with all this information grouped into hypotheses.

Tractus' interactive visualization is shown in the right panel.

**Visualizing dependencies and hypotheses:** Tractus aims to provide an overview of the analysis, and helps data workers transition from exploration to writing production code and reports. To support this, we chose a tree visualization instead of a graph. This tree visualization is constructed as follows:

Tractus uses a tree visualization, the structure of which is determined by execution dependencies.

1. If the statement has no dependencies, it is placed under the root node.
2. If the statement has one dependency, it becomes a child of the dependent statement's node in the tree, e.g., Figure 6.5a.
3. If the statement has multiple dependencies, it becomes a child of the chronologically most recent parent.

The resulting tree visualization captures the dependencies among statements. Tractus then uses the information about each statement's hypothesis, obtained from the parser, for grouping statements. Statements that belong to a hypothesis

Branches in tree are color coded to indicate the hypothesis that is tested. Statements that belong to a block are grouped.

validation are placed under the branch corresponding to the hypothesis as shown in Figure 6.5c and color coded. Statements that do not belong to a particular hypothesis, e.g., code used for loading datasets, are thus distinguishable by color from other code. As a second level of grouping, statements that belong to the same code block are grouped, e.g., Figure 6.5f. Block comments are explicitly shown to the user as they usually describe the code block at a high level. Inline comments are shown only upon request as discussed below in Section 6.2.2. Inside each group of code that corresponds to a hypothesis validation, statements are displayed top-down in their source code order.

Tractus visualizes contextual information, such as execution output and comments.

**Visualizing contextual information:** To reduce visual clutter, Tractus uses progressive disclosure [Nielsen, 2006] of information. For variable assignment statements, Tractus displays only the variable and function names by default; additional information, such as the execution output, the statement's line number in an R code file, tail comment (if any), and the complete statement, are shown when hovering with the mouse pointer as shown in Figure 6.5h. Users can collapse and expand branches in the visualization to shift focus to specific code groups, both at a hypothesis-level and code block-level.

Tractus' visualization is designed to help data workers find information faster.

Statements that had changed a variable's value and statements that do not contribute to the business logic, e.g., `print()` and `cat()`, are displayed, but are intentionally made less noticeable. Through this, by color coding hypothesis groups, and through indentation of nodes, Tractus uses visual cues to help data workers find information faster.

### Data injection

As exploration has a standard routine, Tractus automates this via data injection.

To semi-automate the exploration routine, Tractus supports *data injection.* Data injection works as follows. The data worker selects the base code in the visualization, clicks on a button to inject data, and selects, from a list that Tractus creates by analyzing existing code, the measure and factor(s). Tractus then generates the code with new variables and copies it to the clipboard. This can immensely benefit data workers who often create new code alternatives from existing code by avoiding the need to manage data dependencies manually.

#### Result reproduction

A cornerstone of good scientific practice is to be able to publish code that reproduces consistent results. To support data workers achieve this, Tractus supports generating code that can reproduce the result of a statement. While this is not a novel feature [Weiser, 1981; Head et al., 2019], we included it in Tractus for the benefit it provides. When the user selects one or more desirable statements, Tractus uses dependency information to retrieve all statements necessary to reproduce the expected result, and displays them.

By tracking execution dependencies, Tractus can generate code that will reproduce an execution output.

#### Architecture

Tractus is built so that its components can be modified in an independent manner. This allows for easier extensions, e.g., support for more metadata, for additional visual artifacts, and other scripting languages like Python. The parser is written in Rust[7], a high performance, robust programming language, and returns a structured JSON tree that can be visualized differently if desired. The visualization is built using D3.js[8] and can be run in a web browser. The RStudio addin is written in R.

Tractus is built to support extensions.

### 6.2.3 Evaluation

We used two user studies to evaluate Tractus. In the first study, we aimed to understand how data workers use Tractus to understand R code written by others. We used the findings from this study to improve Tractus, and then conducted a second study to understand how Tractus can help data workers across various stages in analysis, i.e., exploratory and confirmatory phases.

[7]`https://www.rust-lang.org`
[8]`https://d3js.org`

**Study 1: Can Tractus help understand source code?**

In our first study, our participants used Tractus to understand and describe R scripts.

Three participants (1 female; P1–P3) tested an earlier prototype of Tractus (Figure 6.6). We recruited the participants via emails sent to university mailing list and an R users group[9]. Participants were not offered any incentives to take part in the study. Two participants self-reported as intermediate users of RStudio to perform analysis, and one identified as a novice.

Participants used Tractus to understand and explain different R scripts.

Participants were tasked with understanding and then describing three R scripts. We sampled scripts of three different sizes (small: 25 LOC; large: over 500 LOC) from OSF[10]. Participants used a local machine that had RStudio with the Tractus-addin installed on it. On the left pane, participants could see the source code from the sampled scripts, and, on the right pane, they could view and use Tractus. Participants were asked to explain the code verbally using the prompt, *"Could you explain what is going on in the code?"*. Participants were informed that they could take as much time as they want and allowed to use any resources, e.g., use a search engine to check what an R function does. The experimenter intervened very little, but answered any questions the participant had. We collected the video and took field notes. Study sessions lasted 40 minutes on average.

We used thematic analysis to analyze experimental data.

**Analysis:** We analyzed the video and field notes using informal, reflexive thematic analysis [Braun and Clarke, 2021]. We partially transcribed the audio from the session, and immersed ourselves with the screen recording and field notes. We assigned codes to the transcript, developed potential themes, and evaluated these themes based on data from screen recording, field notes, and subsequent data from transcripts. Due to limited number of users, many themes could not be evaluated with confidence. The resulting findings discussed below should be considered preliminary.

Tractus has potential to help data workers understand source code in scripts, especially in large script files.

**Key findings:** All participants commented that Tractus helped them understand source code better than when navigating code without Tractus. This effect is pronounced for larger analyses that validate several hypotheses. P1 suggested better ways

[9]https://www.meetup.com/de-DE/koelnrug/
[10]https://osf.io

to group information in the visualization. P2 liked the hover-interaction, and mentioned that the visualization helped them easily spot which statistical model was used for each hypothesis.

The results also motivated several design improvements. The symmetrical tree structure in this version of Tractus required horizontal scrolling and was hard for the study participants to navigate, even for files that were only moderately long. Participants also mentioned that the visualization had too many details that added to the visual clutter. We fixed these issues, and improved the underlying architecture of Tractus to make extensions to Tractus easier.

Following a user-centered, iterative design process, we used the feedback from our participants to improve Tractus.

**Study 2: Can Tractus improve data science workflow?**

We conducted a second study that validated Tractus across experimentation and production phases. Seven data workers (3 female, median age = 29, P1–P7) took part in the study. They were recruited through mailing lists and social media. Participants were compensated with a 20 € gift card. P1, P5, and P6 self-identified as beginners, P2, P4, and P7 as intermediates, and P3 as an expert R analyst.

Our second study sought to validate how Tractus is used across experimentation and production stages of data science.

To establish a status quo of our participants' workflows, we asked them to first use RStudio without our Tractus addin before they used RStudio with Tractus. Before the session with Tractus, participants were given an overview of Tractus interface. Participants were given datasets[11] to analyze. These datasets contain adequate measurements and factors, and many potential hypotheses could be validated. To maintain ecological validity, participants were asked to first perform exploratory analysis to *generate hypotheses* by themselves, and then perform confirmatory analyses. Based on their findings, participants were asked to write a report of their analysis. After the analysis, participants had the opportunity to provide open-ended feedback about Tractus. All participants analyzed at least two datasets, and sessions were 100 minutes long on average.

In the study, participants generated hypothesis, validated them, and wrote a report with the aid of Tractus.

[11]Source: `https://github.com/fivethirtyeight/data`; see supplements for dataset details.

Participants were encouraged to think aloud during the study.

Throughout the session, participants were encouraged to think aloud. The experimenter prompted the participants little, and allowed sufficient time for participants to do an open-ended exploration of the dataset. In situations where participants had difficulty using the interface, the experimenter intervened to help. The participant was not guided in the analysis process, since this would equally affect their workflow across both conditions.

### Analysis

We partially transcribed the audio recordings, and applied selective reflexive thematic analysis [Braun and Clarke, 2021]. As with study 1, we relied on field notes, screen recordings, and artifacts we collected from the study to evaluate potential themes. The screen recordings were selectively analyzed using the ELAN annotation tool[12].

### Findings

Tractus' visualization proved effective in helping participants track variables created earlier.

*Execution dependencies:* Tractus' visualization of execution dependencies received positive reviews from participants. P1, P3, P4, and P6 reported that the visualization of execution dependencies was useful during the initial experimentation phase. The visualization proved to be particularly effective in helping participants track variables that were created a while ago. P3 compared the visualization to the Environment pane in RStudio, which is one approach used by participants to track variables when using RStudio without Tractus, mentioning that the ability to understand the origins of a variable was useful (*"[The execution dependency graph] reminds of the Environment pane, [but] it is just better as it [also] shows where [a] variable came from."* – P3). In this situation, the participant had not named the variable appropriately, but the dependency graph helped him infer the context (in this case, the variable was the result of a subset function).

Tractus encouraged our participants to remove scratchpad code and rename variables.

*Code curation and code quality:* One unintended side effect of Tractus was that it encouraged participants to *curate* their code. After performing exploratory analysis, P2 and P3 used the visualization to remove scratchpad code from their script so that the visualization would become *less messy.* For example, P2

[12] `https://archive.mpi.nl/tla/elan`

found that there were several nodes in the visualization that represented his explorations to fix a bug; since this did not contribute towards the analysis, he wanted to delete these lines of code. P2 also mentioned that he would not have removed these lines of code when using RStudio without Tractus, indicating that the visualization improves *awareness* of source code. In contrast to removing source code, three participants (P3, P4, and P6) used the visualization to improve the quality of their R code, e.g., by renaming variables.

*Exploration states:* Since the visualization groups code according to hypotheses, it helped participants notice *patterns* across analyses. Several participants (P1, P3, P4, and P6) were able to compare the states of hypotheses to understand similarities and differences (*"[Using Tractus, it is] easier to compare analyses side by side to say 'yeah, it's the same' or find [out] what is different."* – P1). This also proved to be useful when writing reports later, since participants could easily detect differences between explorations.

Tractus allowed participants to compare states of their explorations, an important task that helps progress of analysis process.

*Orientation and navigation:* Tractus can help data workers be more oriented during analysis. For example, when analyzing their data, P2 wanted to test several hypotheses. They selected one and tested it, but while doing so, they identified another hypothesis and set off on a different analysis path. When this did not lead to promising results, P2 used Tractus to *backtrack* to the initial hypothesis to continue the analysis. The benefits of Tractus do not cease after analysis. P4 mentioned that the visualization was useful to kick-start new analyses, since the visualization captures the analysis procedure more succinctly and is more easily understandable than source code.

Tractus allowed participants backtrack to previous code and be more oriented in their analysis.

*Design improvements:* We also identified several areas of improvement based on this study. Three participants (P1, P4, and P5) found the visual notation, especially execution dependencies, hard to understand initially. We redesigned the visualization to reduce clutter by reducing the information shown and by making some changes to the layout. Overall, participants were mostly positive about Tractus and looked forward to using it. During all ten sessions, Tractus was able to detect the hypotheses accurately except for two instances. In both these instances, the participant specified the hypothesis in an unexpected manner, e.g.:

Our study motivated several improvements to the design of Tractus.

```
read.csv("~/data.csv")$measure ~
read.csv("~/data.csv")$factor
```

While this is valid, it is uncommon and our parser failed to detect the hypothesis. (The parser is programmed to only expect variables in a formula notation.)

### 6.2.4 Discussion

#### Towards reproducible, transparent data science

By capturing and producing source code that produces execution output, Tractus promotes reproducible data science.

For future analysts to be able to replicate an analysis, executing the analysis source code should reproduce the expected results. Tractus makes this possible by using the execution dependencies to capture source code. Tractus can also be extended to work with R packages like `reprex`[13], which provide out-of-the-box functionalities to track and capture all dependencies in source code.

Tractus' visualization can be readily shared in reports to promote transparency.

The visualization in Tractus provides a nice overview of the analysis, including all the alternative paths that were explored during the course of the analysis. This can be readily shared in research papers or as supplements to promote transparency of analysis. Tractus could be extended to capture Markdown[14] from R Notebooks, allowing integration of narratives in the visualization.

#### Increased awareness of forking paths

Tractus' visualization keeps a record of all paths taken during an analysis, even the ones that do not end up getting reported in publications or reports.

HARKing or the forking paths problem is a prominent issue with statistical practice. Since Tractus makes these paths visible to the data worker, it can improve data worker's awareness of their forks in hypothesis testing. This could be an antidote to over-testing, and help data workers be more oriented and structured in their analysis. Additionally, Tractus can be extended to track all significance tests the data worker conducts,

[13]`https://github.com/tidyverse/reprex`
[14]`https://daringfireball.net/projects/markdown/`

warn against over-testing, and automatically apply $p$-value corrections.

### 6.2.5 Limitations and Future Work

Tractus' parser can be extended to certain programming constructs and other programming languages.

Tractus could parse most (82.4%) of the R scripts we tested it with, as well as the scripts from our user studies. However, the parser still has certain limitations. Complex structures like deeply nested statements are supported, but slow down the parser significantly. Tail comments that occur before a statement or expression is complete, e.g., `for(i in 1:n) #Comment`, are not supported. More details about limitations are available in the Appendix. Other programming languages used for hypothesis-driven data science, like Python, have syntax similar to R that can be leveraged to detect hypotheses. For example, Python uses the following syntax for selecting data: `variable = data[data['factor'] == level]['measure']`.

Tractus can be extended to other data science tasks that also use syntactic signals.

Tractus could be extended to support other data science tasks, as long as the task involves explicit notations for explorations, similar to the formula notation for hypothesis testing. Tasks that do not fit this criteria, e.g., machine learning, would require a different method to detect explorations. Further, since this depends on syntactic signals in source code, implementations will be programming language-specific.

As with most of our previous studies, most of our study participants are from academia, which limits the external validity of our findings.

## 6.3 Addressing the Dual Use of Scripts and Notebooks

In the previous chapter, we discussed how scripts and notebooks cater to various phases in data science programming, and discussed some problems data workers face by having a need for both modalities. Source code often gets cloned within or across files requiring data workers to maintain links between clones to propagate changes, e.g., by linked editing [Hartmann et al., 2008b].

### Reproducibility

As cluttered scripts can affect code navigation, data workers tend to sacrifice reproducibility of data science. Tracking dependencies in notebooks can be particularly problematic.

We found many data workers record only the source code that produces the results they will use. Participants are aware of the importance of conducting reproducible science, but do not want to clutter their scripts as this might affect code navigation. Further hidden dependencies are prevalent in computational notebooks. Some of our study participants reported encountering notebooks that do not execute because they had not copied all the dependencies when migrating code. Potential solutions include dependency managers, e.g., Drake[15], and tools that help find code snippets, e.g., [Head et al., 2019] and [Kery et al., 2019].

### Design recommendations for notebooks

Despite their share of issues, notebooks are gaining traction among data workers, and researchers are constantly working to improve their interaction design, e.g., with [Kery et al., 2019; Head et al., 2019] and [Rule et al., 2018]. It is also reasonable to expect computational notebooks to have better support for use in production pipeline in future.

We discuss some ideas to extend notebooks in order to reduce the need for scripts.

Here are some ways to redesign notebooks to bridge the difference to scripts: a) *lock* code cells so that they are immune to further changes once the data worker finishes experimentation, b) *merge* code cells after they reach a level of maturity to help avoid executing small code cells one at a time, and c) allow users to *switch* between experimentation and refinement modes; in experimentation mode, cell-based programming is active, but in the refinement mode, cells of flattened into scripts—users can switch between the modes as desired. However, given how prevalent scripts currently are, these are long-term solutions.

[15]https://github.com/ropensci/drake

# Chapter 7

# Conclusion

In this thesis, we discussed data workers' workflows when preparing for and performing data science tasks. We focussed on data workers, analysts who have limited training and expertise in data science. We focussed on hypothesis testing due to its prominence in research and industry. In this chapter, we present four takeaways based on what we learned from this work in Section 7.1. We discuss how our findings apply to other data science tasks in Section 7.2, before adding some concluding remarks in Section 7.4.

## 7.1 Takeaways: What We Learned About Data Science and Data Workers

We present four takeaways aimed at researchers, developers, and educators.

We identified the following takeaways from our user studies that sought to understand data workers' workflows as well as the studies that sought to validate our artifacts. These takeaways aim to lower the barriers for hypothesis testing, and are relevant for researchers, software developers, and statistical educators.

**Data workers underestimate the time needed to prepare for and perform analysis:** We believe that a lack of practical training and negative connotations towards hypothesis testing contribute towards data workers' just-in-time learning practice.

Just-in-time learning leads to satisficing in procedure selection.

Such a practice has several consequences. An importance consequence is that data workers underestimate the time needed to prepare for and perform analysis. This leads to satisficing when selecting statistical procedures (although there are other factors, which we will get to shortly), and an inability to address deep issues in analysis method and experimental design.

Rewarding data workers for preregistration can limit problems.

The research community needs to reward data workers for their preregistration practices, e.g., by giving them a badge or special recognition. Although preregistration may not always be possible, we need to encourage data workers to select procedures early, e.g., after a pilot test. This can motivate data workers to determine statistical procedures in advance, which can limit satisficing behavior and allow them to address problems in their analysis method and experimental design.

Lack of clear mental models contributes towards stagnation of existing procedures.

**Established statistical procedures continue to stagnate:** We find that there is a general reluctance to adopting new statistical methodologies among data workers. This is not limited to novices, although experienced data workers are more inclined to confidently adopt new techniques. We posit that lack of clear mental models about hypothesis testing and knowledge of alternative methods, such as Bayesian inference and estimation statistics, contribute to this.

We need to train data workers to be more critical about analysis techniques, and teach alternative methods during formal education.

These are problems that cannot be solved in short-term and requires statistical educators to a) train data workers to be more critical of methods and b) teach alternative methods. Recent works across communities, e.g., [Dragicevic et al., 2019; Kay et al., 2016; Cumming, 2014], have began to spread awareness among the research community. From our analysis of CHI 2019 papers, we find that 8% of papers that use quantitative methods use these alternative methods, such as Bayesian inference and estimation statistics. One can expect that over time, adoption rates would increase due to the significant benefits of these alternative methods.

Using messy experimental code to write production code or kickstart a new analysis can be difficult.

**There is a high price for working with experimental code:** Experimentation is essential in data science. Since data workers write messy code, there is a lot of effort needed to work with experimental code later when writing production code and reports. We find that some data workers consider the experimental code to be even more important than the final production code, as they use experimental code to kickstart new analysis.

We need to keep the cost of experimentation low, but lower the effort of working with experimental code later. We outlined a few ideas in our work. A hybrid programming interface can encourage the use of reusable modules, and the visual programming interface can facilitate better navigation post-experimentation. Interactive source code visualizations like Tractus and Verdant [Kery et al., 2019] capture and present information that can help situate experimental code. Additionally, structuring source code to match data workers' mental model like we did in Tractus (by grouping code into the hypothesis that it validated) can reduce the effort of working with experimental code.

We outlined some solutions to lower the effort of working with experimental code.

**Diverse tools may address diverse needs but contribute to data workers' problems:** The duality in programming interfaces with the need for both scripts and computational notebooks causes problems. We have many specialized data science tools, e.g., based on the type of data it analyzes, target users, and stage of the data science task at which it is used, and can expect to have more such tools as data science becomes more widespread. Although these tools work well for specific use cases, data workers may need to work with many of these tools. This leads to the variety problem that Biehler discussed in work [Biehler, 1997], with no cohesion among existing tools.

The variety problem is still prevalent in data science programming.

Before new tools and interfaces are developed for analysis, researchers and developers need to be carefully consider its place in data workers' current workflows. Which task does it help with? How can the user graduate to other steps? Is the transition well integrated? Answering such questions can help reduce the cost of using multiple powerful but incohesive tools. We outlined some ideas to improve notebooks to bridge the gap between notebooks and scripts, but the underlying idea of tool unification extends beyond notebooks and scripts.

New tools and interfaces for analysis need to consider how it would integrate into data workers' current workflows.

## 7.2 Generalizability of Findings

Our studies were focussed on understanding data workers' workflows with hypothesis testing. In our research, we briefly explored related data science tasks, such as neural networks, principle component analysis, and so on. We learn that there are several similarities among these tasks that allow us to be hopeful of the external validity of our work. For example:

Many of our insights generalizes to other data science tasks.

- Usage of information sources for procedure selection has few variations among tasks.
- Explicitly visualizing the data science process to mimic the analyst's mental model, as we did with Tractus, can benefit other data science tasks. For example, Patel et al.'s work with *Gestalt,* a tool to support machine learning process, applies this technique to machine learning [Patel et al., 2010].
- Simulation tools like StatPlayground that allow users to learn concepts by discovery can help learn other data science tasks. For example, *CNN Explainer* is a tool that uses interactive visualizations to help users learn Convolutional Neural Networks through active learning [Wang et al., 2021].

This thesis can serve as a first step into understanding workflows of other data science tasks.

However, there are also noticeable differences that limit generalization to other tasks. For example, our approach with StatPlayground is limited to hypothesis testing; and from our interviews, we find that data workers who perform machine learning tasks have lesser reluctance to adopt new techniques—indeed, a lot of contributions in machine learning research are new techniques, e.g., [Ueda, 2000]. Nevertheless, we hope that our work serves as a useful first step in lowering the barriers to other data science tasks.

## 7.3 Do Significance Tests Have a Place in Future?

Significance testing plays a central role in this thesis. With criticisms against the use of significance tests, there is a question mark over their place in future. Will this work be obsolete in ten years? Are we encouraging the use of significance tests despite their shortcomings?

Our work extends beyond significance tests, as we discussed in the previous section. Additionally, we took steps toward adopting alternative methods in our artifacts: StatPlayground encourages estimation statistics even though it reports *p*-values; Tractus can capture any data models, e.g., Bayesian, by capturing syntactical signals in source code; and StatWire's hybrid programming approach extends to all data-centric tasks, e.g., audio signal processing and video editing. We believe that significance tests will remain relevant in future albeit with modifications, such as bootstrapping and increased preference of estimation statistics over *p*-values.

The findings from this thesis are not limited to significance tests.

## 7.4 Concluding Remarks

Through interviews, observations, surveys, and content analyses, we presented data workers' workflows across various tasks in hypothesis testing: selecting statistical procedures, learning practical statistics, using programming IDEs to experiment with ideas in code, refactoring code, and writing production code for storage and dissemination. Based on our findings, we presented design recommendations as well as three artifacts: *StatPlayground,* which helps data workers learn practical statistics; *StatWire,* which encourages data workers to write modular code thereby lowering the cost of working with experimental code; and *Tractus,* which supports the hypothesis testing workflow by exposing contextual information about experimentation. We hope that this thesis sheds light on current workflows in hypothesis testing, and that our artifacts help improve these workflows.

# Own Publications

## Papers

Chat Wacharamanotham, Krishna Subramanian, Sarah Theres Völkel, and Jan Borchers. Statsplorer: Guiding Novices in Statistical Analysis. *In CHI '15: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems,* pages 2693–2702, New York, NY, USA, April 2015.

Acceptance rate: 25%

Krishna Subramanian, Jeanine Bonot, Radu A. Coanda, and Jan Borchers. StatPlayground: A Sandbox for Learning Practical Statistics. *In INTERACT '19: Human-Computer Interaction – INTERACT 2019,* pages 156–165, Springer International Publishing, Cham, September 2019.

Acceptance rate: 29%

Krishna Subramanian, Johannes Maas, and Jan Borchers. TRACTUS: Understanding and Supporting Source Code Experimentation in Hypothesis-Driven Data Science. *In CHI '20: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems,* pages 10, ACM, New York, NY, USA, April 2020.

Acceptance rate: 24.3%

Krishna Subramanian, Nur Hamdan, and Jan Borchers. Casual Notebooks and Rigid Scripts: Understanding Data Science Programming. *In VL/HCC '20: Proceedings of the 2020 IEEE Symposium on Visual Languages and Human-Centric Computing,* IEEE Computer Society, Los Alamitos, CA, USA, August 2020.

Acceptance rate: 30%

Krishna Subramanian, Johannes Maas, Jan Borchers, and James Hollan. From Detectables to Inspectables: Understanding Qualitative Analysis of Audiovisual Data. *In CHI '21: Proceedings of the 2021 ACM Conference on Human Factors in Computing Systems,* ACM, May 2021.

Acceptance rate: 26.3%

This paper received an *Honorable Mention Award* (top 5%)

## Late Breaking Works and Demonstrations

Krishna Subramanian. VisiStat: Visualization-driven, Interactive Statistical Analysis. *In CHI '14: Extended Abstracts on Human Factors in Computing Systems,* pages 987—992, April 2014.

Student Research Competition

Krishna Subramanian and Jan Borchers. StatPlayground: Exploring Statistics through Visualizations. *In CHI '17: Extended Abstracts of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems,* pages 401–404, ACM, New York, NY, USA, May 2017.

Interactivity Research Demo

Krishna Subramanian, Johannes Maas, Michael Ellers, Chat Wacharamanotham, Simon Voelker and Jan Borchers. StatWire: Visual Flow-based Statistical Programming. *In CHI '18: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems,* pages LBW104:1–LBW104:6, ACM, New York, NY, USA, April 2018.

Late Breaking Work

Krishna Subramanian, Ilya Zubarev, Simon Voelker and Jan Borchers. Supporting Data Workers to Perform Exploratory Programming. *In CHI '19: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems,* pages 6, ACM, New York, NY, USA, May 2019.

Late Breaking Work

## Workshop Papers

Krishna Subramanian. *In CHI '17 Workshop on Moving Transparent Statistics Forward,* ACM, New York, NY, USA, May 2017.

Krishna Subramanian. *In CHI '19 Workshop on Human-Centered Study of Data Science Work Practices,* ACM, New York, NY, USA, May 2019.

## Theses

Krishna Subramanian. VisiStat: Visualization-driven, Interactive Statistical Analysis. *Master's Thesis,* RWTH Aachen University, Aachen, March 2014.

# Bibliography

Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y. Lim, and Mohan Kankanhalli. Trends and trajectories for explainable, accountable and intelligible systems: An HCI research agenda. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 582:1–582:18, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5620-6. doi: 10.1145/3173574.3174156. URL http://doi.acm.org/10.1145/3173574.3174156.

Robert P. Abelson. *Statistics as principled argument*. Psychology Press, 2012. ISBN 0805805281.

William C. Adams. Conducting semi-structured interviews. *Handbook of Practical Program Evaluation*, pages 492–505, 2015. doi: https://doi.org/10.1002/9781119171386.ch19. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119171386.ch19.

Htrotugu Akaike. Maximum likelihood identification of Gaussian autoregressive moving average models. *Biometrika*, 60(2):255–265, 08 1973. ISSN 0006-3444. doi: 10.1093/biomet/60.2.255. URL https://doi.org/10.1093/biomet/60.2.255.

American Psychological Association. *Publication manual of the American Psychological Association*. American Psychological Association, 1994. ISBN 978-1-4338-0561-5.

Rexer Analytics. Rexer Analytics - A decade of surveying analytic professionals: 2017 survey highlights, 2017. https://www.rexeranalytics.com/data-science-survey# (last accessed on May 16, 2021).

J Scott Armstrong. Significance tests harm progress in forecasting. *International Journal of Forecasting*, 23(2):321–327, 2007. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2007.03.004. URL https://doi.org/10.1016/j.ijforecast.2007.03.004.

Timur Bachschi, Aniko Hannak, Florian Lemmerich, and Johannes Wachs. From asking to answering: Getting more involved on stack overflow. *arXiv*, pages 886–896, 2020. URL https://arxiv.org/abs/2010.04025.

Brenda S Baker. On finding duplication and near-duplication in large software systems. In *Proceedings of 2nd Working Conference on Reverse Engineering*, pages 86–95. IEEE Computer Society Press, 07 1995. ISBN 0-8186-711-43. doi: 10.1109/WCRE.1995.514697. URL http://citeseer.nj.nec.com/baker95finding.html.

Marjan Bakker and Jelte M Wicherts. The (mis) reporting of statistical results in psychology journals. *Behavior research methods*, 43(3):666–678, 2011. doi: 10.3758/s13428-011-0089-5. URL https://doi.org/10.3758/s13428-011-0089-5.

Prasanta S. Bandyopadhyay and Malcolm R. Forster. Philosophy of statistics: An introduction. In Prasanta S. Bandyopadhyay and Malcolm R. Forster, editors, *Philosophy of Statistics*, volume 7 of *Handbook of the Philosophy of Science*, pages 1–50. North-Holland, Amsterdam, 2011. doi: https://doi.org/10.1016/B978-0-444-51862-0.50001-0. URL https://www.sciencedirect.com/science/article/pii/B9780444518620500010.

Ed Baroth and Chris Hartsough. Visual programming in the real world. In Margaret M. Burnett, Adele Goldberg, and Ted G. Lewis, editors, *Visual Object-oriented Programming*, pages 21–42. Manning Publications Co., Greenwich, CT, USA, 1995. ISBN 0-13-172397-9. URL http://dl.acm.org/citation.cfm?id=213388.213393.

Ira D Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant'Anna, and Lorraine Bier. Clone detection using abstract syntax trees. *In Proceedings of the International Conference on Software Maintenance*, pages 368–377, 1998. ISSN 1063-6773. doi: 10.1109/icsm.1998.738528. URL https://doi.org/10.1109/ICSM.1998.738528.

Andrew Begel, Jan Bosch, and Margaret-Anne Storey. Social networking meets software development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder. *IEEE Software*, 30(1):52–66, 2013. doi: 10.1109/MS.2013.13. URL `https://doi.org/10.1109/MS.2013.13`.

C Glenn Begley and John PA Ioannidis. Reproducibility in science: Improving the standard for basic and preclinical research. *Circulation Research*, 116(1):116–126, 2015. doi: 10.1161/CIRCRESAHA.114.303819. URL `https://doi.org/10.1161/CIRCRESAHA.114.303819`.

Irad Ben-Gal. Outlier detection. In *Data mining and knowledge discovery handbook*, pages 131–146. Springer, 2005. ISBN 978-0-387-25465-4. doi: 10.1007/0-387-25465-X_7. URL `https://doi.org/10.1007/0-387-25465-X_7`.

Ilias Bergstrom and Alan F Blackwell. The practices of programming. *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 190–198, 2016. doi: 10.1109/vlhcc.2016.7739684. URL `https://doi.org/10.1109/VLHCC.2016.7739684`.

Jennifer M Berryman. Judgements during information seeking: A naturalistic approach to understanding the assessment of enough information. *Journal of Information Science*, 34(2):196–206, April 2008. ISSN 0165-5515. doi: 10.1177/0165551507082589. URL `https://doi.org/10.1177/0165551507082589`.

Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. Knime - the konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656280. URL `http://doi.acm.org/10.1145/1656274.1656280`.

Lonni Besançon and Pierre Dragicevic. The continued prevalence of dichotomous inferences at chi. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359719. doi: 10.1145/3290607.3310432. URL `https://doi.org/10.1145/3290607.3310432`.

Hugh Beyer and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 9780080503042.

Ruth Beyth-Marom, Fiona Fidler, and Geoff Cumming. Statistical cognition: Towards evidence-base practice in statistics and statistics education. *Statistics Education Research Journal*, 7(2):20–39, 2008. ISSN 1570-1824. URL `http://www.stat.auckland.ac.nz/~iase/serj/SERJ7(2).pdf`.

Rolf Biehler. Software for learning and for doing statistics. *International Statistical Review*, 65(2):167–189, 1997. ISSN 0306-7734. doi: 10.1111/j.1751-5823.1997.tb00399.x. URL `https://doi.org/10.1111/j.1751-5823.1997.tb00399.x`.

Ekaba Bisong. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64. Springer, 2019. ISBN 978-1-4842-4470-8. URL `https://doi.org/10.1007/978-1-4842-4470-8`.

Alan F. Blackwell. First steps in programming: a rationale for attention investment models. In *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, pages 2–10, 2002. doi: 10.1109/HCC.2002.1046334. URL `https://doi.org/10.1109/HCC.2002.1046334`.

Hennie Boeije. A purposeful approach to the constant comparative method in the analysis of qualitative interviews. *Quality and Quantity*, 36(4):391–409, Nov 2002. ISSN 1573-7845. doi: 10.1023/A:1020909529486. URL `https://doi.org/10.1023/A:1020909529486`.

Robert Bogdan and Sari Knopp Biklen. *Qualitative Research for Education: An Introduction to Theory and Methods*. Pearson, 5 edition, 2007. ISBN 9780205375561.

Jeanine Marian Bonot. Fine-level control of infoviz with foreshadowing via direct manipulation. Master's thesis, RWTH Aachen University, Aachen, February 2019. URL `https://hci.rwth-aachen.de/publications/bonot2019a.pdf`.

Ted Boren and Judith Ramey. Thinking aloud: Reconciling theory and practice. *IEEE Transactions on Professional Communication*, 43(3):261–278, 10 2000. doi: 10.1109/47.867942. URL `https://doi.org/10.1109/47.867942`.

Christine L Borgman and Jonathan Furner. Scholarly communication and bibliometrics. *Annual review of information science and technology*, 36(1):2–72, 2002. ISSN 0066-4200. doi: 10.1002/aris.1440360102. URL `https://doi.org/10.1002/aris.1440360102`.

Lutz Bornmann and Hans-Dieter Daniel. What do citation counts measure? A review of studies on citing behavior. *Journal of documentation*, 64(1):45–80, 2008. doi: 10.1108/00220410810844150. URL `https://doi.org/10.1108/00220410810844150`.

Nadia Boukhelifa, Marc-Emmanuel Perrin, Samuel Huron, and James Eagan. How data workers cope with uncertainty: A task characterisation study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3645–3656, 2017. ISBN 9781450346559. doi: 10.1145/3025453.3025738. URL `https://doi.org/10.1145/3025453.3025738`.

Nadia Boukhelifa, Anastasia Bezerianos, Ioan Cristian Trelea, Nathalie Méjean Perrot, and Evelyne Lutton. An exploratory study on visual exploration of model simulations by multiple types of experts. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–14, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300874. URL `https://doi.org/10.1145/3290605.3300874`.

Alexandre Bovet, Flaviano Morone, and Hernán A. Makse. Validation of Twitter opinion trends with national polling aggregates: Hillary Clinton vs Donald Trump. *Scientific Reports*, 8(1):1–16, Jun 2018. ISSN 2045-2322. doi: 10.1038/s41598-018-26951-y. URL `http://dx.doi.org/10.1038/s41598-018-26951-y`.

Hamparsum Bozdogan. Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, 09 1987. doi:

10.1007/BF02294361. URL `https://doi.org/10.1007/BF02294361`.

Andrew Bragdon, Robert Zeleznik, Steven P. Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola, Jr. Code bubbles: A working set-based interface for code understanding and maintenance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2503–2512, New York, NY, USA, 2010a. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753706. URL `http://doi.acm.org/10.1145/1753326.1753706`.

Andrew Bragdon, Robert Zeleznik, Steven P. Reiss, Suman Karumuri, William Cheung, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola, Jr. Code bubbles: A working set-based interface for code understanding and maintenance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2503–2512, New York, NY, USA, 2010b. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753706. URL `http://doi.acm.org/10.1145/1753326.1753706`.

Virginia Braun and Victoria Clarke. *Thematic analysis: A practical guide*. SAGE Publications Inc.; 1st edition (December 15, 2021), 2021. ISBN 1473953243.

Erik Brynjolfsson, Lorin M Hitt, and Heekyung Hellen Kim. Strength in numbers: How does data-driven decision making affect firm performance? *SSRN Electronic Journal*, 2011. doi: 10.2139/ssrn.1819486. URL `http://dx.doi.org/10.2139/ssrn.1819486`.

Kenneth P Burnham and David R Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer-Verlag New York, 2002. ISBN 978-0-387-22456-5.

Paul Cairns. HCI... not as it should be: Inferential statistics in HCI research. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 1*, BCS-HCI '07, pages 195–201, Swinton, UK, UK, 2007. British Computer Society.

ISBN 978-1-902505-94-7. URL `http://dl.acm.org/citation.cfm?id=1531294.1531321`.

Paul Cairns. *Chapter 13: Exploring, over-testing and fishing*, page 155–166. Cambridge University Press, 2019. doi: 10.1017/9781108685139.014.

Fabio Calefato, Filippo Lanubile, and Nicole Novielli. How to ask for technical help? Evidence-based guidelines for writing questions on Stack Overflow. *Information and Software Technology*, 94:186–207, Feb 2018. ISSN 0950-5849. doi: 10.1016/j.infsof.2017.10.009. URL `http://dx.doi.org/10.1016/j.infsof.2017.10.009`.

David D Carlsen and Thomas Andre. Use of a microcomputer simulation and conceptual change text to overcome student preconceptions about electric circuits. *Journal of computer-based instruction*, 19(4):105–9, 1992. ISSN 0098-597X.

Donald Owen Case. *Looking For Information: A Survey of Research on Information Seeking, Needs, and Behavior*. Emerald Group Publishing, 2016. ISBN 978-1780526546.

George Casella and Elías Moreno. Objective Bayesian variable selection. *Journal of the American Statistical Association*, 101(473):157–167, 2006. ISSN 01621459. URL `http://www.jstor.org/stable/30047446`.

Shi Kuo Chang. *Visual Languages*, volume 41. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1987. doi: 10.1007/978-1-4613-1805-7. URL `http://dx.doi.org/10.1007/978-1-4613-1805-7`.

Kathy Charmaz. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. SAGE Publications, London, United Kingdom, 2006.

Joey Cheung, Grace Ngai, Stephen Chan, and Winnie Lau. Filling the gap in programming instruction: A text-enhanced graphical programming environment for junior high students. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, pages 276–280, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-183-5. doi: 10.1145/1508865.1508968. URL `http://doi.acm.org/10.1145/1508865.1508968`.

Kwok Cheung and Jane Hunter. Provenance Explorer: Customized provenance views using semantic inferencing. In *The Semantic Web - ISWC 2006*, pages 215–227, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-49055-5. doi: 10.1007/11926078_16. URL https://doi.org/10.1007/11926078_16.

Radu-Andrei Coandă. Cheno: Computing datasets from inference statistics. Bachelor's thesis, RWTH Aachen University, Aachen, February 2019. URL https://hci.rwth-aachen.de/publications/coanda2019a.pdf.

Michael Coblenz, Andrew Ko, and Brad Myers. Jasper: An eclipse plug-in to facilitate software maintenance tasks. In *Proceedings of the 2006 OOPSLA Workshop on Eclipse Technology eXchange*, eclipse '06, pages 65–69, New York, NY, USA, 2006. ACM. ISBN 1-59593-621-1. doi: 10.1145/1188835.1188849. URL http://doi.acm.org/10.1145/1188835.1188849.

Andy Cockburn, Carl Gutwin, and Alan Dix. HARK no more: On the preregistration of CHI experiments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3173715. URL https://doi.org/10.1145/3173574.3173715.

Jacob Cohen. A power primer. *Psychological bulletin*, 112(1):155, 1992. doi: 10.1037/0033-2909.112.1.155. URL https://doi.org/10.1037/0033-2909.112.1.155.

Jacob Cohen. The earth is round ($p < 0.05$). *American Psychologist*, 49(12):997, 1994. doi: 10.1037/0003-066X.49.12.997. URL https://psycnet.apa.org/doi/10.1037/0003-066X.49.12.997.

Juliet Corbin and Anselm Strauss. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 2014. ISBN 1483315681. doi: 10.4135/9781452230153.

John W. Creswell. *Research design: Qualitative, quantitative,* *and mixed methods approaches*. SAGE Publications Inc.; 4th edition (14 Mar. 2013), 2013. ISBN 1452226105.

Geoff Cumming. The new statistics: Why and how. *Psychological Science*, 25(1):7–29, 2014. doi: 10.1177/0956797613504966. URL `https://doi.org/10.1177/0956797613504966`. PMID: 24220629.

Nancy Cunniff and Robert Taylor. *Graphical vs. Textual Representation: An Empirical Study of Novices' Program Comprehension*, page 114–131. Ablex Publishing Corp., USA, 1987. ISBN 0893914614. doi: 10.1037/030774. URL `https://dl.acm.org/doi/book/10.5555/54968`.

Allen Cypher and Daniel Conrad Halbert. *Watch what I do: Programming by demonstration*. MIT press, Cambridge, MA, USA, 1993. ISBN 0262032139. doi: 10.5555/168080. URL `https://dl.acm.org/doi/10.5555/168080`.

Roger B. Dannenberg. Combining visual and textual representations for flexible interactive audio signal processing. In *Proceedings of the 2004 International Computer Music Conference, ICMC 2004, Miami, Florida, USA, November 1-6, 2004*, 2004. URL `http://hdl.handle.net/2027/spo.bbp2372.2004.090`.

Thomas H Davenport. *Thinking for a Living: How to Get Better Performances and Results From Knowledge Workers*. Harvard Business Press, 09 2005. ISBN 1-59139-423-6. doi: 10.5860/choice.43-3471.

Thomas H Davenport and DJ Patil. Data scientist: The sexiest job of the 21st century. *Harvard Business Review*, 90(5):70–76, 10 2012.

Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. *2008 ACM SIGMOD International Conference on Management of Data 2008, SIGMOD'08*, pages 1345–1350, 2008. doi: 10.1145/1376616.1376772. URL `https://doi.org/10.1145/1376616.1376772`.

Kate E DeCleene and Jennifer Fogo. Publication manual of the American Psychological Association. *Occupational therapy in health care*, 26(1):90–92, 03 2012. doi: 10.3109/07380577.2011.629024. URL `https://doi.org/10.3109/07380577.2011.629024`.

Robert DeLine, Mary Czerwinski, Brian Meyers, Gina Venolia, Steven Drucker, and George Robertson. Code Thumbnails: Using spatial memory to navigate source code. In *Proceedings of the Visual Languages and Human-Centric Computing*, VLHCC '06, pages 11–18, Washington, DC, USA, 09 2006. IEEE Computer Society. ISBN 0-7695-2586-5. doi: 10.1109/VLHCC.2006.14. URL `https://doi.org/10.1109/VLHCC.2006.14`.

Janez Demšar, Blaž Zupan, Gregor Leban, and Tomaz Curk. Orange: From experimental machine learning to interactive data mining. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD '04, pages 537–539, New York, NY, USA, 2004. Springer-Verlag New York, Inc. ISBN 3-540-23108-0. doi: 10.1007/978-3-540-30116-5_58. URL `https://doi.org/10.1007/978-3-540-30116-5_58`.

Vasant Dhar. Data science and prediction. *Communications of the ACM*, 56(12):64–73, 12 2013. ISSN 0001-0782. doi: 10.1145/2500499. URL `https://doi.org/10.1145/2500499`.

Kay Dickersin, SS Chan, TC Chalmersx, HS Sacks, and H Smith Jr. Publication bias and clinical trials. *Controlled Clinical Trials*, 8(4):343–353, 12 1987. doi: 10.1016/0197-2456(87)90155-3. URL `https://doi.org/10.1016/0197-2456(87)90155-3`.

Zoltan Dienes. Bayesian versus orthodox statistics: Which side are you on?, 05 2011. URL `https://doi.org/10.1177/1745691611406920`.

Zoltan Dienes. Using Bayes to get the most out of non-significant results. *Frontiers in Psychology*, 5:781, 07 2014. doi: 10.3389/fpsyg.2014.00781. URL `https://doi.org/10.3389/fpsyg.2014.00781`.

Pierre Dragicevic. *Fair Statistical Communication in HCI*, pages 291–330. Springer International Publishing, Cham, Switzerland, 03 2016. ISBN 978-3-319-26633-6. doi: 10.1007/978-3-319-26633-6_13. URL `https://doi.org/10.1007/978-3-319-26633-6_13`.

Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay, and Fanny Chevalier. Increasing the transparency of research papers with explorable multiverse analyses. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–15, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300295. URL `https://doi.org/10.1145/3290605.3300295`.

Alexander Eiselmayer, Chat Wacharamanotham, Michel Beaudouin-Lafon, and Wendy E. Mackay. Touchstone2: An interactive environment for exploring trade-offs in hci experiment design. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300447. URL `https://doi.org/10.1145/3290605.3300447`.

Michael Richard Ellers. Statlets: Improving statistical analysis with R. Bachelor's thesis, RWTH Aachen University, Aachen, March 2017. URL `https://hci.rwth-aachen.de/publications/ellers2017a.pdf`.

David Ellis. A behavioural model for information retrieval system design. *Journal of Information Science*, 15(4-5):237–247, 1989. ISSN 0022-0418. doi: 10.1177/016555158901500406. URL `https://doi.org/10.1177/016555158901500406`.

K. Anders Ericsson and Herbert A. Simon. *Protocol analysis: Verbal reports as data*. MIT Press; revised edition (13 April 1993), 1980. ISBN 9780262550239.

Len Erlikh. Leveraging legacy system dollars for e-business. *IT Professional*, 2(3):17–23, 2000. doi: 10.1109/6294.846201. URL `https://doi.org/10.1109/6294.846201`.

Martin Erwig and Bernd Meyer. Heterogeneous visual languages-integrating visual and textual programming. In *Proceedings of Symposium on Visual Languages*, pages 318–325, Sep 1995. doi: 10.1109/VL.1995.520825. URL `https://doi.org/10.1109/VL.1995.520825`.

Ilker Etikan and Kabiru Bala. Sampling and sampling methods. *Biometrics & Biostatistics International Journal*, 5(6):215–

217, 2017. ISSN 2378-315X. doi: 10.15406/bbij.2017.05.00149. URL `https://doi.org/10.15406/bbij.2017.05.00149`.

James Evans and Christián Carlos Cannan. *Mechanical Astronomy: A Route to the Ancient Discovery of Epicycles and Eccentrics*, pages 145–174. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-36736-6. doi: 10.1007/978-3-642-36736-6_7. URL `https://doi.org/10.1007/978-3-642-36736-6_7`.

Sebastian S Feger, Sünje Dallmeier-Tiessen, Albrecht Schmidt, and Paweł W Woźniak. Designing for reproducibility: A qualitative study of challenges and opportunities in high energy physics. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300685. URL `https://doi.org/10.1145/3290605.3300685`.

Jean-Daniel Fekete, Danyel Fisher, Arnab Nandi, and Michael Sedlmair. Progressive Data Analysis and Visualization (Dagstuhl Seminar 18411). *Dagstuhl Reports*, 8(10):1–40, 2019. ISSN 2192-5283. doi: 10.4230/DagRep.8.10.1. URL `http://drops.dagstuhl.de/opus/volltexte/2019/10346`.

Andy Field. *Discovering Statistics Using IBM SPSS Statistics*. Sage Publications Ltd., 4th edition, 2013. ISBN 1446249182. doi: 10.5555/2502692.

Dustin Fife. The eight steps of data analysis: A graphical framework to promote sound statistical analysis. *Perspectives on Psychological Science*, 15(4):1054–1075, 2020. doi: 10.1177/1745691620917333. URL `https://doi.org/10.1177/1745691620917333`.

Dmitry Filippov. Feature spotlight: Refactoring Python code, February 2015. `https://blog.jetbrains.com/pycharm/2015/02/feature-spotlight-refactoring-python-code/` (last accessed on May 16, 2021).

L Dee Fink. *Creating Significant Learning Experiences: An Integrated Approach to Designing College Courses*. John Wiley & Sons, 2013. ISBN 978-1118124253.

William Finzer and Laura Gould. Rehearsal world: Programming by rehearsal. *Watch what I do: Programming by demonstration*, pages 79–100, 1993.

Noleine Elizabeth Fitzallen. *Reasoning about covariation with TinkerPlots*. PhD thesis, University of Tasmania, 2012. URL `https://eprints.utas.edu.au/14717/`.

Rich FitzJohn and Daniel Falster. Nice R code: Functions, 2013. URL `https://nicercode.github.io/guides/functions/`.

Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1997. ISBN 978-0134757599.

Karin S Frey and Diane N Ruble. What children say about classroom performance: Sex and grade differences in perceived competence. *Child Development*, 58(4):1066–1078, 08 1987. doi: 10.2307/1130547. URL `https://doi.org/10.2307/1130547`.

Paul L Gardner and Ingrid Hudson. University students' ability to apply statistical procedures. *Journal of Statistics Education*, 7(1):1–18, 1999. ISSN 1069-1898. URL `http://www.amstat.org/publications/jse/secure/v7n1/gardner.cfm`.

Joan Garfield. How students learn statistics. *International Statistical Review*, 63(1):25–34, 1995. ISSN 0306-7734. doi: 10.2307/1403775. URL `https://doi.org/10.2307/1403775`.

Joan Garfield and Andrew Ahlgren. Difficulties in learning basic concepts in probability and statistics: Implications for research. *Journal for Research in Mathematics Education*, 19(1):44–63, 1988. doi: 10.5951/jresematheduc.19.1.0044. URL `https://doi.org/10.5951/jresematheduc.19.1.0044`.

Joan Garfield and Dani Ben-Zvi. How students learn statistics revisited: A current review of research on teaching and learning statistics. *International Statistical Review*, 75(3):372–396, 2007.

doi: 10.1111/j.1751-5823.2007.00029.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1751-5823.2007.00029.x.

Reto Geiger, Beat Fluri, Harald C Gall, and Martin Pinzger. Relation of code clones and change couplings. In *International Conference on Fundamental Approaches to Software Engineering*, volume 3922, pages 411–425. Springer, 2006. ISBN 978-3-540-33094-3. doi: 10.1007/11693017_31. URL https://doi.org/10.1007/11693017_31.

Andrew Gelman. Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian Analysis*, 1(3):515–534, 09 2006. ISSN 1936-0975. doi: 10.1214/06-ba117a. URL https://doi.org/10.1214/06-ba117a.

Andrew Gelman and Eric Loken. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no “fishing expedition” or “*p*-hacking” and the research hypothesis was posited ahead of time. *Department of Statistics, Columbia University*, 348, 11 2013.

Barney G. Glaser and Anselm L. Strauss. *Discovery of Grounded Theory: Strategies for Qualitative Research*. Routledge, USA, 07 2017. ISBN 0-202-30260-1.

Glassdoor Inc. Best Jobs in America – Glassdoor, 2021. https://www.glassdoor.com/List/Best-Jobs-in-America-LST_KQ0,20.htm (last accessed on May 16, 2021).

Mrudulla Gnanadesikan, Richard L Scheaffer, Ann E Watkins, and Jeffrey A Witmer. An activity-based statistics course. *Journal of Statistics Education*, 5(2), 07 1997. doi: 10.1080/10691898.1997.11910531. URL https://doi.org/10.1080/10691898.1997.11910531.

David Goldberg and Cate Richardson. Touch-typing with a stylus. In *Proceedings of the INTERACT ’93 and CHI ’93 Conference on Human Factors in Computing Systems*, CHI ’93, page 80–87, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897915755. doi: 10.1145/169059.169093. URL https://doi.org/10.1145/169059.169093.

Leo A. Goodman. Snowball sampling. *The annals of mathematical statistics*, 32(1):148–170, 1961. ISSN 00034851. URL `http://www.jstor.org/stable/2237615`.

Anne S Goodsell, Michelle Maher, and Vincent Tinto. *Collaborative Learning: A Sourcebook for Higher Education*. Natl Center on Postsecondary, 1992. ISBN 978-9994081820.

Peter Goos and David Meintrup. *Statistics With JMP: Hypothesis Tests, ANOVA and Regression*. John Wiley & Sons, 04 2016. ISBN 978-1-119-09715-0.

Vincent Granville. 40 Techniques Used by Data Scientists, 2016. `https://www.datasciencecentral.com/profiles/blogs/40-techniques-used-by-data-scientists` (last accessed by May 16, 2021).

Thomas Green and Marian Petre. When visual programs are harder to read than textual programs. In *Human-Computer Interaction: Tasks and Organisation, Proceedings of ECCE-6 (6th European Conference on Cognitive Ergonomics)*, pages 167–180, 1992. doi: 10.1.1.57.1633. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.1633`.

Thomas Green and Marian Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages & Computing*, 7 (2):131–174, 6 1996. ISSN 1045-926X. doi: 10.1006/jvlc.1996.0009. URL `https://doi.org/10.1006/jvlc.1996.0009`.

Thomas Green, Marian Petre, and Rachel Bellamy. Comprehensibility of visual and textual programs: A test of superlativism against the 'match-mismatch' conjecture. *Fourth Workshop on Empirical Studies of Programmers.*, 91(743): 121–146, 1991.

Guide2Research. Top Conferences for Human Computer Interaction, 2020. `https://www.guide2research.com/topconf/human-computer-interaction` (last accessed on May 16, 2021).

Philip Guo. *Software Tools to Facilitate Research Programming*. PhD thesis, Stanford University, Stanford, CA, USA, 2012.

Philip Guo and Margo Seltzer. Burrito: Wrapping your lab notebook in computational infrastructure. In *Proceedings of the 4th USENIX Conference on Theory and Practice of Provenance*, TaPP'12, page 7, Berkeley, CA, USA, 2012. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=2342875.2342882`.

Frank G Halasz and Thomas P Moran. Mental models and problem solving in using a calculator. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 212–216, New York, NY, USA, 12 1983. Association for Computing Machinery. ISBN 0897911210. doi: 10.1145/800045.801613. URL `https://doi.org/10.1145/800045.801613`.

Heiko Haller and Stefan Krauss. Misinterpretations of significance: A problem students share with their teachers. *Methods of Psychological Research*, 7(1):1–20, 2002. doi: 2002-14044-001. URL `https://psycnet.apa.org/record/2002-14044-001`.

Harlan Harris, Sean Murphy, and Marck Vaisman. *Analyzing the Analyzers: An Introspective Survey of Data Scientists and Their Work*. O'Reilly Media, Inc., 2013a. ISBN 1449371760. doi: 10.5555/2555869. URL `https://dl.acm.org/doi/10.5555/2555869`.

Harlan Harris, Sean Murphy, and Marck Vaisman. *Analyzing the analyzers: An introspective survey of data scientists and their work*. O'Reilly Media, Inc., 2013b. ISBN 1449371760.

Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. Design as exploration: Creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, page 91–100, New York, NY, USA, 2008a. Association for Computing Machinery. ISBN 9781595939753. doi: 10.1145/1449715.1449732. URL `https://doi.org/10.1145/1449715.1449732`.

Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. Design as exploration: Creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st Annual ACM Symposium on User*

*Interface Software and Technology*, UIST ’08, pages 91–100, New York, NY, USA, 2008b. ACM. ISBN 978-1-59593-975-3. doi: 10.1145/1449715.1449732. URL `http://doi.acm.org/10.1145/1449715.1449732`.

Andrew Head, Fred Hohman, Titus Barik, Steven M. Drucker, and Robert DeLine. Managing messes in computational notebooks. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI ’19, pages 270:1–270:12, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300500. URL `http://doi.acm.org/10.1145/3290605.3300500`.

M Heithaus, L Dill, G Marshall, and B Buhleier. Habitat use and foraging behavior of tiger sharks (galeocerdo cuvier) in a seagrass ecosystem. *Marine Biology*, 140(2):237–248, 2002. doi: 10.1007/s00227-001-0711-7. URL `https://doi.org/10.1007/s00227-001-0711-7`.

Bradley M Hemminger, Dihui Lu, KTL Vaughan, and Stephanie J Adams. Information Seeking Behavior of Academic Scientists. *Journal of the American society for information science and technology*, 58(14):2205–2225, 2007. doi: 10.1002/asi.20686. URL `https://doi.org/10.1002/asi.20686`.

Miguel A. Hernán, John Hsu, and Brian Healy. A second chance to get causal inference right: A classification of data science tasks. *CHANCE*, 32(1):42–49, 2019. ISSN 0933-2480. doi: 10.1080/09332480.2019.1579578. URL `https://doi.org/10.1080/09332480.2019.1579578`.

Morten Hertzum and Annelise Mark Pejtersen. The information-seeking practices of engineers: searching for documents as well as for people. *Information Processing & Management*, 36(5):761–778, 09 2000. ISSN 0306-4573. doi: 10.1016/S0306-4573(00)00011-X. URL `https://doi.org/10.1016/S0306-4573(00)00011-X`.

Yoshiki Higo and Shinji Kusumoto. Enhancing quality of code clone detection with program dependency graph. In *2009 16th Working Conference on Reverse Engineering*, pages 315–316. IEEE Computer Society, 2009. ISBN 978-0-7695-3867-9. doi: 10.1109/WCRE.2009.39. URL `https://doi.org/10.1109/WCRE.2009.39`.

TR Hodgson. The effects of hands-on activities on students' understanding of selected statistical concepts. In *Proceedings of the Eighteenth Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, pages 241–246. ERIC Clearinghouse for Science, Mathematics, and Environmental Education, 1996.

Markus Hofmann and Ralf Klinkenberg. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. Chapman & Hall/CRC, 2013. ISBN 9781482205497.

Karen Holtzblatt, Jessamyn Burns Wendell, and Shelley Wood. *Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 1558608702. doi: 10.5555/2891404.

Yue Hu. Statistics in the wild: How practitioners choose statistical procedures. Master's thesis, RWTH Aachen University, Aachen, September 2019. URL `https://hci.rwth-aachen.de/publications/hu2019a.pdf`.

Raymond Hubbard and María Jesús Bayarri. Confusion over measures of evidence (p's) versus errors ($\alpha$'s) in classical statistical testing. *The American Statistician*, 57(3):171–178, 08 2003. doi: 10.1198/0003130031856. URL `https://doi.org/10.1198/0003130031856`.

Raymond Hubbard and Patricia Ryan. The historical growth of statistical significance testing in psychology and its future prospects. *Educational and Psychological Measurement*, 60(05):661–681, 2000. ISSN 0013-1644. doi: 10.1177/00131640021970808. URL `https://doi.org/10.1177/00131640021970808`.

John Hughes. Why functional programming matters. *The Computer Journal*, 32(2):98–107, 1989. doi: 10.1093/comjnl/32.2.98. URL `https://doi.org/10.1093/comjnl/32.2.98`.

Macartan Humphreys, Raul Sanchez de la Sierra, and Peter van der Windt. Fishing, commitment, and communication: A proposal for comprehensive nonbinding research registration. *Political Analysis*, 21:1–20, 2013. ISSN 1047-1987. doi: 10.1093/pan/mps021. URL `https://www.jstor.org/stable/23359687`.

Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. doi: 10.2307/1390807. URL `https://doi.org/10.2307/1390807`.

Valerie Illingworth. *Dictionary of Computing*. Oxford University Press, Inc., 1997. ISBN 978-0198538554.

Kenneth E. Iverson. *A Programming Language*. John Wiley & Sons, Inc., New York, NY, USA, 1962. ISBN 0-471430-14-5.

Menglin Jin and Robert E. Dickinson. New observational evidence for global warming from satellite. *Geophysical Research Letters*, 29(10):39–1–39–4, 2002. doi: 10.1029/2001GL013833. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2001GL013833`.

David W Johnson et al. *Cooperative Learning: Increasing College Faculty Instructional Productivity. ASHE-ERIC Higher Education Report No. 4, 1991.* ERIC, 1991.

J Howard Johnson. Identifying redundancy in source code using fingerprints. In *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering-Volume 1*, pages 171–183, 1993.

Steven A. Jones. Steps in planning a research experiment, 2004. `http://www2.latech.edu/~sajones/Senior%20Design%20Web%20Pages/Steps%20in%20Planning%20a%20Research%20Experiment.htm` (last accessed on May 16, 2021).

Eunice Jun, Maureen Daum, Jared Roesch, Sarah Chasins, Emery Berger, Rene Just, and Katharina Reinecke. Tea: A high-level language and runtime system for automating statistical analysis. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 591–603, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368162. doi: 10.1145/3332165.3347940. URL `https://doi.org/10.1145/3332165.3347940`.

Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, 2012. ISSN 1077-2626. doi: 10.1109/tvcg.2012.219. URL `https://doi.org/10.1109/TVCG.2012.219`.

Cory Kapser and Michael Godfrey. "Cloning considered harmful" considered harmful: Patterns of cloning in software. *Empirical Software Engineering*, 13(6):645–692, December 2008. ISSN 1382-3256. doi: 10.1007/s10664-008-9076-6. URL `http://dx.doi.org/10.1007/s10664-008-9076-6`.

Cory Kapser and Michael W Godfrey. Toward a taxonomy of clones in source code: A case study. *Evolution of Large Scale Industrial Software Architectures*, 16:107–113, 2003. doi: 10.1.1.5.6056. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.6056`.

Maurits Kaptein and Judy Robertson. Rethinking statistical analysis methods for CHI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 1105–1114, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310154. doi: 10.1145/2207676.2208557. URL `https://doi.org/10.1145/2207676.2208557`.

Matthew Kay, Gregory Nelson, and Eric Hekler. Researcher-centered design of statistics: Why Bayesian statistics better fit the culture and incentives of HCI. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 4521–4532, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858465. URL `https://doi.org/10.1145/2858036.2858465`.

Douglas Kell and Stephen Oliver. Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. *BioEssays*, 26(1):99–105, 2003. ISSN 0265-9247. doi: 10.1002/bies.10385. URL `https://doi.org/10.1002/bies.10385`.

Norbert L. Kerr. HARKing: Hypothesizing after the results are known. *Personality and Social Psychology Review*, 2:196–217, 1998. ISSN 1088-8683. doi: 10.1207/s15327957pspr0203_4. URL `https://doi.org/10.1207/s15327957pspr0203_4`.

Mary Beth Kery and Brad A. Myers. Exploring exploratory programming. *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 25–29, 2017. doi: 10.1109/vlhcc.2017.8103446. URL `https://doi.org/10.1109/VLHCC.2017.8103446`.

Mary Beth Kery, Amber Horvath, and Brad Myers. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 1265–1276, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025626. URL `http://doi.acm.org/10.1145/3025453.3025626`.

Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E. John, and Brad A. Myers. The story in the notebook: Exploratory data science using a literate programming tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 174:1–174:11, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5620-6. doi: 10.1145/3173574.3173748. URL `http://doi.acm.org/10.1145/3173574.3173748`.

Mary Beth Kery, Bonnie E. John, Patrick O'Flaherty, Amber Horvath, and Brad A. Myers. Towards effective foraging by data scientists to find past analysis choices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 92:1–92:13, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300322. URL `http://doi.acm.org/10.1145/3290605.3300322`.

Rex B Kline. What's wrong with statistical tests–and where we go from here. *Beyond Significance Testing: Reforming Data Analysis Methods in Behavioral Research*, 2004. doi: 10.1037/10693-003. URL `https://psycnet.apa.org/doi/10.1037/10693-003`.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle

Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. *Jupyter notebooks - A publishing format for reproducible computational workflows*. IOS Press, 2016. URL https://eprints.soton.ac.uk/403913/.

Andrew J. Ko and Brad A. Myers. Designing the whyline: A debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, page 151–158, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581137028. doi: 10.1145/985692.985712. URL https://doi.org/10.1145/985692.985712.

Rainer Koschke. Frontiers of software clone management. In *2008 Frontiers of Software Maintenance*, pages 119–128, Sept 2008. doi: 10.1109/FOSM.2008.4659255. URL https://doi.org/10.1109/FOSM.2008.4659255.

Jan-Peter Krämer, Thorsten Karrer, Jonathan Diehl, and Jan Borchers. Stacksplorer: Understanding dynamic program behavior. In *Adjunct Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 433–434, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304627. doi: 10.1145/1866218.1866257. URL https://doi.org/10.1145/1866218.1866257.

John Kruschke. *Doing Bayesian Data Analysis*. Elsevier, 2015. ISBN 9780124058880.

Carol Collier Kuhlthau. *Seeking Meaning: A Process Approach to Library and Information Services*, volume 2. Libraries Unlimited Westport, CT, 2004.

J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174, 1977. doi: 10.2307/2529310. URL https://doi.org/10.2307/2529310.

Loet Leydesdorff and Stasa Milojevic. Scientometrics. *CoRR*, abs/1208.4566, 2012. URL http://arxiv.org/abs/1208.4566.

Jiali Liu, Nadia Boukhelifa, and James R. Eagan. Understanding the role of alternatives in data analysis practices. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):66–76, 2020. doi: 10.1109/TVCG.2019.2934593. URL `https://doi.org/10.1109/TVCG.2019.2934593`.

Julia Lowndes, Benjamin Best, Courtney Scarborough, Jamie Afflerbach, Melanie Frazier, Casey O'Hara, Ning Jiang, and Benjamin Halpern. Our path to better science in less time using open data science tools. *Nature ecology & evolution*, 1(6): 160, 2017. doi: 10.1038/s41559-017-0160. URL `https://doi.org/10.1038/s41559-017-0160`.

LP StataCorp. Stata Data Analysis and Statistical Software. *Special Edition Release*, 10:733, 2007. URL `https://www.stata.com/order/maintenance-agreement/`.

David Lubinsky and Daryl Pregibon. Data analysis as search. *Journal of Econometrics*, 38(1): 247–268, 1988. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(88)90035-8. URL `https://www.sciencedirect.com/science/article/pii/0304407688900358`.

M Leigh Lunsford, Ginger Holmes Rowell, and Tracy Goodson-Espy. Classroom research: Assessment of student understanding of sampling distributions of means and the central limit theorem in post-calculus probability and statistics classes. *Journal of Statistics Education*, 14(3), 2006. doi: 10.1080/10691898.2006.11910587. URL `https://doi.org/10.1080/10691898.2006.11910587`.

Johannes Maas. Statwire: Visual flow-based programming for statistical analysis. Bachelor's thesis, RWTH Aachen University, Aachen, August 2017. URL `https://hci.rwth-aachen.de/publications/maas2017a.pdf`.

Martin Mächler. Good Practices in R Programming, 2014. `https://stat.ethz.ch/Teaching/maechler/R/useR_2014/Maechler-2014-pr.pdf` (last accessed on May 16, 2021).

I. Scott MacKenzie and Shawn X. Zhang. The immediate usability of Graffiti. In *Proceedings of the Conference on Graphics Interface '97*, page 129–137, CAN, 1997. Canadian Information Processing Society. ISBN 0969533861. URL https://dl.acm.org/doi/10.5555/266774.266792.

Robert C. Martin. *Clean code: A handbook of agile software craftsmanship*. Pearson; 1st edition (August 1, 2008), 2008. ISBN 9780132350884.

Justin Matejka and George Fitzmaurice. Same stats, different graphs: Generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 1290–1294, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025912. URL https://doi.org/10.1145/3025453.3025912.

Scott Maxwell. The persistence of underpowered studies in psychological research: Causes, consequences, and remedies. *Psychological Methods*, 9(2):147, 2004. doi: 10.1037/1082-989x.9.2.147. URL https://doi.org/10.1037/1082-989x.9.2.147.

J. Mayrand, C. Leblanc, and E. M. Merlo. Experiment on the automatic detection of function clones in a software system using metrics. In *1996 Proceedings of International Conference on Software Maintenance*, pages 244–253, Nov 1996. doi: 10.1109/ICSM.1996.565012. URL https://doi.org/10.1109/ICSM.1996.565012.

Deirdre N McCloskey and Stephen T Ziliak. The standard error of regressions. *Journal of Economic Literature*, 34(1):97–114, 1996. URL https://www.jstor.org/stable/2729411.

Amelia Ahlers McNamara. *Bridging the gap between tools for learning and for doing statistics*. PhD thesis, UCLA, 2015.

Malte Meng. StatHunt - Supporting novice researchers seek statistical procedures. Bachelor's thesis, RWTH Aachen University, Aachen, June 2020. URL https://hci.rwth-aachen.de/publications/meng2020a.pdf.

Philip Miller, John Pane, Glenn Meter, and Scott Vorthmann. Evolution of novice programming environments: The structure editors of Carnegie Mellon University. *Interactive Learning Environments*, 4(2):140–158, 1994. doi: 10.1080/1049482940040202. URL `https://doi.org/10.1080/1049482940040202`.

Victor Minichiello, Rosalie Aroni, and Terrence Neville Hays. *In-depth interviewing: Principles, techniques, analysis*. Pearson Education Australia, 2008. ISBN 9780733980121.

A. Monden, D. Nakae, T. Kamiya, S. Sato, and K. Matsumoto. Software quality analysis by code clones in industrial legacy software. In *Proceedings Eighth IEEE Symposium on Software Metrics*, pages 87–94, 2002. doi: 10.1109/METRIC.2002.1011328. URL `https://doi.org/10.1109/METRIC.2002.1011328`.

Nick Montfort. *Exploratory Programming for the Arts and Humanities*. MIT Press, 2016. ISBN 9780262034203.

Kelly Moore. Politics of nature: How to bring the sciences into democracy. *Contemporary Sociology*, 34(2):168, 2005. doi: 10.1177/009430610503400236. URL `https://doi.org/10.1177/009430610503400236`.

Brad A. Myers. Taxonomies of visual programming and program visualization. *Journal of Visual Languages and Computing*, 1(1):97–123, March 1990. ISSN 1045-926X. doi: 10.1016/S1045-926X(05)80036-9. URL `http://dx.doi.org/10.1016/S1045-926X(05)80036-9`.

Brad A. Myers, Andrew J. Ko, and Margaret M. Burnett. Invited research overview: End-user programming. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, page 75–80, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595932984. doi: 10.1145/1125451.1125472. URL `https://doi.org/10.1145/1125451.1125472`.

Raymond S. Nickerson. Null hypothesis significance testing: A review of an old and continuing controversy. *Psychological Methods*, 5:241, 2000. ISSN 1082-989X. doi: 10.1037/1082-989x.5.2.241. URL `https://doi.org/10.1037/1082-989x.5.2.241`.

Jakob Nielsen. Progressive disclosure, 2006. `http://nngroup.com/articles/progressive-disclosure/` (last accessed on May 16, 2021).

Markus Nivala, Alena Seredko, Tanya Osborne, and Thomas Hillman. Stack Overflow – Informal learning and the global expansion of professional development and opportunities in programming? *2020 IEEE Global Engineering Education Conference (EDUCON)*, 00:402–408, 2020. doi: 10.1109/educon45650.2020.9125165. URL `https://doi.org/10.1109/EDUCON45650.2020.9125165`.

Michael Oaks. Statistical inference: A commentary for the social and behavioral sciences, 1986.

Michael P O'Brien and Jim Buckley. Modelling the information-seeking behaviour of programmers: An empirical approach. In *13th International Workshop on Program Comprehension (IWPC'05)*, pages 125–134. IEEE, 2005. doi: 10.1109/WPC.2005.24. URL `https://doi.org/10.1109/WPC.2005.24`.

Cedric Okinda, Innocent Nyalala, Tchalla Korohou, Celestine Okinda, Jintao Wang, Tracy Achieng, Patrick Wamalwa, Tai Mang, and Mingxia Shen. A review on computer vision systems in monitoring of poultry: A welfare perspective. *Artificial Intelligence in Agriculture*, 4:184–208, 2020. ISSN 2589-7217. doi: 10.1016/j.aiia.2020.09.002. URL `https://doi.org/10.1016/j.aiia.2020.09.002`.

Nigini Oliveira, Nazareno Andrade, and Katharina Reinecke. Participation differences in Q&A sites across countries: Opportunities for cultural adaptation. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, NordiCHI '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450347631. doi: 10.1145/2971485.2971520. URL `https://doi.org/10.1145/2971485.2971520`.

Open Science Collaboration. Estimating the reproducibility of psychological science. *Science*, 349(6251), 2015. ISSN 0036-8075. doi: 10.1126/science.aac4716. URL `https://science.sciencemag.org/content/349/6251/aac4716`.

Norm O'Rourke, Larry Hatcher, and Edward J Stepanski. *A Step-by-Step Approach to Using SAS for Univariate & Multivariate Statistics*. SAS Publishing, 2005. ISBN 978-1590474174.

Julie Pallant. *SPSS Survival Manual: A Step by Step Guide to Data Analysis Using IBM SPSS*. Open University Press, 2020. ISBN 978-0335242399.

Judith Palmer. Scientists and information: I. using cluster analysis to identify information style. *Journal of Documentation*, 47(2):105–129, May 1991. ISSN 0022-0418. doi: 10.1108/eb026873. URL `https://doi.org/10.1108/eb026873`.

R. K. Pandey and M. M. Burnett. Is it easier to write matrix manipulation programs visually or textually? an empirical study. In *Proceedings 1993 IEEE Symposium on Visual Languages*, pages 344–351, Aug 1993. doi: 10.1109/VL.1993.269621. URL `https://doi.org/10.1109/VL.1993.269621`.

Kayur Patel, Naomi Bancroft, Steven M. Drucker, James Fogarty, Andrew J. Ko, and James Landay. Gestalt: Integrated support for implementation and analysis in machine learning. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 37–46, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0271-5. doi: 10.1145/1866029.1866038. URL `http://doi.acm.org/10.1145/1866029.1866038`.

Jose D. Perezgonzalez. Fisher, Neyman-Pearson or NHST? A tutorial for teaching data testing. *Frontiers in Psychology*, 6:223, 2015. ISSN 1664-1078. doi: 10.3389/fpsyg.2015.00223. URL `https://doi.org/10.3389/fpsyg.2015.00223`.

Thomas J. Pfaff and Aaron Weinberg. Do hands-on activities increase student understanding?: A case study. *Journal of Statistics Education*, 17(3):null, 2009. doi: 10.1080/10691898.2009.11889536. URL `https://doi.org/10.1080/10691898.2009.11889536`.

Gregory Piatetsky. How many data scientists are there?, September 2018. `https://www.kdnuggets.com/2018/09/`

how-many-data-scientists-are-there.html (last accessed on May 16, 2021).

Thomas E Pinelli et al. The relationship between seven variables and the use of US government technical reports by us aerospace engineers and scientists. In *Proceedings of the ASIS Annual Meeting*, volume 28, pages 313–21. ERIC, 1991.

Foster Provost and Tom Fawcett. Data science and its relationship to big data and data-driven decision making. *Big Data*, 1:51–59, 2013. ISSN 2167-6461. doi: 10.1089/big.2013.1508. URL https://doi.org/10.1089/big.2013.1508.

Xiaoying Pu and Matthew Kay. The garden of forking paths in visualization: A design space for reliable exploratory visual analytics. *2018 IEEE Evaluation and Beyond - Methodological Approaches for Visualization (BELIV)*, 00:37–45, 2018. doi: 10.1109/beliv.2018.8634103. URL https://doi.org/10.1109/BELIV.2018.8634103.

Rahul Rathi. Effect of Cambridge Analytica's Facebook ads on the 2016 US Presidential Election. https://tinyurl.com/yhanz6jf, January 2017. Last accessed on May 16, 2021.

Dhavleesh Rattan, Rajesh Bhatia, and Maninder Singh. Software clone detection: A systematic review. *Information and Software Technology*, 55(7):1165 – 1199, 2013. ISSN 0950-5849. doi: 10.1016/j.infsof.2013.01.008. URL http://www.sciencedirect.com/science/article/pii/S0950584913000323.

Stephen K Reed. Effect of computer graphics on improving estimates to algebra word problems. *Journal of Educational Psychology*, 77(3):285, 1985.

Jenny Reinhard. *Participants, incentives and user studies: A survey of CHI 2019*. Bachelor thesis, RWTH Aachen University, Aachen, Germany, 2020. URL https://hci.rwth-aachen.de/publications/reinhard2020a.pdf.

Lauren B Resnick. *Education and learning to think*. National Academies, 1987. ISBN 978-0309037853.

Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. Scratch: Programming for all. *Communications of the ACM*, 52(11):60–67, November 2009. ISSN 0001-0782. doi: 10.1145/1592761.1592779. URL `http://doi.acm.org/10.1145/1592761.1592779`.

Martin Robillard. What makes APIs hard to learn? Answers from developers. *IEEE Software*, 26(6):27–34, 2009. ISSN 0740-7459. doi: 10.1109/ms.2009.193. URL `https://doi.org/10.1109/MS.2009.193`.

Samuel G Robson, Myriam A Baum, Jennifer L Beaudry, Julia Beitner, Hilmar Brohmer, Jason Chin, Katarzyna Jasko, Chrystyna D Kouros, Dr. Laukkonen, Ruben, David Moreau, and et al. Nudging open science. *Web*, Apr 2021. doi: 10.31234/osf.io/zn7vt. URL `psyarxiv.com/zn7vt`.

Jochen Rode and Mary Beth Rosson. Programming at runtime: Requirements & paradigms for nonprogrammer web application development. In *IEEE Symposium on Human Centric Computing Languages and Environments, 2003.*, pages 23–30, 2003. doi: 10.1109/HCC.2003.1260198. URL `https://doi.org/10.1109/HCC.2003.1260198`.

Mary Beth Rosson and John M. Carroll. Active programming strategies in reuse. In Oscar M. Nierstrasz, editor, *ECOOP' 93 — Object-Oriented Programming*, pages 4–20, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg. ISBN 978-3-540-47910-9.

Mary Beth Rosson, Julie Ballin, and Heather Nash. Everyday programming: Challenges and opportunities for informal web development. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 123–130, 2004. doi: 10.1109/VLHCC.2004.26. URL `https://doi.org/10.1109/VLHCC.2004.26`.

Adam Rule, Ian Drosos, Aurélien Tabard, and James D. Hollan. Aiding collaborative reuse of computational notebooks with annotated cell folding. *Proceedings of ACM Human Computer Interaction*, 2(CSCW), November 2018. doi: 10.1145/3274419. URL `https://doi.org/10.1145/3274419`.

Thomas Rutledge and Cathy Loh. Effect sizes and statistical testing in the determination of clinical significance in behavioral medicine research. *Annals of Behavioral Medicine*, 27 (2):138–145, 2004.

Johnny Saldaña. *The Coding Manual for Qualitative Researchers*. SAGE Publications Ltd., London, UK, 2013. ISBN 978-1-44624-736-5.

David W. Sandberg. Smalltalk and exploratory programming. *ACM SIGPLAN Notices*, 23(10):85–92, October 1988. ISSN 0362-1340. doi: 10.1145/51607.51614. URL `http://doi.acm.org/10.1145/51607.51614`.

Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLOS Computational Biology*, 9(10):1–4, 2013. doi: 10.1371/journal.pcbi.1003285. URL `https://doi.org/10.1371/journal.pcbi.1003285`.

Michel Sanner. Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1): 57–61, 1999.

Mark Saunders, Philip Lewis, and Adrian Thornhill. *Research methods for business students*. Pearson Education Limited; 8th edition (3 Jun. 2019), 2019. ISBN 1292208783.

Reijo Savolainen. Everyday life information seeking: Approaching information seeking in the context of "way of life". *Library & Information Science Research*, 17(3):259–294, 1995. doi: 10.1016/0740-8188(95)90048-9. URL `https://doi.org/10.1016/0740-8188(95)90048-9`.

D. A. Scanlan. Structured flowcharts outperform pseudocode: An experimental comparison. *IEEE Software*, 6(5):28–36, Sept 1989. ISSN 0740-7459. doi: 10.1109/52.35587. URL `https://doi.org/10.1109/52.35587`.

Frank Schmidt and John Hunter. *Eight common but false objections to the discontinuation of significance testing in the analysis of research data*. Taylor & Francis, 1997. ISBN 9781315629049. doi: 10.4324/9781315629049. URL `https://doi.org/10.4324/9781315629049`.

Daniel Schwartz, Susan Goldman, Nancy Vye, and Brigid Barron. Aligning everyday and mathematical reasoning: The case of sampling assumptions. *Reflections on Statistics: Learning, Teaching, and Assessment in Grades K-12*, pages 233–273, 1998.

Carolyn Seaman. The information gathering strategies of software maintainers. In *Proceedings of the International Conference on Software Maintenance*, pages 141–149. IEEE, 2002. doi: 10.1109/ICSM.2002.1167761. URL `https://doi.org/10.1109/ICSM.2002.1167761`.

Yuliya Sergiyenko. Investigating statistical analysis workflows in HCI. Master's thesis, RWTH Aachen University, Aachen, October 2018.

Michael Shaughnessy. *Research in probability and statistics: Reflections and directions.* Macmillan Publishing Co, Inc, 1992. doi: 1992-97586-019. URL `https://psycnet.apa.org/record/1992-97586-019`.

Beau Sheil. Readings in artificial intelligence and software engineering. *XI: Artificial Intelligence Programming*, pages 573–580, 1983. doi: 10.1016/b978-0-934613-12-5.50048-3.

Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22:1359–1366, 2011. ISSN 0956-7976. doi: 10.1177/0956797611417632. URL `https://doi.org/10.1177/0956797611417632`.

Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. An examination of software engineering work practices. In *Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research*, CASCON '97, page 21. IBM Press, 1997.

Meredith Skeels, Bongshin Lee, Greg Smith, and George Robertson. Revealing uncertainty for information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, page 376–379, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581415. doi: 10.1145/1385569.1385637. URL `https://doi.org/10.1145/1385569.1385637`.

Jonathan A. Smith, Rom Harre, and Luk van Langenhove. *Rethinking Methods in Psychology*. SAGE Publications Ltd, London, UK, 1995. ISBN 0-8039-7732-8.

Society for the Teaching of Psychology Statistical Literacy Taskforce. Statistical literacy in the undergraduate psychology curriculum, 2014. `https://teachpsych.org/Resources/Documents/otrp/resources/statistics/STP_Statistical%20Literacy_Psychology%20Major%20Learning%20Goals_4-2014.pdf` (last accessed by May 16, 2021).

Aaron Springer and Steve Whittaker. Progressive disclosure: Empirically motivated approaches to designing effective transparency. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 107–120, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362726. doi: 10.1145/3301275.3302322. URL `https://doi.org/10.1145/3301275.3302322`.

Stack Overflow. Tour - Stack Overflow, 2021. `https://stackoverflow.com/talent/en` (Last accessed on May 16, 2021.

Susan S Stodolsky, Scott Salk, and Barbara Glaessner. Student views about learning math and social studies. *American Educational Research Journal*, 28(1):89–116, 1991. doi: 10.3102/00028312028001089. URL `https://doi.org/10.3102/00028312028001089`.

Anselm Strauss and Juliet Corbin. Grounded Theory Methodology: An Overview. *Handbook of Qualitative Research*, 17:273–285, 1994.

Anselm Strauss and Juliet Corbin. *Basics of Qualitative Research Techniques*. SAGE Publications, 1998. ISBN 9780803959408.

Krishna Subramanian and Jan Borchers. Statplayground: Exploring statistics through visualizations. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, page 401–404, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346566. doi: 10.

1145/3027063.3052970. URL `https://doi.org/10.1145/3027063.3052970`.

Krishna Subramanian, Johannes Maas, Michael Ellers, Chat Wacharamanotham, Simon Voelker, and Jan Borchers. Statwire: Visual flow-based statistical programming. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, pages LBW104:1–LBW104:6, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5621-3. doi: 10.1145/3170427.3188528. URL `http://doi.acm.org/10.1145/3170427.3188528`.

Krishna Subramanian, Jeanine Bonot, Radu A. Coanda, and Jan Borchers. Statplayground: A sandbox for learning practical statistics. In David Lamas, Fernando Loizides, Lennart Nacke, Helen Petrie, Marco Winckler, and Panayiotis Zaphiris, editors, *Human-Computer Interaction – INTERACT 2019*, pages 156–165, Cham, 2019a. Springer International Publishing. ISBN 978-3-030-29384-0.

Krishna Subramanian, Ilya Zubarev, Simon Völker, and Jan Borchers. Supporting data workers to perform exploratory programming. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, page 1–6, New York, NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450359719. doi: 10.1145/3290607.3313027. URL `https://doi.org/10.1145/3290607.3313027`.

Krishna Subramanian, Nur Hamdan, and Jan Borchers. Casual notebooks and rigid scripts: Understanding data science programming. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–5, 2020a. doi: 10.1109/VL/HCC50065.2020.9127207. URL `https://doi.org/10.1109/VL/HCC50065.2020.9127207`.

Krishna Subramanian, Johannes Maas, and Jan Borchers. Tractus: Understanding and supporting source code experimentation in hypothesis-driven data science. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020b. Association for Computing Machinery. ISBN 9781450367080. doi:

10.1145/3313831.3376764. URL https://doi.org/10.1145/3313831.3376764.

Matúš Sulír, Michaela Bačíková, Sergej Chodarev, and Jaroslav Porubän. Visual Augmentation of Source Code Editors: A Systematic Mapping Study. *Journal of Visual Languages & Computing*, 49:46–59, 2018. ISSN 1045-926X. doi: 10.1016/j.jvlc.2018.10.001. URL http://dx.doi.org/10.1016/j.jvlc.2018.10.001.

Tim Teitelbaum and Thomas Reps. The cornell program synthesizer: A syntax-directed programming environment. *Communications of the ACM*, 24(9):563–573, September 1981. ISSN 0001-0782. doi: 10.1145/358746.358755. URL https://doi.org/10.1145/358746.358755.

Alexander H. Toledo, Robert Flikkema, and Luis H. Toledo-Pereyra. Developing the research hypothesis. *Journal of Investigative Surgery*, 24(5):191–194, 2011. doi: 10.3109/08941939.2011.609449. URL https://doi.org/10.3109/08941939.2011.609449. PMID: 21867386.

Transparent Statistics in Human–Computer Interaction Working Group. Transparent Statistics Guidelines, Jun 2019. (Available at https://transparentstats.github.io/guidelines).

Christoph Treude and Martin P Robillard. Understanding stack overflow code fragments. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 509–513. IEEE, 2017. doi: 10.1109/ICSME.2017.24. URL https://doi.org/10.1109/ICSME.2017.24.

John W. Tukey. *Exploratory Data Analysis*, volume 2. Pearson, 1977. ISBN 978-0201076165.

John W Tukey. We Need Both Exploratory and Confirmatory. *The American Statistician*, 34(1):23–25, 1980. ISSN 0003-1305. doi: 10.1080/00031305.1980.10482706. URL https://doi.org/10.2307/2682991.

Kimmo Tuominen, Sanna Talja, and Reijo Savolainen. *Discourse, Cognition, and Reality: The Social Constructionist Viewpoint on Information Practices*. Information Today, 2005.

N. Ueda. Optimal linear combination of neural networks for improving classification performance. *IEEE Transactions on*

*Pattern Analysis and Machine Intelligence*, 22(2):207–215, 2000. doi: 10.1109/34.825759. URL `https://doi.org/10.1109/34.825759`.

Pedro M Valero-Mora, Ruben Ledesma, et al. Graphical user interfaces for R. *Journal of Statistical Software*, 49(1):1–8, 2012. doi: 10.18637/jss.v049.i01.

Michael L. Van De Vanter. The Documentary Structure of Source Code. *Information and Software Technology*, 44 (13):767–782, 2002. ISSN 0950-5849. doi: 10.1016/s0950-5849(02)00103-9. URL `https://doi.org/10.1016/S0950-5849(02)00103-9`. Special Issue on Source Code Analysis and Manipulation (SCAM).

JP Verma. *Data analysis in management with SPSS software*. Springer Science & Business Media, 2012. ISBN 78-81-322-0786-3.

Rajesh Vikraman. Global Report on State of Data Science & Machine Learning - 2018 Based on Kaggle Survey, 2018. `https://rpubs.com/cvrajesh/kagglesurvey2018`, (last accessed on May 16, 2021).

Ernst Von Glasersfeld. Learning as a constructive activity. *Problems of representation in the teaching and learning of mathematics*, pages 3–17, 1987. doi: 10.1.1.458.7301. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.458.7301`.

Chat Wacharamanotham, Krishna Subramanian, Sarah Theres Völkel, and Jan Borchers. Statsplorer: Guiding novices in statistical analysis. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 2693–2702, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450331456. doi: 10.1145/2702123.2702347. URL `https://doi.org/10.1145/2702123.2702347`.

Chat Wacharamanotham, Lukas Eisenring, Steve Haroz, and Florian Echtler. Transparency of chi research artifacts: Results of a self-reported survey. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367080. doi:

10.1145/3313831.3376448. URL https://doi.org/10.1145/3313831.3376448.

Dennis Wackerly and J Lang. ExplorStat. In *Joint Statistics Meeting, Chicago*, 1996. URL http://www.stat.ufl.edu/users/dwack.

Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Polo Chau. CNN Explainer: Learning Convolutional Neural Networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1396–1406, 2021. doi: 10.1109/TVCG.2020.3030418. URL https://doi.org/10.1109/tvcg.2020.3030418.

Claire Warwick, Jon Rimmer, Ann Blandford, Jeremy Gow, and George Buchanan. Cognitive economy and satisficing in information seeking: A longitudinal study of undergraduate information behavior. *Journal of the American Society for Information Science and Technology*, 60(12):2402–2415, 2009. doi: 10.1002/asi.21179. URL https://doi.org/10.1002/asi.21179.

Mark Weiser. Program slicing. In *Proceedings of the 5th International Conference on Software Engineering*, ICSE '81, pages 439–449, Piscataway, NJ, USA, 1981. IEEE Press. ISBN 0-89791-146-6. URL http://dl.acm.org/citation.cfm?id=800078.802557.

Kirsten Whitley. Visual programming languages and the empirical evidence for and against. *Journal of Visual Languages & Computing*, 8(1):109 – 142, 1997. ISSN 1045-926X. doi: 10.1006/jvlc.1996.0030. URL https://doi.org/10.1006/jvlc.1996.0030.

Ethelene Whitmire. Information Seeking in the Online Age: Principles and Practice. *Journal of the Association for Information Science and Technology*, 51(10):964, 2000. doi: 10.1002/1097-4571. URL https://doi.org/10.1002/1097-4571.

Moritz Wittenhagen, Christian Cherek, and Jan Borchers. Chronicler: Interactive exploration of source code history. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3522–3532, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi:

10.1145/2858036.2858442. URL http://doi.acm.org/10.1145/2858036.2858442.

Jacob O. Wobbrock. Practical Statistics for HCI, 2011. http://depts.washington.edu/acelab/proj/ps4hci/ (last accessed on May 16, 2021).

Jacob O. Wobbrock, Brad A. Myers, and John A. Kembel. EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, page 61–70, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581136366. doi: 10.1145/964696.964703. URL https://doi.org/10.1145/964696.964703.

Harry F. Wolcott. *Transforming Qualitative Data: Description, Analysis, and Interpretation*. SAGE Publications, London, United Kingdom, 1994. ISBN 9780803952812.

Edward N Wolff. The growth of information workers in the U.S. economy. *Communications of the ACM*, 48 (10):37, 2005. ISSN 0001-0782. doi: 10.1145/1089107.1089134. URL https://doi.org/10.1145/1089107.1089134.

K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mané, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. Visualizing dataflow graphs of deep learning models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):1–12, 2017. ISSN 1077-2626. doi: 10.1109/TVCG.2017.2744878. URL https://doi.org/10.1109/TVCG.2017.2744878.

Patricia Wright and Fraser Reid. Written information: Some alternatives to prose for expressing the outcomes of complex contingencies. *Journal of Applied Psychology*, 57(2):160, 1973. doi: 10.1037/h0037045. URL https://doi.org/10.1037/h0037045.

Berna Yazici and Senay Yolacan. A comparison of various tests of normality. *Journal of Statistical Computation and Simulation*, 77(2):175–183, 2007. doi: 10.1080/10629360600678310. URL https://doi.org/10.1080/10629360600678310.

Moshe Yitzhaki and Gloria Hammershlag. Accessibility and use of information sources among computer scientists and software engineers in Israel: Academy versus industry. *Journal of the American society for information science and technology*, 55(9):832–842, 07 2004. ISSN 1532-2882. doi: 10.1002/asi.20026. URL `https://doi.org/10.1002/asi.20026`.

YoungSeok Yoon and Brad A. Myers. A longitudinal study of programmers' backtracking. *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 101–108, 2014. doi: 10.1109/vlhcc.2014.6883030. URL `https://doi.org/10.1109/VLHCC.2014.6883030`.

YoungSeok Yoon, Brad A. Myers, and Sebon Koo. Visualization of fine-grained code change history. *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 119–126, 2013. doi: 10.1109/vlhcc.2013.6645254. URL `https://doi.org/10.1109/VLHCC.2013.6645254`.

Forrest W. Young and Carla M Bann. ViSta: The Visual Statistics System. Technical report, 94–1 (c), UNC LL Thurstone Psychometric Laboratory Research Memorandum, 1996.

Kang Zhang. *Visual Languages and Applications*. Springer Science & Business Media, 2013. ISBN 978-0-387-29813-9. doi: 10.1007/978-0-387-68257-0.

Andrew Zieffler, Joan Garfield, Shirley Alt, Danielle Dupuis, Kristine Holleque, and Beng Chang. What does research suggest about the teaching and learning of introductory statistics at the college level? A review of the literature. *Journal of Statistics Education*, 16(2), 2008. doi: 10.1080/10691898.2008.11889566. URL `https://doi.org/10.1080/10691898.2008.11889566`.

George Kingsley Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Ravenio Books, 2016. ISBN 978-1614273127.

Christopher Zita. Is Data Science Still a Rising Career in 2021, 2021. `https://tinyurl.com/mry9h9k7` (last accessed on May 16, 2021).

# Index

Typeset January 16, 2023