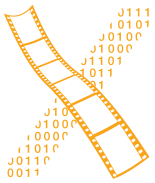


Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Cheno: Computing Datasets from Inference Statistics

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

by
Radu-Andrei Coandă

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Dr. Torsten Trimborn

Registration date: 16. January 2019
Submission date: 27 February 2019

Eidesstattliche Versicherung

Name, Vorname

Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Contents

List of Figures

List of Tables

Abstract

In an attempt to solve an underlying problem of inadequate statistical knowledge among researchers, Krishna et al. introduced StatPlayground, an exploratory learning tool, with the potential to help users improve certain statistical literacy skills, in particular, skills regarding the widely adopted Null Hypothesis Significance Tests (NHST). The tool allows users to control descriptive statistics of the data (e.g. mean, the variance of distributions) by directly manipulating visualizations (e.g., box plots) to see the effect on the resulting inferential statistics (e.g. p-value, effect size) and vice versa.

Augmenting the current tool with the computational functionality we introduce CHENO. As a statistical computations library, CHENO concerns itself with the forward pass (from datasets to descriptive statics to inferential statistics), but more importantly with the correct, swift and flexible computation of the inverse pass (from inferential statistics to descriptive statistics to datasets). CHENO achieves this by use of algebraic transformations of the NHST's equations. Furthermore, to permit the flexible search of datasets, the library allows the user to constrain and bound descriptive statics of the datasets.

As part of our evaluation, we then consider the validity of our produced results. We benchmark the computation times. And discuss the flexibility of the library in computing datasets from inferential statistics.

The developed system currently implements 9 significance tests but permits the simple implementation of other significance tests, by preserving the same interface. Such extensions to the tool will be pointed out at the end.

Überblick

In einem Versuch, ein zugrunde liegendes Problem unzureichenden statistischen Wissens unter Forschern zu lösen, haben Krishna et al. StatPlayground eingeführt, ein Exploratives Lernen Werkzeug, mit dem potentiellen Nutzer bestimmte statistische Fähigkeiten verbessern könnten, insbesondere in Bezug auf die weit verbreiteten Nullhypothese Significant Tests (NHST). Mit dem Tool können Benutzer deskriptive Statistiken der Daten (z. B. Mittelwert, die Varianz der Verteilungen) steuern, indem Visualisierungen (z. B. Box-Plots) direkt bearbeitet werden, um die Auswirkungen auf die resultierenden Folgerungsstatistiken (z. B. p-Wert, Effektgröße) und umgekehrt sehen zu können.

Wir führen CHENO ein, um das aktuelle Werkzeug um die Rechenfunktionalität zu erweitern. Als statistische berechnungs Bibliothek, erlaubt CHENO den Vorwärtsdurchlauf (von Datensätzen über beschreibende Statistiken bis hin zu Inferenzstatistiken), aber noch wichtiger ist die korrekte, schnelle und flexible Berechnung des Rückwärtsdurchgangs (von Inferenzstatistiken über beschreibende Statistiken zu Datensätzen). CHENO erreicht dies durch die Verwendung algebraischer Transformationen der NHST Gleichungen. Um eine flexible Suche nach Datensätzen zu erlauben, ermöglicht die Bibliothek den Benutzern außerdem die Einschränkung und Grenzen der beschreibenden Statik der Datensätzen.

Bei unserer Bewertung berücksichtigen wir die Gültigkeit unserer Ergebnisse. Wir vergleichen die Rechenzeiten and besprechen des Weiteren auch die Flexibilität der Bibliothek bei der Berechnung von Datensätzen aus Inferenzstatistiken.

Das entwickelte System implementiert derzeit 9 Signifikanztests, ermöglicht jedoch die einfache Implementierung anderer Signifikanztests, indem dieselbe Schnittstelle beibehalten wird. Solche Erweiterungen des Werkzeugs werden am Ende gezeigt.

Acknowledgements

I would firstly like to thank Krishna Subramanian, M.Sc. I am immensely grateful to him for his support and advice during the course of this Thesis.

I would like to thank, Dr. Torsten Trimborn, my second examiner, for his insightful comments that kick-started the finding of the solutions published in this work.

I would like to thank Prof. Dr. Jan Borchers, my thesis advisor, for his time and support.

Finally, I would like to thank my parents, and friends for their moral and critical support during the course of this Thesis.

Thank you!
Radu Coanda.

Conventions

Throughout this thesis we use the following conventions.

Text conventions

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

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

Definition:
Excursus

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

```
myClass
```

The whole thesis is written in British English.

Download links are set off in coloured boxes.

File: myFile[a]

[a] http://hci.rwth-aachen.de/public/folder/file_number.file

Chapter 1

Introduction

1.1 Statistical Literacy and StatPlayground

Statistical illiteracy

Statistical analysis plays an important role in many human-centered sciences such as HCI (Cairns [2007]), psychology and medicine (Chavalarias et al. [2016]), because it allows researchers to validate their hypothesis and communicate their results to the community. A commonly used method of statistical analysis, Null Hypothesis Significance Testing (NHST), has been widely criticized over the years for having several shortcomings (Cumming [2014]). An underlying cause of this problem is inadequate statistical literacy (Gal [2016]): "The problem is not that people use P-values poorly, it is that the vast majority of data analysis is not performed by people properly trained to perform data analysis"(Leek [2014]).

For addressing many of the problems at pedagogical level, researchers have identified frequent misunderstood statistical topics and proposed principles for improving teaching (Garfield and Ben-Zvi [2007]).

Learning statistical analysis through exploratory learning with StatPlayground

As a potential didactical augmentation solution for correcting problems with the practice of NHST, StatPlayground (Subramanian and Borchers [2017]) was proposed. Stat-

Playground adopts an exploratory learning approach (De Freitas and Oliver [2006]) towards the teaching of NHST methods, thus encouraging users through its design to discover by themselves relationships between existing background knowledge of statistics and unfamiliar content and concepts about NHST.

Definition: *Inferential Statistics*

INFERENTIAL STATISTICS:
In the context of NHST inferential statistics are statistics (i.e. numbers) drawn from statistics of the datasets (e.g. mean, variance, ranks), which assist in the decision whether to accept or reject different hypothesis.

StatPlayground consists of a dataset selection screen (see Figure 1.1) where the user can manipulate statistical properties of the datasets (i.e. mean, variance, sample size) by dragging sliders to reshape the data. Furthermore, the statistical properties of datasets can be locked (i.e. constrained) and/or bounds can be specified for them. In the top panel, options for the choice of experiment type are presented, allowing users to pick from an array of statistical tests. In the footer of the interface, the inference statistics resulting from data is presented.

Described by the authors, StatPlayground, should allow the following two functionalities:

1. Manipulation of statistical properties of the datasets to then see the effect on the resulting inferential statistics with regard to the selected statistical test.
2. The reverse operation, the manipulation of the inferential statistics to see datasets that could yield those inferential statistics with regard to the selected statistical test.

For the rest of this research work we will use the term forward pass to describe the first functionality and inverse pass to describe the second.

Forward Pass

The forward pass describes the functionality of comput-

Figure 1.1: Main Interface of StatPlayground

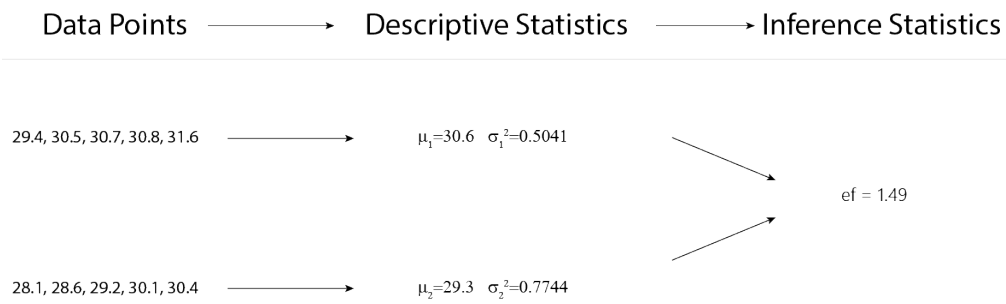
ing the inference statistics resulting from the user provided datasets. From a computational standpoint, this is done by simple algorithms that follow the steps for the selected statistical test (Field [2009], SciPy). From a mathematical standpoint, the forward pass can be described as a function of the data. Therefore for each combination of datasets there is only one resulting inference statistics.

We illustrate this in more detail in Figure 1.2. Suppose we have two datasets X_1 and X_2: $\{29.4, 30.5, 30.7, 30.8, 31.6\}$ and $\{28.1, 28.6, 29.2, 30.1, 30.4\}$. Suppose now that we can calculate a value, with which can gauge the difference between the two datasets, given by the inferential statistics d given by Equation 1.1, where μ_1 and μ_2 are the means (averages) of the two datasets and σ_1^2 and σ_2^2 are the variances (spread).

$$d = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}} \tag{1.1}$$

The calculation can be described as a two step process, where we first transform the data into descriptive statistics and then from descriptive statistics to inferential statistics.

Figure 1.2: Example of a Significance Test



1.2 Difficulties of the Inverse Pass

Definition:
Descriptive Statistics

DESCRIPTIVE STATISTICS:
Statistical tools which aid in quantitatively describing and summarizing of features of groups of data. Common descriptive statistics are mean, variance, sample size and frequency distribution.

As the name implies the inverse pass is the mathematical inverse of the function of the forward pass. The problem of computing the inverse pass can be divided in the following two parts:

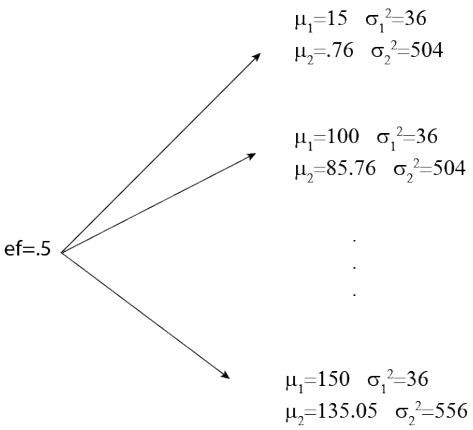
- **From inferential statistics to descriptive statistics:** The functions we use in the forward pass to calculate the inference statistics, are in part multivariate functions, therefore inverting them results in functions that map single values on a multi-dimensional

codomain. Our first obstacle is therefore finding reliably and efficiently as many and as varied of solutions of descriptive statistics that could thus yield a given inferential statistic. For the rest of the paper we will use the terminology of a **suitable statistics configuration** to denote a solution of the first part of the inverse pass problem.

- **From descriptive statistics to data:** Moreover, all functions used in the forward pass await as input descriptive statistics of the data, and as descriptive statistics are summarized properties of data taken as a group, different groups could lead to the same descriptive statistics and thus the same inference statistic. Our second obstacle is therefore finding sets of data with given descriptive statistics.

We illustrate the first part of the problem with the following example: suppose we start from an inference statistics d=0.5. Three possible solutions are displayed in Figure 1.3, but it is important to note that there are infinitely many solutions for this one value of the inference statistic d.

Figure 1.3: Motivation for the Difficulties of the Inverse Pass



Constraining parameters and parameter bounds

Furthermore, as the platform allows the user to constrain (fix) or/and bound certain values of descriptive statistics, the first part of the inverse pass would need to also accommodate such functionality, thus narrowing the search space.

Dependent variables	Two Datasets	Parametric	Student's dependent t-test
		Nonparametric	Wilcoxon Signed-Rank Test
	More than two datasets	Parametric	rANOVA
		Nonparametric	Friedman Analysis
Independent variables	Two Datasets	Parametric	Student's independent t-test
			Welch's t-test
		Nonparametric	Mann-Whitney-Wilcoxon W-test
	More than two datasets	Parametric	ANOVA
		Nonparametric	Kruskal-Wallis H-test

Table 1.1: Implemented Significance Tests.

So far, the development of StatPlayground's inverse pass has been confined to a simple algorithm as a proof of concept. The current implementation finds only neighbouring configurations of descriptive statistics, therefore the suitable statics configurations are relatively similar, and thus not permitting the user to "play-around" to explore the domain of possible values to visualize the impact of vastly different statistical configurations on the result.

1.3 Contributions of This Thesis

Cheno, out of respect, is an anagram of the word Cohen with reference to the statistician Jacob Cohen.

This research work aims to improve over the existing inverse pass functionality for the StatPlayground platform, by increasing the variability for the suitable statistical configurations. The resulting algorithms will be gathered in a library, entitled Cheno, which brings with it the following functionality:

System Requirements

- **Algorithms performing the inverse pass for 9 commonly used significance tests in HCI.** (see Table 1.1)

- **Close to real-time computation.** Finding a suitable statics configuration should not take more than the users uninterrupted flow of thought, therefore we require a maximum computation time of 1.0 seconds (Nielsen). This leads also to a more pleasing user experience.

- **Control over the choice of results being computed.** Fine-grained control over the computed suitable sta-

tistical configurations, by allowing constraining and bounding part of the resulting statistics. Moreover feedback about the space of possible values, should also be prompted to the user (i.e. values for the second mean exist in the range [lower bound, upper bound]).

1.4 Roadmap

We have organized the rest of this paper in the following way:

- **Chapter 2**, Foundation, will go over prerequisites for understanding the algorithms of the proposed library as well present a review of related work, that inspired us.
- **Chapter 3**, Development, will explain the algorithms behind each test, as well as obstacles encountered during their development .
- **Chapter 4**, Results, will present the benchmarks and accuracy test done on the algorithms together with their results.
- **Chapter 5**, Conclusion and Future Work, were we will point out improvements to the library as well as summarize the study.

Chapter 2

Foundation

In the previous chapter, we introduced the exploratory statistical learning tool, StatPlayground, and motivated the need for a statistical analysis library for performing the inverse pass. As the subject of this research work is a software engineering product which relies on statistical methods and algorithmical computations, we dedicate this chapter to prerequisites we expect as necessary for easier understanding of the material to follow, as well as to take a closer look at research work done for solving similar problems to ours.

2.1 Descriptive Statistics

To understand and analyse the data we are processing, we employ the use of statistical tools, called descriptive statistics, which describe and summarize properties of said data. In the following we will overview the basic descriptive statistics we are employing for our problem.

2.1.1 Mean, Variance and Standard Deviation

The *mean* is a measure for the center point or the most aver-

The mean of a dataset

age value in a set of data. Suppose we have a dataset with points $x_1, \dots x_n$. The mean of the dataset is

$$\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{2.1}$$

The variance of a dataset

The *variance* is a measure of the spread or dispersal of data around the mean. The higher the variance the further apart individual data points exist around the mean. Suppose we have a dataset with points $x_1, \dots x_n$ and mean μ. The variance of the dataset is

$$\sigma^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \mu)^2 \tag{2.2}$$

The standard deviation of a dataset

As the variance relies on squared units it is difficult to interpret the result, thus we take the square root of the variance, and obtain the *standard deviation*, σ.

2.1.2 Ranks

Ranking a set of values represents the transformation of data, in which each value is replaced by its order in the list after it has been already sorted. For example, for the sequence $3.4, 4.3, 2.7, 5$ we have the ranks $2, 3, 1, 4$.

2.1.3 Distribution

The distribution of a dataset, sometimes called, the *frequency distribution*, is a list of tuples for each value in our dataset and its respective count (i.e. the first entry in the tuple is given by a datapoint value, whereas the second represents the number of times, that value comes up in the dataset). Visualizing the list of pairs of values and counts gives us the graph of the distribution.

Some unique distributions, can be described by functions, which are often dependent only on a number of parameters (i.e. if we know the parameters we can describe the

underlying data), these are called *parametric distributions*. One such parametric distribution which is of interest to us is the normal distribution (often called Gaussian distribution), which is dependent on the mean and variance.

Normal Distribution

2.2 Inferential Statistics

Another branch of statistics, inferential statistics, leverages descriptive statistics and builds upon tools of assumption and reasoning to draw conclusion about a population (the entire group) from a randomly picked sample (smaller in comparison, a group randomly picked from the population).

The kinds of research questions one can answer with these tools of inferential statistics are of the following form:

- "is there a relationship between two (or more) properties of a group?"
- "is there a difference between two (or more) groups?"

We will focus for the remainder of this research work on the latter point.

Process of research

The act of researching whether or not a difference exists, could be sketched as a multistage process as pointed out by Field [2009]: A researcher would:

1. Generate a research question through initial observation and pose a theory explaining it.
2. Generate hypothesis: breaking your theory down into a set of testable predictions.
3. Collect data that could test your predictions.
4. Analyze the data.

In the second step of generating a testable hypothesis, researchers formulate the *alternative hypothesis* H_a (stating that the predicted effect will be present) and the *null hypothesis* H_0 (stating the opposite, that the effect is absent). The latter is the statement we test, as the methods of statistical inference are not able to prove the validity of our alternative hypothesis, but are able to reject the null hypothesis. It is important to note that even if we reject the null hypothesis, this still does not prove the alternative hypothesis - it just gives us more confidence in its validity.

2.2.1 Null Hypothesis Significance Test

Measure probability of a difference between datasets

One widely used method to aid researchers at rejecting or accepting the null hypothesis, H_0, is the *Null Hypothesis Significance Test* **(NHST)**. The method works by gauging the probability of a difference between two or more datasets. It does so, by computing the probability of the null hypothesis being true, where high probability values represent more confidence that the datasets are similar, whereas low probability value give us more confidence that they differ, thus allowing us to reject H_0.

2.2.2 Effect Size

Measure size of a difference between datasets

One other method for analyzing differences between groups is the use of *effect sizes*. If the NHST assists in the decision whether a difference exists between groups, the effect size informs on the magnitude of the difference. The advantage of utilizing effect sizes lies in the fact, that it positions the measurement of the size of the difference on a standardized scale, thus allowing us for example to compare the size of the effect of different experiments.

All the effect size used in this thesis are based on Tomczak and Tomczak [2014] work.

2.2.3 The Tests

Multiple NHST exist and their uses are based on the experimental setup. We discriminate between:

1. **Dependent and Independent tests** Dependent tests assume that the groups being tested are related, meaning that the subjects are the same across groups. In contrast independent tests, are applied when groups are independent from one another, meaning no subject appears more than in one group.
2. **Parametric and nonparametric tests** Parametric tests assume that the data being tested follows a known distribution (in our cases the normal "Gaussian" distribution) which can be described by a handful of parameters (i.e. mean, standard deviation and sample sizes). In contrast, the non parametric, although not as powerful as the parametric tests, don't make the assumption about the distribution of data. Instead they look at the order of values and are therefore based upon the descriptive statistics of ranks.
3. **Two groups and more than two groups tests**

We will look in the following paragraphs at the steps for computing each significance test of the 9 we have chosen. We note that we will not go into much detail regarding the theory behind the tests as this is outside the scope of this work and clarify only the method of each one, but we make the following sugestions: Field [2009].

From here on out, if not stated otherwise, we are going to use the following notation:

- X_i - denotes the i-th dataset, with entries $x_{i,1}, x_{i,2}, ...$
- p - denotes the p-value.
- ef - denotes the effect size.
- k - denotes the total number of groups.

- μ and μ_i - denotes the mean of the only group respectively of the i-th group.
- μ_{grand} - denotes the global mean: $\mu_{grand} = \frac{1}{Nk}\sum_{i=1}^{k} n_i\mu_i$
- σ and σ_i - denotes the standard deviation of the only group respectively of the i-th group.
- n and n_i - denotes the sample size of the only group respectively of the i-th group.
- $N := \sum_{i=1}^{k} n_i$ - denotes the total sample size.
- R_i with $i \in \{1, ...k\}$ - denotes the sum of the ranks belonging to group i, considering the values in all k groups together. (e.g. let $X_1 = \{11, 21, 43\}$ and $X_2 = \{21, 36, 53\}$ then the ranks associated with group 1, considering ties, are: $1, 1.5, 4$ [the sum is 6.5] and for group 2: $1.5, 3, 5$ [their sum is 9.5])

It can also be assumed that the descriptive statistics belong to each individual group.

Student's dependent *t*-test The test is applicable to depended datasets in experiments with two groups. The descriptive statistics computed (i.e. mean, standard deviation, sample size) belong to the dataset comprised of the differences between measurements of the participant across groups (i.e. let $X_1 = \{x_{1,1}, x_{1,2}, ...\}$ and $X_2 = \{x_{2,1}, x_{2,2}, ...\}$ then $D = X_1 - X_2 = \{x_{1,1} - x_{2,1}, x_{1,2} - x_{2,2}\}$).

t-statistics

$$t := \frac{\mu_D}{\sqrt{\frac{\sigma_D^2}{n}}} = T^{-1}(p|n-1) \tag{2.3}$$

The test statistics t follows a t-distribution [1], denoted by T. Plugging t in the aforementioned distribution, along with $n - 1$ yields the p-value.

[1] Student's t-distribution. (2019). En.wikipedia.org. Retrieved 19 January 2019, from https://en.wikipedia.org/wiki/Student%27s_t-distribution

Effect Size

$$ef = d := \frac{\mu}{\sigma} = \frac{t}{n} \tag{2.4}$$

Student's independent *t*-test This test, similar to the previous one works in experiments with two groups, though the datasets are supposed to be independent and the variances similar.

t-statistics

$$t := \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_{pooled}}{n_1} + \frac{\sigma_{pooled}}{n_2}}} = T^{-1}(p|n_1 + n_2 - 2) \tag{2.5}$$

Similarly to the previous test, T denotes the t-distribution, the test statitics t follows.

$$\text{where } \sigma_{pooled} = (n_1 - 1)\sigma_1^2 + (n_2 - 1)\sigma_2^2 \tag{2.6}$$

Effect Size

$$ef = d := \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}} = t\sqrt{\frac{1}{n_1} + \frac{1}{n_2}} \tag{2.7}$$

Welch's *t*-test This test, similar to the previous one works in experiments with two groups where the datasets are supposed to be independent, though the variances do not have to be similar. The distribution the test statistics t follows is again the t-distribution.

t-statistics

$$t := \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} = T^{-1}(p|df) \tag{2.8}$$

Welch–Satterthwaite equation

$$\text{where } df = \frac{(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2})^2}{\frac{\sigma_1^4}{n_1^2 \cdot (n_1 - 1)} + \frac{\sigma_2^4}{n_2^2 \cdot (n_2 - 1)}} \tag{2.9}$$

Effect Size

$$ef = d := \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}} \tag{2.10}$$

ANOVA This test is applicable in experiments with two or more groups, where the datasets are supposed to be independent and the variances similar.

$$G_{mean} := \frac{1}{N} \sum_{i=0}^{N} x_i = \frac{1}{k} \sum_{i=0}^{k} \mu_i \tag{2.11}$$

$$SS_M = \sum_{i=1}^{k} n_i \cdot (\mu_i - G_{mean})^2 \tag{2.12}$$

$$SS_R = \sum_{i=1}^{k} (n_i - 1) \cdot \sigma_i^2 \tag{2.13}$$

F-statistics

$$F := \frac{SS_M}{k-1} / \frac{SS_R}{N-k} = F^{-1}(p|k-1, N-k) \tag{2.14}$$

The F-statistics follows a F-distribution[2].

Effect Size

$$ef := eta^2 = \frac{SS_M}{SS_M + SS_R} \tag{2.15}$$

Repeared-measures ANOVA Similarly to the normal ANOVA, this test is applicable in experiments with two or more groups, though the datasets are supposed to be dependent. The assumption of similarity of variance, is adapted to suite the repeated-measures design (dependent datasets), so that the variances of the differences of pairwise different groups has to be similar.

[2]Snedecor's F distribution or the Fisher–Snedecor distribution. F-distribution. (2019). En.wikipedia.org. Retrieved 19 January 2019, from https://en.wikipedia.org/wiki/F-distribution

$$SS_W = \sum_{i=1}^{n}(n_i - 1) \cdot \sigma_i^2 \tag{2.16}$$

$$SS_M = \sum_{i=1}^{k} n_i \cdot (\mu_i - G_{mean})^2 \tag{2.17}$$

$$SS_R = SS_W - SS_M \tag{2.18}$$

F-statistics

$$F := \frac{\frac{SS_M}{k-1}}{\frac{SS_R}{(n-1)(k-1)}} = F^{-1}(p|k-1,(k-1)(n-1)) \tag{2.19}$$

The same effect size equation — Equation 2.15 — as with the normal ANOVA is used.

Mann-Whitney-Wilcoxon *U*/*W*-test This test is applicable to independent datasets in experiments with two groups, without any assumption of normality of the datasets.

Mann-Whitney Test Statistics

$$W := \begin{cases} min(R_1, R_2), & \text{if } n_1 = n_2. \\ R_1, & \text{if } n_1 < n_2. \\ R_2, & \text{otherwise.} \end{cases} \tag{2.20}$$

Z-score

$$z := \frac{W - \overline{W}}{SE_{\overline{W}}} = \mathcal{N}^{-1}(p|0,1) \tag{2.21}$$

$\mathcal{N}$ denotes the normal distribution.

$$\overline{W} = \frac{n_1 \cdot (n_1 + n_2 + 1)}{2} \tag{2.22}$$

$$SE_{\overline{W}} = \frac{n_1 n_2 \cdot (n_1 + n_2 + 1)}{12} \tag{2.23}$$

Effect Size

$$r = \eta = \frac{Z}{\sqrt{N}} \tag{2.24}$$

Wilcoxon Signed Rank Test Similarly to the previous test, this one is applicable to any datasets, without an assumption of normality about the datasets, though the datasets have to be dependent. As in the Student's dependent *t*-test, we first compute the difference of the measurements of a participant across the two groups. Following that, we compute the sum of ranks of the negative differences (R_-) and the sum of ranks of the positive differences (R_+). n denotes in this case the differences unequal 0.

Wilcoxon Test Statistics

$$T = min(R_-, R_+) \tag{2.25}$$

Z-score

$$z := \frac{T - \overline{T}}{SE_{\overline{T}}} = \mathcal{N}^{-1}(p|0,1) \tag{2.26}$$

$$\overline{T} = \frac{n \cdot (n+1)}{4} \tag{2.27}$$

$$SE_{\overline{T}} = \frac{n \cdot (n+1) \cdot (2n+1)}{24} \tag{2.28}$$

Kruskal-Wallis *H*-test This test is the equivalent of the Mann-Whitney *W*-test applied to two or more groups.

Kruskal-Wallis Test Statistics

$$H = \frac{12}{N \cdot (N-1)} \cdot \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3 \cdot (N-1) \tag{2.29}$$

The H-statistics follows a χ^2-distribution[3].

Effect Size

$$ef = \eta^2 := \frac{H - k + 1}{N - k} \tag{2.30}$$

Friedman Analysis This test is the equivalent of the Wilcoxon Signed Rank test applied to two or more groups.

Friedman Test Statistics

[3] Chi-squared distribution. (2019). En.wikipedia.org. Retrieved 21 January 2019, from https://en.wikipedia.org/wiki/Chi-squared_distribution

$$F_r = \frac{12}{N \cdot k \cdot (k+1)} \cdot \sum_{i=1}^{k} R_i^2 - 3 \cdot N \cdot (k+1) \tag{2.31}$$

Effect Size

$$W = \frac{\chi^2}{N \cdot (k-1)} \tag{2.32}$$

2.3 Related Work

2.3.1 Anscombe's Quartet

In 1973 Anscombe et al. [2007] called attention to the importance of plotting the data in the processes of statistical analysis. He explained how statistical quantities (e.g mean, variance, regression coefficient) are sometimes not enough to analyze the data, instead certain properties of the underlying data can only appreciated by visualizing it. With his now famous datasets and their respective graphs he pointed out that although the datasets had similar statistics, their graphs were unique and distinctive.

2.3.2 Computing Data from Descriptive Statistics

Building on Anscombe's work Chatterjee and Firat [2007] devised a randomized searching algorithm to automatically search for a new dataset starting from an initial one, while constraining certain statistics of the datasets to stay the same. With a different randomized searching algorithm Matejka and Fitzmaurice [2017] were able to generate target datasets which also had a unique (i.e. a circle, a t-rex sketch) and distinct graph. Similar searching algorithms are also employed in our implementation.

Chapter 3

Design and Development

"The theory of probabilities is at bottom nothing but common sense reduced to calculus."

—Laplace, *Théorie analytique des probabilités*, 1820

In the previous chapter, we briefly overview the prerequisites for the algorithms to come, as well as existing related work. In this chapter, we examine each algorithm for computing the inverted significance tests, discussing the approach and the implementation for each one.

The structure we planed for this chapter follows from the same idea presented in figure 1.2, namely that the significance test can be seen as a two stage process. The same thing can be said of the inverse of a significance test. Therefore we will structure this chapter in two major sections: the stage *From Inference Statistics to Descriptive Statistics* and *From Descriptive Statistics to Data*.

3.1 From Inference Statistics to Descriptive Statistics

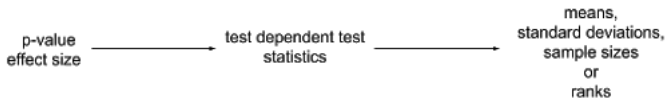
In the first stage of the algorithm we start from a pair of test statistics — **the *p*-value and the effect size** — and compute by means of *algebraic transformations* and *randomized sampling* a suitable statistics configuration (means, standard deviation and sample size, or ranks).

Definition: *Sampling*

SAMPLING:
Sampling is a process used in statistical analysis in which a predetermined number of observations are taken from a larger population.

One observation we made early on during the development process, was that the test specific test statistic (e.g. ***t***-value, ***F***-value, ***H***-value, ***W***-value, etc.) acts as a bridge between the input pair of test statistics and the suitable statistics configuration. This idea is represented in figure 3.1. Transforming the *p*-value and the effect size to this test statistics allows us to recover suitable statistics configurations from the input.

Figure 3.1: Workflow for the Inverse Pass of a Significnat Test



Sampling missing values

Another observation about the significance tests, that we noticed, is that there are considerably less equations than unknowns to simply (i.e. system of equations) be able to solve for the suitable statistics configuration. Therefore we use methods of randomized sampling of values, for the unknown descriptive statistics, from predefined ranges.

Constrained bounds: The predefined ranges for each variable are specified in a configuration file. The user is able to overwrite the values before when he starts a com-

putation. We use q^{lb} to denote the lower bound and q^{ub} to denote the upper bound for the descriptive statistics q.

Fixed variables: The user is also able to suggest values for unknown descriptive statistics. The algorithms take these values into account. If, thought, the user provided descriptive statistics don't fit the provided inferential statistics, we either raise an error, auditioning the user of the incompatibility between the provided descriptive statistics and the provided inferential statistics, or we don't take the provided value into consideration. This choice is specified by the user before the start of the computation.

In the following we will continue using the same notation introduced in Chapter 2.

3.1.1 Student's Dependent *t*-test

Inputs: p, ef, μ, σ, n
Outputs: μ, σ, n

The first unknown we solve for is the sample size, similarly to all the other parametric tests besides *Welch's t-test*, as it is the most constrained unknown of the 3 descriptive statistics. We achieve this, by constructing a loss function, dependent only on n, which we build with the help of equation 2.3 and 2.4.

LOSS FUNCTION:
is a function that maps a cost value to an input. The goal of many optimization problems is the minimization of said cost value.

Definition:
Loss Function

Multiplying equation 2.4 by the sample size, n, and together with 2.3 we get:

Loss function

$$\varphi(n) = T^{-1}(p|n-1) - ef \cdot n \tag{3.1}$$

Minimizing equation 3.1 gives us the optimal sample size, n^*.

To accustom the fact that, as the minimum of equation 3.1 might not be found in 0 and thus it would mean that either the p-value or the effect size would not have been produced by aforementioned n^*, we undertake a recalculation of the effect size as posed in equation 2.4 using t and n^*.

Moving on to the rest of the unknowns, μ and σ, we note that they are constrained by equation 2.4. We find the two values with the following algorithm 1.

Algorithm 1 Finding a suitable mean and standard deviation

Input d, μ, σ, lower and upper bound for the standard deviation σ_{lb}, σ_{ub}

Output suitable mean and standard deviation μ, σ

```
1: function FINDMEANSTD(d, μ, σ, σ_lb, σ_ub)
2:     if μ = NULL & σ = NULL then
3:         σ ← uniform(σ_lb, σ_ub)
4:         μ ← d · s
5:     if m ≠ NULL & s = NULL then
6:         σ ← |μ · d|
7:     if μ = NULL & σ ≠ NULL then
8:         μ ← d · σ
9:     return μ, σ
```

3.1.2 Student's Independent *t*-Test

Inputs: p, *ef*, μ, σ, n
Outputs: μ, σ, n

The first unknowns we solve for are the n_i with $i \in \{1, ...k\}$. For this we construct a loss function with the help of equation 2.6 and 2.7. In contrast to the previous significance test we now have to solve across pairs of values.

Loss Function

$$\varphi(ns_1, ns_2) = F^{-1}(p|n_1 + n_2 - 2) - \frac{d}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{3.2}$$

As n_1 and n_2 are integers, the same note regarding the effect size applies to the independent *t*-test similarly to the dependent t-test. We therefore recompute the effect size value with equation 2.7.

Finding the mean and standard deviation

We move onto the next stage, in finding values for the means and standard deviations. We have a total of 9 cases to cover depending on the number of constrained values the user provided.

1. If the user provided constrained variables are provided for both means and both standard deviations, we check them against equation 2.7, informing the user if they don't fit.
2. If user provided constrained variables are available for both means but not for both standard deviations, we use the means to solve for σ_{pooled}
   (a) If one of the two standard deviations is provided, we use the spooled equation to solve for the remaining standard deviation
   (b) If none of the two is provided we randomly sample with equal probability from a predefined range of standard deviation values and use that to solve for the second one
3. If user provided constrained variables are available for both standard deviations but not for both means, we use them to solve for the difference between the means, $\mu_1 - \mu_2$, which we in turn use to solve for the individual means:
   (a) If one of the two means is provided, we use it to solve for the remaining mean
   (b) If none of the two is provided we randomly sample with equal probability from a predefined range for mean values and use that to solve for the second one

4. If user provided constrained variables are available for one standard deviations, we will then randomly sample with equal probability from a predefined range of standard deviation values, use then both standard deviations to compute σ_{pooled} which then puts us back in case (3) (a) or (b).

5. If no user provided constraints are available for the standard deviations we randomly sample with equal probability from a predefined range of standard deviation values, which brings us back to case (4)

6. If one or none user provided constraints are available for the means:

   (a) We still turn to case (4) or (5) and find values for the standard deviations first.

   (b) We then recover the mean difference from equation 2.7

   (c) Either we compute the missing mean from the mean difference if the user has provided one constrained variable for the mean or we randomly sample one value for the first mean from a predefined range and use it to solve for the second mean

It is important to note that one critical assumption of Student's independent t-test is the homoscedasticity of the variances. We ensure this by either checking the validity of the user provided constraints with the f-test[1] of equality of variances, when both standard deviations are provided, or use the same test to adjust the bounds on the predefined range of values for the standard deviation when randomly sampling them.

3.1.3 Welch's *t*-Test

Inputs: p, ef, μ, σ, n
Outputs: μ, σ, n

[1] F-test. (2019). En.wikipedia.org. Retrieved 21 January 2019, from https://en.wikipedia.org/wiki/F-test

Similarly to Student's independent t-test, we first build our loss function, though in this case the function is dependent on more than only the sample sizes. We use equations 2.8 and 2.10 and construct equation 3.3.

Difficulties in computing the sample sizes for Welch's *t*-test

$$\varphi(n_1, n_2, \sigma_1, \sigma_2) = T^{-1}(p|df) - \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2+\sigma_2^2}{2}}} \tag{3.3}$$

Our approach to address this issue is to search for a combination of sample sizes and standard deviations that best fit equations 2.8 and 2.10.

Algorithm 2 Finding suitable sample sizes and standard deviations

Input μ, n, G_μ
Output suitable means μ

function FINDCOMB(μ, σ)
  $n_1^r, n_2^r, \sigma_1^r, \sigma_2^r \leftarrow$ Empty Lists
  for i in range(0, RUNS) **do**
    $n_1^r \leftarrow n_1^r$ append $n_1^{lb} + i \cdot \frac{(n_1^{ub} - n_1^{lb})}{RUNS}$
    $n_2^r \leftarrow n_2^r$ append $n_2^{lb} + i \cdot \frac{(n_2^{ub} - n_2^{lb})}{RUNS}$
    $\sigma_1^r \leftarrow \sigma_1^r$ append $\sigma_1^{lb} + i \cdot \frac{(\sigma_1^{ub} - \sigma_1^{lb})}{RUNS}$
    $\sigma_2^r \leftarrow \sigma_2^r$ append $\sigma_2^{lb} + i \cdot \frac{(\sigma_2^{ub} - \sigma_2^{lb})}{RUNS}$
  S $\leftarrow n_1^r \times n_2^r \times s_1^r \times s_2^r$
  return $\mathrm{argmin}_{i \in S}\, \varphi(i)$

Moving onto finding the means, we recover the mean difference from equation 2.8, then take the same approach as with Student's independent t-test for cases (6) and (7) point (c).

3.1.4 ANOVA

Inputs: p, ef, μ, σ, n, G_{mean}
Outputs: μ, σ, n

As with the previous tests, we try solve for the most constrained variables first. Therefore we utilize equation 2.14 and 2.15 to construct our loss function. We do this by transforming both equations by solving in both for $\frac{SS_M}{SS_R}$. Taking their difference results in equation 3.4.

Loss function

$$\varphi(N) = f \cdot \frac{k-1}{N-k} - \frac{ef}{1-ef} \tag{3.4}$$

Knowing N, we move onto figuring out a configuration of sample sizes, n. To solve this we either:

1. Check the user provided constrained variables for the sample sizes to sum up to N, if all of them are provided
2. If less than k user provided constrained variables are provided, we check to see whether their sum is less than N, then move onto case (3).
3. For the remaining l variables we assign the value $n^* = \frac{N^*}{l}$, where N^* results from subtracting the user provided constrained variables from the total sample size, N.

Finding means and standard deviations

We move onto the next stage, in finding values for the standard deviations:

1. We first check whether the user provided all constrained variables for the means. If so we:
   (a) compute the global mean, regardless for its previous value.
   (b) employ equation 2.13 to compute SS_M and then utilize the result as well as equation 2.15 to solve for SS_R.
   (c) calculate the standard deviations sequentially utilizing equation 2.13. We do this by computing a factor $c = \frac{rest}{k^*}$ where rest represents the difference between SS_R and all the known components $(n_i - 1)\sigma_i^2$ and k^* represents the number of known components.

   (d) we assign the value of *c* multiplied by a random value in the range .9 to 1.1, to the first unknown standard deviation.

2. We do steps (c) to (d) until we have assigned value to all standard deviations.

3. If the user has not provided all constrained variables for the means, carry out step repeatedly (1) (d), where c is in this case is the mean over the known standard deviations. If no user constrained variable was provided for the standard deviations we randomly sample with equal probability from a predefined range.

We move onto the next stage, in finding values for the means:

1. We compute SS_R from the afore computed standard deviations utilizing equation 2.13. Employing equation 2.15 we solve for SS_M.

2. If more than two user constrained variables were not provided we generate the rest in a similar fashion to step (1) (c) to (e) for the standard deviations. We compute a factor $c = \frac{rest}{k^*}$, where rest represents the difference between $G_{global} \cdot N$ and all the known components $n_i \cdot \mu_i$ and k^* represents the number of known components.

3. As we have two equations (2.12 and equation 2.11) to constrain the means, we utilize a nonlinear equation solver to solve for the remaining means, if there are only two means missing.

4. If it is the case, though, and we have all but one mean missing, we utilize equation 2.11 to solve for the remaining variable, and then test the afore computed mean on correctness if it fits equation 2.12.

To summarize the algorithm:

1. We start by making use of the equations of the significance test and effect size to find the f-value that fits

both well, by cycling through multiple values for the total sample size, N, thus finding also a solution for N.

2. We use the afore computed total sample size, N, to check the user provided constrained variables for the sample sizes or we randomly sample suitable values.

3. We check the user provided constrained variables for the standard deviations or we randomly sample suitable values.

4. We check the user provided constrained variables for the means or we randomly sample suitable values.

3.1.5 rANOVA

Inputs: p, ef, μ, σ, n, G_{mean}
Outputs: μ, σ, n

The first unknown we solve for is the sample size, as in this case it is the most constrained unknown of the 3 descriptive statistics. We achieve this, by constructing a loss function, dependent only on n, which we build with the help of equation 2.19 and 2.15.

By multiplying equation 2.19 with $\frac{1}{n-1}$ we get:

$$f* := \frac{f}{n-1} = \frac{SS_M}{SS_R} = \frac{F^{-1}(p|k-1,(n-1)(k-1))}{n-1} \quad (3.5)$$

Solving equation 2.15 for $\frac{SS_M}{SS_R}$ we get:

$$\frac{SS_M}{SS_R} = \frac{ef^2}{1-ef^2} = f* \quad (3.6)$$

Bringing them together results in our loss function 3.7. Minimizing it results in a suitable sample size value.

Loss function

$$\phi(n) = \frac{f}{n-1} - \frac{ef^2}{1-ef^2} \tag{3.7}$$

We proceed on finding the second unknown, namely the list μ. As there is only one equation constraining the means, equation 2.11, for all the cases the user provides less than all but one mean value, we randomly sample (uniform probability) the remaining values.

Algorithm 3 Finding suitable means

Input μ, n, G_μ
Output suitable means μ

```
1: function FINDMEANS(μ, n, G_μ, lb, ub)
2:     for μ_j in μ do
3:         if μ_j = NULL then
4:             rest ← k · G_μ − Σ_{i=0,μ_i≠NULL}^{k} μ_i
5:             nulls ← # NULL-entries in μ
6:             factor ← rest/nulls
7:             if μ_j last entry in μ then
8:                 μ ← factor
9:             else
10:                μ ← factor · uniform(lb, ub)
11:    return μ
```

Our algorithm 3 fills the empty means in μ, while ensuring the constraints from equation 2.11.

Lastly we press on to finding the last unknown, the list σ containing the standard deviations of each group. Before doing so, we take a detour, and first find the standard deviations of each participant across groups, σ_{part}. To do so we employ equation 2.16, where we compute the missing value SS_W, by use of equations 2.18, 2.17 and 3.5.

Algorithm 4 Finding suitable standard deviations for the participants across groups

Input n, SS_W

Output suitable standard deviations for the participants σ_{part}

function FINDSTDSPART(n, SS_W)
  $\sigma_{part} \leftarrow$ List of size n with NULL entries
  for σ_j in σ_{part} **do**
    $rest \leftarrow k \cdot G_\mu - \sum_{i=0, mu_i \neq \text{NULL}}^{k} \mu_i$
    $nulls \leftarrow$ # NULL-entries in σ_{part}
    $factor \leftarrow rest/nulls$
    if μ last entry in μ **then**
      $\sigma_j \leftarrow factor$
    else
      $\sigma_j \leftarrow factor \cdot uniform(.5, 1.5)$
  return σ_{part}

In our analysis we have noticed that the mean of the standard deviations for the participants across each group is relatively similar to the mean of the standard deviations of each group, therefore we use this as an anchor point for figuring out the unknown σ.

3.1.6 Mann-Whitney *U*-Test

Inputs: *p, ef, n*
Outputs: *ranks*

The first unknown we solve for is the total sample size, N. We utilize equation 2.21 to transform the user provided p-value to a z-score and then equation 2.24 to transform aforementioned z-score to the total sample size value, N.

Knowing the sum of samples sizes, *N*, we either:

1. Check the user provided constrained variables for the sample sizes, if both are provided

2. Use one of the user provided constrained variables for the sample size to solve for the other

3. or we randomly sample from a uniform probability from a predefined range of values for the sample size

We compute $\overline{W}$ and $SE_{\overline{W}}$ via equations 2.22 and 2.23 and utilize them in equation 2.20 to finally figure out the test statistics *W*.

We move onto figuring out the ranks from our afore computed test statistics, *W*.

ARITHMETIC PROGRESSION:
A sequence of numbers, where each two consecutive numbers differ by a constant. For example 1, 2, 3, 4, 5... where consecutive numbers differ by a constant 1 or 3, 6, 9, 12, 15... where consecutive numbers differ by a constant 3.

Definition: *Arithmetic Progression*

The test statistics, *W*, can be described as the sum over a subset of the arithmetic progression defining the ranks. We draw the following conclusions from this observation:

- The test statistics, *W*, has the complement W^c. Together they sum up to $\frac{N \cdot (N+1)}{2}$.
- There is a fixed range of values W and W^c can take.

Finding the ranks

We can now formulate the problem of finding ranks from the test statistics, W. From an arithmetic progression, S, of constant 1, from 1 up to N, find subsets S_1 and S_2 of size n_1 and n_2, where $S_1 = W$ and $S_2 = W^c$ if $n_1 < n_2$ or $W < W^c$ else $\sum S_1 = W^c$ and $\sum S_2 = W$ and $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \varnothing$.

We implement a simple randomized searching algorithm:

1. It starts off with a list, S_{total}, with values from 1 to N.

2. Without loss of generality it samples n_1 pairwise independent values from S_{total}, resulting in a list S. Let W^* be the sum over the values of S.
3. It then reduces the distance between W and W^*, by iteratively improving our W^*:
   (a) If $W^* < W$, randomly pick a value, v, in S, and increment it by 1 resulting in v^*, such that v^* is pairwise independent from all the values in S and less than N. It then goes back to step (3).
   (b) If $W^* > W$, randomly pick a value, v, in S, and decrements it by 1 resulting in v^*, such that v^* is pairwise independent from all the values in S and greater than 1. It then goes back to step (3).
   (c) If $W^* = W$, it stops the algorithm.
4. The algorithm returns a list containing S and S^c, where $S^c = \{1, 2, ...N\} \backslash S$

It is important to note, that as in step (3) of our algorithm we employ randomized searching method to find solutions to our problem, we don't expect to always find a close approximation to a solution. We try increase our chances by doing multiple runs of the entire algorithm.

To summarize the algorithm:

1. We compute the *z*-score from the *p*-value, and the sum of sample sizes, *N*, from the aforementioned *z*-score and the effect size, *r*.
2. From the sum of sample sizes we compute a suitable combination of samples sizes, n_1 and n_2 either from the user provided input or we randomly sample values.
3. We compute $\overline{W}$ and $SE_{\overline{W}}$ from the sample sizes, and thus the test statistics, *W*.
4. We utilize *W*, and the sample sizes to randomly search for independent subsets S_1 and S_2 of the arithmetic progression of constant 1, from 1 to *N*, that sum up to *W* and W^c respectively.

3.1.7 Wilcoxon Signed Rank Test

The algorithm for the Wilcoxon signed rank test is similar to the Mann-Whitney *U/W*-test in that we also utilize equation 2.21 to transform the user provided *p*-value to a *z*-score and then equation 2.24 to transform aforementioned *z*-score to the total sample size value, *N*. In this case, though, as this significance test assumes dependence of the datasets, we find the sample size, *n*, for both datasets, by simply dividing *N* by two. We check this value against the user provided one, if the user has provided any sample size, else we continue with our resulting value.

Then we compute the test statistic, T, with equation 2.25 and the value of $\overline{T}$ and $SE_{\overline{T}}$, which in turn we compute by utilizing equation 2.27 and equation 2.28.

We move onto figuring out the ranks from our afore computed test statistics, T.

In section 2.2.3 we observed, that the test statistics, T, results from the smallest sum of either the negative ranks or the positive ones. From this observation we conclude the following:

- Again we need to generate two lists of ranks, S_1 and S_2, where the sizes n_1 and n_2, respectively and $n_{total} = n_1 + n_2$.
- Without loss of generality we assume that $n_1 < n_2$. The test statistics, T, has then to be greater or equal than the sum of the first n_1 elements of the arithmetic progression of constant 1, from 1 to n.
- Without loss of generality we assume that $n_1 < n_2$. The test statistics, T, has to be less than or equal to the the sum of the last n_2 elements of the arithmetic progression of constant 1, from 1 to n.

To find the two lists of ranks, S_1 and S_2, we first need to find the number of ranks, n_1 and n_2, that go into each of them them. To do so we cycle through values from 0 to n,

and check if the test statistics, T, respects inequality 3.8 we obtain from conclusions (2) and (3), where i, represents the smaller value between n_1 and n_2.

$$i(i+1) \leq T \leq i(n+(n-i+1)) \cdot \frac{1}{2} \tag{3.8}$$

If we find values that respect inequality 3.8 we can apply the randomized searching algorithm from section 3.1.6 to find a solution of lists of ranks.

To summarize the algorithm:

1. We compute the z-score from the p-value, and the sum of sample sizes, N, from the aforementioned z-score and the effect size, r.
2. From the sum of sample sizes, N, we determine a suitable samples size, n, which we compare against the user provided value, if the user has provided any value for the sample size.
3. We compute $\overline{T}$ and $SE_{\overline{T}}$ from the sample sizes, and thus the test statistics, T.
4. We utilize T and the sample size, n, to find a suitable combination of sizes, n_1 and n_2, for rank lists, S_1 and S_2.
5. We employ the randomized searching algorithm, discussed for the Mann-Whitney algorithm, to find independent subsets S_1 and S_2 of the arithmetic progression of constant 1, from 1 to N, that sum up to T and T^c respectively.

3.1.8 Kruskal-Wallis *H*-test

As with the previous algorithms for nonparametric significance tests, we try to first solve for the test dependent test statistics, H, then use it to search for a configuration of

ranks, which fits that test statistics. We find H, by employing equation 2.29.

We then solve for the total sample size, N, in equation 2.4.3.3. Knowing N, we move onto figuring out a configuration of sample sizes, n. To solve this we either:

1. Check the user provided constrained variables for the sample sizes to sum up to N, if all of them are given
2. If less than k user provided constrained variables are given, we control to see whether their sum is less than N, then move onto case (3)
3. For the remaining, l, variables we assign the value $n^* = \frac{N^*}{l}$, where N^* results from subtracting the user provided constrained variables from the total sample size, N

We move onto figuring out the ranks from our afore computed test statistics, H.

We noticed in section 2.4.3 that only equation 2.4.3.1 imposes a direct constraint on the distribution of the ranks between the k groups. We can now formulate the problem as an optimization problem:

PARTITION OF A SET:
Is a grouping of the elements of a set into non-empty subsets, such that every element belongs to only one subset. A k-partition denotes a partitioning into k subsets.

Definition:
Partition of a set

Finding ranks

Let S denote an arithmetic progression of constant 1 from 1 to N, where N is a positive integer, and H is a positive real constant. Find a k-partition of S, such that:

1. $H - y^*$ is minimal
2. Let $y^* = \frac{12}{N \cdot (N-1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3 \cdot (N+1)$, where R_i is the sum over the elements in the i-th partition

Our inclination was to adapt the algorithm previously proposed in section 3.1.6, though we needed a way to reproduce stage (3) of the algorithm, where we took the step in the direction of the solution.

1. The first issue had to do with the fact, that we were working with more than two groups. To address this, we thought of generating an initial partitioning, S, and work directly on the elements of the individual subsets, exchanging between subsets.

2. The second issue we encountered had to do with the choice of which elements to exchange. To address this, we select the group with the smallest or the greatest weighted sum $\frac{R_i}{n_i}$ where $i \in \{1, ...k\}$ depending on whether the difference $H - y^*$ is positive or negative, which acts as the sender. We select another group at random from the remaining groups, which acts as the receiver. For the exchange we select one element, x_{sender} in the sender's group and one element in the receiver group, the closest to, but less than x_{sender}.

We propose the following simple randomized search algorithm:

1. We initialize a list of lists, S, with a random partitioning of the aforementioned arithmetic progression

2. We iterate e times in which we:

   (a) First compute $y*$

   (b) If $H - y^*$is:

      i. Positive, we select the group with the smallest $\frac{R_i}{n_i}$, $i \in \{1, ...k\}$ as the sender and from the remaining groups we randomly sample one other group for the receiver

      ii. Negative, we select the group with the greatest $\frac{R_i}{n_i}$, $i \in \{1, ...k\}$ as the sender and from the remaining groups we randomly sample one other group for the receiver

iii. Zero, we have found a partitioning that fits our equation, therefore we stop.

(c) We proceed with the swap of elements between sender and receiver and we attempt to exchange the smallest element of the sender, x_{sender}, with an element of the receiver, whose value is closest but less than x_{sender}. If we can't find any we proceed onto the next smallest element of the sender. Independent of whether we make the swap we continue onto the next iteration.

3. We return the partitioning, S, that minimized $H - y^*$

To summarize the algorithm:

1. We compute the H test statistics
2. We use the newly computed value of H to solve for the total sample size, N.
3. We find a combination of individual sample sizes, that sum up to N, by checking whether the user provided values for the sample sizes fit the the sum or we use a factor for the sample sizes the users has not provided
4. We proceed onto finding a k-partitioning of the ranks with our randomized search algorithm

3.1.9 Friedmann Analysis

Similarly to the Kruskal-Wallis algorithm in section 3.1.8, we transform the p-value to a F-value, utilizing equation 2.29 and we extract the sample size, n, from the p-value and the effect size, W, by means of equation 2.32. We check this value against the user provided one, if the user has provided any sample size, else we continue with our resulting value.

We move onto figuring out the ranks from our afore computed test statistics, F.

Definition: *Permutation*

PERMUTATION:
Represents on ordering of the elements of a sequence. To permute a sequence is the act of rearranging the elements of said sequence.

Similarly to the Kruskal-Wallis test, the only constraint imposed on the ranks is equation 2.4.4.1. As with the problem in the previous section this observation helps us to formulate the following optimization problem:

Finding ranks

Let X be a matrix of size $n \times k$ where each of the n-rows is a permutation of the arithmetic progression of constant 1 from 1 to k and F is a real constant. Find X such that:

1. $F - y^*$ is minimal
2. Let $y^* - H = \frac{12}{nk(k+1)} \sum_{i=1}^{k} R_i^2 - 3 \cdot n \cdot (k+1)$, where R_i is the sum over the elements in the i-th column of X.

We propose the following simple randomized search algorithm:

1. We initialize each row of X with the arithmetic progression of constant 1 from 1 to k.
2. We iteratively try to improve our solution:
   (a) We compute y^*
   (b) If $F - y^* > 0$ we randomly permute the $imodn$ row, where i is the current iteration.
   (c) If $F - y^* = 0$ we return X

   We return the X that minimized $F - y^*$

To summarize the algorithm:

1. We compute the F test statistics
2. We use the newly computed value of F to solve for the sample size, n.

3. We proceed onto finding a matrix X of ranks our randomized search algorithm

3.2 From Descriptive Statistics to Data

In the previous section we have discussed the first stage of the inverse pass for statistical significance tests and were able to compute approximations for the descriptive statistics of the data sets that could yield our target inferential statistics. In this section we are going to look at the second half of the inverse pass, and utilizing the afore computed descriptive statistics we examine one method for generating data for statistical test, from descriptive statistics, by means of sampling.

As the tests use different kinds of descriptive statistics we are going to differentiate between two methods for computing the data from descriptive statistics.

3.2.1 Parametric Distributions

One of the main assumptions of the parametric significance tests is that they follow a known distribution, which can be described by a fixed number of parameters. For our choice of significance tests, the distribution we expect from our data is the normal distribution, which is described by only two parameters — the mean and the standard deviation.

As we have already computed these descriptive statistics in the previous stage, we can use them to sample data points from a normal distribution described by the mentioned statistics.

Algorithm 5 Sample data for parametric distributions

Input k, μ, σ, n

Output *data* a list of list, where $data_i$ contains the data points for the i-th group

1: **function** SAMPLEPARAM(k, μ, σ, n)
2: $data \leftarrow$ Empty List
3: **for** i in $1, ...k$ **do**
4: $cur \leftarrow sampleNormal(\mu_i, \sigma_i, n_i)$
5: $cur \leftarrow adjustVariance(cur, \sigma_i)$
6: $cur \leftarrow adjustMean(cur, \mu_i)$
7: $data \leftarrow$ append cur to $data$
8: **return** $data$

Algorithm 5 describes the method we used. The subroutine $sampleNormal$ employs a sampling function of the *Numpy*-package[2]. As there is a low chance the generated data from $sampleNormal$ will have exactly the asked for descriptive statistics we undertake a correction to the standard deviation and the mean in subroutines $sampleVariance$ and $sampleMean$.

rANOVA: From Descriptive Statistics to Data

One special case of data generation for a parametrized significance test, is the repeated measures ANOVA, where not only the mean and the standard deviation of each individual group is specified, but also the standard deviation of each individual participant is of importance. Therefore the generation of data for this particular test, is subjected to one additional constrained on the standard deviation of each individual participant.

We start off by initializing the k groups, by using algorithm 5, according to the the k means ms and standard deviations

[2] numpy.random.normal — NumPy v1.15 Manual. (2019). Docs.scipy.org. Retrieved 25 January 2019, from https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.normal.html

ss of the groups we computed in the previous stage. Although the means and the standard deviations of the newly generated datasets match the ones computed in the previous stage it still does not mean that the standard deviations of the participants ss_{part} match.

Step 1: To measure the difference, $\delta \in \mathcal{R}^{n\times 1}$, between the current participant standard deviations $y_{ss_{part}}$ and and the target ss_{part} we construct the loss function in equation 3.9.

Loss function

$$\delta = \varphi(y_{ss_{part}}, ss_{part}) = (y_{ss_{part}} - ss_{part})^2 \tag{3.9}$$

Step 2: From the resulting vector δ, we pick the n_{max} greatest differences and the corresponding indices $i_1, ...i_{n_{max}}$, to correct, where $0 \leq n_{max} \leq n$.

Let X_j^* be the dataset of the $j-th$ group, where the indices $i_1, ...i_{n_{max}}$ have been removed.

Step 3: We compute the remaining standard deviation σ_j^* from the target mean ms_j with equation 3.10.

Step 4: Utilizing the afore computed standard deviation σ_j^* and the target mean ms_j along with the the sample size of n_{max} we generate, with algorithm 5, new values to replace the ones removed in X_j^*.

Remaining standard deviation

$$\sigma_j^* := \sqrt{\frac{1}{n - n_{max}} \sum_{t=0}^{n-n_{max}} (X_{j,t}^* - ms_j)^2} \tag{3.10}$$

If the datasets with the new values produce a *p*-value closet to our input *p*-value, we accept the new datasets as a better approximation to the previous one, replacing them. We repeate steps 1 through 4 for a predefined number of rounds, Continuesly trying to improve our solution.

3.2.2 Nonparametric Distributions

The same assumption of a known distribution for the underlying data does not apply to the nonparametric tests. For these significance tests we convert the ranks, previously computed, to data points, while still maintaining the same order of the data values. We leave it to the user to chose a distribution for the data which best suits him, offering currently the alternatives of a normal and a uniform distribution.

Algorithm 6 Sample data for nonparametric distributions

Input $k, N, n, ranks$ - list of list, where $ranks_i$ contains all the ranks belonging to i-th group

Output *data* a list of list, where $data_i$ contains the data points for the i-th group

```
1: function SAMPLENONPARAM(k, N, n, ranks)
2:     data ← Empty List
3:     s ← sample(N)          ▷ s is sorted in ascending order
4:     for i in 1, ...k do
5:         cur ← Empty List
6:         for j in 1, ...n_i do
7:             cur ← append s_{ranks_{i,j}} to cur
8:         data ← append cur to data
9:     return data
```

Algorithm 6 describes our method, where the subroutine $sample$ denotes one of the chosen distributions.

Chapter 4

Evaluation

In the previous chapter we discussed the algorithms we developed to compute the inverse pass for the significance tests. In this chapter, we discuss the evaluation we conducted on said algorithms to assess their performance with respect to their specification, as well as analyse the results for validity and speed.

4.1 System Requirements

We focused on implementing the following system requirements into our system.

SR1: Algorithms for the inverse of statistical tests that correctly generate valid data points from a pair of inference statistics. StatPlayground already has 9 commonly used NHST implemented, therefore we are adding functionality to them for computing the inverse of NHST.

SR2: Time Efficiency below 1 second.

SR3: Flexibility in generating data points. Computing the inverse of a NHST is not a one-to-one correlation between input and output values, as for multiple configurations of data points can yield the same pair of inferential statistics. Therefore, we are looking for a procedure to better segment the search space by specifying constraints or boundaries for the data we produce.

4.2 Structure of the Evaluation

Our method for testing the validity is as follows:

1. Start from a pair of inference statistics (p-value and effect size) which we use as input to the algorithm
2. Compute data sets that could yield aforementioned value pairs
3. Calculate the inferential statistics (p-value and effect size) for the afore computed generated data sets
4. Calculate the relative error between the two pairs of values from step (3) and the original inference statistics from step (1)

The data we used for input (i.e. pairs of inference statistics) is split into two categories:

- Pairs of values scrapped from different papers, articles and other publications
- Generated values pair

In benchmarking the algorithms we measure only the time it took to compute a set of suitable data points from a pair of inferential statistics. We run a total of 1000 different inferential statistics pairs for each algorithm. The results can be visualized in table 4.2.

System specifications

All benchmarks were run on a *Lenovo Y50-70, 12Gb of RAM,*

Intel Core i7-4700HQ CPU @ 2.40GHz 3 0:1 3400,00 MHz, Nvidia GTX 860m GPU and an ATA Kingston SUV400S 256Gb SSD.

4.3 Results

This section presents the results of the tests.

Test	Relative Error from Data		
	Minimum (s)	Average (s)	Maximum (s)
Student Dependendent	0.00E+00	1.50E-03	3.30E-01
Student Independent	0.00E+00	1.23E-03	2.91E-01
Welch's	2.10E-16	3.12E+00	8.29E+02
ANOVA	0.00E+00	9.75E-02	1.34E+00
rANOVA	5.09E-07	3.57E-03	7.07E-02
Mann Whitney	7.60E-04	6.30E-03	1.39E-01
Wilcoxon	0.00E+00	0.00E+00	0.00E+00
Kruskal Wallis	1.18E-04	1.49E-01	1.39E+00
Friedman	0.00E+00	1.90E-04	1.67E-03

Table 4.1: Relative Error of the Algorithms

Test	Relative Error from Data		
	Minimum (s)	Average (s)	Maximum (s)
Student Dependendent	1.09E-02	1.26E-02	2.75E-02
Student Independent	9.81E-01	1.00E+00	1.16E+00
Welch's	5.25E-01	7.63E-01	1.87E+00
ANOVA	1.61E-02	8.83E-02	1.64E-01
rANOVA	6.27E-01	8.87E-01	1.95E+00
Mann Whitney	2.32E-04	2.18E-03	2.71E-02
Wilcoxon	2.16E-04	1.03E-02	1.03E-01
Kruskal Wallis	1.70E-01	3.49E-01	8.39E-01
Friedman	5.70E-02	2.08E-01	4.12E-01

Table 4.2: Runtime of the Algorithms

4.4 Discussion

This section discusses the data analysis process and elaborates the findings made in detail.

SR1: Algorithms for the inverse of statistical tests, that correctly generate valid data points from a pair of inference statistics. Not surprisingly the tests with more equations to constrain them, have a greater accuracy in generating data, with an average relative error of about 0.1%. Interestingly, even though we only used basic algebraic operations like addition and multiplication, we still have a relative error of 0.1% instead of a lower one, which is due to machine error.

SR2: Time Efficiency below 1 second. We can observe that all algorithms achieve the 1 second runtime Nielsen. Of interest might be the fact that the less constrained tests (i.e. ANOVA as well as the nonparametric tests) achieved on average lower benchmark times. At times even with an order of magnitude lower than the more constrained tests.

SR3: Flexibilty The flexibility of the library is offered through the use of constraints and bounds on the target descriptive statistics. Although generating a specific dataset, on the data level, is not possible with the current state of the library, on the descriptive statics level is. Therefore the more constraints the user specifies the better the approximated result gets.

Chapter 5

Summary and Future Work

In the previous chapter, the evaluation of the system was conducted to validate the system requirements, followed by the analysis and the resultant research findings were discussed. In this chapter, we summarize the thesis upto this point and potential future works, such as additional features that can be added to enhance Cheno, are discussed.

5.1 Summary

In the first chapter, we motivated the need for Cheno as an extension for the existing StatPlayground project (Subramanian and Borchers [2017]) and we introduced the topic of the inverse pass in the context of statistical significance tests — computing datasets from inference statistics. Then we tried to analyze the difficulties of computing the aforementioned inverse and ended the chapter by overviewing the proposed system requirements of the system: accuracy, speed and flexibility.

In the second chapter, we reviewed basic statistical concepts like descriptive statistics and hypothesis tests. Following that we discussed existing research related to

Cheno.

In the third chapter, we discussed the design and development of the algorithms for the various tests supported by Cheno. We presented the first part of the inverse pass — from inferential statistics to descriptive statistics — by describing the inputs and outputs of each algorithm and then elaborating on the method each algorithm takes to compute said descriptive statistics.

In this chapter we also discussed the second part of the inverse pass — from descriptive statistics to datasets — by briefly describing one method to generate data from descriptive statistics.

In the fourth chapter, we enumerated the system requirements and discussed the structure of the evaluation for testing aforementioned requirements. We presented afterwards the results of the evaluation and discussed them. We found out that Cheno respects the target bounds for all but one test in terms of accuracy. All the tests perform on average in the proposed target speed bounds, with only two tests violating in the worst case the proposed time of 1 second.

5.2 Future Work

One of the main features left out of the final implementation is a proper system for informing the user if the solutions he is looking for are even possible given his constraints and bounds. This could be from a user perspective something of interest, to better understand for which configuration of descriptive statistics are the restulting inference statistics realistic.

One a further note, as Cheno was intended as an extension for StatPlayground, Cheno had only those tests developed that were required by StatPlayground. Additional tests can be developed based on the same design.

Some of the algorithms were designed for parallel comput-

ing, lowering the speed by a considerable factor (i.e. in the case of Welch's t-test by a factor of 4). Writing the algorithms with parallel computing in mind, could lower the computing time even more, though, given the GIL [1] of our implementation of python, the multiprocessing can be further improved by changing to a more performant multiprocessing programming language. Furthermore, the current implementation was developed in Python, which doesn't compare in terms of performance to compiled languages such as C++ or Java. Implementing Cheno in a compiled language could also boost performance considerably.

[1] Global interpreter lock. (2019). En.wikipedia.org. Retrieved 15 February 2019, from https://en.wikipedia.org/wiki/Global_interpreter_lock

Appendix A

Implementation

Cheno was developed with `Python 3.7`. The numerical computation were done with the python packages: `numpy 1.15`[1] and `scipy 1.1`[2].

The software design is modular, with each significance test falling into its own class, with the same public interface shared across all the 9 classes.

The two public methods are `forward` and `solve`, whereas the former computes the normal forward pass, the latter computes the inverse pass of the specific significance test.

[1] http://www.numpy.org/
[2] https://www.scipy.org/

Bibliography

F J Anscombe, The American Statistician, and No Feb. Anscombe1973. 27(1):1–6, 2007. ISSN 15372731. doi: 10.1007/978-3-540-71915-1_35. URL papers3://publication/uuid/40F5CDD9-AEF9-4ACE-B982-4004DEC844B8.

Paul Cairns. HCI... not as it should be: inferential statistics in HCI research. *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 1*, pages 195–201, 2007. URL http://dl.acm.org/citation.cfm?id=1531294.1531321.

Sangit Chatterjee and Aykut Firat. Generating data with identical statistics but dissimilar graphics: A follow up to the anscombe dataset. *American Statistician*, 61 (3):248–254, 2007. ISSN 00031305. doi: 10.1198/000313007X220057.

D Chavalarias, J Wallach, A Li, and JA Ioannidis. Evolution of reporting p values in the biomedical literature, 1990-2015. *JAMA*, 315(11):1141–1148, 2016. doi: 10.1001/jama.2016.1952. URL +http://dx.doi.org/10.1001/jama.2016.1952.

Geoff Cumming. The New Statistics: Why and How. *Psychological Science*, 25(1):7–29, 2014. ISSN 14679280. doi: 10.1177/0956797613504966.

Sara De Freitas and Martin Oliver. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated? *Computers and Education*, 46(3):249–264, 2006. ISSN 03601315. doi: 10.1016/j.compedu.2005.11.007.

Andy Field. *Discovering Statistics using SPSS*. 2009. ISBN 9781847879066.

Iddo Gal. Adults ' Statistical Literacy : Meanings , Components , Responsibilities. 70(1):1–25, 2016.

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. ISSN 03067734. doi: 10.1111/j.1751-5823.2007.00029.x.

Jeff Leek. On the scalability of statistical procedures: why the p-value bashers just don't get it. · simply statistics. https://simplystatistics.org/2014/02/14/on-the-scalability-of-statistical-procedures-why-the-p-value-bashers-just-dont-get-it/, 02 2014. (Accessed on 12/05/2018).

Justin Matejka and George Fitzmaurice. Same Stats, Different Graphs. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, (August):1290–1294, 2017. doi: 10.1145/3025453.3025912. URL http://dl.acm.org/citation.cfm?doid=3025453.3025912.

Jakob Nielsen. Response time limits: Article by jakob nielsen. https://www.nngroup.com/articles/response-times-3-important-limits/. (Accessed on 12/05/2018).

SciPy. Statistical functions (scipy.stats) — scipy v1.1.0 reference guide. https://docs.scipy.org/doc/scipy/reference/stats.html. (Accessed on 12/05/2018).

Krishna Subramanian and Jan Borchers. StatPlayground: Exploring Statistics Through Visualizations. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 401–404, 2017. doi: 10.1145/3027063.3052970. URL http://doi.acm.org/10.1145/3027063.3052970.

Maciej Tomczak and Ewa Tomczak. The need to report effect size estimates revisited. An overview of some recommended measures of effect size. Technical Report 21, 2014.

Index

Typeset February 25, 2019