Karlsruhe Institute of Technology

Department of Economics and Management
Institute of Operations Research
Chair of Analytics and Statistics
Prof. Dr. Oliver Grothe

Concerning Levels of Significance – Analysing Methods to Detect and Estimate P-Hacking

Bachelor's Thesis

Philip Müller
1981302

25th March 2021

Advisor: Fabian Kächele

Abstract

Results from empirical analyses can be forged with ease. Currently, such analyses are based on the methodology of Null Hypothesis Significance Testing. Too often, conclusions are drawn solely from the level of significance, also called the p-value. Scientists express concern about the ease at which the p-value can be manipulated to support any result, conclusion and thus narrative. This practice, called p-hacking leads to an inflation of false findings in the scientific literature. In the long run, nothing less than the credibility of entire scientific domains is at stake. The debate on how to properly adapt the scientific enterprise to the problem at hand has been going on for about a century now. This thesis attempts to develop methods to detect p-hacking, and to estimate the extent to which single p-hacking techniques can distort significance levels. The simulations presented indicate that any finding can be labelled as statistically significant. Using real world data, it will furthermore be demonstrated, how the current practise of empirical analysis and communicating findings allows to almost fully disguise p-hacking. These results can help to identify the potential of individual proposed solutions, which are also discussed here. Furthermore, the results underline the urgency for the debate on solutions to reach consensus soon.

Contents

List of Figures

List of Tables

1 Introduction

In March 2016, the American Statistical Association issued a "Statement on Statistical Significance and P-Values", in which p-hacking – a practice best described as "the search for small p-values" – is being observed with increasing concern (cf. Wasserstein, 2016). P-hacking methods can be as simple as stating statistically significant relationships, without worrying much about causal interpretation. For instance, the correlation between human birth rates and the population of stork pairs in 17 European countries turns out to be statistically significant (cf. Matthews, 2000). In other cases, p-hacking can go as far as actively manipulating experimental data to derive at statistical significance. Diederik Stapel admitted to committing data fraud in 55 papers, prompting suspension from his position as lecturer at Tilburg University (cf. Enserink, 2012). Investigators, tasked with analysing the extent of data fraud in the case of Stapel, identified the root cause to be a research culture in which "some scientists don't understand the essentials of statistics, ... leave unwelcome data out of their papers and even the most prestigious journals print results that are obviously too good to be true" (Enserink, 2012). Head et al. (2015) conclude that p-hacking is indeed a widespread phenomenon in sciences.

The consequence of p-hacking is an increasing number of false positives (e.g., falsely rejecting the null hypothesis) in the scientific literature. The publication bias – the tendency to publish positives over null findings – only adds to the extent of the problem. The debate over solutions is in full swing.

Meanwhile, papers are still being published. This thesis investigates the possibility of detecting p-hacked findings in published papers. Being able to detect p-hacking in the current publishing process, would partially solve the problem at hand. If it turns out that detection is not possible, solutions should be based on a proper understanding of p-hacking, which this thesis is aiming to provide. Furthermore, the aim is to develop estimates of the extent to which single p-hacking methods produce false findings.

The subsequent thesis is organised as follows: Chapter 2 aims to provide a theoretical background. An introduction to Null Hypothesis Significance Testing will be given, with emphasis on historical origins (chapter 2.1). This methodology represents a basis for p-hacking, which will be explained in detail in chapter 2.2. chapter 3 contains analyses. The three main p-hacking methods will be analysed, referred to here as flexibility in sample size (chapter 3.1), flexibility in variables (chapter 3.2), and hypothesising after the results are known (HARKing, chapter 3.3). Particular emphasis will be laid on detecting and estimating the analysed p-hacking techniques. Chapter 3.4 aims to

demonstrate p-hacking on real-world data. Those p-hacking methods not possible to analyse on simulated data will be introduced here. Findings will be discussed in chapter 4, before deriving at a final conclusion in chapter 5.

2 Theoretical Background

2.1 Null Hypothesis Significance Testing

Statistical tests differ from one another with regards to their design and application purpose. However, they share some key features. A statistical test is used to study a specific hypothesis regarding a population. To either verify or falsify the hypothesis, a sample is drawn from the population under study. A sample function then consolidates the information which can be extracted from the sample itself. Based on the sample function, one can decide whether to reject or confirm the hypothesis (cf. Mittag, 2017, p.227).

There are multiple ways to set up the hypothesis. A commonly accepted approach is to set up a pair of two hypotheses, namely the null (H_0) and alternative hypothesis (H_1). Combined, they cover the entire sample space and are mutually exclusive. Every possible observation in the sample can then be associated to either one of the two hypotheses. One can never truly confirm a hypothesis. For this reason, researchers rather try to reject the null hypothesis, in favour of the alternative hypothesis. In that case, the alternative hypothesis contains what is to be shown. Suppose one wants to study the effect of incentives on work performance. In treatment A, probands were given a work task and a standard pay-off. Work performance was measured. In treatment B, probands were offered a non-monetary pay-off, additionally to the standard compensation and regardless of their work performance. Probands were given the same task as in treatment A, performance was measured similarly. The researcher expects work performance to be higher in treatment B:

H_0: "Work performance in treatment B is not higher than in treatment A."
H_1: "Work performance in treatment B is higher than in treatment A."

Each proband must be associated to one of the hypotheses, but cannot be associated to both simultaneously. If the experiment yields the desired outcome, one can claim that since the test rejected H_0 in favour of H_1, incentives did increase work performance. If the test outcome is negative, no clear statement can be made.

The sample function consolidates information from the sample (cf. Mittag, 2017, p.227). Most sample functions follow a distribution type, which is independent of the sample distribution. The distribution parameters depend on whether H_0 or H_1 is correct. Given H_0 is true, the so-called null distribution can be computed at least approximately (cf. Bamberg, Baur and Krapp, 2017, p.166).

The sample function uses information from the sample to calculate the test statistic. Hence, the test statistic can be represented as a single value within the distribution of the sample function under the null hypothesis (e.g., the null distribution). Since the mean value of this distribution corresponds to the null hypothesis being true, the further the test statistic deviates from the distribution mean, the more likely the null hypothesis is false, based on the sample data.

The null hypothesis is formally rejected if the test statistic lies within the area of rejection. In a normal distribution, for example, the area of rejection corresponds to the area under the curve on both tails (two-tailed test). One widely used threshold is a significance level $\alpha = 5\%$, meaning that the area of rejection is equal to of the area under the curve on each tail. α is the probability of rejecting the null hypothesis, although it is true: $\alpha = P(\text{"}H_1\text{"}|H_0)$. There is always a chance of the null hypothesis and therefore the assumed distribution of the sample function under H_0 to be false. Due to that, one can never calculate a probability of the null hypothesis being generally false. The test statistic lying within the 5% area of rejection means, there is a 5% chance of observing such – or more extreme – data, given the assumed distribution is true. The use and interpretation of significance levels will be discussed in more detail in chapters 2.1.1 and 2.1.2.

Since a statistical test only operates on a sample of the entire population, there is a chance of it making the wrong decision, for instance, due to noise in the data. That leads to four possible scenarios which are commonly summarised in a confusion matrix. A false positive is also referred to

	true effect	no effect
positive (effect detected)	true positive (TP)	false positive (FP)
negative (no effect detected)	false negative (FN)	true negative (TN)

Table 2.1: *Confusion matrix.* Columns depict, whether there is a true effect or no effect in the population. Rows depict test decisions. (Author's illustration)

as a Type I Error and the probability of obtaining such error is equal to α. $(1-\alpha)$ corresponds to the probability of obtaining a true negative. A false negative, also known as a Type II Error, occurs with a probability of β. The study power is equal to $(1-\beta)$ and corresponds to the probability of obtaining a true positive. Table 2.2 summarises the terms.

scenario:	context:	probability:	term:
true positive	correctly rejecting H_0	$(1-\beta)$	power
false positive	falsely rejecting H_0	α	Type I Error
true negative	correctly not rejecting H_0	$(1-\alpha)$	
false negative	falsely not rejecting H_0	β	Type II Error

Table 2.2: *The four possible scenarios of a test decision.* "scenario" refers to the confusion matrix (Table 2.1). "context" relates scenario to test decision. "probability" refers to the probability of occurrence. "term" depicts alternatively used terms. (Author's illustration)

A researcher should aim to control for the probability of obtaining either one error. That is done by predefining levels of α and β. Decreasing levels of α lead to increasing levels of β. Hence, there

is a trade-off between the probabilities of a Type I and Type II Error. Therefore, the significance level α is determined first because a false positive usually poses a more costly thread. Next, a power analysis should be conducted, in order to determine an appropriate sample size, given the chosen β-level. For more information on the α-β trade-off, power analysis, and appropriate sampling, please refer to chapter 3.1.2.

The range of possible applications for statistical tests is large. Statements, although generally true, may therefore not be true with regards to every test. One can test multiple attributes regarding one population, as well as multiple populations regarding one attribute. Furthermore, one can distinguish between parametric and non-parametric tests. Unlike parametric tests, non-parametric tests do not require assumptions about the distribution of the population under study. For example, the parametric χ^2-test of variance assumes the population to be normally distributed. Non-parametric tests can also be used to study the distribution type of the population itself. The non-parametric Pearson's χ^2-test does not require assumptions about the probability distribution of the population as that is what is being investigated. Furthermore, there are sample functions that do not follow a distribution. Consequently, a null distribution cannot be computed. The value of the test statistic is then compared to critical values from look-up tables. The sample function in the non-parametric Kolmogorov-Smirnov test does not follow a null distribution, whereas the test statistic in the non-parametric Pearson's χ^2-test is defined on a χ^2-distribution.

2.1.1 P-Value and Significance Testing

Although statistical testing is commonly treated as one universal methodology, it is rooted in two rival approaches (cf. Biau, Jolles and Porcher, 2009). Ronald A. Fisher introduced Significance Testing in the late 1920s, prompting Egon S. Pearson and Jerzy Neyman to publish an alternative approach, known as Hypothesis Testing.

Fishers' methodology is built around the p-value, which is credited to the mathematician Karl Pearson (cf. Kennedy-Shaffer, 2019). First, a hypothesis is formulated, and an adequate sample function and probability distribution under that hypothesis are determined. Second, data is collected. As summarised by Biau, Jolles and Porcher (2009):

> "... after an experiment was conducted, we can compute a test statistic that measures the difference between what is observed and the ... hypothesis. This test statistic may be converted to a probability, namely the p-value, using the probability distribution of the test statistic under the ... hypothesis"(p.886).

A small p-value indicates that "either an exceptionally rare chance has occurred or the [hypothesis] is not true" (Fisher, 1956). The p-value corresponds to the probability of obtaining the sample at hand – or a more extreme one – given, the hypothesis is true. It is calculated by computing the probability mass under the curve, between the test statistic and every viable, more extreme value. In a two-tailed test – given the null distribution is symmetrical – if the test statistic equals the

estimated effect, p = 1. With the test statistic deviating from that value, p decreases. Given a small p-value, the corresponding test statistic will be located on one flank of the null distribution, far from the centre. This can be due to two distinct reasons: Either the collected data is exceptionally rare, or the distribution under the null hypothesis – and thus the hypothesis itself – is not true. Therefore, the p-value merely represents a quantitative measure of evidence against the null hypothesis, in that it only computes the incompatibility of the observed data with the hypothesised null distribution. It does not provide any information about the overall probability of the null hypothesis being true because the chance of simply having observed exceptionally rare data is still present.

Fisher intended the p-value to be a part of a fluid, non-numeric process of scientific inference (cf. Goodman, 1999). The actual conclusion was meant to be done by the scientist, weighing all facts, of which the p-value is only one (cf. Biau, Jolles and Porcher, 2009). "If [p] is below 0.02 it is strongly indicated that the hypothesis fails to account for the whole of the facts" (Fisher, 1992).

2.1.2 α-Value and Hypothesis Testing

According to Biau, Jolles and Porcher (2009), Pearson and Neyman criticised the ease at which one can mistakenly use the p-value as single measure of scientific reasoning. They proposed to predefine error probabilities, to control for consequences of potentially false decisions. Furthermore, they introduced the concept of two competing hypotheses, namely the null and alternative hypothesis. In that way, one is willing to only reject one hypothesis, given an alternative explanation for the observed data (cf. Biau, Jolles and Porcher, 2009). Hypothesis Testing can thus be interpreted as an attempt to create a methodology that allows for more sound decision making, solely based on statistical theory – as opposed to the subordinated role of Significance Testing.

As stated by Huberty (1987), Hypothesis testing requires the scientist to articulate a pair of hypotheses. Next, the sample function and probability distribution under the null hypothesis are determined. Prior to computing the test statistic, the significance level α shall be determined (cf. Huberty, 1987). The test decision is based on the value of the test statistic, relative to α (cf. Goodman, 1999). Neyman and Pearson recommended a general significance level of 5%.

α and p are mathematically similar. Both describe the probability of receiving a false positive. The difference only lies in their respective use. α represents a fixed threshold. A test decision based on a significance level α guarantees the false positive probability to not exceed that significance level α. The smaller α is set, the less (more) likely it is to obtain a false positive (false negative). The p-value – which is not defined a priori – can be interpreted as the minimum α level, at which the test decision is still positive. How much one is willing to control for a Type I Error must therefore be determined post hoc, when using p.

2.1.3 Modern Methodology

Today, the term "Null Hypothesis Significance Testing" mostly summarises the whole of statistical testing methodology. Statistical methods are applied to various contexts and disciplines in science (cf. Kennedy-Shaffer, 2019). Applied researchers, therefore, do not necessarily have a background in statistics. That the distinction between the two approaches is mostly forgotten is not a surprise, especially since the differences seem marginal, at first glance. Like Neyman and Pearson, for instance, Fisher also recommended a significance threshold of 5%, meaning that test decisions with $p \geq 0.05$ should not be deemed "significant". He chose this value for its ease of calculation, as it reflects a two standard deviation difference between the test statistic and the null hypothesis (cf. Kennedy-Shaffer, 2019). The present practice is still holding on to that cut-off value, although computational advancements have eradicated its utility. Statistics lectures and textbooks usually start with a narrative close to the Neyman-Pearson approach, just as the beginning of this chapter. Comparing two different introductory textbooks, one sees that both introduce two competing hypotheses and predetermined significance levels (cf. Mittag, 2017), (cf. Bamberg et al., 2017). Following explanations of Type I and Type II Errors, Mittag (2017) explains the concept behind the p-value, calling it an "empirical significance level". Bamberg, Baur and Krapp (2017) do not mention the p-value. In Practice, however, results are communicated using p-values, rather than α-levels. That alone demonstrates a huge difference in how statistical analyses are conducted in empirical sciences, compared to how they are taught at universities. Improper understanding of statistical theory – among a variety of other reasons – leads to p-hacking. I will introduce p-hacking in the following chapter, which will serve as a foundation for the subsequent analyses in chapters 3.1, 3.2, 3.3 and 3.4.

2.2 P-Hacking

What is p-hacking? Definitions differ from one another. The American Statistical Association defines p-hacking as "practices ... that emphasise the search for small p-values over other statistical and scientific reasoning" (Wasserstein, 2016, p.1). Any practice that does not emphasise general scientific inference "enough" is thus p-hacking, as well as is any analysis that is based "too much" on the reported p-value and "too little" on other statistical measures. This definition is vague. Brodeur, Cook and Heyes (2020) define p-hacking as "a variety of practices that ... generate better p-values ..." (p.2). Therefore, any method through which the reported p-value can be influenced is p-hacking. Usually, the goal is to achieve a reduction in p. However, as will be demonstrated in chapter 3.4.4.2, it is sometimes necessary to increase the p-value, to produce publishable results. Thus, p-hacking refers to manipulating p in either direction. In doing so, researchers are exploiting several insufficiencies of the p-value, which will be introduced in chapter 2.2.1. Chapter 2.2.2 explains why p-hacking exists to begin with. Chapter 2.2.3 presents the most common p-hacking methods, which will be further investigated in chapter 3.

2.2.1 Insufficiencies of the P-Value

One problem which has already been discussed in chapter 2.1.1 is that the p-value only measures the incompatibility of the null hypothesis with the data. It does not provide any evidence about whether the null hypothesis is true or false with regards to the entire population. Basically, two equally conceivable inferences can be made from a small p-value. Firstly, one can infer that the null hypothesis is simply false. After all, that is what the collected sample data suggests. Secondly, one can infer that the collected sample data is biased. That might be the case, given one is particularly convinced that the null hypothesis holds true. As explained in chapter 2.1, researchers intentionally set up the null hypothesis in such a way that it can be falsified. The latter inference is thus unlikely to be observed. Nevertheless, that shows another insufficiency: The p-value is by no means a measure of scientific significance. To derive at a scientifically sound conclusion, a researcher must practice genuine inference, which should be based on much more than just a p-value. For instance, inference should always include the observed effect size, as it provides an estimate of the true effect size in the population. The p-value fails to incorporate a measure of the effect size. Vanishingly small effects may not be of scientific interest, even though the corresponding p-value can be well below 5%. Particularly large effects might be of scientific interest, even if the reported p-value is above 5%. Ziliak and McCloskey (2008) coined this phenomenon the "sizeless stare", which stems from flexibility in the acquired sample size. Whereas small samples might fail to detect true effects, large samples might capture effects that are just noise, rather than true relationships. The p-value does not account for the sizeless stare.

Researchers leverage these insufficiencies. The consequence is p-hacking. Another problem emerges from that. Sterling (1959) remarks that "... research which yields nonsignificant results is not published"(p.30). Mueller-Langer et al. (2019) find that because journals prefer publishing statistic-

ally significant findings over null findings, the number of false positive findings in the literature is inflated. By random chance, a statistical test at the 5% significance level is expected to produce 5 positive results out of 100 replications, given there is no true effect in the population. The tendency to not publish negative results, combined with the tendency to publish new findings rather than replications causes a bias towards false positives in the literature, called the publication bias. For instance, estimates of false positive rates range from 14% in the medical literature in general to 37% in clinical trial meta-analyses (cf. Jager and J. T. Leek, p. 2013; cf. Pereira and J. P. Ioannidis, 2011). The impact of the publication bias is potentially big.

2.2.2 Why Does P-Hacking Exist?

Several factors promote p-hacking. One factor is the present incentive structure. Head et al. (2015) state that a researcher's performance is usually measured by her publication rate. The publishing process, which heavily focuses on significance thresholds as a measure of good scientific conduct, incentivises p-hacking (cf. Head et al., 2015). Moreover, because human nature systematically underestimates the impact of subconscious desire, Simmons, Nelson and Simonsohn (2011) conclude that scientists tend to be blind sighted by their desire to find significant results, often causing them to make advantageous, but questionable design decisions. Researchers who commit p-hacking might thus do so both intentionally and unintentionally. However, at the root of the problem is a historically grown testing practice (cf. Kennedy-Shaffer, 2019). J. Leek et al. (2017) criticise methodological mimicry within the applied sciences. This mimicry complicates the transition to a publication process that is free of incentives to forge findings. Furthermore, applied researchers appear to often lack proper understanding of statistical tools (cf. Krämer, 2012). That might be explained by the dissimilar and sometimes probably inadequate teaching of statistics fundamentals at universities, described in chapter 2.1.3.

It is obvious that p-hacking poses a multifaceted and deeply rooted problem. Finding solutions that account for the whole of the problem will be challenging. The American Statistical Association intends to "steer research into a 'post $p<0.05$ era'" (Wasserstein, 2016, p.1). "What we hope will follow is a broad discussion ... that leads to a more nuanced approach of interpreting, communicating, and using the results of statistical methods in research" (Wasserstein, 2016, p.2). The debate is indeed in full swing. Proposals reach from additional measures of confidence (cf. Wasserstein, 2016; cf. Goodman, 2018) to fundamental discussions over school of thought (cf. Goodman, 2001;) and beyond (cf. J. Leek et al., p. 2017). Meanwhile, papers are still being published, some of which are p-hacked. Being able to evaluate the quality of current studies is necessary. In order to do so, it is important to understand how p-hacking works. The following chapter therefore attempts to summarise the numerous p-hacking methods.

2.2.3 P-Hacking Methods

The spectrum of insufficiencies regarding the p-value, introduced in chapter 2.2.1 is alarmingly large. Consequently, the number of emerging techniques exploiting these weak spots is alarmingly large too. In fact, it appears impossible to capture the whole of p-hacking methods. Furthermore, it is not possible to attribute individual p-hacking methods to distinct categories. Doing so would rather disguise the diffuse nature of p-hacking. It is to some extent possible to find generic terms, however. Chapters 3.1, 3.2, and 3.3 are named accordingly.

Chapter 3.1 will focus on flexibility in sample size. The connection between effect sizes and the sizeless stare will be shown. Furthermore, p-hacking methods such as choosing arbitrary sample sizes, intermediate testing, and deleting outliers will be investigated – as well as whether one can detect such techniques. Head et al. (2015) say that p-hacking occurs, "... when researchers try out several statistical analyses ... and then selectively report those that produce significant results" (p.1). Chapter 3.2 summarises such practices under flexibility in variables. HARKing (Hypothesizing After the Results are Known) partly touches on the same issue. However, chapter 3.3 analyses a particular HARKing technique which exploits flexibility in hypothesis design. Further HARKing methods, such as exploratory analysis and multiple testing – the latter is closely related to flexibility in variables – are being demonstrated in chapters 3.4.1 and 3.4.3. A p-hacking method – that here is assigned to flexibility in sample size but could just as well be assigned to HARKing – is conduction multiple small studies, which will be demonstrated in chapter 3.4.4.2. Violating statistical assumptions and model choice are being reviewed in chapter 3.4.2. Finally, chapter 3.4.5 looks at inference from results. This touches on the problem of researchers not considering the shortcomings of the p-value as a sound basis for scientific reasoning.

3 Analysis

3.1 Flexibility in Sample Size

Determining an appropriate sample size has crucial implications for the robustness of a study. A study that is based on an insufficiently small sample size is called underpowered. Failing to conduct a proper power analysis when designing a study, is thus considered to be p-hacking (cf. Wicherts et al., 2016). However, overpowered studies are problematic as well (cf. Bhardwaj et al., 2004). In general, drawing correct conclusions from insufficiently powered studies is difficult. As a result, research resources will be wasted, as findings are being replicated that turn out to be false conclusions. Chapter 3.1.1 sums up some of the most severe consequences of studies on flexible sample sizes that lead to the ambiguity of conclusions. Chapter 3.1.2 introduces strategies to determine a sufficient sample size and chapter 3.1.3 analyses different types of flexibility in sample sizes and the extent to which they distort empirical findings. Chapter 3.1.4 presents an approach to determining the power of a published study. Knowing if a study is underpowered, allows the reader to make conclusions about the deployed sample size.

3.1.1 Consequences of Flexibility in Sample Size

Insufficiently small sample sizes increase the false negative rate (FNR) of statistical test decisions. There are two possible test outcomes related to an effect existing in the population. The decision represents either a false negative or a true positive. A false negative occurs with the probability β (see chapter 2.1). Since both scenarios cover the entire sample space, their summed probability is 100%. Therefore, the probability of obtaining a true positive equals $(1-\beta)$ – which is also referred to as the study power. Accordingly, a low power – stemming from a small sample size (see chapter 3.1.2) – corresponds to a large probability β of obtaining a false negative and a low probability $(1-\beta)$ of obtaining a true positive.

$$P(FN) = \beta \tag{3.1}$$

$$P(TP) = (1-\beta) = power \tag{3.2}$$

$$FNR = \frac{FN}{FN + TP} = P(FN|H_1) \tag{3.3}$$

A low study power also corresponds to an increase in the false discovery rate (FDR) (cf. Tong and Zhao, 2008). The false discovery rate computes the probability of obtaining a false positive, given the test rejects the null hypothesis. It thus computes the number of cases in which an effect was falsely detected (e.g., false positives), relative to the overall number of cases in which an effect was detected. This is not to be confused with the false positive rate (FPR), which computes the number of cases in which an effect was falsely detected, relative to the number of cases in which there is no effect in the population (e.g., false positives and true negatives).

$$FDR = \frac{FP}{FP + TP} = P(FP|positive) \tag{3.4}$$

$$FPR = \frac{FP}{FP + TN} = P(FP|H_0) \tag{3.5}$$

Depending on the practice a researcher uses to manipulate the sample size, the false positive rate can be affected as well (cf. Simmons, Nelson and Simonsohn, 2011). This will be demonstrated in chapter 3.1.3.4.

Furthermore, the accuracy of estimates is low on low sample sizes (Bamberg, Baur and Krapp, 2017, p.140). That renders statistical abuse more feasible as, for instance, point estimates of population means can vary much at low sample size and can thus be manipulated with ease.

Underpowered studies tend to overestimate effect sizes. The Z-test, for instance, uses the z-score transformation as sample function.

$$z = (\frac{\bar{x} - \mu_0}{se}) = (\frac{\bar{x} - \mu_0}{sd}) \cdot \sqrt{n} \tag{3.6}$$

$$se = \frac{sd}{\sqrt{n}} \tag{3.7}$$

The resulting null distribution equals a standard normal distribution. The test statistic is being computed by applying this transformation to the sample mean. If the test statistic – plotted onto the null distribution – is within the proximity of roughly two standard deviations from 0, the test fails to reject the null hypothesis. The central limit theorem allows to express the standard error (se)

through the standard deviation, normed by the square root of the sample size ($\frac{sd}{\sqrt{n}}$). Thus, a small sample size leads to a large standard error *se* and a small deviation between test statistic and null hypothesis. Due to that, the test will only recognise specifically large effects, when performed on small samples.

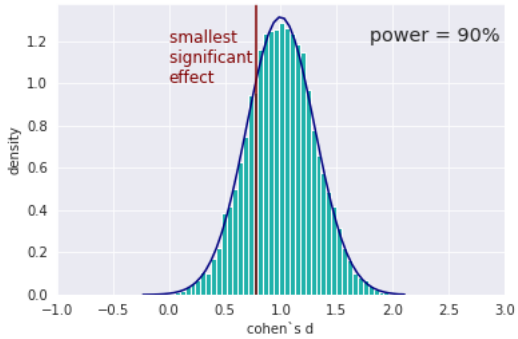

Figure 3.1: *Inflated effect size.* Smallest significant effect at power = 90%. (Author's illustration)

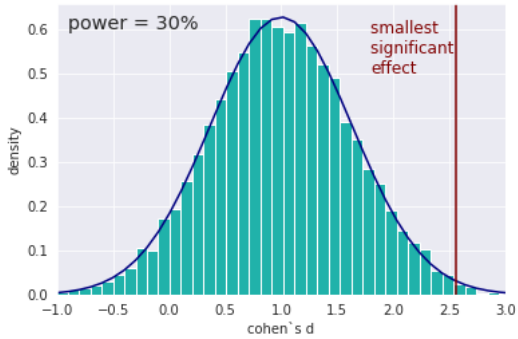

Figure 3.2: *Inflated effect size.* Smallest significant effect at power = 30%. (Author's illustration)

Figures 3.1 and 3.2 show distributions of measured mean effect sizes at power = 90% and power = 30%, respectively. The true effect size is cohen's d = 1. I computed 10000 simulations on independent samples of corresponding sizes. The red line depicts the smallest effect size whose two-sample t-test is still positive at α = 5%. As can be seen in Figure 3.2, at a power of 30%, the test even fails to detect the true effect size. At power = 90%, the test can detect the true effect size. However, when taking the mean of significant means as an estimate, the true effect size will still be overestimated.

Small sample sizes negatively affect replication rates. Because underpowered studies tend to capture inflated effect sizes, reproducing such a study successfully requires to repeatedly draw an equally underpowered sample. A test on a properly powered sample will tend to find a smaller significant effect. That it is common practice to justify sample sizes on literature research can be observed in the fact that Israel (1992) is even encouraging researchers to "use the same sample size as those of studies similar to the one you plan" (p.2). Consequently, low sample sizes are partially self-reinforcing and inflated effect sizes tend to get anchored in the scientific literature, while replication rates based on properly powered analyses decline.

To counter the downsides of small samples by choosing arbitrarily large sample sizes, is not a sufficient solution, since limitations in data gathering and computing power render that mostly infeasible. Furthermore, a test performed on a "too large" sample might detect "very small" effects that do not exist in the population. The effect does exist in the sample and because the sample is large, it is expected to describe the population "very precisely". Therefore, the test labels the effect "statistically significant". Bhardwaj et al. (2004) remark that small, significant effects may not be of

scientific interest if they are too small. Ziliak and McCloskey (2008) refer to this phenomenon as the sizeless stare (see chapter 2.2.1).

3.1.2 Power Analysis

In order to avoid obtaining a "too large" or "too small" sample, the optimum sample size must be determined, prior to conducting an empirical study. That can be done by using equations such as proposed by Israel (1992). These are also used for calculating look-up tables, which one can use as well. Another common approach is to conduct a power analysis. A power analysis takes three out of four variables as input parameters and computes the fourth variable as output parameter. The variables are: effect size, sample size, Type I Error level "α", and Type II Error level "β". One can thus compute the appropriate sample size, given predefined α- and β-levels and an estimate of the effect size. If there is no size estimate of the effect under study – which usually stems from previous exploratory analysis – the researcher can default to the smallest effect size which would still have real world use and implications.

Chapter 2.1 has introduced the trade-off between α- and β-levels. Although this trade-off indicates otherwise, it is possible to predefine both error levels simultaneously. Suppose one wants to test whether the mean value of a population is zero. A two-tailed one-sample Z-test was conducted to test the according null hypothesis.

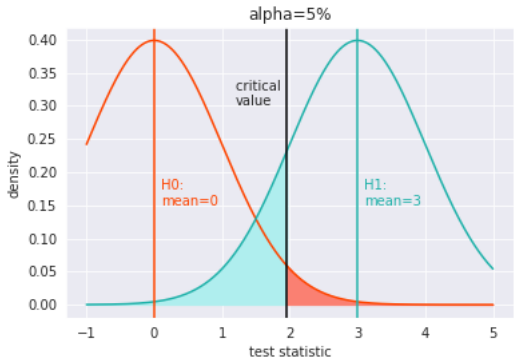

Figure 3.3: *Power analysis.* α-β trade-off, $\alpha = 5\%$. (Author's illustration)

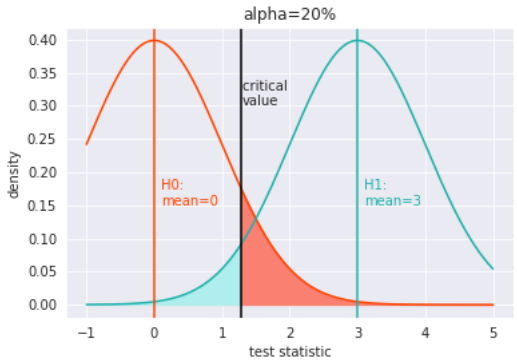

Figure 3.4: *Power analysis.* α-β trade-off, $\alpha = 20\%$. (Author's illustration)

The red curve in Figure 3.3 depicts the probability distribution under the null hypothesis. Since the aim is to test against $H_1 : \mu \neq 0$, each bell curve shifted in x-direction shows one possible scenario under the alternative hypothesis. The blue curve depicts one scenario in which $\mu = 3$. The red area corresponds to the probability of falsely rejecting the null hypothesis (e.g., the probability of obtaining a Type I Error). Because the test statistic which would derive from the blue curve is larger than the upper critical value, the test will reject H_0, even if the data indeed comes from the null distribution. In a two-tailed test at $\alpha = 5\%$, the depicted red shaded area equals 2.5%. Suppose the

blue curve resembles the true probability distribution in the population. The blue shaded area depicts the opposite case, in which the test fails to reject H_0, although the data actually stems from the alternative distribution. Such case occurs with the probability β, which equals the size of the blue area.

The trade-off becomes apparent, once the significance level is altered. In Figure 3.4, for instance, a significance threshold of $\alpha = 20\%$ would correspond to a smaller critical value, thus quadrupling the red shaded area under the curve. Given the sample mean equals 3, the blue curve would depict the probability distribution of the drawn sample and as a result, both curves maintain their depicted shapes. Due to the increased rejection area α, the blue shaded area which corresponds to the probability of receiving a β-error decreases. Vice versa, setting a smaller significance level would increase the probability of a Type II Error.

However, when designing the test, one wants to control for both error probabilities at once. Note that the depicted curves in Figures 3.3 and 3.4 do not represent the probability distribution of the observed variable itself. They depict the probability distribution of the mean value of that variable and thus represent sampling distributions under varying assumptions. The red curve expects the true mean value to be 0. The blue curve expects it to be 3. The latter value can stem from the mean value of a drawn sample, as well as it can be an exemplary value from the alternative hypothesis. Both curves are defined on the standard error (se), which can be expressed through the standard deviation (sd) in the population (see chapter 3.1.1).

A larger sample size would decrease the estimate for the standard error. Therefore, the depicted curves in Figures 3.3 and 3.4 would be narrower. That would lead to a decrease in their overlap, decreasing both error probabilities at the same time and thus increasing the study power. This dynamic allows to determine n, so that α and β satisfy their respective thresholds. The integral inside $[c, \infty]$, under the alternative sampling distribution then equals $(1 - \beta) = power$.

Suppose the Z-test has already been conducted on a sample of given size. The power analysis can then compute the achieved test power post hoc, given α, the measured effect size, and n. In the following, the blue curve in Figure 3.5 shall stem from an observed sample mean. Given that, power equals the survival function (1–cdf) at the critical value. Here, $power = 58.2\%$ and thus $\beta = 41.8\%$. Note that the blue area now depicts power, instead of β.

The function “normal_power” from the statsmodels package in python allows to do the same computations. The command is written as follows:

```
normal_power(effect_size, nobs, alpha, alternative='two-sided', sigma=1)
```

The number of observations, meaning the sample size, is being captured by nobs. The parameter ‘alternative’ defines the test design which is default set to ‘two-sided’ and can furthermore be defined as ‘larger’ given a right-tailed and ‘smaller’ given a left-tailed test design. The computation alters,

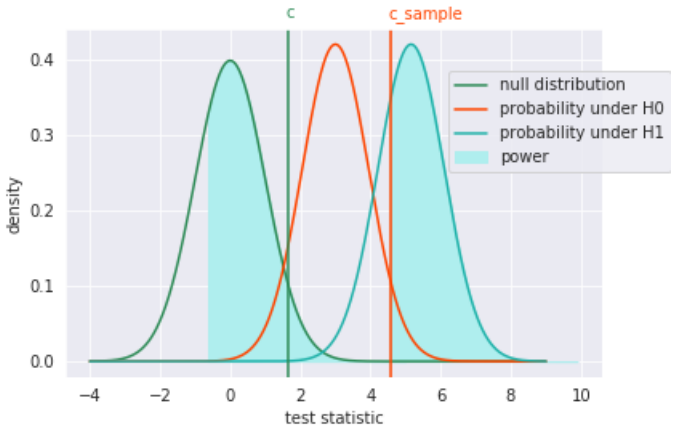

Figure 3.5: *Power analysis.* Power calculations under the null and alternative distribution.

depending on the effect direction. I will explain computations in the context of a positive effect direction. The "normal_power" function expects the effect size to be stated in cohen's d which standardises the absolute effect size (e.g., $\bar{x} - \mu_0$) by the pooled standard deviation.

$$d = \frac{(\bar{x} - \mu_0)}{sd_{pooled}} \tag{3.8}$$

$$sd_{pooled} = \begin{cases} \sqrt{\frac{sd_1^2 + sd_2^2}{n}}, & n_1 = n_2 \\ \sqrt{\frac{(n_1 - 1) \cdot sd_1^2 + (n_2 - 1) \cdot sd_2^2}{n_1 + n_2 - 2}}, & n1 \neq n_2 \end{cases} \tag{3.9}$$

If the post study power shall be determined, the obtained p-value can be handed over as value for α (more information on post hoc power analysis is provided in chapter 3.1.4). Figure 3.5 shows how the sample function in the Z-test transforms the red sampling distribution under the null hypothesis into the null distribution, which here is a standard normal distribution. The Z-test itself would simply compute the Z-transformed mean (e.g., the mean value of the blue sampling distribution) onto the green null distribution. The red critical value c_{sample} is defined under the red distribution, whereas the green critical value c is defined under the green distribution. "scipy.stats.norm.isf" computes the inverse normal survival function.

```
c_sample = scipy.stats.norm.isf(q=alpha, loc=mean, scale=se)
```

$$c = (\frac{c_{sample} - \bar{x}}{sd}) \cdot \sqrt{n} \tag{3.10}$$

Instead of computing the area under the blue sampling curve as done in the introductory example above, "normal_power" performs all computations directly under the standard normal distribution. The critical value in "normal_power" is thus computed similarly to c and is then shifted in the amount of $\left(\frac{-d \cdot \sqrt{n}}{se}\right)$. The standard error is thus default set to 1. The area under the null distribution, right from that value equals the area under the blue distribution, right from c_{sample}. I demonstrate this in "power analysis" (code in Appendix F). Note that d condenses the relative position of both sampling curves to each other – and thus their absolute overlap. Therefore, "normal_power" does not require their actual shape and position.

To sum up, a large sample size corresponds to a small standard error, which itself corresponds to a large cohen's d. Therefore, given the measured absolute effect size (e.g., $\bar{x} - \mu_0$), the standardised effect size d – more specifically its denominator – can be fitted to keep α- and β-levels at their predefined values, through varying the sample size.

Power analysis cannot be performed on tests that do not use a null distribution (e.g., many non-parametric tests). Mumby (2002) provides an approach to determining power of non-parametric tests. If the probability density function (pdf) of the population under study is unknown, it must be determined at first. Based on Monte Carlo simulations, sample data can be generated from the obtained pdf. The respective non-parametric test then can be performed on that data. The number of occurring false negatives, relative to the total number of simulations is an estimate for the β-error level, and $power = 1 - \beta$. Mumby (2002) performed 1000 simulations on varying effect and sample sizes in order to find the optimum set of parameters that allows to control α- and β-error levels.

3.1.3 P-Hacking through Flexible Sample Size

The influence of flexible sample sizes on the p-value is large. It furthermore depends on the type of flexibility that is being applied. For instance, a researcher might justify her sample size by literature research for sample sizes of comparable studies. Through that practice – which, according to Wicherts et al. (2016) is p-hacking– the sample size is rather chosen arbitrarily than analytically. Furthermore, conducting statistical tests while data collection is still ongoing poses another type of flexibility that constitutes p-hacking (cf. John, Loewenstein and Prelec, 2012). A third type of flexibility is excluding observations post hoc. Potential outliers must be addressed and sometimes excluded from the sample. Deciding on that ad hoc is considered to be p-hacking (cf. Wicherts et al., 2016). In the following, I will investigate each of the three presented types of flexibility. The purpose is to demonstrate to what extent error rates and p-values are affected by flexible sample sizes. In this section, simulated data is being used exclusively. That allows to compute error rates, as the truth value of test decisions is known. Chapter 3.1.4 will attempt to detect the extent of flexibility in the sample size, by investigating a published study.

3.1.3.1 Arbitrary Sample Size

Figures 3.6 and 3.7 depict the development of test statistic and p-value over increasing sample sizes, at fixed effects ($d_{measured} = -0.7010$, $d_{true} = -0.5$).

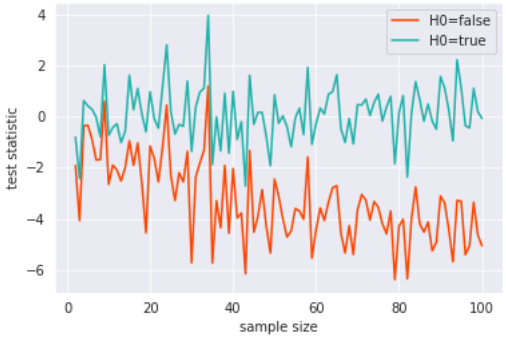

Figure 3.6: *Arbitrary sample size.* Development of the test statistic over increasing sample sizes. (Author's illustration)

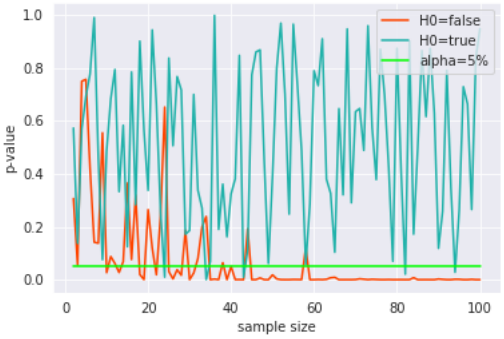

Figure 3.7: *Arbitrary sample size.* Development of the p-value over increasing sample sizes. (Author's illustration)

In order to know the truth value of test decisions, the sample data is simulated, following a standard normal distribution. Two-tailed one-sample t-tests were conducted on samples of sizes ranging from 2 to 100. The blue line depicts test statistic and p-value with the according null hypothesis $H_0 : \mu = 0$. The red line depicts test statistic and p-value with the according null hypothesis $H_0 : \mu = 0.5$. Possible test outcomes are thus true negative and false positive with regards to the blue line, and true positive and false negative with regards to the red line. In the following, I will refer to this relationship as the null hypothesis being true or false. Peaks in Figure 3.6 occur at equal points in both curves of the test statistic, since both curves are computed on the same sample. These peaks resemble rare characteristics of particular samples that occur with random chance. More precisely, the mean value of those samples is particularly large (small), resulting in a particularly large (small) test statistic. Whereas the test statistic steadily decreases in $H_0 = false$, it remains around a value of 0 in $H_0 = true$. Consequently, p in H0 = false drops and remains below the significance threshold for large samples. For $H_0 = true$, however, p does not drop below 5% and remains subject to heavy fluctuation. Due to that fluctuation, it is possible to obtain a false positive at every given sample size, although the according probability appears to be very low. On the other hand, the probability of obtaining a false negative appears to be comparably large at first, but quickly decreases as the sample size increases, which is due to the steady decrease in the test statistic.

The fluctuation disappears once the simulation is being repeated. Figure 3.9 again depicts p-values over increasing sample sizes at similar true effect sizes and null hypotheses. Figure 3.8 now shows the corresponding false negative rate (FNR), false positive rate (FPR), and false discovery

rate (FDR). For more information on the false discovery rate, refer to chapter 3.1.1.

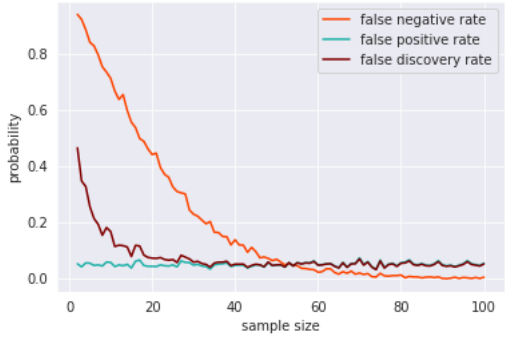

Figure 3.8: *Arbitrary sample size.* Development of error rates over increasing sample sizes. Mean values of 1000 simulations are plotted. (Author's illustration)

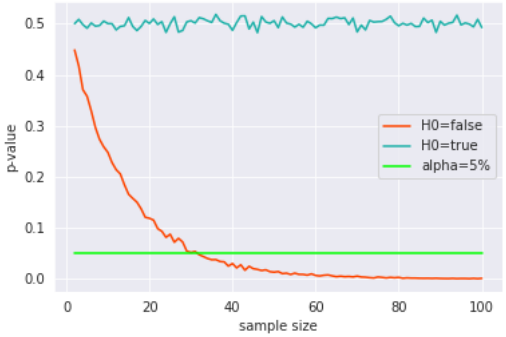

Figure 3.9: *Arbitrary sample size.* Development of p-values over increasing sample sizes. Mean values of 1000 simulations are plotted. (Author's illustration)

The depicted curves are computed as means over 1000 simulations. As a result, the curves are smoothed out, which is especially the case for p in $H_0 = true$. However, the plot does not capture standard deviations, which are $sd_{H_0=true} = 0.29$ and $sd_{H_0=false} = 0.09$. Therefore, the smaller the true effect, the larger the standard deviation of the corresponding p-value. Obtaining a false positive at an arbitrary sample size thus remains a possibility. There is no simple solution to that problem, as it touches on the ASA's statement, introduced in chapter 2.2.1. The p-value measures the incompatibility between sample and hypothesis. The fluctuation in the data, as seen in Figure 3.7 originates from random chance and p directly reflects that randomness. Capturing means of repeated simulations only disguises that characteristic. However, as predicting random chance is impossible, intentionally exploiting that characteristic is also impossible.

The FPR captures the probability of p dropping below 5%. The FPR remains below, but close to 5%. That proves the fact that a false positive will occur with a frequency of 5%, given the significance level is also 5%. As the accuracy in detecting an existing effect increases along an increasing sample size, the FNR eventually approaches 0, which resembles one consequence, introduced in chapter 3.1.1. Furthermore, Figure 3.8 shows that the FDR too decreases, as the sample sizes decreases. This also resembles a consequence, introduced in chapter 3.1.1.

Figure 3.10 provides a comprehensive overview over FNR at varying effects, measured in cohen's d. The presented curves depict means over 1000 simulations again. Sample sizes of 25 and 75 are sufficient for the FNR to approach 0 at large and moderate effects, respectively. However, the FNR is still at 20%, given a sample size of 200, when a small effect is present. That still corresponds to a power of 80%. The sizeless stare becomes apparent. Since doubt about whether the null hypothesis is

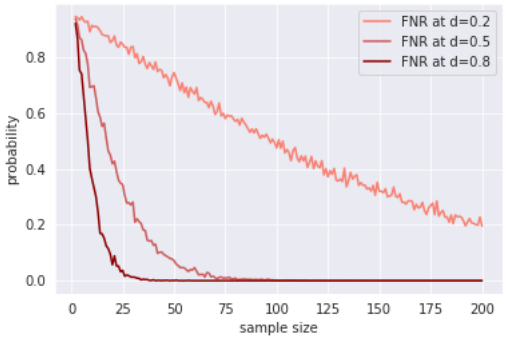

Figure 3.10: *Arbitrary sample size.* False negative rates in dependence of sample size, at varying effect sizes. Mean values of 1000 simulations are plotted. (Author's illustration)

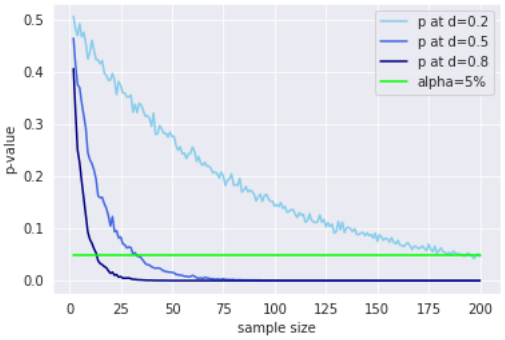

Figure 3.11: *Arbitrary sample size.* P-Values in dependence of sample size, at varying effect sizes. Mean values of 1000 simulations are plotted. (Author's illustration)

true in the population will remain, classifying results as false negatives or false positives is impossible in reality. Therefore, it is not possible either to decide, whether a very small effect can be attributed to random chance or is in fact present in the population. Figure 3.11 provides p-values in dependence of sample size, at small, moderate and large standardised effects. Curves are again depicting means over 1000 simulations. At a sample size of 200, the average p-value of small effects reaches the significance threshold. This implies that any smaller effect with a correspondingly larger sample size will eventually reach a publishable significance level. However, it is uncommon to find sample sizes of 200 and larger in the literature, regarding simple statistical tests. Small effects which are detected in tests on much lower sample sizes do not stem from the sizeless stare. They might rather be product of the fluctuation in p that displays randomness in the data, as discussed in Figure 3.6. That fluctuation is especially large, given small effects, which is why the standard deviation of p is large at small effects, as explained above. Note that more advanced statistical methods like ANOVA and regression analysis generally require more data in order to achieve comparable accuracy. The values for sample size, discussed here, do only apply for simple statistical tests. Even using a two-sample t-tests instead of a one-sample t-test doubles the total required sample size.

3.1.3.2 Intermediate Testing

Another type of flexibility in sample size is intermediate testing. Wicherts et al. (2016) define this practice as repeatedly conducting statistical analysis midway through data collection. Data collection ends upon finding a significant effect. Figures 3.12 and 3.13 depict test statistics and p-values at the same true effect size, as in Figures 3.6 and 3.7 (chapter 3.1.3.1), only that the effect is pointing in positive direction, now ($d_{measured} = 0.5844$, $d_{true} = 0.5$).

The sample size ranges from 2 to 100. A two-tailed two-sample t-test was conducted, after it-

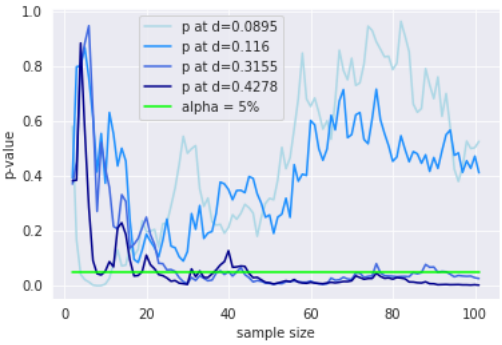

Figure 3.12: *Intermediate Testing.* P-Values in dependence of the sample size, at varying effect sizes. (Author's illustration)

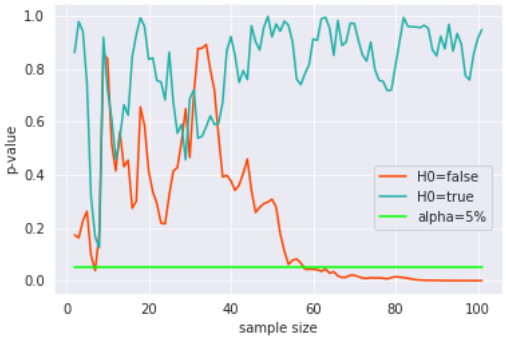

Figure 3.13: *Intermediate Testing.* Development of the p-value over increasing sample sizes. (Author's illustration)

eratively adding one further observation to the existing respective sample. In the following, the x-axis depicts the respective size of either one sample. The total sample size is thus twice as large. The fluctuation in p for $H_0 = true$ is much less, compared to arbitrary sample sizes (see chapter 3.1.3.1). That is not surprising, considering the difference in flexibility. Here, rather than the entire sample, only the latest added observation is subject to random chance. Intermediate testing thus appears to be even less effective when actively searching for significant p-values. However, an arbitrary sample size refers to arbitrarily predefining the size and then drawing the sample once. Here, repeated testing is being conducted, which increases the probability of obtaining a false positive. Assume, the probability of receiving a positive test decision at a random sample size is only $P(positive) = 10\%$. Thus $P(negative) = (1 - P(positive)) = 90\%$. The probability of obtaining one positive result out of 10 repeated random draws is already above 61% and converges towards 100% after 50 draws. Here, the number of draws equals 98. Note that only one significant finding is enough to get published, and how often the test was repeated will not be documented in the study. Intermediate testing can thus pose a very potent method to produce significant p-values.

The sideways trend in p-values in the blue curve replicates the findings of Simmons, Nelson and Simonsohn (2011) who show that p-values do not necessarily drop past the significance threshold if the sample size is iteratively increased. Figure 3.12 compares the trend of p-values at varying effect sizes. Here, p even increases between sample sizes of 50 and 80, given $d = 0.0895$. As the measured effect sizes increase, the development of p approaches the commonly anticipated downwards trend. At $d = 0.4278$, p remains below the significance threshold for most sample sizes. Two noticeable characteristics can be observed. Firstly, in this simulation, the test always fails to detect a small effect ($d_{measured} = 0.116$) and fails detect medium sized effects every time ($d_{measured} = 0.3155$). This suggests that test decisions are prone to random events more often than one might think intuitively. Secondly, the fluctuation in p decreases, as the sample size increases. That is particularly the case with large effects which are more detectable. This fluctuation at low sample sizes can lead to significant findings at every possible effect size. Note that $p < 0.05$ at sample size = [5;12] and $d_{measured} = 0.0895$. This highlights the importance of sufficiently large samples once again.

3.1.3.3 Deleting Outliers

A third type of flexibility derives from the necessity of data pre-processing. The sample size is being influenced by decisions over how to deal with outliers and missing or incomplete data. These types of data can be regarded as measurement errors. For example, a participant in a psychology experiment might have filled out a questionnaire incompletely or not truthfully. In another context, a measuring instrument might deliver incomplete or imprecise data. This type of flexibility stems from a stage in the study, prior to data analysis. However, the analysis stage is what is being investigated here. Because the entire analysis is based on t-tests, outliers refer to data points with maximum proximity to the mean value. Figure 3.14 depicts the development of the test statistics of two-tailed two-sample t-tests in varying circumstances.

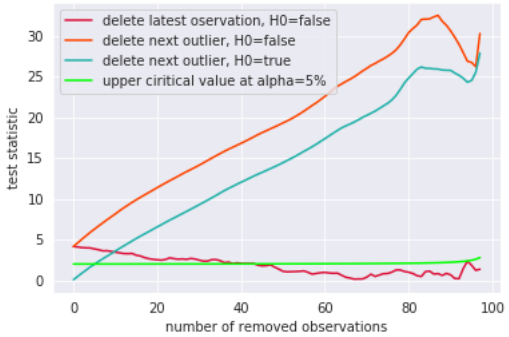

Figure 3.14: *Deleting Outliers.* Development of the test statistic over decreasing sample sizes. (Author's illustration)

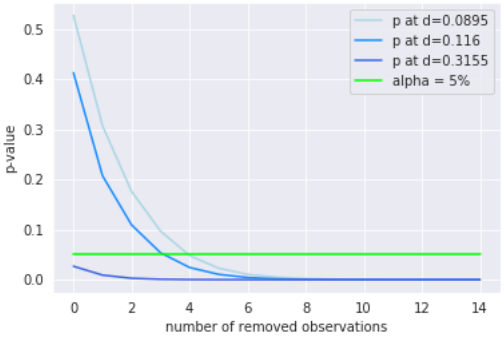

Figure 3.15: *Deleting Outliers.* Development of p-values over decreasing sample sizes, at varying effect sizes. (Author's illustration)

Each of the two samples is iteratively diminished by one observation, starting at a size of 100. The green line depicts the upper critical value of the test statistic at which the significance level of $\alpha = 5\%$ is being reached. Test statistics above this green line correspond to positives, values below correspond to negatives. The red line depicts the naïve case in which the latest observation is being removed and the test is being repeated. As in chapter 3.1.3.2, the effect size on the entire sample of 100 observations is $d_{measured} = 0.5844$. This strategy does produce as many significant findings as intermediate testing since both methods are performed on equal data. After removing 43 observations, the test statistic drops below the upper critical value. The test produces false negatives, as the sample size decreases. Iteratively deleting the latest observation only differs from intermediate testing in two regards. Firstly, test decisions will occur in reversed order. Secondly, randomness is entirely eradicated, as the sample is already drawn when starting the search for significant findings.

The orange line depicts values for the test statistic which are being computed on the same sample as those, shown by the red line. Here, the minimum value from the sample with a larger mean

is deleted ($\bar{x}_{sample1} = 0.6668$), as well as the maximum value from the sample with a lower mean ($\bar{x}_{sample2} = 0.1264$). That iteratively increases the absolute effect size, meaning the deviation between means of both samples, in greatest possible steps. Deleting the opposite outlier in each sample (e.g., the maximum value in sample1 and the minimum value in sample2) will temporarily push the value of the test statistic into the non-rejection area, until the difference in means is large enough for the corresponding test statistic to pass the lower critical value. The minimum p-value by deleting latest observations is $p = 0.00005$, whereas the minimum p-value by deleting next outliers is $p = 1.6969 \cdot 10^{-36}$ – which is achieved by only comparing the last two remaining observations in each sample. These remaining observations are de facto outliers too, but they are located on the side of the data cloud which is beneficial to produce a significant finding. Publishing such a test decision is highly reprehensible and very unlikely since the extreme results will not go undetected in the peer review process. The maximum p-value observed by deleting latest observations is $p = 0.8932$, whereas the maximum p-value observed by deleting next outliers is $p = 0.00005$, which leads to a maximum delta of $\triangle p_{max} = 89.3134\%$. The largest difference in p-values appears on a sample size of 32, which is deemed sufficiently large to perform a t-test. Furthermore, whereas 43% of test decisions by intermediate testing are positive, 100% of test decisions by deleting outliers are positive. This strategy is thus very effective in producing significant findings, particularly compared to intermediate testing.

The blue line depicts the same procedure of deleting next outliers, only on different data. Here, the null hypothesis is true, since both input samples originate from a standard normal distribution. On the full sample, the test correctly decides negative ($p = 0.9494$), but after deleting 7 outliers from each sample, the test decision remains positive ($p = 0.0227$). Deleting outliers here produces false positives in 93% of all cases. Therefore, removing even a small number of outliers from the sample drastically affects the effect size and thus the p-value, regardless of whether the null hypothesis is true or false. Simmons, Nelson and Simonsohn (2011) find great inconsistency in how applied researchers handle outliers. The very small number of deleted outliers might hint to the possibility that applied researchers do forge their analyses by deleting outliers. They probably even do so unintentionally.

Figure 3.15 depicts the development of p-values on varying effect sizes, using the strategy of iteratively deleting outliers again. Already a medium effect size of $d_{measured} = 0.3155$ produces only significant findings. The light blue line shows the development of p with an initially measured effect size of $d_{measured} = 0.0895$. Initially, $p = 0.5275$. After removing 5 observations, $p = 0.0481$. This shows that even on infinitesimal effect sizes, the presented method can produce significant findings. If one is willing to actively forge sample data, producing publishable results is inevitable. Furthermore, this practice might even remain undetected. This becomes apparent in the fact that the employed data with the smallest effect size was only truncated by 5%, resulting in a sample size of 95. Nevertheless, $power = 12.83\%$ using $\alpha = 5\%$ and the true effect size $d_{true} = 0.833$. The sample size is thus far too small in order to make reliable statements about the statistical significance of such small effects. The true effect size is unknown when conducting a study. In many regards, the study is being conducted to receive information about the true effect size. However, the measured

effect size is strongly affected by deleting outliers. Through this relationship, the achieved test power regarding all three depicted lines in Figure 3.15 approaches 100% before 10 observations are removed, when using measured d to compute power. Even, if the $d_{true} = 0$, power is above 80% after removing 10 observations, and reaches 100% before 30 observations are being removed. These findings provide further evidence that deleting outliers indeed remains undetected. Chapter 3.1.3.4 presents FPR, regarding arbitrary sample size, intermediate testing and deleting outliers. Chapter 3.1.4 introduces post hoc power analysis, which further investigates the potential to assess the robustness of published results.

3.1.3.4 False Positive Rates

Simmons, Nelson and Simonsohn (2011) show that p-hacking methods, such as intermediate testing have a severe impact on FPR. By testing after each added observation, starting at a sample size of 10, the reported false positive rate $FPR_{reported} = 22.1\%$. The sample size itself cannot explain that observation, since the FPR should stay unaffected by it. Instead, repeated testing itself causes the rise in FPR. As mentioned in chapter 3.1.3.2, the goal is to produce only one significant finding, although the number of trials is being increased drastically. The FPR in tests on an arbitrary sample size will be 5%, given $\alpha = 5\%$ (see chapter 3.1.3.1). The results from chapter 3.1.3.3 suggest that deleting outliers will result in $FPR = 100\%$.

significance level	minimum sample size	Arbitrary sample size	Intermediate testing	Deleting outliers
$\alpha = 1\%$	$n = 10$	1.0%	9.0%	100.0%
	$n = 20$	1.0%	6.6%	100.0%
	$n = 30$	1.1%	5.9%	100.0%
$\alpha = 5\%$	$n = 10$	5.1%	26.2%	100.0%
	$n = 20$	4.9%	25.4%	100.0%
	$n = 30$	4.9%	18.9%	100.0%
$\alpha = 10\%$	$n = 10$	9.8%	46.6%	100.0%
	$n = 20$	10.4%	37.8%	100.0%
	$n = 30$	10.0%	33.5%	100.0%

Table 3.1: *False positive rates.* Estimates for FPR over 1000 simulations, at varying significance levels and minimum sample sizes. (Author's illustration)

Table 3.1 depicts FPR over 1000 simulations. The maximum sample size is 100, all data is pooled from the standard normal distribution. Paired t-tests were performed after each added observation. FPR is computed as the number of simulations, in which at least one positive was produced, relative to the total number of simulations. This calculation makes sense because the researcher has no incentive to report any of the received insignificant findings. Instead, data collection will end upon finding one significant result, which is also the only one that will be reported. As expected, the FPR on arbitrary sample sizes is very close to the significance level α. Furthermore, at least one false positive was detec-

ted in 100% of cases when deleting outliers. Consequently, if a researcher is willing to forge data, any relationship can be reported as "significant". Intermediate testing does increase the FPR, too. In fact, starting on a sample size of 10 observations at $\alpha = 5\%$, FPR is 4.1% higher, compared to the findings from Simmons, Nelson and Simonsohn (2011). One out of 5 p-hacking attempts will be successful. At $\alpha = 10\%$ almost every second attempt will produce a p-hacked finding. At $\alpha = 1\%$ the FPR is still 6 to 9 times higher. These are alarming findings, considering the widespread use of intermediate testing.

It is not possible to estimate a generally achievable reduction in p by applying flexibility in sample size, for two reasons. Firstly, there are many ways to achieve a reduction in p. Secondly, it is up to the researcher to which extent she is willing to manipulate the sample data. More manipulation leads to lower p-values. On non-manipulated, small samples a small effect will lead to a high FNR and a high p-value. In this situation, intermediate testing and deleting outliers are potent methods to still produce a small p-value. Especially deleting outliers does not necessarily diminish the sample size too much. In fact, it allows to report any effect size and any p-value, on any sample size. As a result, originally small effects are being inflated and corresponding p-values are below 5%. Due to that, measured FPRs are so alarmingly high. As laid out in chapter 3.1.1, small samples already lead to inflated effect sizes. When considering the here presented p-hacking methods, the extent of inflated effect sizes in the literature might be even higher than expected. Since the sample size does not provide any evidence on whether the sample was p-hacked, intermediate testing and deleting outliers remains undetected. Chapter 3.1.4 introduces a method to evaluate the probability of false positives, based on the sample size. It allows to account for inflated effects. However, for the reasons mentioned, intermediate testing or deleting outliers will remain undetected.

3.1.4 Post Hoc Power Analysis

I here present an exemplary approach to conduct a retrospective power analysis. It allows the reader to determine the power of a published study post hoc, providing further information about the conclusiveness of the presented results. Booth et al. (2020) find that the preliminary fatality case rate (pFCR) in patients with haematological malignancy and SARS-CoV-2 infection is significantly higher for those who have been treated with immunosuppressive and cytotoxic medication (pFCR = 70%), than for those who have not (pFCR = 28%). Haematological malignancies describe cancers of the haematopoietic system, such as leukaemia. Cytotoxic and immunosuppressive medication are common treatments for this type of disease. The preliminary fatality case rate is measured in the number of deaths, relative to the total number of cases – including patients, whose symptomatic infection is still ongoing at the end point of data collection. Data was collected from 01 March 2020, until 06 May 2020. 66 patients were included in the study.

The study was conducted as follows: Patients were separated into two groups, which I will refer to as Control and (immunosuppressive and cytotoxic) Treatment. The number of observations account for 29 in Control and 37 in Treatment. A two proportions Z-test was conducted, in order to test the difference in pFCR between both groups. The absolute effect size (difference in pFCR) is reported

to be 42.6%. The test rejected the null hypothesis of equal pFCR in both populations, $p = 0.0013$. The test outcome is thus highly statistically significant. The reported control variables failed to explain the difference in fatality rates. For instance, median age was 73 years in Control and 74 years in Treatment. Nevertheless, the authors did not conclude that immunosuppressive and cytotoxic treatment leads to higher fatality rates, as "other confounding factors might explain this effect" (Booth et al., 2020, p.2). They thus do not infer scientific significance directly from the result.

A two proportions Z-test is applied to dichotomous variables. Here, the characteristic of interest is whether the course of disease results in death. The power analysis requires the p-value ($p = 0.0013$), a measure of effect size (absolute difference in pFCR = 0.426), the sample size ($N = 66$), and the standard deviation in the population, which is taken to be known in the Z-test. The latter can here be determined in two distinct ways. Firstly, dividing the combined number of deaths in each treatment, by the combined number of observations N gives an estimate for the pFCR in the population. From here, the standard error can be computed by: $se = \sqrt{(\hat{p} \cdot (1 - \hat{p}) \cdot (\frac{1}{n_1} + \frac{1}{n_2}))}$, where $\hat{p}$ is the estimate for the pFCR. However, to get the same estimate for the standard error as is used in the study, it offers itself to calculate the standard error from the sample function: $z = \frac{\triangle pFCR}{se}$, $\triangle pFCR =$ difference in pFCR between Treatment and Control. Effect size and Sample size are reported in the paper. The difference in both values for the standard error accounts for 0.0085. The resulting difference in p-values is also negligibly small. From here, one can compute the standard deviation: $sd = se \cdot \sqrt{N}$. The post hoc power analysis was conducted similarly to the procedure, outlined in chapter 3.1.2. I computed cohen's $d = (\frac{\triangle pFCR}{sd})$. The study power equals the area under the standard normal curve inside the interval $[c - d \cdot \sqrt{N}, \infty]$, $c = 1 - cdf(p)$. Given the reported p-value ($p = 0.0013$), $power = 58.10\%$. When using the p-value stemming from the measured standard error ($p = 0.0006, se = 0.1239$), $power = 57.62\%$. The marginal deviation in standard errors thus results in a marginal difference in power (0.48%). Since $power < 80\%$ in both cases, the study appears to be underpowered. The results suggest that data collection has been stopped too early, leaving a too small sample, in order to reach conclusive evidence.

3.1.4.1 Interpretability of Post Hoc Power Analyses

Thomas (1997) states that a post hoc power analysis is of particular interest, given the test fails to reject the null hypothesis. The concern might be that a too small sample size accounted for a particularly large probability of obtaining a false negative (see chapter 3.1.1). The smaller the measured effect is, the more observations are necessary for the test to not attribute the effect to random chance. Nevertheless, flexibility in determining the sample size was shown to increase FPR, as well (see chapter 3.1.3.4). It is therefore useful to conduct a post hoc power analysis, as it contributes to making an informed decision about the conclusiveness of results. However, Thomas (1997) shows that using the additional information obtained in the statistical test rather than estimates as in the a priori power analysis, has a large effect on the outcome. The question should therefore be raised as to what information can be included in the analysis, considering that a power analysis is generally exploratory in its nature, whereas a statistical test aims to confirm observations. Can information –

namely the measured effect size, variance and p-value – from the confirmatory analysis be applied to an exploratory analysis to reinstate the validity of results? When using the standardised effect size, reported in the test, power only depends on the achieved significance. That is due to the α- β trade-off. Power is inversely related to β and thus a small significance level leads to a high β-level and low power. Using the reported study results in the retrospective power computation will yield a small study power when the result is deemed highly significant and vice versa. A low power thus reinforces the achieved significance, but fails to provide a satisfactory estimate of the false negative probability.

A solution, proposed by Thomas (1997) is to simply use an estimate of the effect size in the a posteriori analysis, as would be done in the a priori analysis. If an estimate cannot be provided – for instance, I cannot put forward a reliable estimate of the difference in mortality rates – the minimum effect yielding scientific implications or a range of possible effects can be used.

3.1.4.2 Power Analysis Based on Estimates

The reported d-value equals 0.4231. Using estimates of $\triangle pFCR = \{0.2, 0.4, 0.6, 0.8\}$ results in $d = \{0.1986, 0.3972, 0.5958, 0.7945\}$. Due to a standard deviation close to one, the values deviate only marginally from each other. Using the reported variance and p-value $power$ ={8.11%, 58.54%, 96.63%, 99.97%}. According to these results, the probability of the positive test decision being true in the population is close to 100%, given a true difference of fatality rates of 60%.

Furthermore, the reported variance can also be eliminated from the power analysis, by using a range of standardised effect sizes d instead of dividing a range of absolute effect sizes by the reported standard error, as shown above. Referring to Cohen (1992), I computed power for $d = \{small, medium, large\} = \{0.2, 0.5, 0.8\}$. A medium effect size achieves a power of 85.33%. Figure 3.16 provides estimates for achieved power on the entire range of possible effect sizes.

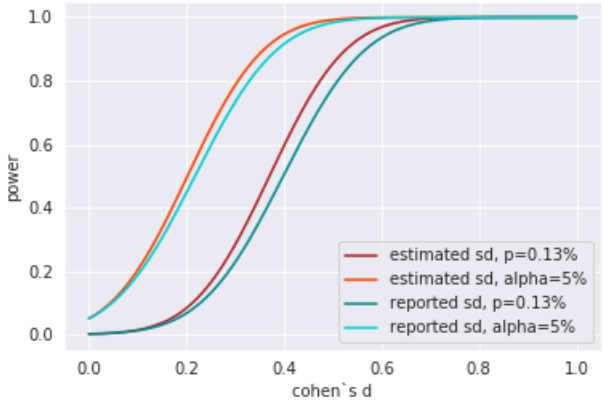

Figure 3.16: *Power analysis.* Post hoc power in dependence of the effect size. (Author's illustration)

Note that a negative effect size would implicate a higher pFCR in Control, compared to Treatment. The maximum difference in pFCR cannot exceed 100% and $d = 1$ is thus the largest possible effect size. Comparing light-coloured to dark-coloured curves reveals the relationship between the significance and power levels. The overall study power is indeed lower at lower significance levels. The orange curves depict power as a function of estimates of d. The blue curves depict power as a function of d, where the absolute effect size is divided by the reported standard deviation. The values for power are marginally lower for estimated standard deviations. That is due to the reported standard deviation being above, but close to 1 ($sd = 1.0761$). Which method is being used to receive an estimate for the achieved power, does therefore play a role. Whether to include the reported standard deviation in the power analysis, depends on how likely that estimate is true in the population. However, determining that is hardly possible for someone who is not a researcher in this specific field. The paper states that reported fatality rates are likely to overestimate the true effect size in the population, since only hospitalised – and thus acutely diseased – patients are captured in the sample. Because $\hat{p}$ is close to 50%, a lower true effect size would imply that the true standard deviation is below the estimate as well. A lower standard deviation results in lower power. Therefore, power based on the estimated variance is also overestimated. In this case, not incorporating the reported variance into the power analysis should lead to more robust estimates.

Another approach which promises to deliver information about the robustness of a study result is reporting confidence intervals. Thomas (1997) suggests them as an alternative to the post hoc power analysis, as they provide an indirect estimate of the power. A confidence interval computes an interval, which the effect size estimate is expected to be in with a certain probability. Booth et al. (2020), for instance, report a confidence interval of $[0.242; 0.61] = 90\%$, meaning nine out of ten samples are expected to show an approximate difference in fatality rate between 24% and 61%. The computation goes as follows: $[\triangle pFCR - c \cdot se; \triangle pFCR + c \cdot se] = [\triangle pFCR - c \cdot \frac{sd}{\sqrt{N}}; \triangle pFCR + c \cdot \frac{sd}{\sqrt{N}}] = (1 - \alpha)$. α corresponds to the probability of obtaining a mean that is outside the stated interval, c depicts the critical value that corresponds to α. The confidence interval thus simply computes the area under the sampling curve with $\triangle pFCR = 0.426, se = 0.1325$.

$$[v_l; v_u] = [0.426 - 0.1325 \cdot c; 0.425 + 0.1325 \cdot c]$$

$$[v_l; v_u] = [0.426 - 0.184; 0.426 + 0.184]$$

$$[v_l; v_u] = [24.2\%; 61.0\%]$$

$$(c_l, c_u) = (-1.3887; 1.3887)$$

This interval estimate thus provides information about the spread of the sampling curve, under which the study power is calculated as well. Although both values deviate from one another, a larger confidence interval will be related to a larger power at the estimated effect size. Because the confidence interval requires the estimated standard error as well, applying it here is not useful either. Nevertheless, it can be useful in other contexts.

3.1.4.3 Concluding Assessment

Is the sample size in the investigated study sufficiently large? The usually applied β-error level is 20%. Just as the α-error level of 5%, this value constitutes an arbitrary convention. Considering the actual achieved power of approximately 58% to 66% (depending on the method of calculating the standard error), the study appears to include too few observations. Whether that is actually the case, depends on the context. A domain expert should rather determine whether a power of approximately 60% can be deemed as sufficient – which means tolerating a β-error level of approximately 40%. Owing to the issues with calculating power from the obtained study results, making a conclusive statement will perhaps not be possible. It is rather advisable, to consult Figure 3.16 instead. That, however, does not make interpretations more accessible for non-domain experts. Nevertheless, fatally underpowered studies can be detected through a post hoc power analysis.

As laid out in chapter 3.1.2, the post hoc power analysis is infeasible given non-parametric studies. For studies using non-parametric tests the here presented procedure can be performed on the strategy using Monte Carlo simulations, proposed by Mumby (2002). A brief overview over the strategy was given at the end of chapter 3.1.2. The formulas, provided by Israel (1992) assume normally distributed sampling distributions, and are thus even more limited in their use. If a paper does not provide all four input parameters necessary for the power calculation, the post hoc study power is infeasible as well. Concluding, whether the sample size was sufficient is then impossible again. The formulas provided by Israel (1992) require rather more than less parameters, e.g., the sampling error. Whether every required parameter can be extracted depends on the applied test design and the information provided in the study. In this case, it was possible to calculate the standard error implicitly. That may not always be the case. A trustworthy study should be expected to report sample size, p-value and either test statistic or effect size. The effect size can often be calculated from the test statistic (Z-test: $(\bar{x}-\mu_0) = t \cdot \frac{sd}{\sqrt{n}}$), ($\chi^2$-test of variance: $s^2 = \frac{\nu \cdot \sigma_0^2}{n-1}$).

Either way, the post hoc power analysis involves a lot effort and is thus not useful to be applied on a large scale. Furthermore, the ambiguity in determining the post hoc power introduces further researcher degrees of freedom that can be exploited as likely as the p-hacking methods presented in this thesis. A scientific field can lose its public credibility once false conclusions are accumulating in the literature body. If disarray about the trustworthiness of published studies arises from improper post hoc power analyses, the credibility of the respective discipline will likely be undermined even more. In addition, chances are that results will rather be compared against a fixed power threshold, than against each other. That practice will reinforce arbitrary error levels. The post hoc power analysis thus appears to be of value only in individual cases, when interest in a specific study result is particularly high.

The analysis cannot provide information about whether intermediate testing or deleting outliers was conducted. Consequently, inflated FPR cannot be detected either (see chapter 3.1.3.4). Intermediate testing seems most feasible if data is being collected over a period of time, rather than, for instance,

in a singular experiment. Booth et al. (2020) had the additional incentive to conduct intermediate testing because in the face a rapidly spreading pandemic, conclusions about the fatality in certain groups are a matter of public interest. Wicherts et al. (2016) expect a well-conducted study to meet several criteria. A prospective power analysis should be conducted, and the targeted sample size should be reported. The starting and end point of data collection must be reported. The population from which is sampled and how the participants are being sampled must be stated. Booth et al. (2020) fail to specify a targeted sample size, along with a power analysis. Trying to reach a targeted sample size may not be feasible because the time-period required to reach the target heavily depends on correctly estimating the near future development of the pandemic. However, starting and end point of actual data collection are reported (01 March 2020, 06 May 2020). The population from which is sampled and how participants qualify for the study are being described vigorously. Booth et al. (2020) say, for instance, that in order to be included in the study,

> "patients were required to be hospitalised and positive for SARS-CoV-2 RNA by ... quantitative polymerase chain reaction ... or with clinical and radiological features consistent with COVID-19, where the clinical team judged COVID-19 was the most likely diagnosis. All patients had a current haematological malignancy under ongoing treatment or in clinical follow-up" (p.477).

All patients were pooled from a "regional cancer network" of 7 hospitals (Booth et al., 2020, p.477).

I conclude that the study shows no signs of intentional or unintentional attempts to p-hack, regarding the sample size and sampling plan. However, as laid out, deleting outliers and intermediate testing are simply impossible to detect. Not being familiar with medical studies in general, I interpret a power of 90% around the measured effect size ($\alpha = 5\%$) to be very high. The power regarding the true effect size in the population might still be insufficiently low. However, there are no conclusions made about the fatality rate in the population. In fact, it is stated that the true fatality rate is expected to be lower than the measured rate in the hospitalised patients. Furthermore, I could replicate the main findings, including the overall pFCR, the reported effect size estimate, confidence interval, and result from the two proportions Z-test. An accurate test design was chosen, given the goal of the study.

A number of control variables was collected, including age, gender, haematological diagnosis, prior and current haematological treatment, method of diagnosis of SARS-CoV-2. The authors themselves conclude that their main findings may be explained by additional variables not captured in the study. They thus conclude that the pFCR in patients with haematological malignancy is significantly higher, but evidence for the pFCR being affected by immunosuppressive and cytotoxic treatment was considered too weak, in order to infer a general relationship. I do not suspect p-hacking in this very conservative set of conclusions. Chapter 3.2 will introduce methods to use additionally measured variables to increase the likelihood of finding significant relationships.

3.2 Flexibility in Variables

In a simple empirical study, the obtained data is usually organised into a simple matrix format. Each row usually contains one unique observation, for instance, a hospitalised patient or proband in a lab experiment. Each column depicts one distinct variable or characteristic, for example, the patients' age, gender, cigarette consumption per day, probands' pay-off in the Ultimatum Game or the laboratory rats' weight. In the previous chapter on flexibility in sample size, only the number of rows in the data matrix was subject to manipulation. In this chapter, only the number of columns in the matrix will be manipulated.

An empirical study employs different types of variables, introducing further flexibility to exploit. Two main types are independent and dependent variables, also known as input and output variables. The independent variable is used to explain observations made in the dependent variable. For example, Booth et al. (2020) observe the preliminary fatality case rate (pFCR), which thus is the dependent variable. The independent variable that attempts to explain the observations made in the pFCR is the respective experimental group. The two distinct groups are Control and (cytotoxic or immunosuppressive) Treatment. The independent variable is thus binary, whereas the dependent variable is continuous.

Membership to the treatment group appears to explain an increase in pFCR. However, other variables may explain the observed difference in pFCR, as well. It is therefore possible that in reality, the observed relationship between the independent and dependent variable is due to another causal relationship that has not been accounted for in the analysis. Such variables are generally referred to as confounders, covariates or control variables. For instance, age is known to explain observations in pFCR, regarding patients infected with the SARS-CoV-2 virus. More precisely, pFCR is higher in high age groups. Therefore, Booth et al. (2020) mention age as one possible confounder, meaning that age might explain the difference in pFCR between Treatment and Control. That would be the case if the mean age of patients differed significantly between Treatment and Control. One should therefore control for age by keeping age constant across all conditions.

How to handle different types of variables depends on the employed analysis. For example, in regression analysis, independent variables are often referred to as predictors, and dependent variables are often called response variables. Whereas the response variables must be continuous, predictors can be of any data type. Here, one controls for covariates by simply including them as predictors in the regression model. Several measures are used to assess the goodness of fit of the resulting regression model. A t-test, for instance, investigates the null hypothesis of the slope of a particular predictor being zero. Consequently, the predictor will fail to explain the response variable if the test fails to reject the null hypothesis. The null hypothesis that the slope of all predictors combined is 0, is being tested by the F-test. Consequently, If the null hypothesis were true, each predictor would fail to explain the response variable. A small p-value on that test would imply a strong relationship between each employed predictor and the response variable.

There are numerous ways to exploit flexibility in variables. Just like the sample size, one can manipulate the number of variables measured. That includes measuring additional variables, as well as discarding variables during analysis. Furthermore, one can assign different roles to measured variables during analysis. Covariates, originally measured to be controlled for, can instead replace the – probably insignificant – independent variable. Therefore, the entire study design can be altered ad hoc. Flexibility does not stop here, however. Concrete decisions over the handling of specific variables are dependent on the analysis itself. In a regression, other choices will be made than in a simple statistical hypothesis test. In order to build sufficient and thus publishable regression models, a battery of statistical tests is usually employed that can be p-hacked just as well. Some, like the F-test are assessing the goodness of fit of the results, others are testing for specific assumptions of the employed model design, like the Goldfeld-Quandt test, Levene's test or White test for heteroskedasticity. Among other factors, the choice of variables strongly influences the outcome of such tests.

To sum up, the range of techniques that exploit flexibility in variables is very broad. The above techniques are providing just a brief overview. Chapter 3.2.1 presents a list of the main p-hacking techniques that incorporate flexibility of variables. Chapter 3.2.2 presents some exemplary simulations that aim to show the extent to which one can forge findings using flexibility in variables.

3.2.1 P-Hacking through Flexible Variables

I here present a list of p-hacking methods that combines parts from Wicherts et al. (2016) and Simmons, Nelson and Simonsohn (2011):

1. Measuring additional independent variables and conditions.

The independent variable in Booth et al. (2020) could differentiate between cytotoxic treatment, immunosuppressive treatment, and no treatment, leading to three experimental conditions. The treatment condition was defined as patients who have received either cytotoxic or immunosuppressive treatment. A more conservative definition would only include patients who have received both treatments at once. By including further types of treatment against haematological malignancies, the number of independent variables can be increased. Consequently, the conditions can be put together ad hoc and very flexibly. The chances of receiving a significant result increase.

2. Measuring additional dependent variables

Suppose one wants to test people's willingness to donate to a certain charity. The researcher sets up a questionnaire, asking for age, gender, political affiliation and further independent and confounding variables. Additional to asking, how much the proband would donate to charity A, one can simply repeat the question for charity B, C and so on. Or one can ask for "willingness to pay"

on a self-report scale. Each of such additional question poses one additionally measured dependent variable. This particular practice is often called “outcome switching” (Wicherts et al., 2016, p.5).

3. Selecting variables as independent variables, dependent variables, covariates afterwards

Booth et al. (2020) reported age as one covariate. The primary goal is to study the effect of cancer treatment on the pFCR. They controlled for age, by testing the effect of the independent variable on age as a dependent variable. If there is a significant observed age difference between Treatment and Control, age could account for the initially observed difference in pFCR. This test could just as well be reported as main study if the initial study did not yield a publishable result. Here, age can thus be handled as covariate or independent variable. Such flexibility increases the likelihood of finding a publishable relationship.

4. Measuring additional covariates

Mehraeen et al. (2020) find that hypertension, older age, and diabetes significantly increase the risk of dying from COVID-19. If Booth et al. (2020) were to increase the likelihood of receiving a publishable result, they could additionally measure blood pressure and blood sugar levels of patients. Either one of the covariates can be reported as independent variable. The likelihood of finding a publishable result thus increases.

Published papers commonly fail to report the used data. Even if data is being published, there is no guarantee that every originally measured variable is still included. Therefore, it is not possible to conclude, whether either one of the four presented methods was exploited to derive at the published results.

3.2.2 Significant Findings Are Certain

Since there is no method to detect or estimate the impact of flexibility in variables, the following simulations rather demonstrate how much the impact can be in some exemplary contexts. Figure 3.17 depicts the probability of positive test decisions ($\alpha = 5\%$) over 1000 simulations. In each iteration, one further variable is added to the data set. Each generated vector follows a normal distribution with $\mu = 0$, $\sigma = 2$. Pearson correlation coefficient tests and independent two-sample t-tests were performed on sample sizes of 30 and 200, respectively. Both tests search for relationships between two distinct variables and thus require two input vectors. At 20 independent variables, the number of possible tests is $\binom{20}{2} = 190$. Figure 3.18 depicts the corresponding total number of positive test decisions.

In either case, the probability of receiving a positive test decision hovers at 5%. That is not

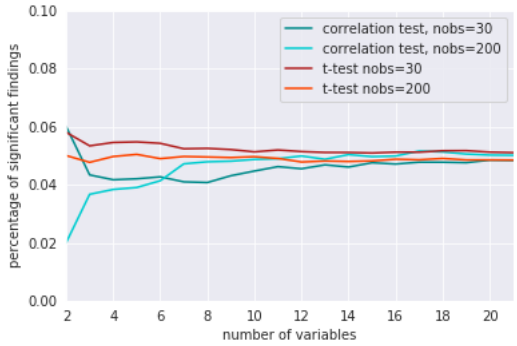

Figure 3.17: *Adding categorical variables.* Probability of obtaining a positive. (Author's illustration)

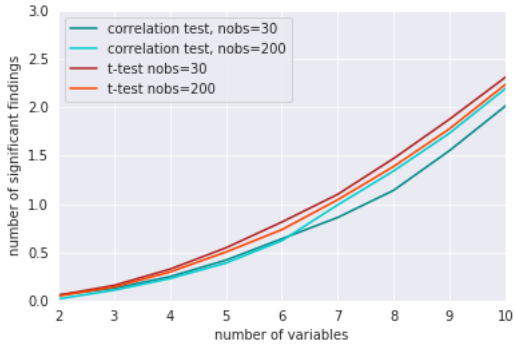

Figure 3.18: *Adding categorical variables.* Absolute frequency of obtained positives. (Author's illustration)

surprising, since the significance level is set to 5%. On average, it takes 7 variables to find one significant relationship if there is no true relationship in the data. There are $\binom{7}{2} = 21$ possible combinations to test. $21 \cdot 5\%$ of outcomes are expected to be positive. Consequently, by simply increasing the number of measured variables, one will almost certainly find a significant result, even at an arbitrarily small significance level. If, for instance, one wants to find a positive relationship at $\alpha = 1\%$, at most 100 possible test combinations will be necessary. Therefore, 15 variables must be collected if none show a true relationship. If using one-sample tests, one requires $\binom{20}{1} = 20$ variables to get $20 \cdot 5\% = 1$ positive finding. The number of observations has no apparent effect on results. Here, a sample size of 30 is sufficient, however. Results might differ on smaller samples. Which test is being performed, has no apparent effect on results, either.

In two-sample tests, the independent variable is binary. In ANOVA the independent variable is at most categorical. The following simulation is taking this into consideration. After generating a vector of 120 random observations, following a standard normal distribution, up to 20 categorical variables are iteratively added to the data matrix. Each categorical variable shall represent an independent variable and has 4 unique values, resulting in 4 possible conditions, including 30 observations each. Two-sample t-tests were performed on each two-subset of conditions, as well as a one-way ANOVA on all conditions simultaneously. The assignment to each condition is random. Positive results are therefore again due to random chance. The probability of receiving a positive result is again close to the significance level of 5%, regarding tests. Whether the added independent variable is continuous or categorical does not alter the rate at which positive findings occur. Nor does the fact that the standard deviation is smaller now. Results can thus be compared to the results from above. Figure 3.19 depicts the development of obtained positives over added independent variables. The blue lines are referring to the correlation test and t-test on 30 observations from Figure 3.18. The dark red line depicts the number of positives in the t-test. The light red line depicts the number of positives in the ANOVA.

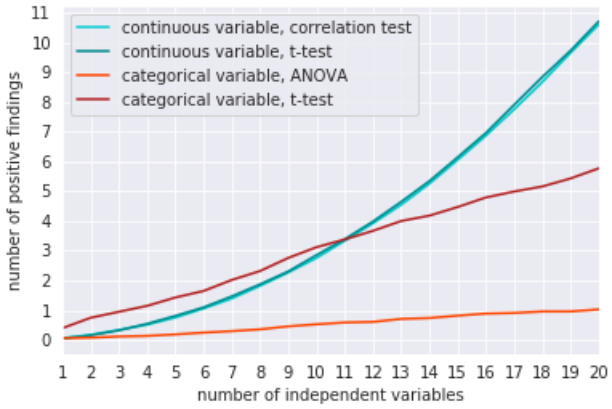

Figure 3.19: *Adding continuous and categorical variables.* Absolute frequency of obtained positives. (Author's illustration)

The number of positive results, given there is no true relationship in the data can be estimated from the number of possible tests. Assume there is no true relationship in the data. Let c be the number of possible combinations, meaning the number of tests, one can perform on the current data set. Let furthermore s be the number of successes, meaning the number of expected positives in the current data set. s is proportional to c, with the proportionality factor α. For continuous independent variables, c is simply the binomial coefficient of $\binom{k}{d}$, where k is the number of independent variables and d is the number of vectors tested (d=2 for two-sample tests, d=1 for one-sample tests). For categorical independent variables, c is $k \cdot \binom{u}{d}$, where u depicts the number of unique values of the independent variable (here, $u = 4$).

$$s = \begin{cases} \alpha \cdot \binom{k}{d}, & continuous \\ \alpha \cdot k \cdot \binom{u}{d}, & categorical \end{cases} \tag{3.11}$$

s = number of successes

k = number of variables

u = number of unique values

α = significance level

d = number of variables per test

Assume, the researcher adds only those variables which guarantee a scientifically interesting relationship. Then, c, at which $s = 1$ is of interest, as it tells the number of combinations in order to find one significant relationship ($1 = c \cdot \alpha \iff c = \frac{1}{\alpha}$). At the usual significance level of 5%, $\frac{1}{0.05} = 20$ combinations are necessary. c is large for large k and u, and small d. Therefore, ANOVA and ANCOVA are less easy to p-hack than two-sample and one-sample tests. Note the slope of the light red line, compared to the dark red line in Figure 3.19. For large k, c does increase quicker by adding

continuous variables. That can be seen in the blue lines compared to the red lines in Figure 3.19. For $d = 2, u = 3$, both methods reach $c = 21$, after including $k = 7$ input variables. For $d = 2, u < 3$ adding continuous variables yields one significant result quicker, and for $d = 2, u > 3$ adding categorical variables yields one significant result quicker. For instance, adding continuous variables yields one positive result at $k = 7, u = d = 2$, whereas adding categorical variables acquired$k = 20$ variables ad $u = d = 2$. Measuring variables that allow to use many conditions is thus most effective.

The simulation focuses on flexibility in the number of variables and does therefore not consider flexibility in variable types. Especially in statistical tests continuous variables can well be used as a dependent variable. That is implicitly considered by simply testing for each subset of two. Consequently, however, categorical variables should have been measured as well to further increase variability of test deigns. Categorical variables can be well used as covariates. The analysis did not consider the possibility of defining one as input variable and then controlling for several others simultaneously, which is common practice, as John, Loewenstein and Prelec (2012) point out. Those practices potentially decrease the number of variables needed to find at least one significant result. Therefore, probably even less variables will be sufficient. Correlations among the data were not taken into consideration either.

All in all, flexibility in variables allows to find publishable results in any study. Without even exploiting the whole spectrum of flexibility, I was able to demonstrate that producing false findings can be done with extraordinary little effort. That is especially true in the behavioural sciences where one additional variable can easily be just an additional question on a questionnaire. In the current publishing practice, flexibility in variables cannot be detected. Estimating the extent to which one can p-hack makes little sense. It is obvious that a researcher will find something publishable if she wants to.

3.3 HARKing

Another major form of p-hacking is Hypothesizing After the Results are Known (HARKing). It refers to the practice of fitting hypotheses to the analysis post hoc, as opposed to first hypothesising about a possible effect and then testing that effect in a confirmatory analysis. There are several reasons, why HARKing poses a threat to the justifiability of results. To mention some of the arguments, proposed by Kerr (1998):

1. HARKed hypotheses are less likely true than predicted hypotheses.

Kerr (1998) concludes that "accommodated hypotheses ... explain the current results, whereas ... a priori hypotheses ... explain a broader set of findings" (p.207). As a result, he expects HARKing to "promote narrow new theory (p.210)" (e.g., theory that fails to make general predictions).

> "One is more likely to tailor a hypothesis to fit a narrow empirical base, dominated by the current result, rather than when one practices genuine prediction, where the in-hand result cannot engulf the attentional field" (Kerr, 1998, p.207).

A machine learning algorithm will never be trained and tested on the same data, as this often leads to overfitting. Whenever training and testing are being performed on one data set, high error rates will occur when the algorithm classifies new data. To counter the problem, engineers split data sets into training and test data. This poses a straightforward analogy to HARKing, as post hoc hypotheses tend to overfit the data at hand.

Another analogy can be found in The Black Swan, a book that investigates highly improbable events (cf. Taleb, 2010). Black swan events have three universal characteristics. Firstly, they have not been predicted by the observer. Secondly, they have a severe impact. Third, post hoc theories will be developed to explain the event and its causes. Such theories incorporate hindsight bias. Their power to predict the event, would they have been developed a priori is generally overestimated. An actual prediction would pose a much more powerful hypothesis than such post hoc hypotheses.

2. HARKing promotes false theory to become scientific consensus.

As explained above, HARKed findings are less likely true than successfully predicted findings. The publication bias promotes HARKed findings to enter the literature. A bias against publishing replications potentially hinders these findings to be falsified. Thus, HARKing promotes false theory to become scientific consensus.

3. HARKed Hypotheses fail Popper's criterion of disconfirmability

Popper (1959) argued that any useful hypothesis must be disconfirmable. Since HARKed hypotheses are tailored to fit already obtained results, these results can obviously not reject the hypothesis.

A HARKed hypothesis is not disconfirmbale – at least until replication – and thus of little scientific use.

4. HARKing promotes information loss.

Besides an apparent bias against publishing null findings per se, exchanging initially stated hypotheses that yield null findings against HARKed hypotheses in the final paper, further promotes information loss. Other researchers are even more likely reproducing unpublished null findings and are therefore more prone to unknowingly publish false positives.

5. HARKing promotes statistical abuse.

Flexibility in the use of one-tailed and two-tailed tests, as analysed below, serves as an example. Exploiting larger rejection areas in directional tests is generally justified upon a priori theory. Disguising HARKed hypotheses as predictions allows for unobservable statistical abuse.

Not every type of HARKing must necessarily be regarded as p-hacking. For instance, THARKing (Transparently Hypothesizing After the Results Are Known) refers to openly reporting that a paper's hypotheses are product of the conducted analysis. Depending on the definition, this practice does not lead to p-hacked results per se. Stating p-values to confirm these hypotheses constitutes statistical abuse, however. Using the same empirical data to produce exploratory findings and to then conduct confirmatory analysis leads to overfitting. As a result, p-values are expected to be lower than if both analyses would have been conducted separately. THARKing can thus be interpreted as p-hacking itself, or as a practice which makes p-hacking visible. Anyway either, HARKing leads to overconfidence in results which are more likely false positives than the p-values suggest.

Detecting HARKing is close to impossible, since there is no way to infer, at what point a hypothesis was being made. A common HARKing practice is, for instance, conducting exploratory analysis. I will demonstrate how this works – and how effective it is as a p-hacking method – in chapter 3.4.1. I will further demonstrate multiple testing, which is another common HARKing technique, in chapter 3.4.3. Measuring additional variables to increase later flexibility in data analysis (as demonstrated in chapter 3.2) poses another HARKing technique. In doing so, one has either advanced multiple hypotheses and reported only those that yield statistical significance, or the reported hypothesis was formulated post analysis. Either way, that is HARKing.

In some cases, however, one might be able to find evidence for a particular HARKing method. In the following chapter, I will produce an estimate for how much p is being affected by that p-hacking method. Furthermore, I will investigate in how far this technique can be detected.

A flexible hypothesis fails to specify effect directions. A2 in Table 3.2 describes a scenario, in

	one-tailed test	two-tailed test
one-sided hypothesis	A1	B1
two-sided hypothesis	A2	B2

Table 3.2: *Flexibility in hypothesis design.* Four possible scenarios. (Author's illustration)

which a one-tailed test is used to study a two-sided hypothesis. B1 describes a scenario, in which a two-tailed test is used to study a one-sided hypothesis. Both represent cases of improper scientific conduct. However, A2 can be particularly exploited.

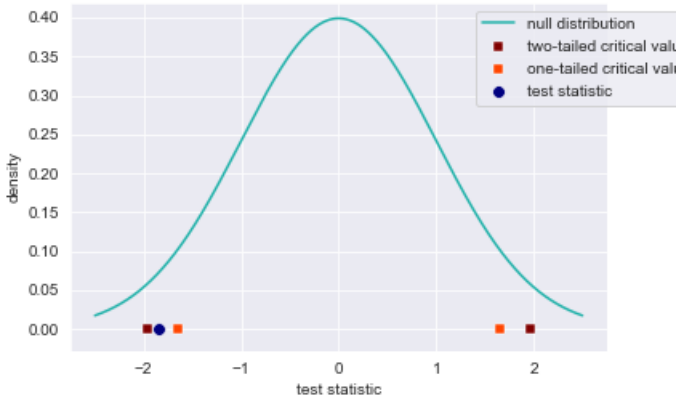

Figure 3.20: *Exploiting flexibility in hypothesis design.* (Author's illustration)

Figure 3.20 shows, how the difference in one-tailed and two-tailed tests can be used to p-hack. A one-sample Z-test was conducted on randomly generated data. The test statistic was computed and compared to the critical values of a one-tailed and two-tailed test design, respectively ($\alpha = 5\%$). Table 3.3 depicts the set of hypotheses, corresponding to the respective test design.

3.3.1 Flexibility in Hypothesis Design

Z-test	H_0	H_1
two-tailed	$\mu = 0$	$\mu \neq 0$
left-tailed	$\mu \geq 0$	$\mu < 0$
right-tailed	$\mu \leq 0$	$\mu > 0$

Table 3.3: *Possible hypothesis designs for the Z-test.* (Author's illustration)

The sample data includes 10000 observations and follows a standard normal distribution. Hence, the null hypothesis is true. In a two-tailed design the area of rejection equals 2.5% on each flank of the distribution. The area of rejection in a one-tailed test is double in size (5%) on the respective flank. In general, there is a 50% chance of randomly hypothesising the correct effect direction and thus choosing

the fitting one-tailed test. That effect and the difference in rejection areas compensate each other, when making genuine predictions. However, that is not the case, once the effect direction is HARKed. Therefore, because the two-tailed critical value is located further away from the distribution centre than the one-tailed critical value, one can exploit A2. Stating a flexible hypothesis allows to conduct all three possible test designs and to report only those that yield significant findings. Furthermore, it is possible to fit an appropriate one-sided hypothesis to the significant one-tailed test, making detection more challenging. Here, the two-tailed test fails to reject the null hypothesis, which results in a true negative. However, the left-tailed test rejects the null hypothesis ($p < 0.05$), resulting in a false positive. Because the effect direction was HARKed by reporting the left-tailed test result, one commits p-hacking.

3.3.1.1 Z-Test Decisions at $\alpha = 5\%$

I started by looking at one of the most common tests in applied research. The Z-test is used for samples, stemming from a normal distribution and computes the deviation from sample mean to expected value under the null hypothesis. In this chapter, I will define this measure as the effect size. Note that effect sizes are usually defined by cohen's d or other means of calculation. The Z-test's very simple design is well-suited to introduce the basic methodology which I use here. The sample function $t = (\frac{\bar{x}-\mu_0}{\frac{sd}{\sqrt{n}}})$ follows a standard normal distribution. The test statistic is thus only dependent on the sample size n and the effect size $(\bar{x} - \mu_0)$. Effect sizes are ranging from (-1) to 1 in 0.01 increments, and sample sizes ranging from 1 to 100. For each combination of effect and sample size, the two- and right-tailed Z-test decisions were computed at the significance level $\alpha = 5\%$. Probabilities were measured by means over 1000 repetitions. In every repetition, a new sample is created, and a new test statistic is computed. In addition to computing the mean, I computed the standard deviation over 1000 test decisions. The null hypothesis is defined as $H_0 : \mu = 0$. Therefore, an effect size of $(\bar{x} - \mu_0) = (-1)$ translates into a sample mean of (-1). Unlike the left- and two-tailed Z-tests, the right-tailed Z-test cannot capture negative effect sizes. For better visualisation, I computed test decisions only at positive effect sizes, however.

Figures 3.21 and 3.22 depict probabilities of positive Z-test decisions at $\alpha = 5\%$ in the one-tailed and two tailed test, respectively. The red area depicts the combination of effect and sample size, in which the test decision is ambiguous. If the position of the red area alters between both Figures, flexibility in hypothesis design can yield publishable results. Effect sizes, in which the test decision is ambiguous can thus be exploited, in general. Although both Figures are largely similar, the range of exploitable effect sizes is tilted towards higher values in Figure 3.22. Hence, there is a deviation between both red areas along the effect size axis, which allows for p-hacking by always preferring one-tailed tests. Furthermore, the two-tailed test contains less positives at low sample sizes. In order to obtain significant results, one should perform one-tailed test, especially on small samples. The range of exploitable effect sizes seems to converge towards a fixed interval with increasing sample size. For one-tailed tests, that interval seems to converge faster.

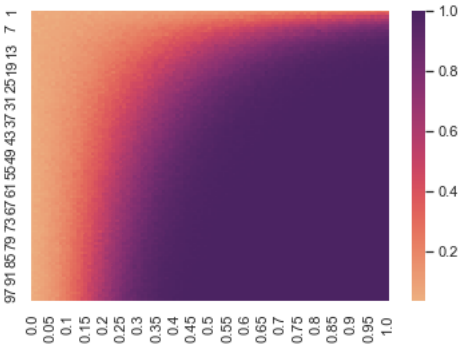

Figure 3.21: *Exploitable effect sizes.* Probability of obtaining a positive in a one-tailed Z-test. Effect size is on the horizontal axis. Sample size is on the vertical axis. (Author's illustration)

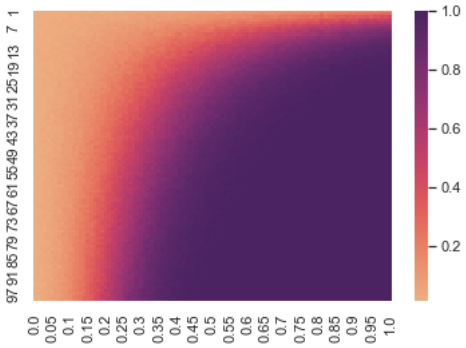

Figure 3.22: *Exploitable effect sizes.* Probability of obtaining a positive in a two-tailed Z-test. Effect size is on the horizontal axis. Sample size is on the vertical axis. (Author's illustration)

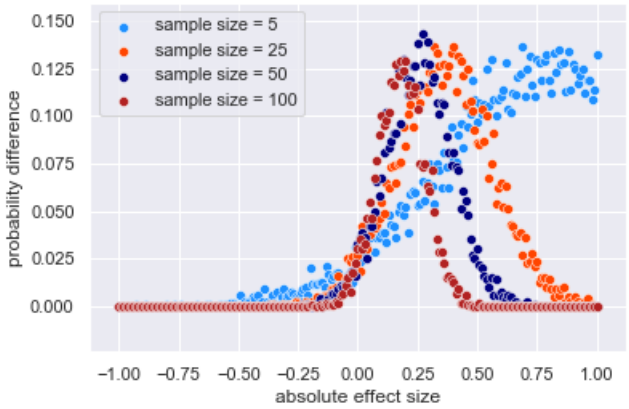

Figure 3.23: *Exploitable effect sizes.* Probability difference of obtaining a positive. (Author's illustration)

Figure 3.23 shows the difference in probability of obtaining a positive, regarding both test designs. A bell-shaped curve can be recognised, whose spread is smaller at larger sample sizes. The null hypothesis of the data originating from a normal distribution could be rejected (Omnibus test at (sample size, test statistic, p) = $\{(5, 189.0364, 8.9385 \cdot 10^{-42}), (25, 32.7869, 7.5932 \cdot 10^{-8}), (50, 62.8609, 2.2383 \cdot 10^{-14}), (100, 97.8254, 5.7210 \cdot 10^{-22})\}$) (cf. Dagostino, 1971). There is a difference in ambiguous effect sizes at sample size = 100, as can be seen in the red scatterplot. This finding corresponds to the deviation between both red areas in Figures 3.21 and 3.22, mentioned before. In the red scatter plot, that deviation can be found inside the $[-0.125, 0.375]$ interval, which can be interpreted as the interval of exploitable effect sizes. The probability difference represents the increase in probability of obtaining a positive result, when using a one-tailed instead of a two-tailed test design. The area left

to the $[-0.125, 0.375]$ interval of exploitable effect sizes, shows no probability difference, because the overall probability of obtaining a positive result equals 0% in both test designs, at a sample size of 100. The area right to the $[-0.125, 0.375]$ interval of exploitable effect sizes, shows no probability difference because the overall probability of obtaining a positive result equals 100% in both test designs. This relationship can be found in the yellow and purple areas in Figures 3.21 and 3.22 at sample size = 100.

Higher sample sizes correlate with smaller exploitable effect size intervals, as well as less noise in the respective curves. For instance, at sample size = 100, using a one-tailed rather than a two-tailed test increases the probability of obtaining a positive by $\leq 15\%$ at an effect size ≤ 0.25. A probability > 0 can be seen in the $[-0.125, 0.375]$ interval. For sample size = 25 the interval is larger. The maximum probability difference is reached at effect size ≥ 0.25.

In order to make an educated guess, at which point the interval of exploitable effect sizes converges, I ran a second computation with sample sizes ranging from 100 to 1000. The interval does seem to converge at sample sizes of 500 and larger. When computing the mean of test decisions over samples of sizes between 500 and 1000, the interval of exploitable effect sizes equals $[-0.08, 0.27]$. The mean maximum probability difference of test decisions between both test designs equals 12.42% at an effect size of 0.07. Reporting definite boundaries – as well as a maximum probability difference – appears to be impossible, due to limited computing power, and too much noise in the data. The maximum probability difference fluctuates, but seems to be independent of the sample size. It seems appropriate to estimate the probability difference to not exceed 15%, although maximum probabilities of 25% and larger were measured. Noise in the data might explain the latter value. Producing p-hacked results by applying flexible hypotheses is generally possible, since the interval of exploitable effect sizes does not converge towards 0.

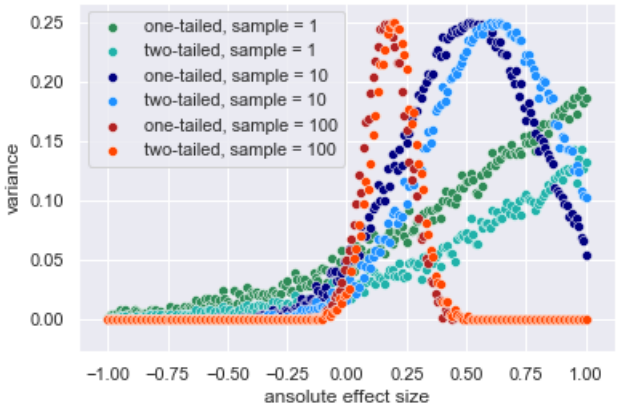

Figure 3.24: *Exploitable effect sizes.* Variance of the probability of obtaining a positive. (Author's illustration)

Additional to mean test decisions, I computed variances of test decisions for all combinations

of effect and sample sizes. Figure 3.24 captures the variances regarding both test designs. The maximum variance appears to be 0.25, independent of the sample size. It is reached at the equal point in t, at which the probability difference between both test decisions peaks. Outside the interval of ambiguous test decisions, the variance is 0.

Findings can be extrapolated to left-tailed tests. Exploitable effect size intervals can be mirrored at effect size = 0. Values for maximum probability and variance remain unchanged.

In the analysis above, I computed the test statistic from randomly created samples. In the following, I will use an array of incrementally increasing values for the test statistic t. In the Z-test, t is dependent on effect and sample size (see chapter 3.1.3.1). In tests using more complex sample functions, t can be dependent on more parameters, making visualisation of results more complicated. Reducing the number of input variables thus helps to interpret and visualise the following results. Furthermore, because t is not dependent on random samples anymore, results are not dependent on probabilities anymore. Each test requires one predefined sample function which defines the probability distribution under the null hypothesis. Given the data in the sample, this sample function computes one unique test statistic. The test outcome solely depends on the position of t under the null distribution. Hence, each unique value for t corresponds to one unique p-value (considering only positive or negative values for t, at once), under one prespecified null distribution. By iterating over all values for t that correspond to p-values = $[0, 1]$, every possible test outcome is captured. Additionally, measuring p-values allows for analysing a spectrum of possible test decisions, instead of probabilities of positives at only one significance threshold. I measured right-tailed (p_1) and two-tailed (p_2) p-values at varying test statistics (t) for the standard normal distribution, t-distribution, χ^2-distribution and F-distribution, respectively.

3.3.1.2 Standard Normal Distribution

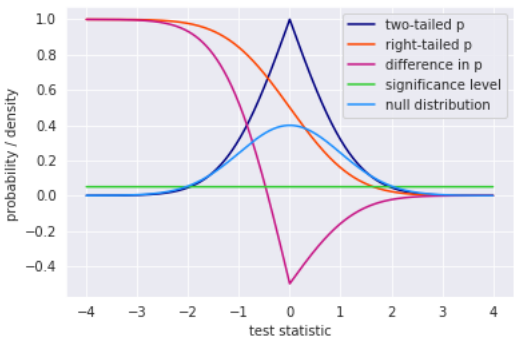

Figure 3.25: *Standard Normal Distribution.* P-Values in dependence of the test statistic. (Author's illustration)

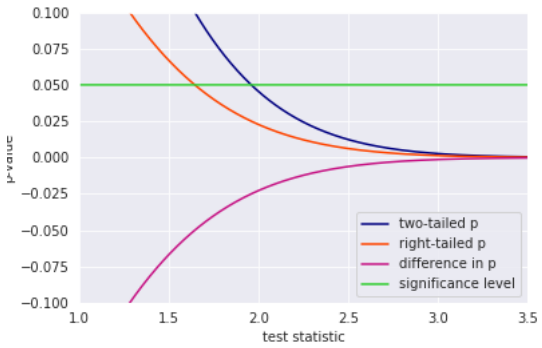

Figure 3.26: *Standard Normal Distribution.* Exert of Figure 3.25 at the significance level. (Author's illustration)

Figure 3.25 shows two-tailed and one-tailed p-values in dependence of the test statistic. Note that

the y-axis depicts density only for the standard normal distribution under the null hypothesis. For all other curves, the y-axis depicts p-values. The expected value of t equals 0 ($E(t) = 0$) under the null hypothesis, at which point $p_2 = 100\%$. The probability of rejecting the null hypothesis, although it is true, is 100% because $t = 0$ indicates that the sample data shows zero discrepancy towards the two-sided null hypothesis. With t deviating from zero, the discrepancy increases and the probability of falsely rejecting the null hypothesis decreases. p_2 therefore decreases accordingly. A right-sided null hypothesis expects $t \leq 0$. Hence, p_1 approaches 1 for decreasing values of t. Higher discrepancies between data and null hypothesis become apparent through larger absolute values of t. As a result, the probability of obtaining a false positive decreases and p_1 decreases as well. Here, p_1 depicts a right-tailed curve of p-values and thus mirrors the curve of a left-tailed test at $t = 0$. The difference curve shows a maximum difference of 50%, at $t = 0$, at which both p-values are insignificant, however. A general relationship between both p-values can be observed. For right-tailed tests: $p_2 = 2 \cdot p_1, t \geq 0$. For left-tailed tests: $p_2 = 2 \cdot p_1, t \leq 0$.

Figure 3.26 shows an extract of Figure 3.25. The difference in p-values is exploitable at the point in t, where p_1 drops below the significance level α. p_1 converges towards 0 for increasing t. At a test statistic of $t = 1.6450$, p_1 is significant, whereas $p_2 = 10\%$. Reporting two-tailed test decisions results in significant p-values at $t \geq 1.9600$, at which point $p_1 = 2.5\%$. Thus, reporting p_1, instead of p_2 pays off inside the interval $t = [1.6450; 1.9600]$. P can thereby be cut by 2.5% to 5%. Note that $p_2 = 2 \cdot p_1$. In the following, I will refer to this type of interval as the interval of exploitable test statistics – similar to the interval of exploitable effect sizes, mentioned above. For larger values of t, the potential costs of misreporting might outweigh the benefit of obtaining more significant results, since both test designs are generally publishable. Nevertheless, p can still be reduced by $\leq 2.5\%$. The difference in p accounts for $(p_2 - p_1) = 0.13$, at $t = 3$. The interval of exploitable test statistics occupies an area under the curve of the null distribution of 5%, meaning there is a 5% chance that a flexible hypothesis is possible, given the null hypothesis is true. Unfortunately, this value has little significance, as it only holds true, given the null hypothesis is true. In that way, it inherits the same questionable characteristic as the p-value: It is not a definitive probability value. In reality, no conclusive statement can be made about the true probability distribution of t in any experiment.

3.3.1.3 t-Distribution

I analysed the t-distribution on 0 to 30 degrees of freedom. The output parameter p is now dependent on two input variables: test statistic (continuous) and sample size (categorical). Note, that the t-distribution closely approximates a standard normal distribution at $df \geq 30$. t was computed from 0 upwards, to capture right-sided test decisions. Performing left-tailed tests is also possible, however.

Figure 3.27 shows p-values of one- and two-tailed p-values, regarding different degrees of freedom. The exploitability for $df = 1$ – in which case the sample holds only two observations – lies inside the interval of $t = [6.314, 12.707]$. This interval has a range of 6.393. The interval boundaries

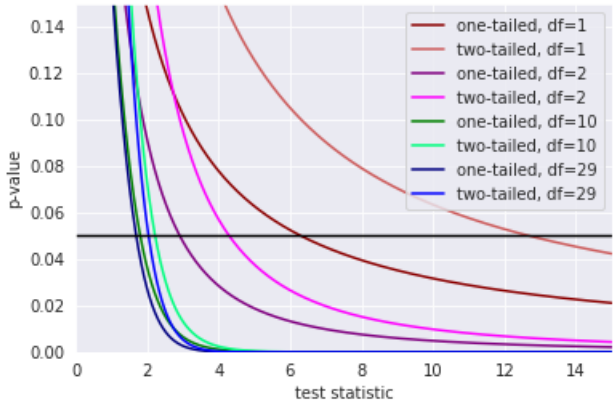

Figure 3.27: *t-Distribution.* P-Values in dependence of the test statistic. (Author's illustration)

and range quickly decrease for larger degrees of freedom. At $df = 30$ the exploitability remains inside $t = [1.698, 2.043]$, which closely resembles the interval for the standard normal distribution. p_2 is again 5% higher when $p_1 = 0.05$ and p_1 is again 2.5% higher when $p_2 = 0.05$. That finding is independent of the sample size. The relationship, observed in the normal distribution holds again: $p_2 = 2 \cdot p_1, t \geq 0$, right-tailed tests. A table which contains all relevant values for $df = [0, 30]$ can be found in Appendix A. For tests using a t-distribution of higher degrees, please refer to the results of the standard normal distribution.

3.3.1.4 χ^2-Distribution

The χ^2-distribution requires the same input variables as the t-distribution. The analysis is conducted similarly. Note, that the χ^2-distribution is only defined on t $\geq$ 0.

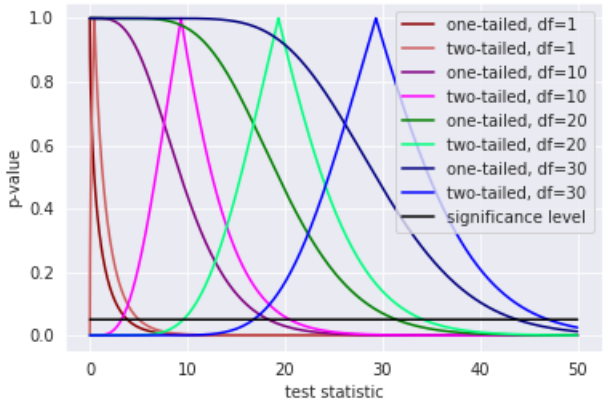

Figure 3.28: χ^2*-Distribution.* P-Values in dependence of the test statistic. (Author's illustration)

Figure 3.28 shows one- and two-tailed p-values, regarding different degrees of freedom again. As

can be seen again: $p_2 = 2 \cdot p_1, t \geq 0$. This relationship holds true only for values of t, right from the corresponding expected value in the null distribution – which is at the point in t, where p_2 peaks in Figure 3.28. Unlike the null distributions which are being normally or t-distributed, the χ^2-distributed null distribution does not necessarily have its expected value at $t = 0$. Unlike the t-distribution, the boundaries and range of the interval of exploitable test statistics increase with increasing degrees of freedom. The χ^2-distribution can be converted into the normal distribution, at least approximately for $df \geq 30$. That is crucial, because the relevant interval range does not converge towards 0. For $df = 1$ – in which case the sample holds only two observations – reporting p_1 instead of p_2 produces significant results inside the $t = [3.842, 5.042]$ interval. That interval equals $[43.773, 46.980]$ for $df = 30$. Hence, the range increases from 1.182 up to 3.207. p_2 is again 5% higher when $p_1 = 0.05$ and p_1 is 2.5% higher when $p_2 = 0.05$. Like in the t-distribution, the delta in p is independent of degrees of freedom. I created a table that contains all relevant values for $df = [0, 30]$ (see Appendix B). For tests using a χ^2-distribution of higher degrees, please refer to the results of the standard normal distribution.

3.3.1.5 F-Distribution

When analysing the F-distribution, a second degree of freedom is added to the number of input variables: t (continuous), df1 (categorical), df2 (categorical). Furthermore, it is not possible to approximate the distribution – and thus any findings – through the normal distribution at certain degrees of freedom. However, effects become marginal with increasing degrees of freedom. I computed curves of p-values, regarding every combination of df1 and df2 between 0 and 30, and test statistics = $[1, 50]$. Like the χ^2-distribution, the F-distribution is only defined on $t \geq 0$.

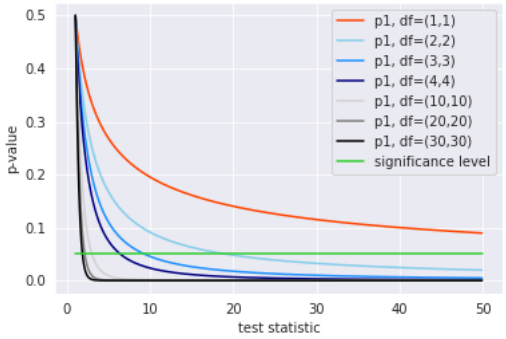

Figure 3.29: *F-Distribution.* One-tailed p-values in dependence of the test statistic. (Author's illustration)

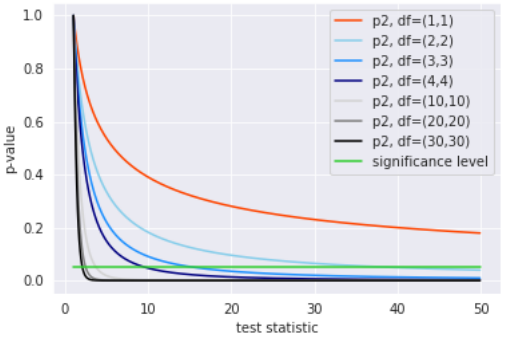

Figure 3.30: *F-Distribution.* Two-tailed p-values in dependence of the test statistic. (Author's illustration)

Figures 3.29 and 3.30 show one-tailed and two-tailed curves of p-values at varying degrees of freedom. The behaviour of p_1 and p_2 is similar, if not explicitly mentioned otherwise. The p-value decreases with increasing values for t and df, in general. For $df1 = df2 = 1$ p never reaches significance

levels of 5% or less. The difference in curves is marginal at $df \geq 30$.

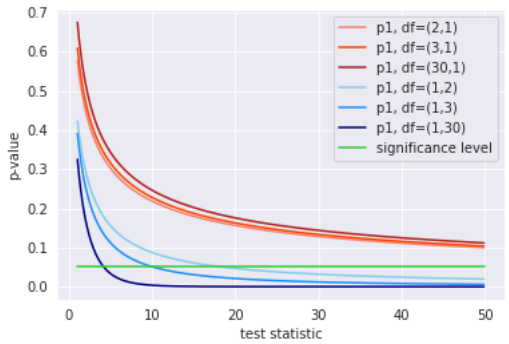

Figure 3.31: *F-Distribution.* One-tailed p-values in dependence of the test statistic. (Author's illustration)

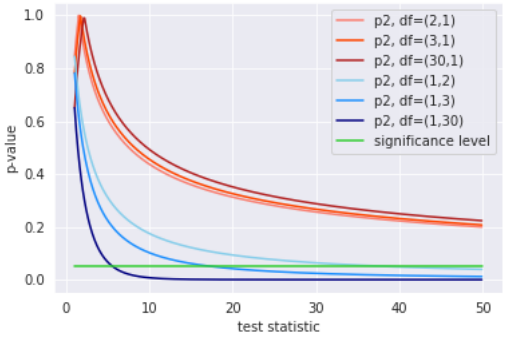

Figure 3.32: *F-Distribution.* Two-tailed p-values in dependence of the test statistic. (Author's illustration)

Figures 3.31 and 3.32 show one-tailed and two-tailed curves of p-values for $df1 = 1$ and $df2 = 1$, respectively. At $df2 = 1$, p never reaches the significance level α, whereas at $df1 = 1$, p always does. The peak in two-tailed curves resembles the expected value in the corresponding null distribution. The asymmetrical shape of the curve indicates that the F-distribution is asymmetrical as well – which is particularly the case at lower degrees of freedom. Because measurements start at $t = 1$, the peak is often cut out in the plot. There is no significant drop in p-values for$t \geq 50$.

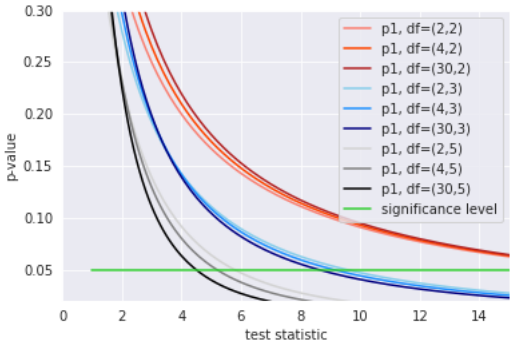

Figure 3.33: *F-Distribution.* One-tailed p-values in dependence of the test statistic. df1 increasing, df2 constant. (Author's illustration)

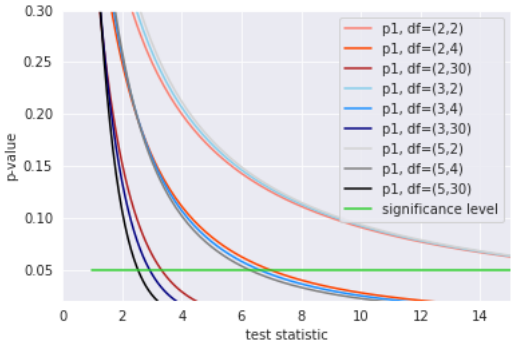

Figure 3.34: *F-Distribution.* One-tailed p-values in dependence of the test statistic. df1 constant, df2 increasing. (Author's illustration)

Figure 3.33 shows one-tailed p-values at increasing df1, given df2 is held constant. For $df2 = 2$, p increases with increasing df1. This behaviour reverses at a certain value for t, given $df2 = 3$. For most values of t, p decreases with increasing df1, at $df2 = 5$. The same findings can be seen for

p_2. However, this is not the case vice versa. Figure 3.34 shows p-values at increasing df2, given df1 is held constant. Here, increasing df2 always correspond to decreasing p-values, regardless of df1. In the following, I will focus on values at the significance threshold. This allows to further reduce complexity in presenting results, leaving out the irrelevant behaviour of p-values that are too large to be published by any means. Given p_1 is significant, I measured p_2 at equal t and vice versa. For every drop of p_1 (p_2) below 5%, I therefore computed p_1 (p_2) at t and the corresponding other p-value p_2 (p_1) at t.

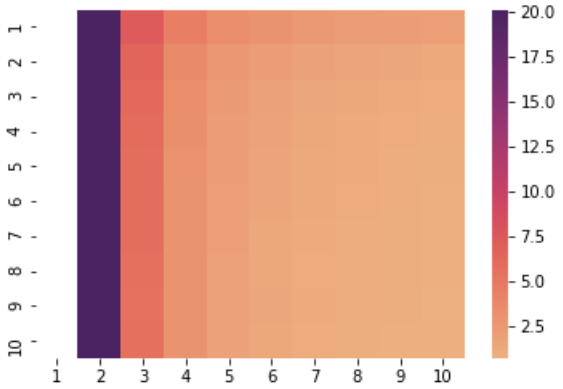

Figure 3.35: *F-Distribution.* Interval range of exploitable test statistics. df1 on vertical axis, df2 on horizontal axis. (Author's illustration)

Figure 3.35 depicts the interval range of exploitable values, by subtracting t for $p_1 = 5\%$ from t for $p_2 = 5\%$. The horizontal axis depicts df2, whereas the vertical axis depicts df1. Note, that for $df2 = 1$ there are no significant p-values. The interval range roughly equals 20 at $df2 = 2$ and drops to values between 5 and 10 at $df2 = 3$. The range decreases along the df1 axis as well, although not much. There is nearly no change in interval size for $df \geq (10, 10)$. The actual interval limits can be visualised in a multi-grid plot, whose size would exceed practical use, however. I thus decided to condense the information from ten times ten degrees of freedom into one plot.

Figure 3.36 depicts a multi-plot grid, containing all 900 combinations of degrees of freedom, with df1 along the vertical axis and df2 along the horizontal axis. Each cell contains all combinations of degrees of freedom inside an interval range of 10. For instance, the cell in the centre depicts $df1 = [10, 19]$ and $df2 = [10, 19]$, the cell below depicts $df1 = [20, 29], df2 = [10, 19]$. The blue plot depicts all values for $p_1 = 5\%$ and the dark blue plot depicts the corresponding p2-values at $p_1 = 5\%$. The orange plot shows values for p2 = 5% and the dark red plot depicts the corresponding p_1-values at $p_2 = 5\%$. p_2 at $p_1 = 5\%$ appears to be around 2.5% and p_1 appears to be around 10% at $p_2 = 5\%$. The findings suggest that again, $p_2 = 2 \cdot p_1$ which turns out to hold true. This characteristic can thus be observed in all four distributions which is no surprise, considering that the t-, χ^2- and F-

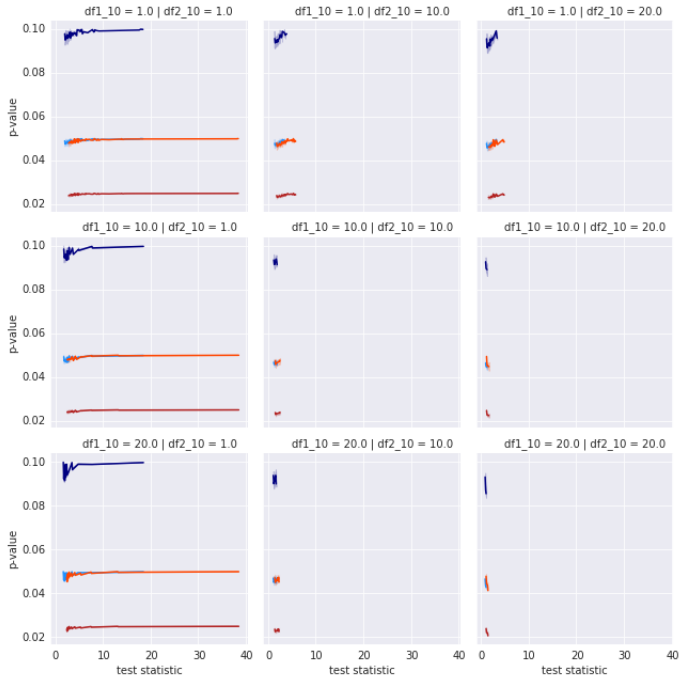

Figure 3.36: *F-Distribution.* P-Values in dependence of the test statistic. Each cell depicts combinations of 10 degrees of freedom. (Author's illustration)

distribution are based on the normal distribution. For $df2 < 10$ the test statistic, at which $p = 5\%$ ranges from 0 to 40. For $df1 < 10$ and $df2 = [10, 30]$ the test statistic, at which $p = 5\%$ ranges from 0 to 10. At $df \geq (20, 20)$ there is nearly no more movement in t. Just as the range of exploitable values for t decreases quicker along the df2 axis, the values for boundaries of this interval decrease along the df2 axis quicker as well. As mentioned before, t is large at small degrees of freedom.

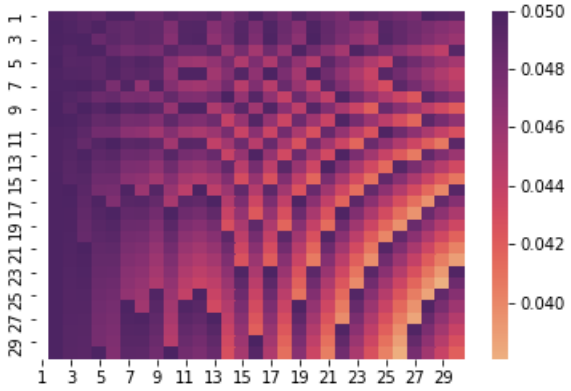

Figure 3.37: *F-Distribution.* Values of fetched p-values from Figure 3.36. df1 on vertical axis, df2 on horizontal axis. (Author's illustration)

Figure 3.37 shows the actual values of plotted p_1-values in the multi-plot grid. The plotted values are

computed for each t in 0.1 increments. The data in the plot is therefore not continuous. If a p-value of 5% lies between two computed values of t, the lower value of p is plotted. However, the orange plot mostly overlays the blue plot, showing captured p_1- and p_2-values to be very close to each other and 5%. There is still noise in the data, especially for small values of t, as can be seen in the fluctuations of line plots. Figure 3.37 confirms that finding. p_1-values only deviate by roughly 1% for particular degrees of freedom. That is true for p_2 as well (see code in Appendix U). Fluctuation in the blue plot automatically translates into fluctuation in the dark blue plot and fluctuation in the orange plot automatically translates into fluctuation in the dark red plot, as the dark and light plot take p-values at equal t. If, for instance, p_2 deviates from 5% and therefore corresponds to a slightly too high t, the corresponding point in the p_1-plot is taken at the same high value of t and will thus also deviate downwards. The fact that fluctuations especially occur around low test statistics, can be traced back to the shape of curves at high degrees of freedom. p-values drop below 5% for comparably low values of t with a comparably steep slope. Therefore, fluctuations are much larger at high df.

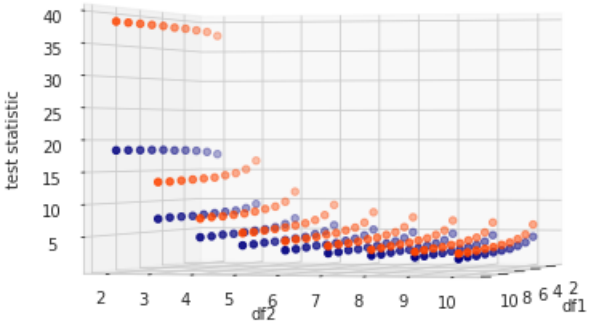

Figure 3.38: *F-Distribution.* Boundaries of interval of exploitable test statistics for low degrees of freedom.

A maximum t of 40, combined with a maximum range of 20 indicates that t is close to $[20, 40]$ for low df. Figure 3.38 confirms this finding. Blue points depict t at $p_1 = 5\%$ and red points depict t at $p_2 = 5\%$. At $df1 = df2 = 10$, t ranges from 2 to 2.8. For a comprehensive list of boundaries and range of the interval of exploitable test statistics, please refer to Appendix C. At $p = 5\%$ and $df2 = 2$, t increases along df1. At $df > 2$, t decreases along df1. That resembles the ambiguous dynamic of t along df1, presented in Figures 3.33 and 3.34.

3.3.1.6 How to Detect Flexibility in Hypothesis Design

In general, if t is not inside the interval of exploitable test statistics, exploiting a flexible hypothesis was not possible. Otherwise, p could have been cut by up to 5%. There are three ways to exploit the differences in the two-tailed and one-tailed test design. One way is to state a two-sided hypo-

thesis and to then conduct a one-tailed test. Wicherts et al. (2016) list this practice as a separate p-hacking method. If the test design is not explicitly mentioned in the study, one can consult the tables, provided in the Appendices A, B, and C. If t is inside the interval of exploitable test statistics, p is only significant in the right-tailed test. p in the according two-tailed test is twice as big.

The second approach is to simply hypothesise about the effect direction, although a two-sided hypothesis would make more sense. For instance, the F-test in regression analysis advances a two-sided hypothesis, since a deviation from 0 in either direction has equal implications. A researcher should therefore choose the scientifically more useful hypothesis design. Furthermore, whereas the one-tailed test has more statistical power, positives from the two-tailed test are more certain, due to the more restrictive critical values (for more information on test power, see chapter 3.1.2). A researcher should therefore weigh both characteristics when planning the study. Simply choosing the hypothesis that yields a significant outcome is p-hacking. This practice can also be detected by consulting the tables in the Appendices A, B, and C. Again, if t is inside the interval of exploitable test statistics, p is only significant in the right-tailed test and p in the two-tailed test is twice as big.

The third approach is to perform the test in either effect direction. The hypothesis can then be fitted to either test that yields a significant finding. This approach is a typical form of HARKing and does therefore differ, compared to the second approach. However, especially if the one-sided hypothesis does make sense, the reader will find no noticeable difference. Unfortunately, even if t is inside the interval of exploitable test statistics, one cannot conclude that HARKing was conducted. The researcher could have genuinely predicted the reported hypothesis.

In the case of a normally or t-distributed null hypothesis, the negative interval of exploitable test statistics can be used to check for a left-tailed test design. For the χ^2- and F-distribution this is not possible, unfortunately. However, left-tailed tests using these null distributions are rather uncommon. For tests which do not use a null distribution, the presented method to detect flexible hypotheses is infeasible.

It is impossible to guess, whether an individual result was further manipulated, using additional HARKing and p-hacking methods. In fact, by combining p-hacking methods, the likelihood of producing significant results increases drastically. By combining four separate p-hacking methods which produced an individual, average false positive rate (FPR) of close to 10%, Simmons, Nelson and Simonsohn (2011) were able to achieve FPR $\geq$ 60%, at the common significance level of 5%. In chapter 3.4, I will demonstrate that if there is a significant relationship in the data, HARKing will certainly detect it. I will thereby use exploratory analysis (chapter 3.4.1), multiple testing (chapter 3.4.3), and flexibility in hypothesis design. I will furthermore prove that by combining several p-hacking methods, anything can be presented as “statistically significant”.

3.4 P-Hacking Demonstration

In this chapter the discussed p-hacking methods will be demonstrated on a real-world data set, available on (UCI Machine Learning Repository, Retrieved February 15, 2021, from https://archive.ics.uci.edu/ml/datasets/Heart+Disease). Here, a pre-processed version is being used which is available on (Kaggle, Retrieved February 15, 2021, from https://www.kaggle.com/ronitf/heart-disease-uci). This data set has 303 instances and 14 distinct variables – compared to 75 variables in the initial data set. All missing values are replaced. Table 3.4 summarises the used variables in the data set.

variable	description	data type	values
age	age of hospitalised patient	numeric	Range = [29;77] Mean = 54.4 Std = 9.07
sex	Gender of hospitalised patient	binary	0 = female, 1 = male
trestbps	Resting blood pressure in mm Hg on admission to the hospital	numeric	Range = [94;200] Mean = 132 Std = 17.5
chol	Cholesterol levels in mg/dl	numeric	Range = [126;564] Mean= 246 Median = 51.7
fbs	Fasting blood sugar	binary	>120mg/dl: 0 = false, 1 = true
thalach	Maximum achieved heart rate	numeric	Range = [71;202] Mean = 150 Std = 22.9
exang	Exercise induced angina	binary	0 = no, 1 = yes
oldpeak	ST depression induced by exercise (relative to rest)	numeric	Range = [0;6.2] Mean = 1.04 Std = 1.16
target	Presence of heart disease	binary	0 = disease, 1 = no disease

Tab. 3.4: *Variables used in chapter 3.4.*

In chapter 3.4.1 an exploratory analysis will serve as a foundation for the following analysis. Exploratory analysis can also be regarded as straightforward HARKing. This will be demonstrated as well. Another approach to HARKing is multiple testing. Chapter 3.4.3 will demonstrate this approach. Flexible hypothesis design will be incorporated when feasible. Chapter 3.4.3 will exploit flexibility in sample size in various ways. Besides intermediate testing and deleting outliers which are already analysed in chapters 3.1.3.2 and 3.1.3.3, further types of flexibility will be demonstrated. Chapter 3.4.4 will use flexibility in variables to produce significant findings. A newly introduced type of p-hacking will be demonstrated in chapter 3.4.5.

3.4.1 Exploratory Analysis

Exploratory analysis allows to search for relationships in the data. It is used to extract information from data and can thus be compared to data mining. The latter includes further techniques like Bayes Classifiers, Neural Networks and more (cf. Han, Kamber and Pei, 2012). These techniques search for relationships in the data as well. In contrast, confirmatory analysis aims to confirm relationships in the data. Statistical tests do just that. As written in chapter 3.3, searching and confirming findings on one dataset drastically increases the likelihood of finding false positives, due to overfit in the data. Instead, relationships which were detected by exploratory analysis must be confirmed on new data. Failing to do so therefore represents a particularly straightforward HARKing method.

In order to find relationships, one can start with looking at boxplots of numerical variables, confidence intervals of binary variables, and counts of categorical variables. The confidence interval for sex reveals that the about 70% of probands are male. On the other hand, between 10% and 20% of cases are expected to show $fbs = 1$, with 95% confidence.

Histogram plots indicate that age, trestbps, chol, and thalach are approximately normally distributed. Oldpeak is heavily right-skewed, revealing that exercise induced ST-depression is below 0.62 for most patients.

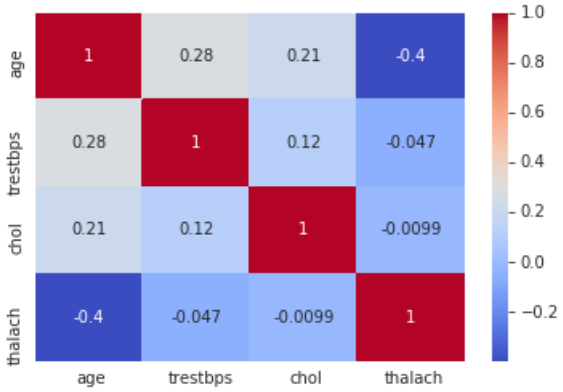

Figure 3.39: *Correlation heat map for selected variables.* (Author's illustration)

Figure 3.39 shows correlations between apparently normally distributed variables. If one searches for significant correlations in the data set, the heat map indicates to start testing with age and thalach. Using the test for correlation from Coblenz and Grothe (2019, p.53), age and thalach are indeed negatively correlated ($statistic = (-7.5386), p = 2.8140 \cdot 10^{-13}$). Furthermore, the same test produces significant results for a positive correlation between age and trestbps ($statistic = 5.0475, p = 3.8811 \cdot 10^{-7}$), age and chol ($statistic = 3.7948, p = 8.9314 \cdot 10^{-5}$), chol and trestbps ($statistic = 2.1534, p = 0.0160$). Here, in the two-tailed test $p = 0.0321$. By incorporating the gained knowledge about the direction

of the relationship allows to halve the reported p-value. Since the sampling distribution of the here presented test is a t-distribution of $(n-2) = 301$ degrees of freedom, this finding is consistent with the reported findings in chapter 3.3.1.3.

The tested hypotheses were not predicted before looking at the data, nor were they derived from exploratory analysis on other data. That the correlation test produced 4 out of 6 significant findings is therefore not surprising, considering the presented values in Figure 3.39. All four significant findings are HARKed and thus p-hacked. HARKing through exploratory analysis has a practical success rate of 100%. The produced results are seriously fraudulent since confirmatory analysis was simply not conducted at all.

3.4.2 Violating Statistical Assumptions and Model Choice

The type of correlation test employed in the exploratory analysis (chapter 3.4.1) is a parametric test. It assumes the input data to be normally distributed, which was never properly tested. Whereas blood pressure (trestbps) and cholesterol level (chol) are approximately normally distributed in the population (cf. Ané, 2006; cf. Ross, 2009), the maximum achieved heart rate (thalach) and the ST depression induced by exercise, relative to rest (oldpeak) are probably not normally distributed. In any case, it helps to test for the presence of a normal distribution. The omnibus test of normality (cf. Dagostino, 1971) rejects the hull hypothesis of the data stemming from a normal distribution in all cases, at the confidence level $\alpha = 5\%$. The Kolmogorov-Smirnov test, as well as the Shapiro-Wilk test produce similar findings. The largest of all 12 received p-values is $p = 0.0126$ in the omnibus test for normality in age, making an exceptionally rare event as an explanation for the test decisions highly unlikely. Therefore, a non-parametric test should have been conducted. Otherwise, the obtained results are p-hacked, due to violation of statistical assumptions. Suitable non-parametric tests are, for instance, the Kendall rank correlation coefficient and the Spearman rank correlation coefficient (Coblenz and Grothe, 2019, p.57). Since the Kendall coefficient is computed on categorical data, the data would have to be binned a priori, resulting in information loss. The Spearman coefficient is the better choice in the given context. As turns out, the Spearman coefficient test classifies all four relationships as significant, as well. The greatest increase in all four p-values is a factor of 3.4 between $p_{spearman} = 0.0006$ and $p_{normal} = 0.0001$. The largest decrease in p-values is a factor of 0.5 between $p_{spearman} = 4.2617 \cdot 10^{-7}$ and $p_{normal} = 7.7623 \cdot 10^{-7}$. The highly significant results could potentially disguise how effective using one method over the other really is on more ambiguous data.

The test deployed in chapter 3.4.1 assumes a bivariate normal distribution. On the one hand, plotting both tested variables against each other indicates a bivariate normal distribution in all four cases. On the other hand, the histogram plots indicated an univariate normal distribution in all variables as well, which proved to be false. Consequently, the Pearson correlation coefficient – which assumes only univariate normally distributed input data – might have been a more suitable test. It assumes a standard normal distribution under the null hypothesis, as opposed to a t-distribution on (n-2) degrees of freedom. The test statistic is derived directly from the equally

named statistic $r = \frac{s_{xy}}{s_x \cdot s_y}$. The test statistic in chapter 3.4.1 multiplies r by $\sqrt{\frac{(n-2)}{1-r^2}}$. The largest difference in measured p-values between these two tests is in the 16th decimal place. Nevertheless, the choice of statistical models influences test decisions and p-values and might furthermore be more effective in other cases. Formally, the choice of the statistical model often poses a case of violating statistical assumptions, as well. It might be misleading, however, to refer to this example as p-hacking.

An example, in which the branding "p-hacking" is clearly misleading is the study which has been investigated in chapter 3.1.4. The Shapiro-Wilk and Kolmogorov-Smirnov tests, performed on patients' age in Booth et al. (2020) reject the null hypothesis of normality ($p = 5.8250 \cdot 10^{-6}, p = 1.3551 \cdot 10^{-84}$, Shapiro-Wilk test, Kolmogorov-Smirnov test, respectively). However, the omnibus test for normality fails to reject that hypothesis ($p = 0.0721$). Whether or not age is approximately normally distributed seems ambiguous. It is not stated, what test was used to control for age. I was able to replicate an insignificant relationship, using t-test ($p = 0.3454$), Z-test ($p = 0.3418$), and U-test ($p = 0.2683$). Wicherts et al. (2016) call the choice of statistical models – and, in fact, every other p-hacking method – a researcher degree of freedom. The choice of model by which the researchers controlled for age, as well as not stating that choice, should be called a researcher degree of freedom, rather than p-hacking, as it does not affect the results. Even in face of these results, other confounding factors are not ruled out to account for the main findings. The researcher degree of freedom does therefore not affect the conclusion either.

One more comment must be made: Conducting several tests for normality and selectively reporting those that fail to reject the null hypothesis constitutes p-hacking, more specifically HARKing through multiple testing.

3.4.3 Multiple Testing

In HARKing, multiple testing and exploratory analysis are closely related techniques. Multiple tests were already conducted in chapter 3.4.1. Nevertheless, it is possible to produce significant findings without even knowing the data, by iteratively testing all possible combinations of variables for significant findings. Here, all possible combinations of two-proportions Z-tests are being conducted. The data is split into two mutually exclusive groups for every binary variable separately. The remaining binary variables can now be tested for difference of means, regarding both groups – and regarding each single group, compared to the entire data set. Out of 27 possible tests, 17 produced significant findings. Some of the significant findings are: Male patients are more likely to suffer from heart disease than female patients ($statistic = (-8.6982), p = 1.6856 \cdot 10^{-18}$). Note that disease = 0, no disease = 1. Therefore, a negative test statistic relates to an effect pointing towards disease. Exercise induced angina (exang) is more likely to be found in male probands than female probands ($statistic = 4.2512, p = 1.0632 \cdot 10^{-5}$). Besides exang = "yes" being detected more often in male than female patients, male patients are also being detected more frequently in exang = "yes" than in exang = "no" ($statistic = 3.3631, p = 3.8533 \cdot 10^{-4}$). Diseased patients are also more likely to be detected in exang = "yes" than in exang = "no" ($statistic = (-1.0926), p = 4.3175 \cdot 10^{-28}$). Male patients

were detected significantly more frequently in the group of diseased patients than in the group of non-diseased patients ($statistic = 8.1341, p = 2.0755 \cdot 10^{-16}$). These findings are suggesting a positive relationship between exercise induced angina, male patients and occurrence of heart disease. The correlations, already computed in the exploratory analysis hinted to that relationship, beforehand. A combination of multiple tests and exploratory analysis seems most successful to produce significant findings. At this point, the researcher can freely decide, which of the findings to report in the paper. That renders detection of such HARKed findings practically impossible.

Furthermore, exercise induced angina is less likely to be found in female patients than in the complete data set ($statistic = (-2.2745), p = 0.0115$), although it is not more likely to be found in male patients, compared to the whole of the data ($statistic = 1.3469, p = 0.0890$). Such findings have little practical use, since one should compare mutually exclusive groups to each other. However, these particular two tests are interesting here because as will be demonstrated in the following chapter, by manipulating the sample size, the p-value on the latter test can be pushed below 5% as well.

3.4.4 Flexible Sample Size

A selection of tests yielding insignificant findings from chapter 3.4.2 is repeated in chapter 3.4.4.1. Chapter 3.4.4.2 attempts to forge new findings. The following methods pose rather intentional p-hacking attempts, as it is highly unlikely that a researcher is not aware of the problem with actively manipulating data in this manner. However, Simmons, Nelson and Simonsohn (2011) observe that this practice can be found with ease, at least in the psychological literature.

3.4.4.1 Deleting Latest Observations and Outliers

Since the data is already available, there is no point in carrying out intermediate tests. Instead, one can iteratively delete the latest observation in the dataset, which is being demonstrated in the following. As pointed out in chapter 3.1.3.3, this method leads to similar results in reversed order. Additionally, significant findings can be found on the largest possible data set, making detection practically impossible. In addition to deleting latest observations, deleting outliers is being demonstrated as well. This particular method does not iteratively delete one outlier in each sample, because binary variables are tested, here. Instead, in every iteration the next observation is being checked for being either 0 or 1. The observation is being removed if that pushes the effect size into the beneficial direction. In the previous tests on the entire sample, one could observe what test direction would be beneficial to lower the p-value. If $p = 1$ in the right-tailed test, then $p = 0$ in the left-tailed test, and vice versa (see chapter 3.3.1.1). Consequently, the effect points into the direction of the smaller p-value. This knowledge about effect directions is being incorporated, here. The analysis thus uses two distinct p-hacking methods at once: Deleting outliers and flexibility in hypothesis design. Conducting two-tailed tests instead, would lead to the same findings, but on potentially smaller samples. Because every employed sampling distribution is a standard normal distribution and the algorithm stops upon receiving $p \leq 5\%$, the test statistic will always be around

1.645 and the two-tailed p-value will always be around 10%. The reported test power is computed on the 5% significance level, using the measured effect size, standardised by the measured standard error.

After deleting the latest 26 observations in each sample, exang is more likely to be found in male patients than in the full data set ($p = 0.0481, power = 100$. Deleting outliers yields a significant finding ($p = 0.0486, power = 100$ on sample sizes of 202 and 302. This relationship was not significant ($p = 0.0890$) in chapter 3.4.2., although exercise induced angina was shown to be less likely in female patients than in the complete data set. That casted doubt on the assumption that there is a true relationship between gender and exang. Now, a researcher could report a "significant relationship". Deleting latest observations removed a total of 52 observations, whereas deleting outliers removed total of 195 observations. Deleting latest observations is thus more than 3 times more efficient, here.

For the remaining tests, the significance level cannot be reached by deleting latest observations. However, deleting outliers always results in a significant finding, even with initially very large p-values. A fasting blood sugar $> 120\frac{mg}{dl}$ (fbs) is present more often in diseased patients ($p = 0.048, samplesizes = (121, 162), power = 100$ than in non-diseased patients. Furthermore, fbs is more often present in male than female patients ($p = 0.0312, samplesizes = (263, 133), power = 100$. By deleting opposite outliers, the exact opposite relationship can be proven. Now, fbs is present more often in non-diseased ($p = 0.0493, samplesizes = (124, 133), power = 100\%$) and female patients ($p = 0.04761, samplesizes = (114, 297), power = 100$.

Whereas intermediate testing appears to be an effective p-hacking method on numerical data, it fails to properly work on binary data. Deleting outliers is a very effective p-hacking method on numeric and binary data. It appears that practically any relationship can be proven "significant" – in both effect directions. Power was always 100%, due to the large sample sizes. Thus, the samples are heavily manipulated, but post hoc power analysis fails to detect the manipulation completely.

3.4.4.2 Multiple Small Studies

Another approach to exploit flexible sample sizes which has not been introduced yet, is to split the data set into multiple subsets and to conduct a given analysis on each subset (cf. Wicherts et al., 2016). This p-hacking method is closely related to multiple testing.

The first example is a test for mean difference in age, regarding the groups of diseased and non-diseased patients. A non-parametric Mann-Whitney U-test is being performed, which relaxes the assumption of normally distributed data, but assumes equal variance in both groups. The Levene's test rejects the null hypothesis of equal variance in both groups ($p = 0.0050$). Performing a U-test on this data is thus not feasible. Instead of ignoring statistical assumptions, I binned the data into three mutually exclusive subsets and performed Levene's test and U-test on each subset, separately. The sample size is 46 in each subset in the group of diseased patients and 55 in each subset in the

group of non-diseased patients. In subset 2, the Levene's test failed to reject the null hypothesis of equal variances ($p = 0.7983$). The corresponding Mann-Whitney U-test rejects the null hypothesis of equal means in both groups ($p = 0.0005$). In this scenario, running multiple small studies yielded a significant finding. Age can now be reported to be higher in diseased probands than in non-diseased probands. Although this conclusion seems unsurprising, it is based on a p-hacked result. The original data does not support the conclusion.

As seen in the exploratory analysis, oldpeak is highly skewed towards small values. Testing for mean difference in oldpeak, regarding the groups of diseased and non-diseased patients, requires performing a non-parametric test again. The assumption of equal variance in both groups, as required for the Mann-Whitney U-test is again being tested, using the Levene's test. It rejects the null hypothesis of equal variance. Figure 3.40 shows histogram plots of oldpeak in both groups. Note that target = 0 refers to diseased probands.

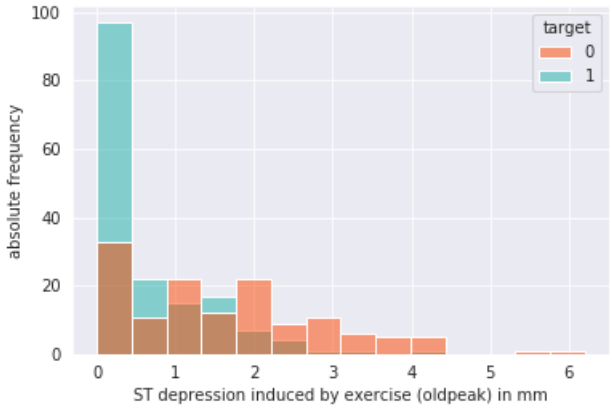

Figure 3.40: *Histogram plot of oldpeak.* Target = 0 corresponds to diseased patients. Target = 1 corresponds to non-diseased patients. (Author's illustration)

The skew is much less in diseased patients, giving support to the hypothesis that on average, oldpeak is higher in diseased people. However, to underpin the hypothesis with a significant p-value, the Levene's test must be p-hacked into deciding negative. Because the strong skew in the data can be found in each subset again, all three Levene's tests are positive. An easy way to still achieve a negative result is to iteratively delete the minimum value in the group of diseased people in each subset. This assures to diminish the standard deviation in the group of diseased people, whilst increasing the mean difference of both groups. Note that multiple small studies, deleting outliers and exploratory analysis are being combined, here. Furthermore, this case shows that p-hacking can also be performed to increase p-values. Achieving that is just as easy. After deleting 7 out of 46 observations in the group of diseased probands, subset 1 produces a negative Levene's test ($statistic = 3.8745, p = 0.0520$). The corresponding U-test on subset 1 is significant ($statistic = 446.5, p = 7.5033 \cdot 10^{-7}$). After deleting a total of 17 observations in the group of diseased probands, subset 2 produces a neg-

ative Levene's test too ($statistic = 3.7698, p = 0.05546$), the corresponding U-test on subset 2 is significant ($statistic = 155, p = 1.8206 \cdot 10^{-11}$). The Levene's tests are negative, on a remaining sample size of 17 in subset 1 ($statistic = 0.2743, p = 0.6021$), subset 2 ($statistic = 0.9982, p = 0.3212$) and subset 3 ($statistic = 3.5924, p = 0.0622$). The corresponding U-tests are positive in subset 1 ($statistic = 33, p = 3.8939 \cdot 10^{-9}$), subset 2 ($statistic = 21.5, p = 4.2821 \cdot 10^{-10}$) and subset 3 ($statistic = 14.5, p = 1.6505 \cdot 10^{-10}$).

The algorithm could have ended upon receiving at least one negative Levene's test decision with a corresponding positive U-test decision. A publishable result was already produced after not more than 7 iterations. However, ending upon producing publishable findings in each subset demonstrates that, if a researcher is willing to actively manipulate the data, any result can be produced. The same algorithm achieves publishable results in each subset, regarding age difference in diseases and non-diseased patients.

3.4.5 Inference from Results

Some of the presented significant relationships are not surprising. The mean age in diseased patients, for instance, is significantly higher than the mean age in non-diseased patients. Patients showing symptoms of exercise induced angina, do suffer from heart disease more often than those who do not show such symptoms. Inferring a general relationship from those findings is reasonable. General inference from other presented findings seems inappropriate, however. For example, male probands are more likely diseased than female probands. There is no reason to assume that this holds true in the population. Furthermore, male patients are more likely to show exercise induced angina, than female patients. That finding certainly stems from the interaction of confounding relationships: Firstly, there are more diseased male than female probands in the data. Secondly, patients showing exercise induced angina are more likely to suffer from heart disease. Inferring this exercise induced angina is more likely to be detected in the male population, rather proves that none of the authors controlled for covariates. The question therefore arises, whether and how general conclusions can be drawn from statistically significant results. Chapter 2.2.1 already touched on the problem of premature inference. The American Statistical Association condemns "practices ... that emphasise the search for small p-values over ... scientific reasoning" as p-hacking (Wasserstein, 2016, p.1). General scientific reasoning seems to be the only reliable method to hinder false inference. After all, Fisher intended this to be part of Significance Testing(cf. Goodman, 1999). The importance of a correct scientific inference becomes particularly clear when one considers that in chapter 3.4.4.1, I produced completely contradictory results on the same data – twice. The following paragraph is written in the format of a typical results section in a published paper. Among other p-hacking methods, especially untrustworthy inference will be demonstrated.

3.4.6 Results and Conclusion in Form of a Published Study

The following two paragraphs are written in the style of typical results and conclusion section from a published paper:

Cholesterol levels, age and resting blood pressure of 96 female and 207 male probands were observed. The measured cholesterol levels are higher in female probands, compared to male probands: $261.3021\frac{mg}{dl}$ versus $239.7545\frac{mg}{dl}$, $difference = 22.0122\frac{mg}{dl} \pm 5.73\frac{mg}{dl}$ (95% confidence interval). The difference in cholesterol levels is highly significant (Welch's t-test, $statistic = 3.0244, p = 0.0015$). Neither of the two additionally measured confounding factors could be proven to account for the difference: The median age of female probands in the sample is 57 years, the median age of male probands in the sample is 54 years. That difference is insignificant (Welch's t-test, $statistic = 1.7163, p = 0.0871$). The observed resting blood pressure is 133.08 mm Hg versus 130.95 mm Hg for female and male probands, respectively. The difference is insignificant again (Welch's t-test, $statistic = 0.9865, p = 0.3247$).

I could prove a true difference in cholesterol levels, between men and women. Two of the typical confounding factors failed to explain the observed difference. Other confounding factors that were not captured in this study might account for the difference in cholesterol levels, however. Further studies will be necessary to investigate the relationship. At present, one possible explanation might lie in different gender-specific diets. Many studies have investigated dieting differences in men and women. Kiefer, Rathmanner and Kunze (2005), for instance, find that "... women are more often affected by problems with their eating behaviour, such as craving for special foods ..." (p.1). The here reported difference in cholesterol levels adds support to that finding, suggesting that the typical female diet is indeed characterised by more cholesterol-rich food, such as dairy and meat products.

An array of p-hacking methods was employed to derive the rather preposterous conclusion that women eat more meat: Exploratory analysis detected no surprisingly different cholesterol levels in any other binary variable than gender. Flexibility in variables allowed to report chol as dependent variable, instead of target. Age and trestbps were reported as confounders, rather than independent variables. Flexible hypotheses were employed to make both covariates appear insignificant. Since high age should correlate with high cholesterol, a right-tailed test would be suitable. I employed a two-tailed test, however. Because the test statistic and sample size were reported, the flexibility in hypothesis design might be detected in the peer-review process. The probability under the null hypothesis follows a standard normal distribution. The reported test statistic lies inside the interval of exploitable test statistics $t = [1.6450; 1.9600]$, reported in chapter 3.3.1.2. The inference from the statistical analysis is prompt and empty of scientific reasoning. As a result, the conclusion is simply false. The cited study from claims that "women eat more fruits, vegetables, cereals, milk, dairy products and whole grain products, whereas the consumption of red meat, particularly pork, sausages, ... is higher in men", which draws an entirely different picture (Kiefer, Rathmanner and Kunze, 2005, p.1). That weak-spot might be detected in the peer-review process. The data itself was not manipulated in any kind.

It seems unrealistic for such a study to end up being published, at first. However, an equally p-hacked and actually published study concludes that "consumption of chocolate with a high cocoa content can significantly increase the success of weight-loss diets" (J. Bohannon et al., 2015, p.2). The point of this study – other than the title "chocolate with high cocoa content as a weight-loss accelerator" might suggest – is to demonstrate how easy it really is to publish seriously p-hacked studies. The author documented everything in an article, published later (Gizmodo, Retrieved February 19, 2021, from https://io9.gizmodo.com/i-fooled-millions-into-thinking-chocolate-helps-weight-1707251800). Bohannon is referencing to a list of questionable publishers, as well (Science, Retrieved February 19, 2021, from https://science.sciencemag.org/content/suppl/2013/10/03/342.6154.60.DC1). He sent the study to some of these journals himself. Papers published in trustworthy journals – especially in prestigious ones – are certainly not p-hacked in such an aggressive manner, as presented here. However, since most p-hacking methods are practically undetectable, forged studies probably slip through every peer-review process, occasionally. Note that p-hacking may not necessarily be conducted intentionally (see chapter 2.2.2).

4 Discussion

P-hacking allows researchers to label irrelevant and even false relationships “statistically significant”, thereby producing publishable findings. Most p-hacking methods turn out to be very effective and hardly detectable.

Power calculations are capable of detecting insufficient sample sizes. They allow the reader to rule out the sizeless stare as an explanation for the reported results. Furthermore, based on the determined post hoc power, one can easily develop estimates for the expected FNR and FDR in a reported finding. FNR is inversely related to TPR. Power is an estimate for TPR (see chapter 3.1.1). FDR takes FPR and TPR as arguments. On non-manipulated samples, p is an estimate for FPR (see chapter 3.1.3.1). The consequences of a flexible sample size can thus be controlled through post hoc power analysis. However, since the sample size itself gives no indication of whether the sample has been manipulated, post-hoc power analysis cannot detect intermediate testing or deleting outliers. Such p-hacking methods have a strong effect on FPR. Between 40% and 60% of psychologists admitted to engaging in such practices (cf. John, Loewenstein and Prelec, 2012).

I thus do not recommend to derive estimates for FDR and FNR from the post hoc power analysis. More importantly, false positives derived from manipulated sample data cannot be detected with the methods presented here. Detecting them requires full transparency about the sampling plan, the sampling procedure, and the sample itself. This is the only way to detect manipulations that ultimately lead to a biased sample and thus to significant p-values.

Flexibility in variables refers to trying out a certain type of analysis on several combinations of variables. The impact of such flexibility on the probability of finding a false positive is hard – and probably even impossible – to estimate. The root cause is the variety of possible techniques in combination with the option to always measure further constructs. Close to 70% of psychologists admitted to selectively report dependent constructs, making it the most common flexibility in variables (cf. John, Loewenstein and Prelec, 2012). Again, only full transparency about the sampling plan, the sampling procedure, and the sample itself allows to detect such p-hacking methods.

Even more transparency is needed to detect HARKing. Readers cannot assess, whether a hypothesis was developed post analysis, based on reported results from that very analysis. Therefore, the hypothesis must be made public even before the actual experiment is being conducted. Flexibility in hypothesis design can be detected, based on the interval of exploitable test statistics. If a one-tailed test has been leveraged, the reported p-value must be corrected by a factor of 2. Whether the

hypothesis was HARKed after searching for an effect in either direction, however, can again only be detected when the hypothesis is made public prior to analysis.

The findings suggest that in order to detect and thereby prevent p-hacking, the current practice of reporting and publishing empirical studies has to change. As Kupferschmidt (2018) points out, preregistered studies are gaining popularity in clinical trials and elsewhere. Preregistrations on the Open Science Framework are roughly doubling each year (cf. Kupferschmidt, 2018). A preregistration can be a loose document addressing fundamental questions about the research question, hypothesis design, sampling plan, and analysis design. In its most advanced form, a preregistered study can pose a fully written introduction and methodology section. The yet incomplete paper will be handed in for peer-review. If it is being admitted, the researchers can then start the actual data gathering, experiment and analysis. The published paper is then only extended by results, discussion and conclusion. Once, such a preregistered study has passed peer-review, it will be published regardless of what findings the analysis will yield. Hence, incentives to p-hack fade. This method appears very effective to detect p-hacking. Flexibility in variables and sample size would be detected prior to publication. If J. Bohannon et al. (2015) had to preregister their study, readers could assess the trustworthiness of the presented findings by themselves, even if the paper were published in a questionable journal. Dal-Ré et al. (2014) hope that transparency and credibility of research will improve through preregistration. Furthermore, because null findings were much easier to be published, this solution addresses the publication bias as well. In fact, the idea of preregistration was initially invented in the face of the publication bias (cf. Couzin-Frankel, 2018).

J. Leek et al. (2017) advocate for a two-stage analysis design to be implemented in the scientific publishing process, as does Andrew Gelman (cf. Nuzzo, 2014). Two-stage analysis clearly separates exploratory from confirmatory analysis and explicitly labels both as such in an online preregistration, such as the Open Science Framework. HARKing would thus become impossible, as readers could simply look up, at what point the reported hypothesis has been advanced. Susan Golding-Maedow reckons that: “preregistration will stifle discovery” (Kupferschmidt, 2018). However, researchers are simply asked to mark their HARKed hypotheses as such, which compares to THARKing (see chapter 3.3). Any hypothesis should be investigated through confirmatory analysis anyways, which does not change through either preregistration or a two-stage analysis.

I suggest one further approach to detect and thus prevent p-hacking in the long run. The truth about a certain null hypothesis will eventually be uncovered, through repeated analysis. The influences of Hypothesis Testing, being a more frequentist and mathematically rigorous methodology (see chapter 2.1.2), make inference from individual results particularly challenging. Pearson and Neyman concluded: “We must abandon our ability to measure evidence ... in an individual experiment” (Goodman, 1999, p.998). Therefore, as long Null Hypothesis Significance Testing remains the groundwork of empirical research, replication rates must be drastically increased. Since there is little incentive for scientists to replicate studies (see chapter 2.2.2), the burden could be handed down to prospective researchers. Graduate or PhD students – on their path to eventually pursuing their

own research interests – could contribute to the overall quality of science by replicating at least one study during their degree program. Incorporating the replication of published studies into the degree program, allows to set the necessary quality standards. After all, a lecturer would oversee the replication study and grade the work accordingly. Students are strongly incentivised by grades. The internet provides the necessary infrastructure to publish, store, manage, and analyse large amounts of replications. The Open Science Framework is basically using the same platform. For this approach to work, a large number of universities would have to adapt to the proposed practice, which is highly unlikely. Otherwise, too few replications would be achieved, and the publication bias would remain.

Besides the challenge to make p-hacking detectable, solutions must also consider the weaknesses of the p-value in communicating empirical findings (see chapter 2.2.1). Preregistrations and two-stage analyses would make HARKing, flexibility in variables and flexibility in sample sizes detectable, thus eradicating a large proportion of false findings. However, they do not change the way, results are being communicated. Statisticians call for confidence intervals to be included in reporting (cf. Wasserstein, 2016), as is already being done in many papers (cf. Booth et al., 2020). Confidence intervals provide readers with a straightforward estimate of the true effect size. Readers can then immediately recognise scientifically useless effects, especially infinitesimal ones. However, this estimate is sharing the p-value's principal assumption. The standard error, on the basis of which the interval range is computed, is derived directly from the null hypothesis and thus only holds true, given the null hypothesis is true.

One fundamental problem remains: the p-value by itself only measures incompatibility between data and hypothesis. Scientific inference from that measure alone is not possible. Fisher himself did not think of p as a sound basis for inference (cf. Biau, Jolles and Porcher, 2009). Replications do not entirely solve that problem. Goodman (1999a) remarks, there is a need for conclusive statements in an individual paper. He proposes Bayesian methods as a solution. J. Leek et al. (2017) suggest to state a false positive risk on results. The false positive risk shall reflect the probability that a positive is false in the population, by incorporating a prior probability of the null hypothesis being true. Goodman (2018) proposes a confidence index which goes partly down the same path. However, such estimates involve subjective judgements when determining the prior odds about the null hypothesis (cf. Goodman, 1999). Simmons, Nelson and Simonsohn (2011) remark that defining prior probabilities only adds further researcher degrees of freedom. However, there are ways to minimise subjectivity and thus flexibility. In fact, estimates can be derived from the p-value itself (cf. Goodman, 2001). The minimum Bayes factor, for instance, reflects the probability of seeing the gathered data, given the null hypothesis is true, normed by the probability of seeing the same data, given the alternative hypothesis is true (cf. Goodman, 1999). It therefore measures how much the study itself has strengthened confidence in the hypothesis.

Nevertheless, one popular proposal must be rejected, based on the here presented findings: introducing flexible p-values. In a survey from 2017, including close to 7000 researchers, 69% of respondents argued for adapting a standard significance level of 0.5% (cf. Chawla, 2017). It appears

that scientists underestimate how effective commonly adapted p-hacking methods such as deleting outliers really are. False findings can be produced at any significance level.

Clearly, another part of the problem is a lack of statistical understanding on part of applied researchers. That "some scientists don't understand the essentials of statistics" (Enserink, 2012), explains how the majority of psychologists find intermediate testing, selectively reporting dependent variables, or "rounding off" p-values defensible (cf. John, Loewenstein and Prelec, 2012). More and better university level education is clearly needed.

Some statisticians go as far as to criticise Null Hypothesis Significance Testing. J. Leek et al. (2017) wish to "move beyond the alchemy of binary statements about 'an effect' or 'no effect'" (p.4). Fisher invented Significance Testing in the face of much more general research questions. Null Hypothesis Significance Testing is perhaps not suitable to study infinitesimal effects as in "biomedicine [or] nutrition studies" (J. Leek et al., 2017, p.3). In such contexts, chasing general inference by advancing a set of binary hypotheses, disguises uncertainty. Researchers should instead "accept uncertainty and embrace variation under different circumstances" (J. Leek et al., 2017, p.4).

5 Conclusion

Fundamental issues such as certain researcher degrees of freedom, occasional false findings, and individual scientists acting in bad faith will always remain in the empirical sciences, regardless of the adopted practices and procedures. A rigid structure that would achieve to eradicate all researcher degrees of freedom is detrimental to the scientific enterprise because it impairs the ability of researchers to make new discoveries. However, the number of researcher degrees of freedom is currently too high. As a result, false findings are amassing to the point that the credibility of entire scientific fields is at stake (cf. Simmons, Nelson and Simonsohn, 2011). P-hacking is at the root cause and is currently even being incentivised by journals. In this thesis I was able to demonstrate how p-hacking can be performed with ease in any empirical study. Currently, practically any finding can be presented as "statistically significant". Furthermore, my analyses prove that in most cases it is impossible to detect p-hacking. In order to drastically diminish the number of false findings, the current practice of conducting empirical studies and communicating findings must change.

Several approaches to the problem seem feasible. It is necessary to increase replication rates, as long as studies focus on objective, frequentist analysis as much as they do currently. I recommend, to therefore involve prospective researchers into the academic enterprise more strongly, but other approaches are tenable too. The Bayesian school of thought would supplement and enhance conclusion from statistical tests. After all, inductive inference is the only way to make conclusive statements about results at hand. A nuanced discussion about how to bring Bayesian and frequentists inference together is needed. Most necessarily, however, transparency along the process of conducting empirical research must be increased. Researchers face severe consequences if they are being caught manipulating studies (cf. Enserink, 2012), which presents a powerful nudge to resist the temptation to p-hack, once preregistrations and two-stage analyses are the de facto norm in sciences. Transparency is furthermore necessary to allow replications.

I conclude that a combination of additionally reporting Bayesian measures of confidence, increasing replication rates and increasing transparency of analyses is a potent mixture to solve the present problem of p-hacking. After all, my findings only add to an increasing body of evidence that the current practice is not sustainable.

Bibliography

[Ané06] Cécile Ané. *The Normal Distribution*. 2006. URL: http://pages.stat.wisc.edu/~ane/st371/notes/chap4_by4.pdf.

[BBK17] Günter Bamberg, Franz Baur and Michael Krapp. *Statistik - Eine Einführung für Wirtschafts- und Sozialwissenschaftler*. Vol. 18. De Gruyter, 2017. ISBN: 978-3-11-049570-6.

[BCH20] Abel Brodeur, Nikolai Cook and Anthony Heyes. "Methods Matter: P-Hacking and Publication Bias in Causal Analysis in Economics". In: *American Economic Review* 110.11 (Nov. 2020).

[Bha+04] Sachin S. Bhardwaj et al. "Statistical Significance and Clinical Relevance - The Importance of Power in Clinical Trials in Dermatology". In: *Archives of Dermatological Research* 140 (Dec. 2004).

[BI16] Stephan B. Bruns and John P.A. Ioannidis. "p-Curve and p-Hacking in Observational Research". In: *PLOS ONE* 11.2 (Feb. 2016).

[BJP09] David Jean Biau, Brigitte Jolles and Raphael Porcher. "P Value and the Thepry of Hypothesis Testing - An Explanation for New Researchers". In: *Clinical Orthopaedics and Related Research* 468.3 (Nov. 2009). DOI: 10.1007/s11999-009-1164-4.

[Boh+15] J. Bohannon et al. "Chocolate with High Cocoa Content as a Weight-Loss Accelerator". In: *International Archives of Medicine* 8.1 (Dec. 2015).

[Boh15] John Bohannon. *I Fooled Millions Into Thinking Chocolate Helps Weight Loss. Here's How.* May 2015. URL: https://io9.gizmodo.com/i-fooled-millions-into-thinking-chocolate-helps-weight-1707251800.

[Boo+20] Stephen Booth et al. "Regional outcomes of severe acute respiratory syndrome coronavirus 2 infection in hospitalised patients with haematological malignancy". In: *European Journal of Haematology* 105 (June 2020). DOI: 10.1111/ejh.13469.

[BT16] Dorothy V.M. Bishop and Paul A. Thompson. "Problems in using p-curve analysis and text-mining to detect rate of p-hacking and evidential value". In: *PeerJ* 4 (Feb. 2016), p. 1715. DOI: 10.7717/peerj.1715.

[BY05] Yoav Benjamini and Daniel Yekutieli. "False Discovery Rate–Adjusted Multiple Confidence Intervals for Selected Parameters". In: *Journal of the American Statistical Association* 100.469 (Mar. 2005), pp. 71–81. DOI: 10.1198/016214504000001907.

[CA05] An-Wen Chan and Douglas G Altman. “Identifying outcome reporting bias in randomised trials on PubMed: review of publications and survey of authors”. In: *BMJ* 330.7494 (Jan. 2005), p. 753. DOI: 10.1136/bmj.38356.424606.8F.

[CG19] Maximilian Coblenz and Oliver Grothe. *Skript zur Vorlesung Analyse multivariater Daten*. 2019.

[CGS20] Tarun Chordia, Amit Goyal and Alessio Saretto. “Anomalies and False Rejections”. In: *The Review of Financial Studies* 33.5 (Feb. 2020). Ed. by Andrew Karolyi, pp. 2134–2179. DOI: 10.1093/rfs/hhaa018.

[Cha17] Dalmeet Singh Chawla. “’One-size-fits-all’ threshold for P values under fire”. In: *Nature* (Sept. 2017). URL: https://www.nature.com/news/one-size-fits-all-threshold-for-p-values-under-fire-1.22625.

[Coh92] Jacob Cohen. “Statistical Power Analysis”. In: *Current Directions in Psychological Science* 1.3 (June 1992).

[Cou18] Jennifer Couzin-Frankel. “’Journalologists’ use scientific methods to sutdy academic publishing. Is their work improving science?” In: *Science Magazine* (Sept. 2018). URL: https://www.sciencemag.org/news/2018/09/journalologists-use-scientific-methods-study-academic-publishing-their-work-improving.

[Dag71] Ralph B. Dagostino. “An omnibus test of normality for moderate and large size samples”. In: *Biometrika* 58.2 (1971), pp. 341–348. DOI: 10.1093/biomet/58.2.341.

[Dal+14] Rafael Dal-Ré et al. “Making Prospective Registration of Observational Research a Reality”. In: *Science Translational Medicine* 6.224 (Feb. 2014).

[Ens12] Martin Enserink. “Final Report: Stapel Affair Points to Bigger Problems in Social Psychology”. In: *Science Magazine* (Nov. 2012). URL: https://www.sciencemag.org/news/2012/11/final-report-stapel-affair-points-bigger-problems-social-psychology#:~:text=Final%20Report:%20Stapel%20Affair%20Points%20to%20Bigger%20Problems%20in%20Social%20Psychology,-By%20Martin%20Enserink&text=Taming%20his%20demons.,almost%20nothing%20ever%20went%20wrong.%22&text=The%20blame%20goes%20far%20beyond,worked%20as%20a%20social%20psychologist..

[Fan09] Daniele Fanelli. “How Many Scientists Fabricate and Falsify Research? A Systematic Review and Meta-Analysis of Survey Data”. In: *PLoS ONE* 4.5 (May 2009). Ed. by Tom Tregenza, p. 5738. DOI: 10.1371/journal.pone.0005738.

[Fis56] Ronald A. Fisher. *Statistical methods and scientific inference*. Hafner Publishing, 1956.

[Fis92] Ronald A. Fisher. “Breakthroughs in Statistics”. In: vol. 2. Springer, 1992. Chap. The Arrangement of Field Experiments, pp. 82–91.

[Ger+15] Will M. Gervais et al. “A Powerful Nudge? Presenting CalculableConsequences of Underpowered ResearchShifts Incentives Toward AdequatelyPowered Designs”. In: *Social Psychological and Personality Science* 6.7 (2015).

[Goo01] Steven N. Goodman. “Of P-Values and Bayes: A Modest Proposal”. In: *Epidemiology* 12.3 (May 2001).

[Goo18] Steven N. Goodman. “How sure are you of your result? Put a number on it”. In: *Nature* (Dec. 2018). URL: https://www.nature.com/articles/d41586-018-07589-2.

[Goo99a] Steven N. Goodman. “Toward Evidence-Based Medical Statistics. 1: The P Value Fallacy”. In: *Annals of Internal Medicine* 130.12 (June 1999). DOI: 10.7326/0003-4819-130-12-199906150-00008.

[Goo99b] Steven N. Goodman. “Toward Evidence-Based Medical Statistics. 2: The Bayes Factor”. In: *Annals of Internal Medicine* 130.15 (June 1999). DOI: 10.7326/0003-4819-130-12-199906150-00019.

[Hea+15] Megan L. Head et al. “The Extent and Consequences of P-Hacking in Science”. In: *PLOS Biology* 13.3 (Mar. 2015). DOI: 10.1371/journal.pbio.1002106.

[HKP12] Jiawei Han, Micheline Kamber and Jian Pei. *Data Mining - Concepts and Techniques*. 3rd ed. Elsevier, 2012. DOI: 10.1016/c2009-0-61819-5.

[Hub87] Carl J. Huberty. “On Statitical Testing”. In: *Educational Researcher* 16.8 (Nov. 1987).

[Ioa05] John P.A. Ioannidis. “Why Most Published Research Findings Are False”. In: *PLOS Medicine* 2.8 (Aug. 2005).

[Ioa08] John P. A. Ioannidis. “Why Most Discovered True Associations Are Inflated”. In: *Epidemiology* 19.5 (Sept. 2008), pp. 640–648. DOI: 10.1097/ede.0b013e31818131e7.

[Isr92] Glenn D. Israel. “Determining Sample Size”. In: (Nov. 1992). PEOD6, one of a series of the Agricultural Education and Communication Department, University of Florida Extension.

[JL13] L. R. Jager and J. T. Leek. “An estimate of the science-wise false discovery rate and application to the top medical literature”. In: *Biostatistics* 15.1 (Sept. 2013), pp. 1–12. DOI: 10.1093/biostatistics/kxt007.

[JLP12] Leslie K. John, George Loewenstein and Drazen Prelec. “Measuring the Prevalence of Questionable Research Practices With Incentives for Truth Telling”. In: *Psychological Science* 23.5 (May 2012).

[Ken19] Lee Kennedy-Shaffer. “Before $p<0.05$ to Beyond $p<0.05$: Using History to Contextualize p-Values and Significance Testing”. In: *The American Statistician* 73.1 (Mar. 2019), pp. 82–90. DOI: 10.1080/00031305.2018.1537891.

[Ker98] Norbert L. Kerr. “HARKing: Hypothesizing After the Results are Known”. In: *Personality and Social Psychology Review* 2.3 (1998).

[Kic13] Michal Kicinski. “Publication Bias in Recent Meta-Analyses”. In: *PLoS ONE* 8.11 (Nov. 2013). Ed. by Daniele Marinazzo, p. 81823. DOI: 10.1371/journal.pone.0081823.

[Krä12] Walter Krämer. “Das Signifikanztest-Ritual und andere Sackgassen des Fortschritts in der Statistik”. In: *AStA Wirtschafts- und Sozialstatistisches Archiv* 5.4 (Feb. 2012), pp. 299–308. DOI: 10.1007/s11943-012-0110-1.

[KRK05] Ingrid Kiefer, Theres Rathmanner and Michael Kunze. “Eating and dieting differences in men and women”. In: *The Journal of Men's Health & Gender* 2.2 (June 2005), pp. 194–201. DOI: 10.1016/j.jmhg.2005.04.010.

[Kup18] Kai Kupferschmidt. “More and more scientists are preregistering their studies. Should you?” In: *Science Magazine* (Sept. 2018). URL: https://www.sciencemag.org/news/2018/09/more-and-more-scientists-are-preregistering-their-studies-should-you.

[Lee+17] Jeff Leek et al. “Five ways to fix statistics”. In: *Nature* (Nov. 2017). URL: https://www.nature.com/articles/d41586-017-07522-z.

[Mat00] Robert Matthews. “Storks Deliver Babies (p=0.008)”. In: *Teaching Statistics* 22.2 (2000).

[Meh+20] Esmaeil Mehraeen et al. “Predictors of mortality in patients with COVID-19–a systematic review”. In: *European Journal of Integrative Medicine* 40 (Dec. 2020), p. 101226. DOI: 10.1016/j.eujim.2020.101226.

[Mit17] Hans-Joachim Mittag. *Statistik - Eine Einführung mit wesentlichen Elementen*. Vol. 5. Springer, 2017. ISBN: 978-3-662-55320-6.

[Mor00] Janice M. Morse. “Determining Sample Size”. In: *Qualitative Health Research* 10.1 (Jan. 2000).

[Mue+19] Frank Mueller-Langer et al. “Replication studies in economics - How many and which papers are chosen for replication, and why?” In: *Research Policy* 48.1 (Feb. 2019).

[Mum02] Peter J. Mumby. “Statistical power of non-parametric tests: A quick guide for designing sampling strategies”. In: *Marine Pollution Bulletin* 44.1 (Jan. 2002). DOI: 10.1016/s0025-326x(01)00097-2.

[Nuz14a] Regina Nuzzo. “Scientific method: Statistical errors”. In: *Nature* (Feb. 2014). URL: https://www.nature.com/news/scientific-method-statistical-errors-1.14700.

[Nuz14b] Regina Nuzzo. “Statistical Errors - P values, the 'gold standard' of statistical validity, are not as reliable as many scientists assume”. In: *Nature* 506 (Feb. 2014).

[Ope15] Open Science Collaboration. “Estimating the reproducibility of psychological science”. In: *Science* 349.6251 (Aug. 2015), pp. 4716–4716. DOI: 10.1126/science.aac4716.

[PBI15] Chirag J. Patel, Belinda Burford and John P.A. Ioannidis. “Assessment of vibration of effects due to model specification can demonstrate the instability of observational associations”. In: *Journal of Clinical Epidemiology* 68.9 (Sept. 2015), pp. 1046–1058. DOI: 10.1016/j.jclinepi.2015.05.029.

[PI11] Tiago V. Pereira and John P.A. Ioannidis. “Statistically significant meta-analyses of clinical trials have modest credibility and inflated effects”. In: *Journal of Clinical Epidemiology* 64.10 (Oct. 2011).

[Pop59] Karl Popper. *The logic of scientific discovery*. New York: Basic Books, Inc, 1959. ISBN: 9781614277439.

[Ros09] Sheldon Ross. *Introduction to probability and statistics for engineers and scientists*. Amsterdam Boston: Academic Press/Elsevier, 2009. ISBN: 9780123704832.

[RT16] James Rockey and Jonathan Temple. "Growth econometrics for agnostics and true believers". In: *European Economic Review* 81 (Jan. 2016), pp. 86–102. DOI: 10.1016/j.euroecorev.2015.05.010.

[Sch+13] Martijn J. Schuemie et al. "Interpreting observational studies: why empirical calibration is needed to correct p -values". In: *Statistics in Medicine* 33.2 (July 2013), pp. 209–218. DOI: 10.1002/sim.5925.

[SNS11] Joseph P. Simmons, Leif D. Nelson and Uri Simonsohn. "False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant". In: *Psychological Science* 22.11 (Nov. 2011).

[SNS14] Uri Simonsohn, Leif D. Nelson and Joseph P. Simmons. "P-curve: A key to the file-drawer." In: *Journal of Experimental Psychology: General* 143.2 (2014), pp. 534–547. DOI: 10.1037/a0033242.

[Son+09] Fujian Song et al. "Extent of publication bias in different categories of research cohorts: a meta-analysis of empirical studies". In: *BMC Medical Research Methodology* 9.1 (Nov. 2009). DOI: 10.1186/1471-2288-9-79.

[Ste59] Theodore D. Sterling. "Publication Decisions and Their Possible Effects on Inferences Drawn from Tests of Significance - Or Vice Versa". In: *Journal of the American Statistical Association* 54.285 (Mar. 1959), p. 30. DOI: 10.2307/2282137.

[Tal10] Nassim Nicholas Taleb. *The Black Swan*. Random House LCC US, May 2010. 400 pp. ISBN: 081297381X. URL: https://www.ebook.de/de/product/7775103/nassim_nicholas_taleb_the_black_swan.html.

[Tho97] Len Thomas. "Retrospective Power Analysis". In: *Conservation Biology* 11.1 (Feb. 1997).

[TZ08] Tiejun Tong and Hongyu Zhao. "Practical guidelines for assessing power and false discovery rate for a fixed sample size in microarray experiments". In: *Statistics in Medicine* 27.11 (May 2008). DOI: 10.1002/sim.3237.

[Was16] Ron Wasserstein. *Statement on Statistical Significance and P-Values*. Mar. 2016. URL: https://www.amstat.org/asa/files/pdfs/p-valuestatement.pdf.

[WD15] Joost CF de Winter and Dimitra Dodou. "A surge of p-values between 0.041 and 0.049 in recent decades (but negative results are increasing rapidly too)". In: *PeerJ* 3 (Jan. 2015), p. 733. DOI: 10.7717/peerj.733.

[Whi07] Halbert White. "A Reality Check for Data Snooping". In: *Econometrica* 68.5 (Sept. 2007).

[Wic+16] Jelte M. Wicherts et al. "Degrees of Freedom in Planning, Running, Analyzing, and Reporting Psychological Studies: A Checklist to Avoid P-Hacking". In: *Frontiers in Psychology* 7 (Nov. 2016). DOI: 10.3389/fpsyg.2016.01832.

[ZM08] Stephen T. Ziliak and Deirdre N. McCloskey. *The Cult of Statistical Significance: How the Standard Error Costs Us Jobs, Justice, and Lives*. University of Michigan Press, 2008. ISBN: 978-0-472-07007-7.

Appendix

Appendix A

Interval of exploitable test statistics in the t-distribution.

(see chapter 3.3.1.3 t-Distribution)

degrees of freedom	statistic at p1=sig.	statistic at p2=sig.	interval range
df=1	6.314	12.707	6.393
df=2	2.92	4.303	1.383
df=3	2.354	3.183	0.829
df=4	2.132	2776	0.645
df=5	2.016	2.571	0.555
df=6	1.944	2.447	0.503
df=7	1.895	2.365	0.47
df=8	1.86	2.307	0.447
df=9	1.834	2.263	0.429
df=10	1.813	2.229	0.416
df=11	1.796	2.201	0.405
df=12	1.783	2.179	0.396
df=13	1.771	2.161	0.39
df=14	1.762	2.145	0.383
df=15	1.754	2.132	0.37
df=16	1.746	2.12	0.374
df=17	1.74	2.11	0.37
df=18	1.735	2.101	0.366
df=19	1.73	2.094	0.364
df=20	1.725	2.086	0.361
df=21	1.721	2.08	0.359
df=22	1.718	2.074	0.356
df=23	1.714	2.069	0.355
df=24	1.711	2.064	0.353
df=25	1.709	2.06	0.351
df=26	1.706	2.056	0.35
df=27	1.704	2.052	0.348
df=28	1.702	2.049	0.347
df=29	1.7	2.046	0.346
df=30	1.698	2.043	0.345

Appendix B

Interval of exploitable test statistics in the χ^2-distribution.

(see chapter 3.3.1.4 χ^2-Distribution)

degrees of freedom	statistic at p1=sig.	statistic at p2=sig.	interval range
df=1	3.842	5.024	1.182
df=2	5.992	7.378	1.386
df=3	7.815	9.349	1.534
df=4	9.488	11.144	1.656
df=5	11.071	12.833	1.762
df=6	12.592	14.45	1.858
df=7	14.068	16.013	1.945
df=8	15.508	17.535	2.027
df=9	16.919	19.023	2.104
df=10	18.308	20.484	2.176
df=11	19.676	21.921	2.245
df=12	21.027	23.337	2.31
df=13	22.363	24.736	2.373
df=14	23.685	26.119	2.434
df=15	24.996	27.489	2.493
df=16	26.297	28.846	2.549
df=17	27.588	30.192	2.604
df=18	28.87	31.527	2.657
df=19	30.144	32.853	2.709
df=20	31.411	34.17	2.759
df=21	32.671	35.479	2.808
df=22	33.925	36.781	2.856
df=23	35.173	38.076	2.903
df=24	36.416	39.365	2.949
df=25	37.653	40.647	2.994
df=26	38.886	41.924	3.038
df=27	40.114	43.195	3.081
df=28	41.338	44.461	3.123
df=29	42.557	45.723	3.166
df=30	43.773	46.98	3.207

Appendix C

Interval of exploitable test statistics in the F-distribution.

(see chapter 3.3.1.5 F-Distribution)

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
1,2	17.6	37.6	20.0
1,3	9.2	16.5	7.3
1,4	6.8	11.3	4.5
1,5	5.7	9.1	3.4
1,6	5.0	7.9	2.9
1,7	4.6	7.1	2.5
1,8	4.4	6.6	2.2
1,9	4.2	6.3	2.1
1,10	4.0	6.0	2.0
1,11	3.9	5.8	1.9
1,12	3.8	5.6	1.8
1,13	3.7	5.5	1.8
1,14	3.7	5.3	1.6
1,15	3.6	5.2	1.6
1,16	3.5	5.2	1.7
1,17	3.5	5.1	1.6
1,18	3.5	5.0	1.5
1,19	3.4	5.0	1.6
1,20	3.4	4.9	1.5
1,21	3.4	4.9	1.5
1,22	3.4	4.8	1.4
1,23	3.3	4.8	1.5
1,24	3.3	4.8	1.5
1,25	3.3	4.7	1.4
1,26	3.3	4.7	1.4
1,27	3.3	4.7	1.4
1,28	3.2	4.7	1.5
1,29	3.2	4.6	1.4
1,30	3.2	4.6	1.4
2,2	18.0	38.1	20.1
2,3	8.6	15.1	6.5
2,4	6.0	9.7	3.7
2,5	4.8	7.5	2.7
2,6	4.2	6.3	2.1
2,7	3.8	5.6	1.8
2,8	3.5	5.1	1.6
2,9	3.3	4.8	1.5
2,10	3.2	4.5	1.3
2,11	3.0	4.3	1.3
2,12	2.9	4.1	1.2
2,13	2.9	4.0	1.1
2,14	2.8	3.9	1.1
2,15	2.7	3.8	1.1
2,16	2.7	3.7	1.0
2,17	2.6	3.7	1.1
2,18	2.6	3.6	1.0
2,19	2.6	3.6	1.0
2,20	2.5	3.5	1.0
2,21	2.5	3.5	1.0
2,22	2.5	3.4	0.9
2,23	2.5	3.4	0.9

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
2,24	2.5	3.4	0.9
2,25	2.4	3.3	0.9
2,26	2.4	3.3	0.9
2,27	2.4	3.3	0.9
2,28	2.4	3.3	0.9
2,29	2.4	3.3	0.9
2,30	2.4	3.2	0.8
3,2	18.2	38.2	20.0
3,3	8.3	14.5	6.2
3,4	5.6	9.0	3.4
3,5	4.5	6.8	2.3
3,6	3.8	5.6	1.8
3,7	3.4	4.9	1.5
3,8	3.1	4.5	1.4
3,9	2.9	4.1	1.2
3,10	2.8	3.9	1.1
3,11	2.6	3.7	1.1
3,12	2.5	3.5	1.0
3,13	2.5	3.4	0.9
3,14	2.4	3.3	0.9
3,15	2.3	3.2	0.9
3,16	2.3	3.1	0.8
3,17	2.2	3.1	0.9
3,18	2.2	3.0	0.8
3,19	2.2	3.0	0.8
3,20	2.1	2.9	0.8
3,21	2.1	2.9	0.8
3,22	2.1	2.8	0.7
3,23	2.1	2.8	0.7
3,24	2.1	2.8	0.7
3,25	2.0	2.7	0.7
3,26	2.0	2.7	0.7
3,27	2.0	2.7	0.7
3,28	2.0	2.7	0.7
3,29	2.0	2.7	0.7
3,30	2.0	2.6	0.6
4,2	18.3	38.3	20.0
4,3	8.2	14.2	6.0
4,4	5.4	8.7	3.3
4,5	4.2	6.4	2.2
4,6	3.6	5.3	1.7
4,7	3.2	4.6	1.4
4,8	2.9	4.1	1.2
4,9	2.7	3.8	1.1
4,10	2.5	3.5	1.0
4,11	2.4	3.3	0.9
4,12	2.3	3.2	0.9
4,13	2.2	3.0	0.8
4,14	2.2	2.9	0.7
4,15	2.1	2.9	0.8
4,16	2.1	2.8	0.7

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
4,17	2.0	2.7	0.7
4,18	2.0	2.7	0.7
4,19	1.9	2.6	0.7
4,20	1.9	2.6	0.7
4,21	1.9	2.5	0.6
4,22	1.9	2.5	0.6
4,23	1.8	2.5	0.7
4,24	1.8	2.4	0.6
4,25	1.8	2.4	0.6
4,26	1.8	2.4	0.6
4,27	1.8	2.4	0.6
4,28	1.8	2.3	0.5
4,29	1.8	2.3	0.5
4,30	1.7	2.3	0.6
5,2	18.3	38.3	20.0
5,3	8.1	13.9	5.8
5,4	5.3	8.4	3.1
5,5	4.1	6.2	2.1
5,6	3.4	5.0	1.6
5,7	3.0	4.3	1.3
5,8	2.7	3.9	1.2
5,9	2.5	3.5	1.0
5,10	2.4	3.3	0.9
5,11	2.3	3.1	0.8
5,12	2.2	2.9	0.7
5,13	2.1	2.8	0.7
5,14	2.0	2.7	0.7
5,15	2.0	2.6	0.6
5,16	1.9	2.6	0.7
5,17	1.9	2.5	0.6
5,18	1.8	2.4	0.6
5,19	1.8	2.4	0.6
5,20	1.8	2.3	0.5
5,21	1.7	2.3	0.6
5,22	1.7	2.3	0.6
5,23	1.7	2.2	0.5
5,24	1.7	2.2	0.5
5,25	1.7	2.2	0.5
5,26	1.6	2.2	0.6
5,27	1.6	2.1	0.5
5,28	1.6	2.1	0.5
5,29	1.6	2.1	0.5
5,30	1.6	2.1	0.5
6,2	18.4	38.4	20.0
6,3	8.0	13.8	5.8
6,4	5.2	8.2	3.0
6,5	4.0	6.0	2.0
6,6	3.3	4.9	1.6
6,7	2.9	4.2	1.3
6,8	2.6	3.7	1.1
6,9	2.4	3.4	1.0

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
6,10	2.3	3.1	0.8
6,11	2.1	2.9	0.8
6,12	2.0	2.8	0.8
6,13	2.0	2.7	0.7
6,14	1.9	2.6	0.7
6,15	1.8	2.5	0.7
6,16	1.8	2.4	0.6
6,17	1.7	2.3	0.6
6,18	1.7	2.3	0.6
6,19	1.7	2.2	0.5
6,20	1.6	2.2	0.6
6,21	1.6	2.1	0.5
6,22	1.6	2.1	0.5
6,23	1.6	2.1	0.5
6,24	1.6	2.0	0.4
6,25	1.5	2.0	0.5
6,26	1.5	2.0	0.5
6,27	1.5	2.0	0.5
6,28	1.5	2.0	0.5
6,29	1.5	1.9	0.4
6,30	1.5	1.9	0.4
7,2	18.4	38.4	20.0
7,3	7.9	13.7	5.8
7,4	5.1	8.1	3.0
7,5	3.9	5.9	2.0
7,6	3.3	4.7	1.4
7,7	2.8	4.0	1.2
7,8	2.6	3.6	1.0
7,9	2.3	3.2	0.9
7,10	2.2	3.0	0.8
7,11	2.1	2.8	0.7
7,12	2.0	2.7	0.7
7,13	1.9	2.5	0.6
7,14	1.8	2.4	0.6
7,15	1.8	2.3	0.5
7,16	1.7	2.3	0.6
7,17	1.7	2.2	0.5
7,18	1.6	2.1	0.5
7,19	1.6	2.1	0.5
7,20	1.6	2.1	0.5
7,21	1.5	2.0	0.5
7,22	1.5	2.0	0.5
7,23	1.5	2.0	0.5
7,24	1.5	1.9	0.4
7,25	1.5	1.9	0.4
7,26	1.4	1.9	0.5
7,27	1.4	1.9	0.5
7,28	1.4	1.8	0.4
7,29	1.4	1.8	0.4
7,30	1.4	1.8	0.4
8,2	18.4	38.4	20.0

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
11,18	1.4	1.9	0.5
11,19	1.4	1.8	0.4
11,20	1.4	1.8	0.4
11,21	1.3	1.7	0.4
11,22	1.3	1.7	0.4
11,23	1.3	1.7	0.4
11,24	1.3	1.6	0.3
11,25	1.2	1.6	0.4
11,26	1.2	1.6	0.4
11,27	1.2	1.6	0.4
11,28	1.2	1.5	0.3
11,29	1.2	1.5	0.3
11,30	1.2	1.5	0.3
12,2	18.5	38.5	20.0
12,3	7.8	13.4	5.6
12,4	5.0	7.8	2.8
12,5	3.7	5.6	1.9
12,6	3.0	4.4	1.4
12,7	2.6	3.7	1.1
12,8	2.3	3.2	0.9
12,9	2.1	2.9	0.8
12,10	2.0	2.7	0.7
12,11	1.8	2.5	0.7
12,12	1.7	2.3	0.6
12,13	1.7	2.2	0.5
12,14	1.6	2.1	0.5
12,15	1.5	2.0	0.5
12,16	1.5	1.9	0.4
12,17	1.4	1.9	0.5
12,18	1.4	1.8	0.4
12,19	1.4	1.8	0.4
12,20	1.3	1.7	0.4
12,21	1.3	1.7	0.4
12,22	1.3	1.7	0.4
12,23	1.3	1.6	0.3
12,24	1.2	1.6	0.4
12,25	1.2	1.6	0.4
12,26	1.2	1.5	0.3
12,27	1.2	1.5	0.3
12,28	1.2	1.5	0.3
12,29	1.2	1.5	0.3
12,30	1.1	1.5	0.4
13,2	18.5	38.5	20.0
13,3	7.8	13.4	5.6
13,4	4.9	7.8	2.9
13,5	3.7	5.5	1.8
13,6	3.0	4.4	1.4
13,7	2.6	3.7	1.1
13,8	2.3	3.2	0.9
13,9	2.1	2.9	0.8
13,10	1.9	2.6	0.7

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
13,11	1.8	2.4	0.6
13,12	1.7	2.3	0.6
13,13	1.6	2.2	0.6
13,14	1.6	2.1	0.5
13,15	1.5	2.0	0.5
13,16	1.4	1.9	0.5
13,17	1.4	1.8	0.4
13,18	1.4	1.8	0.4
13,19	1.3	1.7	0.4
13,20	1.3	1.7	0.4
13,21	1.3	1.6	0.3
13,22	1.2	1.6	0.4
13,23	1.2	1.6	0.4
13,24	1.2	1.6	0.4
13,25	1.2	1.5	0.3
13,26	1.2	1.5	0.3
13,27	1.2	1.5	0.3
13,28	1.1	1.5	0.4
13,29	1.1	1.4	0.3
13,30	1.1	1.4	0.3
14,2	18.5	38.5	20.0
14,3	7.8	13.3	5.5
14,4	4.9	7.7	2.8
14,5	3.7	5.5	1.8
14,6	3.0	4.3	1.3
14,7	2.6	3.6	1.0
14,8	2.3	3.2	0.9
14,9	2.1	2.8	0.7
14,10	1.9	2.6	0.7
14,11	1.8	2.4	0.6
14,12	1.7	2.3	0.6
14,13	1.6	2.1	0.5
14,14	1.5	2.0	0.5
14,15	1.5	1.9	0.4
14,16	1.4	1.9	0.5
14,17	1.4	1.8	0.4
14,18	1.3	1.7	0.4
14,19	1.3	1.7	0.4
14,20	1.3	1.7	0.4
14,21	1.2	1.6	0.4
14,22	1.2	1.6	0.4
14,23	1.2	1.5	0.3
14,24	1.2	1.5	0.3
14,25	1.2	1.5	0.3
14,26	1.1	1.5	0.4
14,27	1.1	1.4	0.3
14,28	1.1	1.4	0.3
14,29	1.1	1.4	0.3
14,30	1.1	1.4	0.3
15,2	18.5	38.5	20.0
15,3	7.8	13.3	5.5

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
18,19	1.2	1.6	0.4
18,20	1.2	1.6	0.4
18,21	1.2	1.5	0.3
18,22	1.1	1.5	0.4
18,23	1.1	1.4	0.3
18,24	1.1	1.4	0.3
18,25	1.1	1.4	0.3
18,26	1.1	1.4	0.3
18,27	1.1	1.3	0.2
18,28	1.0	1.3	0.3
18,29	1.0	1.3	0.3
18,30	1.0	1.3	0.3
19,2	18.5	38.5	20.0
19,3	7.7	13.2	5.5
19,4	4.9	7.6	2.7
19,5	3.6	5.4	1.8
19,6	2.9	4.2	1.3
19,7	2.5	3.5	1.0
19,8	2.2	3.1	0.9
19,9	2.0	2.7	0.7
19,10	1.8	2.5	0.7
19,11	1.7	2.3	0.6
19,12	1.6	2.1	0.5
19,13	1.5	2.0	0.5
19,14	1.5	1.9	0.4
19,15	1.4	1.8	0.4
19,16	1.3	1.7	0.4
19,17	1.3	1.7	0.4
19,18	1.3	1.6	0.3
19,19	1.2	1.6	0.4
19,20	1.2	1.5	0.3
19,21	1.2	1.5	0.3
19,22	1.1	1.5	0.4
19,23	1.1	1.4	0.3
19,24	1.1	1.4	0.3
19,25	1.1	1.4	0.3
19,26	1.1	1.3	0.2
19,27	1.0	1.3	0.3
19,28	1.0	1.3	0.3
19,29	1.0	1.3	0.3
19,30	1.0	1.3	0.3
20,2	18.5	38.5	20.0
20,3	7.7	13.2	5.5
20,4	4.9	7.6	2.7
20,5	3.6	5.4	1.8
20,6	2.9	4.2	1.3
20,7	2.5	3.5	1.0
20,8	2.2	3.0	0.8
20,9	2.0	2.7	0.7
20,10	1.8	2.5	0.7
20,11	1.7	2.3	0.6

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
20,12	1.6	2.1	0.5
20,13	1.5	2.0	0.5
20,14	1.4	1.9	0.5
20,15	1.4	1.8	0.4
20,16	1.3	1.7	0.4
20,17	1.3	1.7	0.4
20,18	1.2	1.6	0.4
20,19	1.2	1.6	0.4
20,20	1.2	1.5	0.3
20,21	1.1	1.5	0.4
20,22	1.1	1.4	0.3
20,23	1.1	1.4	0.3
20,24	1.1	1.4	0.3
20,25	1.1	1.4	0.3
20,26	1.0	1.3	0.3
20,27	1.0	1.3	0.3
20,28	1.0	1.3	0.3
20,29	1.0	1.3	0.3
20,30	1.0	1.2	0.2
21,2	18.5	38.5	20.0
21,3	7.7	13.2	5.5
21,4	4.8	7.6	2.8
21,5	3.6	5.4	1.8
21,6	2.9	4.2	1.3
21,7	2.5	3.5	1.0
21,8	2.2	3.0	0.8
21,9	2.0	2.7	0.7
21,10	1.8	2.5	0.7
21,11	1.7	2.3	0.6
21,12	1.6	2.1	0.5
21,13	1.5	2.0	0.5
21,14	1.4	1.9	0.5
21,15	1.4	1.8	0.4
21,16	1.3	1.7	0.4
21,17	1.3	1.6	0.3
21,18	1.2	1.6	0.4
21,19	1.2	1.5	0.3
21,20	1.2	1.5	0.3
21,21	1.1	1.5	0.4
21,22	1.1	1.4	0.3
21,23	1.1	1.4	0.3
21,24	1.1	1.4	0.3
21,25	1.0	1.3	0.3
21,26	1.0	1.3	0.3
21,27	1.0	1.3	0.3
21,28	1.0	1.3	0.3
21,29	1.0	1.2	0.2
21,30	1.0	1.2	0.2
22,2	18.5	38.5	20.0
22,3	7.7	13.2	5.5
22,4	4.8	7.6	2.8

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
25,20	1.1	1.4	0.3
25,21	1.1	1.4	0.3
25,22	1.1	1.4	0.3
25,23	1.0	1.3	0.3
25,24	1.0	1.3	0.3
25,25	1.0	1.3	0.3
25,26	1.0	1.3	0.3
25,27	1.0	1.2	0.2
25,28	1.0	1.2	0.2
25,29	0.9	1.2	0.3
25,30	0.9	1.2	0.3
26,2	18.5	38.5	20.0
26,3	7.7	13.2	5.5
26,4	4.8	7.5	2.7
26,5	3.6	5.3	1.7
26,6	2.9	4.1	1.2
26,7	2.4	3.4	1.0
26,8	2.2	3.0	0.8
26,9	1.9	2.6	0.7
26,10	1.8	2.4	0.6
26,11	1.6	2.2	0.6
26,12	1.5	2.0	0.5
26,13	1.5	1.9	0.4
26,14	1.4	1.8	0.4
26,15	1.3	1.7	0.4
26,16	1.3	1.7	0.4
26,17	1.2	1.6	0.4
26,18	1.2	1.5	0.3
26,19	1.1	1.5	0.4
26,20	1.1	1.4	0.3
26,21	1.1	1.4	0.3
26,22	1.1	1.4	0.3
26,23	1.0	1.3	0.3
26,24	1.0	1.3	0.3
26,25	1.0	1.3	0.3
26,26	1.0	1.2	0.2
26,27	1.0	1.2	0.2
26,28	0.9	1.2	0.3
26,29	0.9	1.2	0.3
26,30	0.9	1.2	0.3
27,2	18.5	38.5	20.0
27,3	7.7	13.1	5.4
27,4	4.8	7.5	2.7
27,5	3.6	5.3	1.7
27,6	2.9	4.1	1.2
27,7	2.4	3.4	1.0
27,8	2.1	3.0	0.9
27,9	1.9	2.6	0.7
27,10	1.8	2.4	0.6
27,11	1.6	2.2	0.6
27,12	1.5	2.0	0.5

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
27,13	1.4	1.9	0.5
27,14	1.4	1.8	0.4
27,15	1.3	1.7	0.4
27,16	1.3	1.6	0.3
27,17	1.2	1.6	0.4
27,18	1.2	1.5	0.3
27,19	1.1	1.5	0.4
27,20	1.1	1.4	0.3
27,21	1.1	1.4	0.3
27,22	1.1	1.3	0.2
27,23	1.0	1.3	0.3
27,24	1.0	1.3	0.3
27,25	1.0	1.3	0.3
27,26	1.0	1.2	0.2
27,27	1.0	1.2	0.2
27,28	0.9	1.2	0.3
27,29	0.9	1.2	0.3
27,30	0.9	1.2	0.3
28,2	18.5	38.5	20.0
28,3	7.7	13.1	5.4
28,4	4.8	7.5	2.7
28,5	3.6	5.3	1.7
28,6	2.9	4.1	1.2
28,7	2.4	3.4	1.0
28,8	2.1	3.0	0.9
28,9	1.9	2.6	0.7
28,10	1.8	2.4	0.6
28,11	1.6	2.2	0.6
28,12	1.5	2.0	0.5
28,13	1.4	1.9	0.5
28,14	1.4	1.8	0.4
28,15	1.3	1.7	0.4
28,16	1.3	1.6	0.3
28,17	1.2	1.6	0.4
28,18	1.2	1.5	0.3
28,19	1.1	1.5	0.4
28,20	1.1	1.4	0.3
28,21	1.1	1.4	0.3
28,22	1.0	1.3	0.3
28,23	1.0	1.3	0.3
28,24	1.0	1.3	0.3
28,25	1.0	1.2	0.2
28,26	1.0	1.2	0.2
28,27	0.9	1.2	0.3
28,28	0.9	1.2	0.3
28,29	0.9	1.2	0.3
28,30	0.9	1.1	0.2
29,2	18.5	38.5	20.0
29,3	7.7	13.1	5.4
29,4	4.8	7.5	2.7
29,5	3.6	5.3	1.7

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
8,3	7.9	13.6	5.7
8,4	5.1	8.0	2.9
8,5	3.9	5.8	1.9
8,6	3.2	4.6	1.4
8,7	2.8	3.9	1.1
8,8	2.5	3.5	1.0
8,9	2.3	3.2	0.9
8,10	2.1	2.9	0.8
8,11	2.0	2.7	0.7
8,12	1.9	2.6	0.7
8,13	1.8	2.4	0.6
8,14	1.7	2.3	0.6
8,15	1.7	2.2	0.5
8,16	1.6	2.2	0.6
8,17	1.6	2.1	0.5
8,18	1.6	2.1	0.5
8,19	1.5	2.0	0.5
8,20	1.5	2.0	0.5
8,21	1.5	1.9	0.4
8,22	1.4	1.9	0.5
8,23	1.4	1.9	0.5
8,24	1.4	1.8	0.4
8,25	1.4	1.8	0.4
8,26	1.4	1.8	0.4
8,27	1.4	1.8	0.4
8,28	1.3	1.7	0.4
8,29	1.3	1.7	0.4
8,30	1.3	1.7	0.4
9,2	18.4	38.4	20.0
9,3	7.9	13.5	5.6
9,4	5.0	8.0	3.0
9,5	3.8	5.7	1.9
9,6	3.1	4.6	1.5
9,7	2.7	3.9	1.2
9,8	2.4	3.4	1.0
9,9	2.2	3.1	0.9
9,10	2.1	2.8	0.7
9,11	1.9	2.6	0.7
9,12	1.8	2.5	0.7
9,13	1.8	2.4	0.6
9,14	1.7	2.3	0.6
9,15	1.6	2.2	0.6
9,16	1.6	2.1	0.5
9,17	1.5	2.0	0.5
9,18	1.5	2.0	0.5
9,19	1.5	1.9	0.4
9,20	1.4	1.9	0.5
9,21	1.4	1.8	0.4
9,22	1.4	1.8	0.4
9,23	1.4	1.8	0.4
9,24	1.4	1.8	0.4

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
9,25	1.3	1.7	0.4
9,26	1.3	1.7	0.4
9,27	1.3	1.7	0.4
9,28	1.3	1.7	0.4
9,29	1.3	1.6	0.3
9,30	1.3	1.6	0.3
10,2	18.4	38.4	20.0
10,3	7.8	13.5	5.7
10,4	5.0	7.9	2.9
10,5	3.8	5.7	1.9
10,6	3.1	4.5	1.4
10,7	2.7	3.8	1.1
10,8	2.4	3.3	0.9
10,9	2.2	3.0	0.8
10,10	2.0	2.8	0.8
10,11	1.9	2.6	0.7
10,12	1.8	2.4	0.6
10,13	1.7	2.3	0.6
10,14	1.7	2.2	0.5
10,15	1.6	2.1	0.5
10,16	1.5	2.0	0.5
10,17	1.5	2.0	0.5
10,18	1.5	1.9	0.4
10,19	1.4	1.9	0.5
10,20	1.4	1.8	0.4
10,21	1.4	1.8	0.4
10,22	1.3	1.7	0.4
10,23	1.3	1.7	0.4
10,24	1.3	1.7	0.4
10,25	1.3	1.7	0.4
10,26	1.3	1.6	0.3
10,27	1.3	1.6	0.3
10,28	1.2	1.6	0.4
10,29	1.2	1.6	0.4
10,30	1.2	1.6	0.4
11,2	18.5	38.5	20.0
11,3	7.8	13.4	5.6
11,4	5.0	7.8	2.8
11,5	3.8	5.6	1.8
11,6	3.1	4.5	1.4
11,7	2.7	3.8	1.1
11,8	2.4	3.3	0.9
11,9	2.2	3.0	0.8
11,10	2.0	2.7	0.7
11,11	1.9	2.5	0.6
11,12	1.8	2.4	0.6
11,13	1.7	2.2	0.5
11,14	1.6	2.1	0.5
11,15	1.6	2.1	0.5
11,16	1.5	2.0	0.5
11,17	1.5	1.9	0.4

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
15,4	4.9	7.7	2.8
15,5	3.7	5.5	1.8
15,6	3.0	4.3	1.3
15,7	2.6	3.6	1.0
15,8	2.3	3.2	0.9
15,9	2.1	2.8	0.7
15,10	1.9	2.6	0.7
15,11	1.8	2.4	0.6
15,12	1.7	2.2	0.5
15,13	1.6	2.1	0.5
15,14	1.5	2.0	0.5
15,15	1.5	1.9	0.4
15,16	1.4	1.8	0.4
15,17	1.4	1.8	0.4
15,18	1.3	1.7	0.4
15,19	1.3	1.7	0.4
15,20	1.3	1.6	0.3
15,21	1.2	1.6	0.4
15,22	1.2	1.5	0.3
15,23	1.2	1.5	0.3
15,24	1.2	1.5	0.3
15,25	1.1	1.5	0.4
15,26	1.1	1.4	0.3
15,27	1.1	1.4	0.3
15,28	1.1	1.4	0.3
15,29	1.1	1.4	0.3
15,30	1.1	1.4	0.3
16,2	18.5	38.5	20.0
16,3	7.7	13.3	5.6
16,4	4.9	7.7	2.8
16,5	3.7	5.5	1.8
16,6	3.0	4.3	1.3
16,7	2.5	3.6	1.1
16,8	2.3	3.1	0.8
16,9	2.0	2.8	0.8
16,10	1.9	2.5	0.6
16,11	1.8	2.4	0.6
16,12	1.6	2.2	0.6
16,13	1.6	2.1	0.5
16,14	1.5	2.0	0.5
16,15	1.4	1.9	0.5
16,16	1.4	1.8	0.4
16,17	1.3	1.7	0.4
16,18	1.3	1.7	0.4
16,19	1.3	1.6	0.3
16,20	1.2	1.6	0.4
16,21	1.2	1.6	0.4
16,22	1.2	1.5	0.3
16,23	1.2	1.5	0.3
16,24	1.1	1.5	0.4
16,25	1.1	1.4	0.3

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
16,26	1.1	1.4	0.3
16,27	1.1	1.4	0.3
16,28	1.1	1.4	0.3
16,29	1.1	1.3	0.2
16,30	1.0	1.3	0.3
17,2	18.5	38.5	20.0
17,3	7.7	13.3	5.6
17,4	4.9	7.7	2.8
17,5	3.6	5.4	1.8
17,6	3.0	4.3	1.3
17,7	2.5	3.6	1.1
17,8	2.2	3.1	0.9
17,9	2.0	2.8	0.8
17,10	1.9	2.5	0.6
17,11	1.7	2.3	0.6
17,12	1.6	2.2	0.6
17,13	1.5	2.1	0.6
17,14	1.5	2.0	0.5
17,15	1.4	1.9	0.5
17,16	1.4	1.8	0.4
17,17	1.3	1.7	0.4
17,18	1.3	1.7	0.4
17,19	1.2	1.6	0.4
17,20	1.2	1.6	0.4
17,21	1.2	1.5	0.3
17,22	1.2	1.5	0.3
17,23	1.1	1.5	0.4
17,24	1.1	1.4	0.3
17,25	1.1	1.4	0.3
17,26	1.1	1.4	0.3
17,27	1.1	1.4	0.3
17,28	1.1	1.3	0.2
17,29	1.0	1.3	0.3
17,30	1.0	1.3	0.3
18,2	18.5	38.5	20.0
18,3	7.7	13.2	5.5
18,4	4.9	7.6	2.7
18,5	3.6	5.4	1.8
18,6	2.9	4.3	1.4
18,7	2.5	3.6	1.1
18,8	2.2	3.1	0.9
18,9	2.0	2.8	0.8
18,10	1.8	2.5	0.7
18,11	1.7	2.3	0.6
18,12	1.6	2.2	0.6
18,13	1.5	2.0	0.5
18,14	1.5	1.9	0.4
18,15	1.4	1.8	0.4
18,16	1.4	1.8	0.4
18,17	1.3	1.7	0.4
18,18	1.3	1.6	0.3

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
22,5	3.6	5.4	1.8
22,6	2.9	4.2	1.3
22,7	2.5	3.5	1.0
22,8	2.2	3.0	0.8
22,9	2.0	2.7	0.7
22,10	1.8	2.4	0.6
22,11	1.7	2.2	0.5
22,12	1.6	2.1	0.5
22,13	1.5	2.0	0.5
22,14	1.4	1.9	0.5
22,15	1.4	1.8	0.4
22,16	1.3	1.7	0.4
22,17	1.3	1.6	0.3
22,18	1.2	1.6	0.4
22,19	1.2	1.5	0.3
22,20	1.2	1.5	0.3
22,21	1.1	1.4	0.3
22,22	1.1	1.4	0.3
22,23	1.1	1.4	0.3
22,24	1.1	1.3	0.2
22,25	1.0	1.3	0.3
22,26	1.0	1.3	0.3
22,27	1.0	1.3	0.3
22,28	1.0	1.3	0.3
22,29	1.0	1.2	0.2
22,30	1.0	1.2	0.2
23,2	18.5	38.5	20.0
23,3	7.7	13.2	5.5
23,4	4.8	7.6	2.8
23,5	3.6	5.3	1.7
23,6	2.9	4.2	1.3
23,7	2.5	3.5	1.0
23,8	2.2	3.0	0.8
23,9	2.0	2.7	0.7
23,10	1.8	2.4	0.6
23,11	1.7	2.2	0.5
23,12	1.6	2.1	0.5
23,13	1.5	2.0	0.5
23,14	1.4	1.9	0.5
23,15	1.3	1.8	0.5
23,16	1.3	1.7	0.4
23,17	1.2	1.6	0.4
23,18	1.2	1.6	0.4
23,19	1.2	1.5	0.3
23,20	1.1	1.5	0.4
23,21	1.1	1.4	0.3
23,22	1.1	1.4	0.3
23,23	1.1	1.4	0.3
23,24	1.0	1.3	0.3
23,25	1.0	1.3	0.3
23,26	1.0	1.3	0.3

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
23,27	1.0	1.3	0.3
23,28	1.0	1.2	0.2
23,29	1.0	1.2	0.2
23,30	0.9	1.2	0.3
24,2	18.5	38.5	20.0
24,3	7.7	13.2	5.5
24,4	4.8	7.6	2.8
24,5	3.6	5.3	1.7
24,6	2.9	4.2	1.3
24,7	2.5	3.5	1.0
24,8	2.2	3.0	0.8
24,9	2.0	2.7	0.7
24,10	1.8	2.4	0.6
24,11	1.7	2.2	0.5
24,12	1.6	2.1	0.5
24,13	1.5	1.9	0.4
24,14	1.4	1.8	0.4
24,15	1.3	1.8	0.5
24,16	1.3	1.7	0.4
24,17	1.2	1.6	0.4
24,18	1.2	1.6	0.4
24,19	1.2	1.5	0.3
24,20	1.1	1.5	0.4
24,21	1.1	1.4	0.3
24,22	1.1	1.4	0.3
24,23	1.1	1.3	0.2
24,24	1.0	1.3	0.3
24,25	1.0	1.3	0.3
24,26	1.0	1.3	0.3
24,27	1.0	1.2	0.2
24,28	1.0	1.2	0.2
24,29	1.0	1.2	0.2
24,30	0.9	1.2	0.3
25,2	18.5	38.5	20.0
25,3	7.7	13.2	5.5
25,4	4.8	7.6	2.8
25,5	3.6	5.3	1.7
25,6	2.9	4.2	1.3
25,7	2.5	3.5	1.0
25,8	2.2	3.0	0.8
25,9	1.9	2.7	0.8
25,10	1.8	2.4	0.6
25,11	1.7	2.2	0.5
25,12	1.5	2.1	0.6
25,13	1.5	1.9	0.4
25,14	1.4	1.8	0.4
25,15	1.3	1.7	0.4
25,16	1.3	1.7	0.4
25,17	1.2	1.6	0.4
25,18	1.2	1.5	0.3
25,19	1.2	1.5	0.3

df1,df2	statistic at p1=sig.	statistic at p2=sig.	interval range
29,6	2.9	4.1	1.2
29,7	2.4	3.4	1.0
29,8	2.1	3.0	0.9
29,9	1.9	2.6	0.7
29,10	1.8	2.4	0.6
29,11	1.6	2.2	0.6
29,12	1.5	2.0	0.5
29,13	1.4	1.9	0.5
29,14	1.4	1.8	0.4
29,15	1.3	1.7	0.4
29,16	1.2	1.6	0.4
29,17	1.2	1.6	0.4
29,18	1.2	1.5	0.3
29,19	1.1	1.5	0.4
29,20	1.1	1.4	0.3
29,21	1.1	1.4	0.3
29,22	1.0	1.3	0.3
29,23	1.0	1.3	0.3
29,24	1.0	1.3	0.3
29,25	1.0	1.2	0.2
29,26	1.0	1.2	0.2
29,27	0.9	1.2	0.3
29,28	0.9	1.2	0.3
29,29	0.9	1.2	0.3
29,30	0.9	1.1	0.2
30,2	18.5	38.5	20.0
30,3	7.7	13.1	5.4
30,4	4.8	7.5	2.7
30,5	3.5	5.3	1.8
30,6	2.9	4.1	1.2
30,7	2.4	3.4	1.0
30,8	2.1	2.9	0.8
30,9	1.9	2.6	0.7
30,10	1.7	2.4	0.7
30,11	1.6	2.2	0.6
30,12	1.5	2.0	0.5
30,13	1.4	1.9	0.5
30,14	1.4	1.8	0.4
30,15	1.3	1.7	0.4
30,16	1.2	1.6	0.4
30,17	1.2	1.6	0.4
30,18	1.2	1.5	0.3
30,19	1.1	1.4	0.3
30,20	1.1	1.4	0.3
30,21	1.1	1.4	0.3
30,22	1.0	1.3	0.3
30,23	1.0	1.3	0.3
30,24	1.0	1.3	0.3
30,25	1.0	1.2	0.2
30,26	1.0	1.2	0.2
30,27	0.9	1.2	0.3
30,28	0.9	1.2	0.3
30,29	0.9	1.1	0.2
30,30	0.9	1.1	0.2

Appendix D

Code from chapter 3.1.1 Consequences of Consequences of Flexibility in Sample Size:

Inflated effect size.

```
# In[ ]:

import numpy as np
import pandas as pd
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
# from statsmodels.stats.power import TTestIndPower
from statsmodels.stats.power import tt_ind_solve_power

# set parameters
alpha = 0.05
power1 = 0.9
power2 = 0.3
mean1 = 0
mean2 = 1
sd1, sd2 = 1,1
effect_size = (mean2-mean1/sd1)

# In[ ]:

# create function to calculate cohens_d
def cohen(n1, n2, mean1, mean2, s1, s2 ):
#     pooled_sd = (((n1-1)*s1**2)+((n2-1)*s2**2)/(n1+n2-2))**0.5
    pooled_sd = np.sqrt((sd1**2+sd2**2)*0.5)
    cohen_d = ((mean2-mean1)/pooled_sd)
    return cohen_d

# In[ ]:

# compute sample size, according to predefined power levels
# power_analysis = TTestIndPower
size1 = tt_ind_solve_power(effect_size=effect_size, power=power1, alpha=alpha,
                                        alternative='two-sided')
size2 = tt_ind_solve_power(effect_size=effect_size, power=power2, alpha=alpha,
                                        alternative='two-sided')
# size1 = power_analysis.solve_power(effect_size = effect_size, power = power1,
                                        alpha = alpha, alternative = 'two-sided',
                                        self)
# size2 = power_analysis.solve_power(effect_size = effect_size, power = power2,
                                        alpha = alpha, alternative = 'two-sided')
sample_size1 = np.round(size1,0).astype(int)
sample_size2 = np.round(size2,0).astype(int)
print(sample_size1)
print(sample_size2)
```

```
print(effect_size)

# In[ ]:

nsim = 10000 # 10000 simulations
cohen_d1 = []
cohen_d2 = []
results1 = []
results2 = []
for i in range(0,nsim):
    # compute samples
    sample1_1 = np.random.normal(loc=mean1, scale=sd1, size=sample_size1)
    sample1_2 = np.random.normal(loc=mean2, scale=sd2, size=sample_size1)
    sample2_1 = np.random.normal(loc=mean1, scale=sd1, size=sample_size2)
    sample2_2 = np.random.normal(loc=mean2, scale=sd2, size=sample_size2)
    # measure standardized deviation: cohen_d
    d1 = cohen(mean1=np.mean(sample1_1), mean2=np.mean(sample1_2), n1=sample_size1,
                                                    n2=sample_size1,
                    s1=np.std(sample1_1), s2=np.std(sample1_2))
    d2 = cohen(mean1=np.mean(sample2_1), mean2=np.mean(sample2_2), n1=sample_size2,
                                                    n2=sample_size2,
                    s1=np.std(sample2_1), s2=np.std(sample2_2))
    # perform two-tailed two-sample t-test and store p-values
    result1 = stats.ttest_rel(a = sample1_2, b = sample1_1)[1]#, equal_var=True)[1]
    result2 = stats.ttest_rel(a = sample2_2, b = sample2_1)[1]#, equal_var=True)[1]
    # store values
    cohen_d1.append(d1)
    cohen_d2.append(d2)
    results1.append(result1)
    results2.append(result2)

# In[ ]:

df1 = pd.DataFrame([cohen_d1, results1], index=['d1', 'results1']).transpose()
df2 = pd.DataFrame([cohen_d2, results2], index=['d2', 'results2']).transpose()

# In[ ]:

# search insignificant d
min1 = df1.query('results1 > 0.05').results1.min()
for i in range(len(df1)):
    if df1.iloc[i,1] == min1:
        print(df1.iloc[i,:])
        mind1 = df1.iloc[i,0]
min2 = df2.query('results2 > 0.05').results2.min()
for i in range(len(df2)):
    if df2.iloc[i,1] == min2:
        print(df2.iloc[i,:])
        mind2 = df2.iloc[i,0]
```

```
# In[ ]:

max1 = df1.query('results1 <= 0.05').results1.max()
for i in range(len(df1)):
    if df1.iloc[i,1] == max1:
        print(df1.iloc[i,:])
        maxd1 = df1.iloc[i,0]
max2 = df2.query('results2 <= 0.05').results2.max()
for i in range(len(df2)):
    if df2.iloc[i,1] == max2:
        print(df2.iloc[i,:])
        maxd2 = df2.iloc[i,0]

# In[ ]:

lower, upper = -1,3
with sns.axes_style('darkgrid'):
    _, bins, _ = plt.hist(x=df1.d1, density=1, bins=50, color='lightseagreen')
    mu, sigma = stats.norm.fit(df1.d1)
    curve = stats.norm.pdf(bins, mu, sigma)
    plt.plot(bins,curve, color='navy')
    plt.axvline(mind1,color='maroon')
    plt.text(s='smallest \nsignificant \neffect',x=0,y=1, color='maroon', fontsize=
                                                'large')
    plt.xlim(lower,upper)
    plt.xlabel('cohen‘s d')
    plt.ylabel('density')
    plt.text(s='power = 90%',x=1.8,y=1.2,fontsize='x-large')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.1\plot1')

# In[ ]:

lower, upper = -1,3
with sns.axes_style('darkgrid'):
    _, bins, _ = plt.hist(x=df2.d2, density=1, bins=50, color='lightseagreen')
    mu, sigma = stats.norm.fit(df2.d2)
    curve = stats.norm.pdf(bins, mu, sigma)
    plt.plot(bins,curve, color='navy')
    plt.axvline(mind2,color='maroon')
    plt.text(s='smallest \nsignificant \neffect',x=1.8,y=0.5, color='maroon',
                                                fontsize='large')
    plt.xlim(lower,upper)
    plt.xlabel('cohen‘s d')
    plt.ylabel('density')
    plt.text(s='power = 30%',x=-.9,y=.6,fontsize='x-large')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.1\plot2')
```

Appendix E

Code from chapter 3.1.2 Power Analysis:

α-β trade-off.

```
# In[ ]:

import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np

# In[ ]:

# null hypothesis: N(0,1)
mu = 0
sigma = 1
alpha = 0.05

# sample data
mean = 3
sd = 1
num=1000

# beta = 0.2
critical_value = stats.norm.ppf(0.975) # alpha = 5%, two-tailed
rejection_area = np.linspace(critical_value, mu+5*sigma, num=num)
beta_area = np.linspace(mu-1*sigma, critical_value, num=num)

x = np.linspace(mu-1*sigma, mu+5*sigma, num=num)
curve = stats.norm.pdf(x=x, loc=mu, scale=sigma)
sample_curve = stats.norm.pdf(x=x, loc=mean, scale=sd)

# In[ ]:

with sns.axes_style('darkgrid'):
    # plot null distribution
    ax = sns.lineplot(x=x, y=curve, color='orangered')
    # plot sample distribution
    sns.lineplot(x=x, y=sample_curve, color='lightseagreen')
    #shade in error areas
    ax.fill_between(x=beta_area, y1=stats.norm.pdf(x=beta_area, loc=mean, scale=sd)
                                                    ,
                    color='paleturquoise')
    ax.fill_between(x=rejection_area, y1=stats.norm.pdf(x=rejection_area, loc=mu,
                                                scale=sigma),
                    color='salmon')
    plt.axvline(x=critical_value, color='black')
    plt.text(s='critical \nvalue', x=1.2, y=0.3)
    # plot expected value under H0
```

```
    plt.axvline(x=mu, color='orangered')
    plt.text(s='H0:\nmean=0', x=0.1, y=0.15, color='orangered')
    plt.axvline(x=mean, color='lightseagreen')
    plt.text(s='H1:\nmean=3', x=3.1, y=0.15 ,color='lightseagreen')
    ax.set(xlabel='test statistic', ylabel='density')
    plt.title('alpha=5%')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.2\plot1')

# In[ ]:

power = stats.norm.sf(x=critical_value, loc=mean, scale=sd)
beta = stats.norm.cdf(x=critical_value, loc=mean, scale=sd)
print(power)
print(beta)
power + beta

# In[ ]:

# increase alpha area to show trade-off
new_alpha = 0.2
new_crit = stats.norm.isf(new_alpha/2)
print(critical_value)
print(new_crit)

new_alpha_area = np.linspace(new_crit, mu+5*sigma, num=num)
new_beta_area = np.linspace(mu-1*sigma, new_crit, num=num)

# In[ ]:

with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x, y=curve, color='orangered')
    sns.lineplot(x=x, y=sample_curve, color='lightseagreen')
    ax.fill_between(x=new_beta_area, y1=stats.norm.pdf(x=new_beta_area, loc=mean,
                                            scale=sd), color='paleturquoise')
    ax.fill_between(x=new_alpha_area, y1=stats.norm.pdf(x=new_alpha_area, loc=mu,
                                            scale=sigma), color='salmon')
    plt.axvline(x=new_crit, color='black')
    plt.text(s='critical \nvalue', x=1.3, y=0.3)
    plt.axvline(x=mu, color='orangered')
    plt.text(s='H0:\nmean=0', x=0.1, y=0.15, color='orangered')
    plt.axvline(x=mean, color='lightseagreen')
    plt.text(s='H1:\nmean=3', x=3.1, y=0.15 ,color='lightseagreen')
    ax.set(xlabel='test statistic', ylabel='density')
    plt.title('alpha=20%')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.2\plot2')

# In[ ]:
```

```
new_power = stats.norm.sf(x=new_crit, loc=mean, scale=sd)
new_beta = stats.norm.cdf(x=new_crit, loc=mean, scale=sd)
print(new_power)
print(new_beta)
new_power + new_beta
```

Appendix F

Code from chapter 3.1.2 Power Analysis:

Power analysis.

```
# In[ ]:

from statsmodels.stats.power import normal_power
import numpy as np
import scipy.stats as stats
import scipy.integrate as integrate
import matplotlib.pyplot as plt
import seaborn as sns

# In[ ]:

alpha = 0.05
mean_null = 3 # null hypothesis
sd = 3 # standard deviation in the population
n = 10 # sample size
se = sd/np.sqrt(n) # standard error of the sampling distribution
data = np.random.normal(loc=4.5, scale=3, size=n) # sample
d = (np.mean(data)-mean_null)/sd # effect size in cohens' d
z = (np.mean(data)-mean_null)/se # test statistic in one-sample z-test

# In[ ]:

c = stats.norm.isf(q=alpha) # identical to statsmodels computation
c_shifted = c - d*np.sqrt(n)/1 # in statsmodels: sigma=1, shifted in amount of z-
                                score
c_null = stats.norm.isf(q=alpha, loc=mean_null, scale=se)
power = stats.norm.sf(c_shifted)

# In[ ]:

pow_ = normal_power(effect_size=d, nobs=n, alpha=alpha, alternative='larger')

# In[ ]:

# build plot
power_area = np.linspace(c_null, np.mean(data)+5*se, num=1000)
pow_area = np.linspace(c_shifted, 5, num=1000)

x = np.linspace(start=-4, stop=9, num=1000)
null_dist = stats.norm.pdf(x=x, loc=mean_null, scale=se) # sampling distribution
                                under H0
alternative_dist = stats.norm.pdf(x=x, loc=np.mean(data), scale=se) # sampling
                                distirbution under sample
sample_dist = stats.norm.pdf(x=x, loc=0, scale=1) # sample distribution in z-test
with sns.axes_style('darkgrid'):
```

```
    ax1 = sns.lineplot(x=x, y=sample_dist, color='seagreen',label='null
                                            distribution')
    ax1.fill_between(x=pow_area, y1=stats.norm.pdf(x=pow_area, loc=0, scale=1),
                     color='paleturquoise')
    ax2 = sns.lineplot(x=x, y=null_dist, color='orangered', label='probability
                                            under H0')
    ax3 = sns.lineplot(x=x, y=alternative_dist, color='lightseagreen', label='
                                            probability under H1')
    ax3.fill_between(x=power_area, y1=stats.norm.pdf(x=power_area, loc=np.mean(data
                                            ), scale=se),
                     color='paleturquoise', label='power')
    plt.axvline(c_null, color='orangered') # critical value under the null
                                            distribution
    plt.axvline(c, color='seagreen') # critical value under the sample distribution
    plt.text(y=0.455, x=1.6, s='c',color='seagreen')
    plt.text(y=0.455, x=4.5, s='c_sample', color='orangered')
    plt.legend(bbox_to_anchor=(0.7,0.9))
    ax1.set(xlabel='test statistic', ylabel='density')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.2\plot3.png')

# In[ ]:

for i in range(x.size):
    if x[i] > c_null and x[i-1] <= c_null:
        index = i
print(index)
print(f'c={c_null}, x[index]={x[index]}')
print(f'diff={x[index]-c_null}')

# In[ ]:

power_by_integral = integrate.simps(y=alternative_dist[index:alternative_dist.size+
                                        1], x=x[index:x.size+1])

# In[ ]:

print(f'power = {power}, sigma=1')
print(f'pow_ = {pow_}, sigma=1')
print(f'integral = {power_by_integral}')
1-power
```

Appendix G

Code from chapter 3.1.3.1 Arbitrary Sample Size:

Arbitrary sample size.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt

# In[ ]:

size = 101
x = np.linspace(2,size-1,size-2).astype(int)
data = [np.random.normal(loc=0, scale=1, size=i) for i in range(2,size)]
# truth = H1 = positive
# H0: mean = 1
statistic_pos = [stats.ttest_1samp(a=data[i], popmean=0.5)[0] for i in range(size-2
                                                )]
p_pos = [stats.ttest_1samp(a=data[i], popmean=0.5)[1] for i in range(size-2)]
# truth = H0 = negative
# H0: mean = 0
statistic_neg = [stats.ttest_1samp(a=data[i], popmean=0)[0] for i in range(size-2)]
p_neg = [stats.ttest_1samp(a=data[i], popmean=0)[1] for i in range(size-2)]

df = pd.DataFrame(data=[statistic_pos, p_pos, statistic_neg, p_neg], index=['
                                            statistic_pos', 'p_pos', 'statistic_neg',
                                             'p_neg'])
df.columns=x
results = df.transpose()

# In[ ]:

# calculate measured standardized effect sizes
mean = [np.mean(data[i]) for i in range(len(data))]
std = [np.std(data[i]) for i in range(len(data))]
effect_pos = [np.abs(mean[i]-0.5) for i in range(len(data))]
effect_neg = [np.abs(mean[i]-0) for i in range(len(data))]
d_pos = [effect_pos/(np.sqrt(std[i]**2+1)/2) for i in range(len(data))]
d_neg = [effect_neg/(np.sqrt(std[i]**2+1)/2) for i in range(len(data))]
np.mean(d_pos),np.mean(d_neg)

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x, y=results.p_pos, color='orangered', label='H0=false')
    sns.lineplot(x=x, y=results.p_neg, color='lightseagreen', label='H0=true')
```

```
    sns.lineplot(x=x, y=0.05, color='lime', label='alpha=5%')
    g.set(xlabel='sample size', ylabel='p-value')
    plt.legend(loc='upper right')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot2')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x, y=results.statistic_pos,color='orangered', label='H0=
                                        false')
    sns.lineplot(x=x, y=results.statistic_neg, color='lightseagreen', label='H0=
                                        true')
    g.set(xlabel='sample size', ylabel='test statistic')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot1')

# In[ ]:

# repetition
nsim=1000
size = 101
x = np.linspace(2,size-1,size-2).astype(int)
result_p_pos, result_p_neg = np.array([[0.0]*nsim]*(size-2)), np.array([[0.0]*nsim]
                                        *(size-2))

for i in range(nsim):
    data = [np.random.normal(loc=0, scale=1, size=i) for i in range(2,size)]
    statistic_pos = [stats.ttest_1samp(a=data[i], popmean=0.5)[0] for i in range(
                                        size-2)]
    statistic_neg = [stats.ttest_1samp(a=data[i], popmean=0)[0] for i in range(size
                                        -2)]
    p_pos = [stats.ttest_1samp(a=data[i], popmean=0.5)[1] for i in range(size-2)]
    p_neg = [stats.ttest_1samp(a=data[i], popmean=0)[1] for i in range(size-2)]
    result_p_pos[:,i] = p_pos
    result_p_neg[:,i] = p_neg

# In[ ]:

sns.heatmap(result_p_pos)

# In[ ]:

sns.heatmap(result_p_neg)

# In[ ]:

fn, fp = np.array([[0.0]*nsim]*(size-2)), np.array([[0.0]*nsim]*(size-2))
for i in range(size-2):
    for j in range(nsim):
```

```
        if result_p_pos[i,j] < 0.05:
            fn[i,j] = 0 #true positive
        else:
            fn[i,j] = 1 #false negative
        if result_p_neg[i,j] < 0.05:
            fp[i,j] = 1 #false positive
        else:
            fp[i,j] = 0 #true negative

# In[ ]:

mean_p_pos = [np.mean(result_p_pos[i,:]) for i in range(size-2)]
mean_p_neg = [np.mean(result_p_neg[i,:]) for i in range(size-2)]
fn_rate = [np.mean(fn[i,:]) for i in range(size-2)]
fp_rate = [np.mean(fp[i,:]) for i in range(size-2)]
tp_rate = [1-fn_rate[i] for i in range(size-2)]
tn_rate = [1-fp_rate[i] for i in range(size-2)]

FDR = [fp_rate[i]/(tp_rate[i]+fp_rate[i]) for i in range(size-2)]

# In[ ]:

std_p_neg = [np.std(result_p_neg[i,:]) for i in range(size-2)]
std_p_pos = [np.std(result_p_pos[i,:]) for i in range(size-2)]
print(f' std in p_neg: {np.mean(std_p_neg)}')
print(f' std in p_pos: {np.mean(std_p_pos)}')
print(f' ratio={np.mean(std_p_pos)/np.mean(std_p_neg)}')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x, y=mean_p_pos, color='orangered', label='H0=false')
    sns.lineplot(x=x, y=mean_p_neg, color='lightseagreen', label='H0=true')
    sns.lineplot(x=x, y=0.05, color='lime', label='alpha=5%')
    g.set(xlabel='sample size', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot4')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x, y=fn_rate, color='orangered', label='false negative rate'
                                                )
    sns.lineplot(x=x, y=fp_rate, color='lightseagreen', label='false positive rate'
                                                )
    sns.lineplot(x=x, y=FDR, color='maroon', label='false discovery rate')
    g.set(xlabel='sample size', ylabel='probability')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot3')
```

```
# In[ ]:

np.mean(fp_rate)
np.mean(fn_rate[:30])
np.mean(fn_rate[30:])

# In[ ]:

np.mean(fn_rate[30:])

# In[ ]:

# varying effect sizes
nsim=1000
new_size = 201
new_x = np.linspace(2,new_size-1,new_size-2).astype(int)
result1, result2, result3 = np.array([[0.0]*nsim]*(new_size-2)), np.array([[0.0]*
                                        nsim]*(new_size-2)), np.array([[0.0]*nsim
                                        ]*(new_size-2))

for i in range(nsim):
    data = [np.random.normal(loc=0, scale=1, size=i) for i in range(2,new_size)]
    p1 = [stats.ttest_1samp(a=data[i], popmean=-0.2)[1] for i in range(new_size-2)]
    p2 = [stats.ttest_1samp(a=data[i], popmean=-0.5)[1] for i in range(new_size-2)]
    p3 = [stats.ttest_1samp(a=data[i], popmean=-0.8)[1] for i in range(new_size-2)]
    result1[:,i] = p1
    result2[:,i] = p2
    result3[:,i] = p3

# In[ ]:

fn1, fn2, fn3 = np.array([[0.0]*nsim]*(new_size-2)), np.array([[0.0]*nsim]*(
                                        new_size-2)), np.array([[0.0]*nsim]*(
                                        new_size-2))
for i in range(new_size-2):
    for j in range(nsim):
        if result1[i,j] < 0.05:
            fn1[i,j] = 0 #true positive
        else:
            fn1[i,j] = 1
        if result2[i,j] < 0.05:
            fn2[i,j] = 0
        else:
            fn2[i,j] = 1
        if result3[i,j] < 0.05:
            fn3[i,j] = 0
        else:
            fn3[i,j] = 1

# In[ ]:
```

```
FNR1 = [np.mean(fn1[i,:]) for i in range(new_size-2)]
FNR2 = [np.mean(fn2[i,:]) for i in range(new_size-2)]
FNR3 = [np.mean(fn3[i,:]) for i in range(new_size-2)]

p_val1 = [np.mean(result1[i,:]) for i in range(new_size-2)]
p_val2 = [np.mean(result2[i,:]) for i in range(new_size-2)]
p_val3 = [np.mean(result3[i,:]) for i in range(new_size-2)]

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=new_x, y=FNR1, color='salmon', label='FNR at d=0.2')
    sns.lineplot(x=new_x, y=FNR2, color='indianred', label='FNR at d=0.5')
    sns.lineplot(x=new_x, y=FNR3, color='darkred', label='FNR at d=0.8')
    g.set(xlabel='sample size', ylabel='probability')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot5')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=new_x, y=p_val1, color='skyblue', label='p at d=0.2')
    sns.lineplot(x=new_x, y=p_val2, color='royalblue', label='p at d=0.5')
    sns.lineplot(x=new_x, y=p_val3, color='navy', label='p at d=0.8')
    sns.lineplot(x=new_x, y=0.05, color='lime', label='alpha=5%')
    g.set(xlabel='sample size', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot6')
```

Appendix H

Code from chapter 3.1.3.2 Intermediate Testing:

Intermediate testing.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt

# In[ ]:

# generate a sample with H0=true
# generate a sample with H0=false
# perform paired t-test
# measure: p, test statistic = t, false positives = fp, false negatives = fn
# increase sample size by one observation
# repeat test

# !!! use generated data to know truth value of test decision !!!

# In[ ]:

df_pos = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\df1.csv').query('
                                        location==0.5 and scale==1')
df_neg = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\df1.csv').query('
                                        location==0 and scale==1')
df_base = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\df2.csv').query('
                                        scale==1')

# In[ ]:

# truth = H1 = positive
# H0: equal means
# possible outcomes in the population: false negative, true positive
sample1_pos = df_pos.iloc[:,3].values.tolist()
sample2_pos = df_base.iloc[:,2].values.tolist()
effect = (np.mean(sample1_pos)-np.mean(sample2_pos))
d = np.round(effect/np.sqrt((np.std(sample1_pos)**2+np.std(sample2_pos)**2)/2), 4)
print(f'd = {d}')
# truth = H0 = negative
# H0: equal means
# possible outcomes in the population: false positive, true negative
```

```
sample1_neg = df_neg.iloc[:,3].values.tolist()
sample2_neg = sample2_pos

# In[ ]:

group1_pos, group2_pos = [], []
t_pos = np.array([0.0]*len(sample1_pos))
p_pos = np.array([0.0]*len(sample1_pos))
fn = np.array([0.0]*len(sample1_pos))

group1_neg, group2_neg = [], []
t_neg = np.array([0.0]*len(sample1_pos))
p_neg = np.array([0.0]*len(sample1_pos))
fp = np.array([0.0]*len(sample1_pos))

for i in range(len(df_base)):
    # H0 false
    group1_pos.append(sample1_pos[i])
    group2_pos.append(sample2_pos[i])
    statistic_pos = stats.ttest_ind(a=group1_pos,b=group2_pos)
    t_pos[i], p_pos[i] = statistic_pos[0], statistic_pos[1]
    # H0 true
    group1_neg.append(sample1_neg[i])
    group2_neg.append(sample2_neg[i])
    statistic_neg = stats.ttest_ind(a=group1_neg, b=group2_neg)
    t_neg[i], p_neg[i] = statistic_neg[0], statistic_neg[1]

# In[ ]:

x = np.arange(0,101,).astype(int)
df = pd.DataFrame(data=[t_pos, p_pos, t_neg, p_neg], index=['t_pos', 'p_pos', '
                                        t_neg', 'p_neg'], columns=x)
results = df.dropna(axis=1)
results

# In[ ]:

x = np.linspace(2,len(sample1_pos),len(sample1_pos)-1).astype(int)
alpha = 0.05
with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x, y=results.iloc[1,:], color='orangered', label='H0=false'
                                                )
    sns.lineplot(x=x, y=results.iloc[3,:], color='lightseagreen', label='H0=true')
    sns.lineplot(x=x, y=alpha, color='lime', label='alpha=5%')
    ax.set(xlabel='sample size', ylabel='p-value')
    plt.legend(bbox_to_anchor=(0.7,0.2))
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot8')

# In[ ]:
```

```
with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x, y=results.iloc[0,:], color='orangered', label='H0=false')
    sns.lineplot(x=x, y=results.iloc[2,:], color='lightseagreen', label='H0=true')
    g.set(xlabel='sample size', ylabel='test statistic')

# In[ ]:

# varying effects
df1 = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\df1.csv')
df2 = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\df2.csv')

# In[ ]:

sample1 = df1.query('location==0.75 and scale==1').iloc[:,3].values.tolist()
sample2 = df1.query('location==0.5 and scale==1').iloc[:,3].values.tolist()
sample3 = df1.query('location==0.75 and scale==2').iloc[:,3].values.tolist()
sample4 = df1.query('location==0.5 and scale==2').iloc[:,3].values.tolist()
sample5 = df1.query('location==0.25 and scale==2').iloc[:,3].values.tolist()
sample6 = df1.query('location==0.25 and scale==3').iloc[:,3].values.tolist()
baseline1 = df2.query('scale==1').iloc[:,2].values.tolist()
baseline2 = df2.query('scale==2').iloc[:,2].values.tolist()
baseline3 = df2.query('scale==3').iloc[:,2].values.tolist()

# In[ ]:

p1,p2,p3 = np.array([0.0]*(len(sample1)-1)), np.array([0.0]*(len(sample1)-1)), np.
                                        array([0.0]*(len(sample1)-1))
p4,p5,p6 = np.array([0.0]*(len(sample1)-1)), np.array([0.0]*(len(sample1)-1)), np.
                                        array([0.0]*(len(sample1)-1))

cond1_1, cond1_2, cond1_3 = [sample1[0]],[sample2[0]],[sample3[0]]
cond1_4, cond1_5, cond1_6 = [sample4[0]],[sample5[0]],[sample6[0]]

cond2_1, cond2_2, cond2_3 = [baseline1[0]], [baseline2[0]], [baseline3[0]]
for i in range(1,len(sample1)):
    cond1_1.append(sample1[i])
    cond1_2.append(sample2[i])
    cond1_3.append(sample3[i])
    cond1_4.append(sample4[i])
    cond1_5.append(sample5[i])
    cond1_6.append(sample6[i])
    cond2_1.append(baseline1[i])
    cond2_2.append(baseline2[i])
    cond2_3.append(baseline3[i])
    p1[i-1] = stats.ttest_ind(a=cond1_1,b=cond2_1)[1]
    p2[i-1] = stats.ttest_ind(a=cond1_2, b=cond2_1)[1]
    p3[i-1] = stats.ttest_ind(a=cond1_3,b=cond2_2)[1]
```

```
    p4[i-1] = stats.ttest_ind(a=cond1_4,b=cond2_2)[1]
    p5[i-1] = stats.ttest_ind(a=cond1_5,b=cond2_2)[1]
    p6[i-1] = stats.ttest_ind(a=cond1_6,b=cond2_3)[1]

# In[ ]:

d1 = np.round((np.mean(sample1)-np.mean(baseline1)) / np.sqrt((np.std(sample1)**2+
                                        np.std(baseline1)**2)/2),4)
d2 = np.round((np.mean(sample2)-np.mean(baseline1)) / np.sqrt((np.std(sample2)**2+
                                        np.std(baseline1)**2)/2),4)
d3 = np.round((np.mean(sample3)-np.mean(baseline2)) / np.sqrt((np.std(sample3)**2+
                                        np.std(baseline2)**2)/2),4)
d4 = np.round((np.mean(sample4)-np.mean(baseline2)) / np.sqrt((np.std(sample4)**2+
                                        np.std(baseline2)**2)/2),4)
d5 = np.round((np.mean(sample5)-np.mean(baseline2)) / np.sqrt((np.std(sample5)**2+
                                        np.std(baseline2)**2)/2),4)
d6 = np.round((np.mean(sample6)-np.mean(baseline3)) / np.sqrt((np.std(sample6)**2+
                                        np.std(baseline3)**2)/2),4)
print(f'{d1,d2,d3,d4,d5,d6}')

# In[ ]:

print(np.mean(sample1)-np.mean(baseline1))
print(np.mean(sample2)-np.mean(baseline1))
print(np.mean(sample3)-np.mean(baseline2))
print(np.mean(sample4)-np.mean(baseline2))
print(np.mean(sample5)-np.mean(baseline2))
print(np.mean(sample6)-np.mean(baseline3))

# In[ ]:

print(np.mean(baseline3))

# In[ ]:

with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x, y=p6, color='lightblue', label=f'p at d={d6}')
    sns.lineplot(x=x, y=p5, color='dodgerblue', label=f'p at d={d5}')
    sns.lineplot(x=x, y=p4, color='royalblue', label=f'p at d={d4}')
    sns.lineplot(x=x, y=p3, color='darkblue', label=f'p at d={d3}')
    sns.lineplot(x=x, y=0.05, color='lime', label='alpha = 5%')
    ax.set(xlabel='sample size', ylabel='p-value')
    plt.legend(bbox_to_anchor=(.18,1))
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot7')
```

Appendix I

Code from chapter 3.1.3.3 Deleting Outliers:

Deleting outliers.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.power import normal_power

# In[ ]:

df1 = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\df1.csv').query('location==0.5 and
                                        scale==1')
df2 = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\df2.csv').query('scale==1')
df0 = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\df1.csv').query('location==0 and
                                        scale==1')

# In[ ]:

# remove latest added observation
data1 = df1.iloc[:,3].values.tolist()
data2 = df2.iloc[:,2].values.tolist()
t_result = [stats.ttest_ind(a=data1, b=data2)[0]]
p_result = [stats.ttest_ind(a=data1, b=data2)[1]]
# test
while len(data1) > 2:
    del data1[len(data1)-1]
    del data2[len(data2)-1]
    t = stats.ttest_ind(a=data1, b=data2)[0]
    p = stats.ttest_ind(a=data1, b=data2)[1]
    t_result.append(t), p_result.append(p)
# store results
df = pd.DataFrame(data=[t_result, p_result], index=['statistic', 'p_value'])
bac = df.transpose()
#restore data
data1 = df1.iloc[:,3].values.tolist()
data2 = df2.iloc[:,2].values.tolist()

# In[ ]:

# remove next outlier
data0 = df0.iloc[:,3].values.tolist()
```

```
data1 = df1.iloc[:,3].values.tolist()
data2 = df2.iloc[:,2].values.tolist()
t_result = [stats.ttest_ind(a=data1, b=data2)[0]]
t0_result = [stats.ttest_ind(a=data0, b=data2)[0]]
p_result = [stats.ttest_ind(a=data1, b=data2)[1]]
p0_result = [stats.ttest_ind(a=data0, b=data2)[1]]
upper_c = [stats.t.isf(q=0.025, loc=0, scale=1, df=100)]
lower_c = [stats.t.ppf(q=0.025, loc=0, scale=1, df=100)]
mean1, mean2, mean0 = np.mean(data1), np.mean(data2), np.mean(data0)
print(f'mean0={np.mean(data0)}')
print(f'mean1={np.mean(data1)}')
print(f'mean2={np.mean(data2)}')
# test
while len(data1) > 2:
    data1.remove(np.min(data1))
    data2.remove(np.max(data2))
    t = stats.ttest_ind(a=data1, b=data2)[0]
    p = stats.ttest_ind(a=data1, b=data2)[1]
    t_result.append(t), p_result.append(p)
    # compute critical value
    upper_c.append(stats.t.isf(q=0.025, loc=0, scale=1, df=len(data1)))
    lower_c.append(stats.t.ppf(q=0.025, loc=0, scale=2, df=len(data1)))
data1 = df1.iloc[:,3].values.tolist()
data2 = df2.iloc[:,2].values.tolist()
data0 = df0.iloc[:,3].values.tolist()
while len(data0) > 2:
    data0.remove(np.min(data0))
    data2.remove(np.max(data2))
    t0 = stats.ttest_ind(a=data0, b=data2)[0]
    p0 = stats.ttest_ind(a=data0, b=data2)[1]
    t0_result.append(t0), p0_result.append(p0)
# store results
df = pd.DataFrame(data=[t_result, p_result, t0_result, p0_result],
                  index=['statistic', 'p_value', 'statistic_neg', 'p_value_neg'])
out = df.transpose()
#restore data
data0 = df0.iloc[:,3].values.tolist()
data1 = df1.iloc[:,3].values.tolist()
data2 = df2.iloc[:,2].values.tolist()

# In[ ]:

effect = mean1-mean2
std1, std2 = np.std(data1), np.std(data2)

d = np.round(effect / np.sqrt((std1**2+std2**2) / 2), 4)
print(f'H0=false: d={d}')

# In[ ]:
```

```
x = np.arange(0,98)
with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x, y=bac.statistic.iloc[0:98], color='crimson', label='
                                            delete latest oservation, H0=false')
    sns.lineplot(x=x, y=out.statistic.iloc[0:98], color='orangered', label='delete
                                            next outlier, H0=false')
    sns.lineplot(x=x, y=out.statistic_neg.iloc[0:98], color='lightseagreen', label=
                                            'delete next outlier, H0=true')
    sns.lineplot(x=x, y=upper_c[0:98], color='lime', label='upper ciritical value
                                            at alpha=5%')
    g.set(xlabel='number of removed observations', ylabel='test statistic')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot9')

# In[ ]:

x = np.arange(0,100)
with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x, y=bac.p_value, color='crimson', label='delete latest
                                            oservation, H0=false')
    sns.lineplot(x=x, y=out.p_value, color='orangered', label='delete next outlier,
                                             H0=false')
    sns.lineplot(x=x, y=out.p_value_neg, color='lightseagreen', label='delete next
                                            outlier, H0=true')
    sns.lineplot(x=x, y=0.05, color='lime', label='alpha = 5%')
    g.set(xlabel='number of removed observations', ylabel='p-value')

# In[ ]:

bac_max_p = np.max(bac.p_value)
bac_min_p = np.min(bac.p_value)
out_max_p = np.max(out.p_value)
out_min_p = np.min(out.p_value)

print(f'maximum p in backwards: p={bac_max_p}')
print(f'maximum p in outliers: p={out_max_p}')
print(f'delta in maximum p = {bac_max_p-out_max_p}')
print(f'minimum p in backwards: p={bac_min_p}')
print(f'minimum p in outliers: p={out_min_p}')
print(f'(bac/out)-ratio in minumum p = {bac_min_p/out_min_p}')

# In[ ]:

a,b = [],[]
for i in range(2):
    a.append(max(data1))
    b.append(min(data2))
stats.ttest_ind(a=a, b=b)

# In[ ]:
```

```
plt.plot(bac.p_value-out.p_value)
(bac.p_value-out.p_value)[60:80]

# In[ ]:

# varying effects
dfa = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\df1.csv')
dfb = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\df2.csv')

# In[ ]:

# create vectors to capture power
alpha = 0.05
power0, power1, power2, power3, power4, power5, power6 = [],[],[],[],[],[],[]

# In[ ]:

sample1 = dfa.query('location==0.75 and scale==1').iloc[:,3].values.tolist()
sample2 = dfa.query('location==0.5 and scale==1').iloc[:,3].values.tolist()
sample3 = dfa.query('location==0.75 and scale==2').iloc[:,3].values.tolist()
sample4 = dfa.query('location==0.5 and scale==2').iloc[:,3].values.tolist()
sample5 = dfa.query('location==0.25 and scale==2').iloc[:,3].values.tolist()
sample6 = dfa.query('location==0.25 and scale==3').iloc[:,3].values.tolist()
sample = dfa.query('location==0 and scale==3')
sample0 = sample.iloc[:,3].values.tolist()
baseline1 = dfb.query('scale==1').iloc[:,2].values.tolist()
baseline2 = dfb.query('scale==2').iloc[:,2].values.tolist()
baseline3 = dfb.query('scale==3').iloc[:,2].values.tolist()

d1 = np.round((np.mean(sample1)-np.mean(baseline1)) / np.sqrt((np.std(sample1)**2+
                                        np.std(baseline1)**2)/2),4)
d2 = np.round((np.mean(sample2)-np.mean(baseline1)) / np.sqrt((np.std(sample2)**2+
                                        np.std(baseline1)**2)/2),4)
d3 = np.round((np.mean(sample3)-np.mean(baseline2)) / np.sqrt((np.std(sample3)**2+
                                        np.std(baseline2)**2)/2),4)
d4 = np.round((np.mean(sample4)-np.mean(baseline2)) / np.sqrt((np.std(sample4)**2+
                                        np.std(baseline2)**2)/2),4)
d5 = np.round((np.mean(sample5)-np.mean(baseline2)) / np.sqrt((np.std(sample5)**2+
                                        np.std(baseline2)**2)/2),4)
d6 = np.round((np.mean(sample6)-np.mean(baseline3)) / np.sqrt((np.std(sample6)**2+
                                        np.std(baseline3)**2)/2),4)
d0 = np.round((np.mean(sample0)-np.mean(baseline3)) / np.sqrt((np.std(sample0)**2+
                                        np.std(baseline3)**2)/2),4)
print(f'{d1,d2,d3,d4,d5,d6, d0}')

p1 = [stats.ttest_ind(a=sample1, b=baseline1)[1]]
p2 = [stats.ttest_ind(a=sample2, b=baseline1)[1]]
```

```
p3 = [stats.ttest_ind(a=sample3, b=baseline2)[1]]
p4 = [stats.ttest_ind(a=sample4, b=baseline2)[1]]
p5 = [stats.ttest_ind(a=sample5, b=baseline2)[1]]
p6 = [stats.ttest_ind(a=sample6, b=baseline3)[1]]
p0 = [stats.ttest_ind(a=sample0, b=baseline3)[1]]

# In[ ]:

while len(sample1) > 2:
    sample1.remove(min(sample1))
    sample2.remove(min(sample2))
    baseline1.remove(max(baseline1))
    p1.append(stats.ttest_ind(a=sample1, b=baseline1)[1])
    p2.append(stats.ttest_ind(a=sample2, b=baseline1)[1])
    sample3.remove(min(sample3))
    sample4.remove(min(sample4))
    sample5.remove(min(sample5))
    baseline2.remove(max(baseline2))
    p3.append(stats.ttest_ind(a=sample3, b=baseline2)[1])
    p4.append(stats.ttest_ind(a=sample4, b=baseline2)[1])
    p5.append(stats.ttest_ind(a=sample5, b=baseline2)[1])
    sample6.remove(min(sample6))
    sample0.remove(min(sample0))
    baseline3.remove(max(baseline3))
    p6.append(stats.ttest_ind(a=sample6, b=baseline3)[1])
    p0.append(stats.ttest_ind(a=sample0, b=baseline3)[1])
    # compute power
    D1 = (np.mean(sample1)-np.mean(baseline1)) / np.sqrt((np.std(sample1)**2+np.std
                                            (baseline1)**2)/2)
    D2 = (np.mean(sample2)-np.mean(baseline1)) / np.sqrt((np.std(sample2)**2+np.std
                                            (baseline1)**2)/2)
    D3 = (np.mean(sample3)-np.mean(baseline2)) / np.sqrt((np.std(sample3)**2+np.std
                                            (baseline2)**2)/2)
    D4 = (np.mean(sample4)-np.mean(baseline2)) / np.sqrt((np.std(sample4)**2+np.std
                                            (baseline2)**2)/2)
    D5 = (np.mean(sample5)-np.mean(baseline2)) / np.sqrt((np.std(sample5)**2+np.std
                                            (baseline2)**2)/2)
    D6 = (np.mean(sample6)-np.mean(baseline3)) / np.sqrt((np.std(sample6)**2+np.std
                                            (baseline3)**2)/2)
    D0 = (np.mean(sample0)-np.mean(baseline3)) / np.sqrt((np.std(sample0)**2+np.std
                                            (baseline3)**2)/2)
    power1.append(normal_power(effect_size=D1, nobs=len(sample1), alpha=alpha,
                                            alternative='two-sided'))
    power2.append(normal_power(effect_size=D2, nobs=len(sample1), alpha=alpha,
                                            alternative='two-sided'))
    power3.append(normal_power(effect_size=D3, nobs=len(sample1), alpha=alpha,
                                            alternative='two-sided'))
    power4.append(normal_power(effect_size=D4, nobs=len(sample1), alpha=alpha,
                                            alternative='two-sided'))
    power5.append(normal_power(effect_size=D5, nobs=len(sample1), alpha=alpha,
```

```
                                                alternative='two-sided'))
    power6.append(normal_power(effect_size=D6, nobs=len(sample1), alpha=alpha,
                                                alternative='two-sided'))
    power0.append(normal_power(effect_size=D0, nobs=len(sample1), alpha=alpha,
                                                alternative='two-sided'))

# In[ ]:

with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x[:15], y=p6[:15], color='lightblue', label=f'p at d={d6}')
    sns.lineplot(x=x[:15], y=p5[:15], color='dodgerblue', label=f'p at d={d5}')
    sns.lineplot(x=x[:15], y=p4[:15], color='royalblue', label=f'p at d={d4}')
    sns.lineplot(x=x[:15], y=0.05, color='lime', label='alpha = 5%')
    ax.set(xlabel='number of removed observations', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.3\plot10')

# In[ ]:

d = 0.08333
c = stats.norm.isf(q=alpha/2) # identical to statsmodels computation
c_shifted = c - d*np.sqrt(95)/1 # in statsmodels: sigma=1, shifted in amount of z-
                                        score
power = stats.norm.sf(c_shifted)
print(power)
pow_ = normal_power(effect_size=d, alpha=alpha, nobs=95, alternative='two-sided')
pow_
```

Appendix J

Code from chapter 3.1.3.4 False Positive Rates

FPR.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

# In[ ]:

nsim = 1000
size = 100

# minsize = 10
# minsize = 20
minsize = 30

alpha = 0.01
# alpha = 0.05
# alpha = 0.1

# In[ ]:

# deleting outliers
count = []
for n in range(nsim):
    df1 = pd.DataFrame([np.random.normal(0,1,size)]).transpose()
    df2 = pd.DataFrame([np.random.normal(0,1,size)]).transpose()
    data1, data2 = df1.values.tolist(), df2.values.tolist()
    hit = 0
    for i in range(minsize,size+1):
        data1.remove(np.min(data1))
        data2.remove(np.max(data2))
        p = stats.ttest_rel(a=data1, b=data2)[1]
        if p <= alpha:
            hit += 1
    count.append(hit)

# In[ ]:

# intermediate testing
inter_count = []
for n in range(nsim):
    df1 = pd.DataFrame([np.random.normal(0,1,size)]).transpose()
    df2 = pd.DataFrame([np.random.normal(0,1,size)]).transpose()
    data1, data2 = df1.values.tolist(), df2.values.tolist()
    hit = 0
```

```
    for i in range(minsize,size):
        del (data1[len(data1)-1])
        del (data2[len(data2)-1])
        p = stats.ttest_rel(a=data1, b=data2)[1]
        if p <= alpha:
            hit += 1
    inter_count.append(hit)

# In[ ]:

df = pd.DataFrame(data = [inter_count, count],index = ['intermediate','outlier']).
                                                transpose()
FPR_outlier = len(df.query('outlier > 0'))/nsim
FPR_intermediate = len(df.query('intermediate > 0.05'))/nsim

[FPR_intermediate, FPR_outlier]

# In[ ]:

# arbitrary sample size
arb_result = np.array([[0.0]*(size-minsize)]*nsim)
for n in range(nsim):
    arb = []
    for i in range(minsize,size):
        data1 = np.random.normal(loc=0, scale=1, size=i)
        data2 = np.random.normal(loc=0, scale=1, size=i)
        p = stats.ttest_rel(data1,data2)[1]
        arb.append(p)
    arb_result[n,:] = arb
arb_count = []
for i in range(size-minsize):
    hit = 0
    for n in range(nsim):
        if arb_result[n,i] <= alpha:
            hit += 1
    arb_count.append(hit)
np.mean([arb_count[i]/nsim for i in range(size-minsize)])
```

Appendix K

Code from chapter 3.1.4 Post Hoc Power Analysis:

Post hoc power analysis.

```
# In[ ]:

import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
import scipy.integrate as integrate
from statsmodels.stats.power import zt_ind_solve_power
from statsmodels.stats.proportion import proportions_ztest
from statsmodels.stats.power import normal_power

# In[ ]:

n1, n2 = 29, 37
N = n1+n2
x1, x2 = 8, 26
p1_hat, p2_hat = x1/n1, x2/n2
p_hat = (x1+x2)/(n1+n2)

se = (p_hat*(1-p_hat)*((1/n1)+(1/n2)))**0.5 # standard error

mean_null = 0 # H0: mean2-mean1=0
mean_sample = p2_hat-p1_hat # sample: mean2-mean1= 0.7-0.28=0.42

# In[ ]:

# reported effect size = 0.426
# measured effect size = mean_sample = 0.42684
mean_reported = 0.426
z_measured = (mean_sample - mean_null)/se
p_measured = 2*stats.norm.sf(x=z_measured) # assuming two-tailed test was conducted
# reported p=0.0013:
z_reported = stats.norm.isf(q = 0.0013/2) # assuming two-tailed test was conducted
# z = (effect - 0 /se) <=> se = effect/z
se_reported = mean_reported/z_reported
p_reported = 2*stats.norm.sf(x=z_reported) # assuming two-tailed test was conducted

print(f'measured effect size = {mean_sample}')
print(f'reported effect size = {mean_reported}')
print(f'measured z = {z_measured}')
print(f'reported z = {z_reported}')
print(f'measured se = {se}')
print(f'reported se = {se_reported}')
print(f'measured p = {p_measured}')
print(f'reported p = {p_reported}')
```

```
# se_reported - se
p_reported/p_measured
p_reported-p_measured

# In[ ]:

### power analysis: reported effect and se ###

# In[ ]:

sd_reported = se_reported * np.sqrt(N)
sd_measured = se*np.sqrt(N)
d_reported = mean_reported/sd_reported
d_measured = mean_sample/sd_measured
c_reported = stats.norm.isf(q=p_reported) - d_reported*np.sqrt(N)
c_measured = stats.norm.isf(q=p_measured) - d_measured*np.sqrt(N)
d_reported, d_measured

# In[ ]:

x = np.linspace(-5,5,100)
curve = stats.norm.pdf(x=x, loc=0, scale=1)
sns.lineplot(x=x, y=curve)
plt.axvline(c_reported)
plt.axvline(c_measured, color='orange')

# In[ ]:

power1 = stats.norm.sf(c_reported)
power2 = stats.norm.sf(c_measured)
print(power1)
print(power2)
power1-power2

# In[ ]:
### power analysis: estimated effects and reported se ###
# In[ ]:

effects = [.2, .4, .6, .8]
d = [effects[0]/sd_reported, effects[1]/sd_reported, effects[2]/sd_reported,
                                        effects[3]/sd_reported]
c_base = stats.norm.isf(q=p_reported)
c = [c_base-d[0]*np.sqrt(N), c_base-d[1]*np.sqrt(N), c_base-d[2]*np.sqrt(N), c_base
                                        -d[3]*np.sqrt(N)]
d

# In[ ]:

sns.lineplot(x=x, y=curve)
plt.axvline(c[0])
```

```
plt.axvline(c[1])
plt.axvline(c[2])
plt.axvline(c[3])
power = [stats.norm.sf(c[0]),stats.norm.sf(c[1]),stats.norm.sf(c[2]),stats.norm.sf(
                                                    c[3])]
power

# In[ ]:
### power analysis: estimated effects and estimated se ###
# In[ ]:

d_est = [0.2, 0.5, 0.8]
c_est = [c_base-d_est[0]*np.sqrt(N), c_base-d_est[1]*np.sqrt(N), c_base-d_est[2]*np
                                                    .sqrt(N)]
sns.lineplot(x=x, y=curve)
plt.axvline(c_est[0])
plt.axvline(c_est[1])
plt.axvline(c_est[2])
power = [stats.norm.sf(c_est[0]),stats.norm.sf(c_est[1]),stats.norm.sf(c_est[2])]
power
```

Appendix L

Code from chapter 3.1.4.2 Power Analysis Based on Estimates:

Study power function.

```
# In[ ]:

import numpy as np
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt

# In[ ]:

### plot power on effect size axis ###
p, alpha = 0.0013, 0.05
c_p = stats.norm.isf(p)
c_alpha = stats.norm.isf(alpha)

N = 66
mean_reported = 0.426
z_reported = stats.norm.isf(q = p/2) # assuming two-tailed test was performed
se_reported = mean_reported/z_reported
d = np.linspace(0,1,1000)

# based on estimates
c_est_p, c_est_alpha = np.array([0.0]*len(d)), np.array([0.0]*len(d))
pow_est_p, pow_est_alpha = np.array([0.0]*len(d)),np.array([0.0]*len(d))
# based on reported se
d_se = d/(se_reported*np.sqrt(N))
c_se_p, c_se_alpha = np.array([0.0]*len(d)), np.array([0.0]*len(d))
pow_se_p, pow_se_alpha = np.array([0.0]*len(d)), np.array([0.0]*len(d))

for i in range(len(d)):
    c_est_p[i], c_est_alpha[i] = c_p-d[i]*np.sqrt(N), c_alpha-d[i]*np.sqrt(N)
    pow_est_p[i], pow_est_alpha[i] = stats.norm.sf(c_est_p[i]), stats.norm.sf(
                                                c_est_alpha[i])
    c_se_p[i], c_se_alpha[i] = c_p-d_se[i]*np.sqrt(N), c_alpha-d_se[i]*np.sqrt(N)
    pow_se_p[i], pow_se_alpha[i] = stats.norm.sf(c_se_p[i]), stats.norm.sf(
                                                c_se_alpha[i])

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=d, y=pow_est_p, color='firebrick', label='estimated sd, p=0.
                                                13%')
    sns.lineplot(x=d, y=pow_est_alpha, color='orangered', label='estimated sd,
                                                alpha=5%')
    sns.lineplot(x=d, y=pow_se_p, color='darkcyan', label='reported sd, p=0.13%')
    sns.lineplot(x=d, y=pow_se_alpha, color='darkturquoise', label='reported sd,
                                                alpha=5%')
    g.set(xlabel='cohen‘s d', ylabel='power', ylim=(-0.05,1.05))
```

```
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter3.4\plot')
```

Appendix M

Code from chapter 3.2 Flexibility in Sample Size:

Simulation data.

```
# In[36]:

import pandas as pd
import numpy as np

# In[37]:

max_size = 101
max_sigma = 3
min_mean, max_mean = 0, 1
Group1 = np.array([[[0.0]*max_size]*(max_sigma)]*5)
Group2 = np.array([[0.0]*max_size]*(max_sigma))

# In[38]:

# create data
for i in range(5):
    for j in range(1,max_sigma+1):
        Group1[i,j-1,:] = np.random.normal(loc=i/4, scale=j, size=max_size)
for j in range(1,max_sigma+1):
    Group2[j-1,:] = np.random.normal(loc=0, scale=j, size=max_size)

# In[39]:

# reorganize data into vector format
data1 = [Group1[i,j-1,k]
         for i in range(5)
         for j in range(1,max_sigma+1)
         for k in range(max_size)]
data2 = [Group2[j-1,k]
         for j in range(1,max_sigma+1)
         for k in range(max_size)]

# In[40]:

# create multi-level index for dataframe
range_ = np.linspace(0,1,5)
tuples1 = tuple([location,scale,size]
                for location in range_
                for scale in range(1,max_sigma+1)
                for size in range(max_size))
tuples2 = tuple([scale, size]
                for scale in range(1,max_sigma+1)
                for size in range(max_size))
index1 = pd.MultiIndex.from_tuples(tuples1, names=['location', 'scale', 'size'])
index2 = pd.MultiIndex.from_tuples(tuples2, names=['scale','size'])
```

```
# build dataframes
df1 = pd.DataFrame(data1,index=index1)
df2 = pd.DataFrame(data2, index=index2)

# In[41]:

df1.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\data\
                                        df1.csv')
df2.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\data\
                                        df2.csv')
```

Appendix N

Code from chapter 3.2.2 Significant Findings Are Certain:

Adding categorical variables.

```
# In[ ]:

import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# In[ ]:

nobs = 120 # number of observations
nsim = 100 # number of simulations
max_man = 20 # maximum number of additionally measured manipulated variables (
                                        categorical)

# In[ ]:

# ANOVA: all conditions differ at once
denom_res = np.array([[0.0]*max_man]*nsim)
numer_res = np.array([[0.0]*max_man]*nsim)
prob_res = np.array([[0.0]*max_man]*nsim)
numer = []
for a in range(nsim):
    data = np.random.normal(loc=0, scale=1, size=nobs)
    hit, trial = 0, 0
    numer, denom = [],[]
    for i in range(max_man):
        man_var = np.round(np.random.randint(low=0,high=4,size=nobs),1) #
                                                manipulated variable
        df = pd.DataFrame([data,man_var], index=['data', 'conditions']).transpose()
        result = stats.f_oneway(df.query('conditions==0').data,df.query('conditions
                                                ==1').data,
                                df.query('conditions==2').data, df.query('conditions
                                                                    ==3').data
                                                                    ,)
        if result[1] <= 0.05:
            hit += 1
        trial += 1
        numer.append(hit)
        denom.append(trial)
    denom_res[a,:], numer_res[a,:], prob_res[a,:] = denom, numer, [numer[z]/denom[z
                                            ] for z in range(len(numer))]

# In[ ]:
```

```
ANOVA_df = pd.DataFrame([
    [np.mean(prob_res[:,i]) for i in range(max_man)],
    [np.mean(numer_res[:,i]) for i in range(max_man)],
    [np.mean(denom_res[:,i]) for i in range(max_man)]],
    index=['probability','hit_count','trial_count']).transpose()

# In[ ]:

# t-test: detect each single mean difference
denom_res = np.array([[0.0]*max_man]*nsim)
numer_res = np.array([[0.0]*max_man]*nsim)
prob_res = np.array([[0.0]*max_man]*nsim)

for a in range(nsim):
    hit,trial = 0,0
    numer, denom = [],[]
    data = np.random.normal(loc=0, scale=1, size=nobs)
    for i in range(max_man):
        man_var = np.round(np.random.randint(low=0,high=4,size=nobs),1) #
                                                    manipulated variable
        df = pd.DataFrame([data,man_var], index=['data', 'condition']).transpose()
        result1 = stats.ttest_ind(a=df.query('condition==0').data, b=df.query('
                                                    condition==1').data)[1]
        if result1 <= 0.05:
            hit += 1
        result2 = stats.ttest_ind(a=df.query('condition==0').data, b=df.query('
                                                    condition==2').data)[1]
        if result2 <= 0.05:
            hit += 1
        result3 = stats.ttest_ind(a=df.query('condition==0').data, b=df.query('
                                                    condition==3').data)[1]
        if result3 <= 0.05:
            hit += 1
        result4 = stats.ttest_ind(a=df.query('condition==1').data, b=df.query('
                                                    condition==2').data)[1]
        if result4 <= 0.05:
            hit += 1
        result5 = stats.ttest_ind(a=df.query('condition==1').data, b=df.query('
                                                    condition==3').data)[1]
        if result5 <= 0.05:
            hit += 1
        result6 = stats.ttest_ind(a=df.query('condition==2').data, b=df.query('
                                                    condition==3').data)[1]
        if result6 <= 0.05:
            hit += 1
        trial += 6
        numer.append(hit)
        denom.append(trial)
    denom_res[a,:], numer_res[a,:], prob_res[a,:] = denom, numer, [numer[z]/denom[z
```

```
                                                ] for z in range(len(numer))]

# In[ ]:

t_df = pd.DataFrame([
    [np.mean(prob_res[:,i]) for i in range(max_man)],
    [np.mean(numer_res[:,i]) for i in range(max_man)],
    [np.mean(denom_res[:,i]) for i in range(max_man)]],
    index=['probability','hit_count','trial_count']).transpose()

# In[ ]:

with sns.axes_style('darkgrid'):
    plt.plot(t_df.probability)
    plt.plot(ANOVA_df.probability)

# In[ ]:

with sns.axes_style('darkgrid'):
    plt.plot(t_df.hit_count)
    plt.plot(ANOVA_df.hit_count)

# In[ ]:

with sns.axes_style('darkgrid'):
    plt.plot(t_df.trial_count)
    plt.plot(ANOVA_df.trial_count)

# In[ ]:

# import dataframes from additional continuous variables
df_30r = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs30_rtest.
                                        csv')
df_200r = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs200_rtest.
                                        csv')
df_30t = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs30_ttest.
                                        csv')
df_200t = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs200_ttest.
                                        csv')

# In[ ]:

x = np.arange(1,21)
with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x, y=df_30r.hit_count, color='darkturquoise', label='
                                            continuous variable, correlation test
```

```
                                                    ')
    sns.lineplot(x=x, y=df_30t.hit_count, color='darkcyan', label='continuous
                                                    variable, t-test')
    sns.lineplot(x=x, y=ANOVA_df.hit_count, color='orangered', label='categorical
                                                    variable, ANOVA')
    sns.lineplot(x=x, y=t_df.hit_count, color='firebrick', label='categorical
                                                    variable, t-test')
    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.set(xlim=(1,20),xlabel='number of independent variables',ylabel='number of
                                                    positive findings')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                            chapter4\plot3')

# In[ ]:

with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x, y=df_30r.probability, color='darkturquoise', label='
                                                    continuouse variable, correlation
                                                    test')
    sns.lineplot(x=x, y=df_30t.probability, color='darkcyan', label='continuous
                                                    variable, t-test')
    sns.lineplot(x=x, y=ANOVA_df.probability, color='orangered', label='categorical
                                                     variable ANOVA')
    sns.lineplot(x=x, y=t_df.probability, color='firebrick', label='categorical
                                                    variable, t-test')
    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.set(xlim=(1,20),xlabel='number of independent variables',ylabel='probaiblity
                                                     of positive finding')
```

Appendix O

Code from chapter 3.2.2 Significant Findings Are Certain:

Adding continuous variables.

```
# In[ ]:

import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from itertools import combinations

# In[ ]:

nobs = 30
nvar = 20
nsim = 1000

# In[ ]:

denom_res = np.array([[0.0]*nvar]*nsim)
numer_res = np.array([[0.0]*nvar]*nsim)
prob_res = np.array([[0.0]*nvar]*nsim)
for a in range(nsim):
    df,columns = pd.DataFrame(np.random.normal(loc=0,scale=2,size=nobs)),[0]
    denom, numer = [],[]
    for i in range(nvar):
        columns.append(i+1)
        df.insert(loc=i+1,column=i+1,value=np.random.normal(loc=0,scale=2,size=nobs
                                                            ))
        df.columns=columns
        tuples = list(combinations(columns,2))
        denom.append(len(tuples))
        count = []
        for j in range(len(tuples)):
            sample = tuples[j]
            result = (stats.pearsonr(x=df[sample[0]],y=df[sample[1]]))[1] #
                                                              correlation test
#             result = stats.ttest_rel(a=df[sample[0]],b=df[sample[1]])[1] # mean
                                                    difference test
            if result <= 0.05:
                count.append(1)
            else:
                count.append(0)
        numer.append(np.sum(count))
    denom_res[a,:], numer_res[a,:], prob_res[a,:] = denom, numer, [numer[z]/denom[z
                                                ] for z in range(len(numer))]
```

```
# In[ ]:

result_df = pd.DataFrame([
    [np.mean(prob_res[:,i]) for i in range(nvar)],
    [np.mean(numer_res[:,i]) for i in range(nvar)],
    [np.mean(denom_res[:,i]) for i in range(nvar)]],
    index=['probability','hit_count','trial_count']).transpose()

# In[ ]:

nobs = 200
nvar = 20
nsim = 1000

# In[ ]:

denom_res = np.array([[0.0]*nvar]*nsim)
numer_res = np.array([[0.0]*nvar]*nsim)
prob_res = np.array([[0.0]*nvar]*nsim)
for a in range(nsim):
    df,columns = pd.DataFrame(np.random.normal(loc=0,scale=2,size=nobs)),[0]
    denom, numer = [],[]
    for i in range(nvar):
        columns.append(i+1)
        df.insert(loc=i+1,column=i+1,value=np.random.normal(loc=0,scale=2,size=nobs
                                                            ))
        df.columns=columns
        tuples = list(combinations(columns,2))
        denom.append(len(tuples))
        count = []
        for j in range(len(tuples)):
            sample = tuples[j]
            result = (stats.pearsonr(x=df[sample[0]],y=df[sample[1]]))[1] #
                                                              correlation test
#             result = stats.ttest_rel(a=df[sample[0]],b=df[sample[1]])[1] # mean
                                                 difference test
            if result <= 0.05:
                count.append(1)
            else:
                count.append(0)
        numer.append(np.sum(count))
    denom_res[a,:], numer_res[a,:], prob_res[a,:] = denom, numer, [numer[z]/denom[z
                                                ] for z in range(len(numer))]

# In[ ]:

second_result_df = pd.DataFrame([
    [np.mean(prob_res[:,i]) for i in range(nvar)],
    [np.mean(numer_res[:,i]) for i in range(nvar)],
```

```
    [np.mean(denom_res[:,i]) for i in range(nvar)]],
    index=['probability','hit_count','trial_count']).transpose()

# In[ ]:

# mean values:

# saving ttest results as csv
# result_df.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit
                                        \data\added_nobs30_ttest.csv')
# second_result_df.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs200_ttest.
                                        csv')

# saving correlation test results as csv
result_df.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                        data\added_nobs30_rtest.csv')
second_result_df.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs200_rtest.
                                        csv')

# In[ ]:

df_30r = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs30_rtest.
                                        csv')
df_200r = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs200_rtest.
                                        csv')
df_30t = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs30_ttest.
                                        csv')
df_200t = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\
                                        Bachelorarbeit\data\added_nobs200_ttest.
                                        csv')

# In[ ]:

x = np.arange(2,22)
with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x, y=df_30r.hit_count, color='darkcyan', label='correlation
                                            test, nobs=30')
    sns.lineplot(x=x, y=df_200r.hit_count, color='darkturquoise', label='
                                            correlation test, nobs=200')
    sns.lineplot(x=x, y=df_30t.hit_count, color='firebrick', label='t-test nobs=30'
                                            )
    sns.lineplot(x=x, y=df_200t.hit_count, color='orangered', label='t-test nobs=
                                            200')
    ax.set(xlim=(2,10), ylim=(0,3), xlabel='number of variables', ylabel='number of
                                            significant findings')
```

```
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter4\plot2.png')

# In[ ]:

with sns.axes_style('darkgrid'):
    ax = sns.lineplot(x=x, y=df_30r.probability, color='darkcyan', label='
                                            correlation test, nobs=30')
    sns.lineplot(x=x, y=df_200r.probability, color='darkturquoise', label='
                                            correlation test, nobs=200')
    sns.lineplot(x=x, y=df_30t.probability, color='firebrick', label='t-test nobs=
                                            30')
    sns.lineplot(x=x, y=df_200t.probability, color='orangered', label='t-test nobs=
                                            200')
    ax.set(xlim=(2,21),ylim=(0,0.1),xlabel='number of variables', ylabel='
                                            percentage of significant findings')
    ax.xaxis.set_major_locator(ticker.MultipleLocator(2))
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter4\plot1.png')
```

Appendix P

Code from chapter 3.3.1 Flexibility in Hypothesis Design:

Introductory example for flexibility in hypothesis design.

```
# In[ ]:

import seaborn as sns
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

np.random.seed(0)

mu = 0
sigma = 1
size = 10000
data = np.random.normal(mu, sigma, size)

t = [((np.mean(data)-mu)/sigma)*np.sqrt(len(data))]

critical_two = [stats.norm.ppf(loc=mu, scale=sigma, q=0.025), stats.norm.ppf(loc=mu
                                        , scale=sigma, q=0.975)]
critical_one = [stats.norm.ppf(loc=mu, scale=sigma, q=0.05), stats.norm.ppf(loc=mu,
                                         scale=sigma, q=0.95)]

# In[ ]:

x = np.linspace(start=mu-2.5*sigma, stop=mu+2.5*sigma, num=len(data))
curve = stats.norm.pdf(x=x, loc=mu, scale=sigma)
with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=x,y=curve, color='lightseagreen', label='null distribution')
    sns.scatterplot(x=critical_two ,y=0, label='two-tailed critical values', color=
                                            'maroon', marker='s')
    sns.scatterplot(x=critical_one, y=0, label='one-tailed critical values', color=
                                            'orangered', marker='s')
    sns.scatterplot(x=t, y=0, label='test statistic', color='navy', s=60)
    g.legend(loc='upper right', bbox_to_anchor=(1.2,1))
    g.set(xlabel='test statistic', ylabel='density')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot1.png')
```

Appendix Q

Code from chapter 3.3.1.1 Z-test decisions at $\alpha = 5\%$:

Interval of exploitable effect sizes.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import sys
import seaborn as sns
import matplotlib.pyplot as plt

np.set_printoptions(threshold=sys.maxsize)
np.random.seed(0)

mu = 0
sigma = 1
exp_effect = np.round(np.arange(-1, 1.01, 0.01),2)
# exp_effect = [0.1]
sample_size = np.round(np.arange(1,101,1))
# sample_size = [100]
n = 1000 # n iterations over each effect size
upper_one = stats.norm.ppf(q = 0.95)
upper_two = stats.norm.ppf(q = 0.975)

#create matrices to contain T:test statistics, Decision_one, Decision_two: test
                                        decisions for one- and two-sided tests,
                                        respectively
T = np.array([[[0.0]*n]*len(exp_effect)] * len(sample_size))
Decision_one = np.array([[[0]*n]*len(exp_effect)] * len(sample_size))
Decision_two = np.array([[[0]*n]*len(exp_effect)] * len(sample_size))

# In[ ]:

# columns: increase effect size (mu-mean)
# hues: increase sample size
# rows: repeat 10 times, on random data

for j in range(0, len(exp_effect)):
    for k in range(0, len(sample_size)):
        for i in range(0,n):
            sample = np.random.normal(loc=exp_effect[j], size=sample_size[k], scale
                                                    =sigma)
            T[k,j,i] = (((np.mean(sample) - mu)/sigma)*np.sqrt(sample_size[k])).
                                                    astype(float)
            # positive = 1, negative = 0
            if(T[k,j,i] < upper_one):
                Decision_one[k,j,i] = 0
```

```
            else:
                Decision_one[k,j,i] = 1
            if (T[k,j,i] < upper_two):
                Decision_two[k,j,i] = 0
            else:
                Decision_two[k,j,i] = 1

# In[ ]:

def mean(data):
    n = len(data)
    mean = sum(data) / n
    return mean

# In[ ]:

def variance(data):
    # Number of observations
    n = len(data)
    # Mean of the data
    mean = sum(data) / n
    # Square deviations
    deviations = [(x - mean) ** 2 for x in data]
    # Variance
    variance = sum(deviations) / n
    return variance

# In[ ]:

probability_one = np.array([[0.0]*len(exp_effect)] * len(sample_size))
probability_two = np.array([[0.0]*len(exp_effect)] * len(sample_size))
deviation_one = np.array([[0.0]*len(exp_effect)] * len(sample_size))
deviation_two = np.array([[0.0]*len(exp_effect)] * len(sample_size))

for k in range(0,len(sample_size)):
    for j in range(0,len(exp_effect)):
        probability_one[k,j] = mean(Decision_one[k,j,:])
        probability_two[k,j] = mean(Decision_two[k,j,:])
        deviation_one[k,j] = variance(Decision_one[k,j,:])
        deviation_two[k,j] = variance(Decision_two[k,j,:])

# In[ ]:

df_one = pd.DataFrame(probability_one, index=sample_size, columns=exp_effect)
df_two = pd.DataFrame(probability_two, index=sample_size, columns=exp_effect)
df_diff = df_one-df_two
df_std1 = pd.DataFrame(deviation_one, index=sample_size, columns=exp_effect)
df_std2 = pd.DataFrame(deviation_two, index=sample_size, columns=exp_effect)
df_stddiff = df_std1-df_std2
```

```
# In[ ]:

df_one.iloc[:,100:201]

# In[ ]:

sns.heatmap(df_one.iloc[:,100:201], cmap='flare')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot2.png')

# In[ ]:

sns.set_theme(palette='pastel')
g = sns.lineplot(x=exp_effect, y=df_one.iloc[0,:], label='sample size = 1')
sns.lineplot(x=exp_effect, y=df_two.iloc[24,:], label='sample size = 25')
sns.lineplot(x=exp_effect, y=df_one.iloc[39,:], label='sample size = 40')
sns.lineplot(x=exp_effect, y=df_one.iloc[99,:], label='sample size = 100')
g.set(xlabel='effect size', ylabel='p(one-tailed = positive=')

# In[ ]:

sns.heatmap(df_two.iloc[:,100:201], cmap='flare')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot3.png')

# In[ ]:

sns.set_theme(palette='colorblind')
g = sns.lineplot(x=exp_effect, y=df_two.iloc[0,:], label='sample size = 1')
sns.lineplot(x=exp_effect, y=df_two.iloc[24,:], label='sample size = 25')
sns.lineplot(x=exp_effect, y=df_two.iloc[39,:], label='sample size = 40')
sns.lineplot(x=exp_effect, y=df_two.iloc[99,:], label='sample size = 100')
g.set(xlabel='effect size', ylabel='p(two-tailed = positive)')

# In[ ]:

sns.heatmap(df_diff, cmap='flare')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.scatterplot(x=exp_effect, y=df_diff.iloc[4,:], color='dodgerblue',
                                            label='sample size = 5')
    sns.scatterplot(x=exp_effect, y=df_diff.iloc[24,:], color='orangered',label='
                                            sample size = 25')
    sns.scatterplot(x=exp_effect, y=df_diff.iloc[49,:], color='navy', label='sample
                                             size = 50')
    sns.scatterplot(x=exp_effect, y=df_diff.iloc[99,:], color='firebrick', label='
                                            sample size = 100')
    g.set(xlabel='absolute effect size', ylabel='probability difference')
```

```
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot4.png')
print(stats.normaltest(df_one.iloc[:,4]))
print(stats.normaltest(df_diff.iloc[24,:]))
print(stats.normaltest(df_diff.iloc[49,:]))
print(stats.normaltest(df_diff.iloc[99,:]))

# In[ ]:

with sns.axes_style('darkgrid'):
    ax = sns.scatterplot(x=exp_effect, y=df_std1.iloc[0,:], color='seagreen', label
                                        ='one-tailed, sample = 1')
    sns.scatterplot(x=exp_effect, y=df_std2.iloc[0,:], color='lightseagreen', label
                                        ='two-tailed, sample = 1')
    sns.scatterplot(x=exp_effect, y=df_std1.iloc[9,:], color='navy', label='one-
                                        tailed, sample = 10')
    sns.scatterplot(x=exp_effect, y=df_std2.iloc[9,:], color='dodgerblue', label='
                                        two-tailed, sample = 10')
    sns.scatterplot(x=exp_effect, y=df_std1.iloc[99,:], color='firebrick', label='
                                        one-tailed, sample = 100')
    sns.scatterplot(x=exp_effect, y=df_std2.iloc[99,:], color='orangered', label='
                                        two-tailed, sample = 100')
    ax.set(xlabel='ansolute effect size', ylabel='variance')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot5.png')

# In[ ]:

sns.heatmap(df_std1, cmap='flare')

# In[ ]:

sns.heatmap(df_std2, cmap='flare')

# In[ ]:

sample_size = np.arange(start=500, stop=1000, step=1)
effect_size = np.round(np.arange(start=-1, stop=1, step=0.01),2)
iteration = 100
t = np.array([[[0.0]*n]*len(exp_effect)] * len(sample_size))
d_one = np.array([[[0]*n]*len(exp_effect)] * len(sample_size))
d_two = np.array([[[0]*n]*len(exp_effect)] * len(sample_size))

for j in range(0, len(effect_size)):
    for k in range(0, len(sample_size)):
        for i in range(0,iteration):
            sample = np.random.normal(loc=effect_size[j], size=sample_size[k],
                                                scale=sigma)
            t[k,j,i] = (((np.mean(sample) - mu)/sigma)*np.sqrt(sample_size[k])).
                                                astype(float)
```

```
            # positive = 1, negative = 0
            if(t[k,j,i] < upper_one):
                d_one[k,j,i] = 0
            else:
                d_one[k,j,i] = 1
            if (t[k,j,i] < upper_two):
                d_two[k,j,i] = 0
            else:
                d_two[k,j,i] = 1

# In[ ]:

s_one = np.array([[0.0]*len(effect_size)] * len(sample_size))
s_two = np.array([[0.0]*len(effect_size)] * len(sample_size))
prob_one = np.array([[0.0]*len(effect_size)] * len(sample_size))
prob_two = np.array([[0.0]*len(effect_size)] * len(sample_size))

for j in range(0, len(effect_size)):
    for k in range(0,len(sample_size)):
        for i in range(0,iteration):
            s_one[k,j] = s_one[k,j] + d_one[k,j,i]
            s_two[k,j] = s_two[k,j] + d_two[k,j,i]
            prob_one[k,j] = s_one[k,j] / iteration
            prob_two[k,j] = s_two[k,j] / iteration
dfOne = pd.DataFrame(prob_one, index=sample_size, columns=effect_size)
dfTwo = pd.DataFrame(prob_two,index=sample_size, columns=effect_size)
dfDiff = dfOne - dfTwo

# In[ ]:

sns.heatmap(dfDiff, cmap='flare')
```

Appendix R

Code from chapter 3.3.1.2 Standard Normal Distribution:

Interval of exploitable test statistics for the standard normal distribution.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import sys
import seaborn as sns
import matplotlib.pyplot as plt

test_statistic = np.arange(start=-4,stop=4,step=0.001)
n = (len(test_statistic))

decision_one = [0] * n
decision_two = [0] * n

#upper_one: one-sided critical value
#upper_two: two-sided critical value
upper_one = stats.norm.ppf(loc=0, scale=1, q=0.95)
upper_two = stats.norm.ppf(loc=0, scale=1, q=0.975)
print(n)

# In[ ]:

mu = 0
scale = 1

P_one = np.array([0.0] * n)
P_two = np.array([0.0] * n)
Decision_one = np.array([0] * n)
Decision_two = np.array([0] * n)

# In[ ]:

# compute p-values
for i in range(0,n):
    P_one[i] = (1-stats.norm.cdf(x=test_statistic[i], loc=mu, scale=scale))
    if test_statistic[i] > 0:
        P_two[i] = ((1-stats.norm.cdf(x=test_statistic[i], loc=mu, scale=scale))*2)
    else:
        P_two[i] = stats.norm.cdf(x=test_statistic[i], loc=mu, scale=scale)*2
# positive = 1
# negative = 0
    if(test_statistic[i] < upper_one):
        Decision_one[i] = 0
    else:
```

```
        Decision_one[i] = 1
    if (test_statistic[i] < upper_two):
        Decision_two[i] = 0
    else:
        Decision_two[i] = 1

# In[ ]:

curve = stats.norm.pdf(test_statistic, loc=mu, scale=scale, )
with sns.axes_style('darkgrid'):
    g=sns.lineplot(x = test_statistic, y = P_two, color='navy', label='two-tailed p
                                                ')
    sns.lineplot(x = test_statistic, y = P_one, color='orangered', label='right-
                                                tailed p')
    sns.lineplot(x = test_statistic, y = (P_one - P_two), color='mediumvioletred',
                                                label='difference in p')
    sns.lineplot(x = test_statistic, y = 0.05, color='limegreen', label='
                                                significance level')
    sns.lineplot(x = test_statistic, y = curve, color='dodgerblue', label='null
                                                distribution')
    g.legend(loc='upper right')
    g.set(xlabel='test statistic', ylabel='probability / density ')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                            chapter5\plot6.png')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x = test_statistic, y = P_two, color='navy', label='two-tailed
                                                 p')
    sns.lineplot(x = test_statistic, y = P_one, color='orangered', label='right-
                                                tailed p')
    sns.lineplot(x = test_statistic, y = (P_one - P_two), color='mediumvioletred',
                                                label='difference in p')
    sns.lineplot(x = test_statistic, y = 0.05, color='limegreen', label='
                                                significance level')
    g.set(xlim=(1,3.5), ylim=(-0.1,0.1), xlabel='test statistic', ylabel='p-value')
    g.legend(loc='lower right')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                            chapter5\plot7.png')

# In[ ]:

for i in range(0,n):
    if P_one[i-1] > 0.05 and P_one[i] < 0.05 or P_one[i] == 0.05:
        print(f't={test_statistic[i]}, p1={P_one[i]}, p2={P_two[i]}')
        print(f'iteration={i}')
    if P_two[i-1] > 0.05 and P_two[i] < 0.05 or P_two[i] == 0.05:
        print(f't={test_statistic[i]}, p1={P_one[i]}, p2={P_two[i]}')
        print(f'iteration={i}')
```

```
# In[ ]:

print(test_statistic[7000])
print(P_two[7000]-P_one[7000])

# In[ ]:

# compute probability of acchieving vague hypothesis:
(stats.norm.cdf(x=test_statistic[5960])-stats.norm.cdf(x=test_statistic[5645]))*2

# In[ ]:

# testing findings from exploratory analyisis in chapter7: p-hacking demonstration
for i in range(n):
    if test_statistic[i-1] < 2.1533905454694775 and test_statistic[i] >= 2.
                                                    1533905454694775:
        print(i)
        print(test_statistic[i])
        print(P_two[i])
        print(P_one[i])
```

Appendix S

Code from chapter 3.3.1.3 t-Distribution:

Interval of exploitable test statistics for the t-distribution.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import scipy.special as special
import sys
import seaborn as sns
import matplotlib.pyplot as plt

test_statistic = np.arange(start=0,stop=64,step=0.001)
degf = np.arange(start=1, stop=31, step=1)
n = (len(test_statistic))
m = (len(degf))

p_one = np.array([[0.0]*m]*n)
p_two = np.array([[0.0]*m]*n)

print(m)
print(n)

# In[ ]:

for j in range(0,m):
    for i in range(0,n):
        p_one[i,j] = (1-stats.t.cdf(x=test_statistic[i], loc=0, scale=1, df=degf[j]
                                                    ))
        if test_statistic[i] > 0:
            p_two[i,j] = ((1-stats.t.cdf(x=test_statistic[i], loc=0, scale=1, df=
                                                        degf[j]))*2)
        else:
            p_two[i,j] = stats.t.cdf(x=test_statistic[i], loc=0, scale=1, df=degf[j
                                                        ])*2

# In[ ]:

p1_statistic = np.array([0.0]*m)
p2_statistic = np.array([0.0]*m)
statistic_diff = np.array([0.0]*m)
df = pd.DataFrame([p1_statistic,p2_statistic,statistic_diff],
                  index=['p1_statistic', 'p2_statistic', 'statistic_diff'],
                  columns=['df=1', 'df=2', 'df=3', 'df=4', 'df=5', 'df=6', 'df=7',
                                                          'df=8', 'df=9',
                           'df=10', 'df=11', 'df=12', 'df=13', 'df=14', 'df=15', '
                                                                df=16', 'df=17
```

```
                                                                    ',
                          'df=18', 'df=19', 'df=20', 'df=21', 'df=22', 'df=23', '
                                                                  df=24', 'df=25
                                                                    ',
                          'df=26', 'df=27', 'df=28', 'df=29', 'df=30'])

# In[ ]:

for j in range(0,m):
    for i in range(0,n):
        if p_one[i-1,j] > 0.05 and p_one[i,j] < 0.05 or p_one[i,j] == 0.05:
            df.iloc[0,j] = i*0.001
for j in range(0,m):
    for i in range(0,n):
        if p_two[i-1,j] > 0.05 and p_two[i,j] < 0.05 or p_two[i,j] == 0.05:
            df.iloc[1,j] = i*0.001
for j in range(0,m):
    df.iloc[2,j] = df.iloc[1,j]-df.iloc[0,j]

# In[ ]:

df = np.around(df, decimals=8)
# for negative effect size use negative values. null distribution is symmetrical
                                            around 0
df.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\data\
                                            t_results.csv')
df

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x = test_statistic, y = p_one[:,0], color='darkred', label='
                                            one-tailed, df=1')
    sns.lineplot(x = test_statistic, y = p_two[:,0], color='indianred', label='two-
                                            tailed, df=1')
    sns.lineplot(x = test_statistic, y = p_one[:,1], color='purple', label='one-
                                            tailed, df=2')
    sns.lineplot(x = test_statistic, y = p_two[:,1], color='magenta', label='two-
                                            tailed, df=2')
    sns.lineplot(x = test_statistic, y = p_one[:,9], color='green', label='one-
                                            tailed, df=10')
    sns.lineplot(x = test_statistic, y = p_two[:,9], color='springgreen', label='
                                            two-tailed, df=10')
    sns.lineplot(x = test_statistic, y = p_one[:,28], color='navy', label='one-
                                            tailed, df=29')
    sns.lineplot(x = test_statistic, y = p_two[:,28], color='blue', label='two-
                                            tailed, df=29')
    sns.lineplot(x = test_statistic, y = 0.05, color='black')
    g.set(xlim=(0,15), ylim=(0,0.15), xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
```

```
                                                chapter5\plot8.png')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x = test_statistic, y = p_one[:,0], color='darkred', label='
                                                one-tailed, df=1')
    sns.lineplot(x = test_statistic, y = p_two[:,0], color='indianred', label='two-
                                                tailed, df=1')
    sns.lineplot(x = test_statistic, y = p_one[:,1], color='purple', label='one-
                                                tailed, df=2')
    sns.lineplot(x = test_statistic, y = p_two[:,1], color='magenta', label='two-
                                                tailed, df=2')
    sns.lineplot(x = test_statistic, y = p_one[:,9], color='green', label='one-
                                                tailed, df=10')
    sns.lineplot(x = test_statistic, y = p_two[:,9], color='springgreen', label='
                                                two-tailed, df=10')
    sns.lineplot(x = test_statistic, y = p_one[:,28], color='navy', label='one-
                                                tailed, df=29')
    sns.lineplot(x = test_statistic, y = p_two[:,28], color='blue', label='two-
                                                tailed, df=29')
    sns.lineplot(x = test_statistic, y = 0.05, color='black')
    g.set(xlabel='test_statsitic', ylabel='p-value')

# In[ ]:

sns.lineplot(x= test_statistic, y=p_two[:,7])
sns.lineplot(x= test_statistic, y=(p_one[:,7]*2))
```

Appendix T

Code from chapter 3.3.1.4 χ^2-Distribution:

Interval of exploitable test statistics for the χ^2-distribution.

```
# In[ ]:

import pandas as pd
import numpy as np
import scipy.stats as stats
import scipy.special as special
import sys
import seaborn as sns
import matplotlib.pyplot as plt

test_statistic = np.arange(start=0,stop=50,step=0.001)
degf = np.arange(start=1, stop=31, step=1)
n = (len(test_statistic))
m = (len(degf))

#upper_one: one-sided critical value
#upper_two: two-sided critical value
upper_one = np.array([0.0]*m)
upper_two = np.array([0.0]*m)

p_one = np.array([[0.0]*m]*n)
p_two = np.array([[0.0]*m]*n)

print(m)
print(n)

# In[ ]:

for j in range(0,m):
    upper_one[j] = stats.chi2.ppf(df=degf[j], q=0.95)
    upper_two[j] = stats.chi2.ppf(df=degf[j], q=0.975)
    for i in range(0,n):
        p_one[i,j] = (1-stats.chi2.cdf(x=test_statistic[i], df=degf[j]))
        if test_statistic[i] > stats.chi2.ppf(q=0.5,df=degf[j]): #Expected value =
                                                                 df
            p_two[i,j] = ((1-stats.chi2.cdf(x=test_statistic[i], df=degf[j]))*2)
        else:
            p_two[i,j] = stats.chi2.cdf(x=test_statistic[i], df=degf[j])*2

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x = test_statistic, y = p_one[:,0], color='darkred', label='
                                                 one-tailed, df=1')
    sns.lineplot(x = test_statistic, y = p_two[:,0], color='indianred', label='two-
```

```
                                                tailed, df=1')
    sns.lineplot(x = test_statistic, y = p_one[:,9], color='purple', label='one-
                                                tailed, df=10')
    sns.lineplot(x = test_statistic, y = p_two[:,9], color='magenta', label='two-
                                                tailed, df=10')
    sns.lineplot(x = test_statistic, y = p_one[:,19], color='green', label='one-
                                                tailed, df=20')
    sns.lineplot(x = test_statistic, y = p_two[:,19], color='springgreen', label='
                                                two-tailed, df=20')
    sns.lineplot(x = test_statistic, y = p_one[:,29], color='navy', label='one-
                                                tailed, df=30')
    sns.lineplot(x = test_statistic, y = p_two[:,29], color='blue', label='two-
                                                tailed, df=30')
    sns.lineplot(x = test_statistic, y = 0.05, color='black', label='significance
                                                level')
    g.set(xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot9.png')

# In[ ]:

with sns.axes_style('darkgrid'):
   g = sns.lineplot(x = test_statistic, y = p_one[:,0], color='darkred', label='one
                                                -tailed, df=1')
   sns.lineplot(x = test_statistic, y = p_two[:,0], color='indianred', label='two-
                                                tailed, df=1')
   sns.lineplot(x = test_statistic, y = p_one[:,9], color='purple', label='one-
                                                tailed, df=10')
   sns.lineplot(x = test_statistic, y = p_two[:,9], color='magenta', label='two-
                                                tailed, df=10')
   sns.lineplot(x = test_statistic, y = p_one[:,19], color='green', label='one-
                                                tailed, df=20')
   sns.lineplot(x = test_statistic, y = p_two[:,19], color='springgreen', label='
                                                two-tailed, df=20')
   sns.lineplot(x = test_statistic, y = p_one[:,29], color='navy', label='one-
                                                tailed, df=30')
   sns.lineplot(x = test_statistic, y = p_two[:,29], color='blue', label='two-
                                                tailed, df=30')
   sns.lineplot(x = test_statistic, y = 0.05, color='black')
   g.set(xlabel='test statistic', ylabel='p-value',xlim=(0,60), ylim=(0,0.1))

# In[ ]:

p1_statistic = np.array([0.0]*m)
p2_statistic = np.array([0.0]*m)
statistic_diff = np.array([0.0]*m)
df = pd.DataFrame([p1_statistic,p2_statistic,statistic_diff],
                  index=['p1_statistic', 'p2_statistic', 'statistic_diff'],
                  columns=['df=1', 'df=2', 'df=3', 'df=4', 'df=5', 'df=6', 'df=7',
                                                      'df=8', 'df=9',
```

```
                          'df=10', 'df=11', 'df=12', 'df=13', 'df=14', 'df=15', '
                                                              df=16', 'df=17
                                                              ',
                          'df=18', 'df=19', 'df=20', 'df=21', 'df=22', 'df=23', '
                                                              df=24', 'df=25
                                                              ',
                          'df=26', 'df=27', 'df=28', 'df=29', 'df=30'])

# In[ ]:

for j in range(0,m):
    for i in range(0,n):
        if p_one[i-1,j] > 0.05 and p_one[i,j] < 0.05 or p_one[i,j] == 0.05:
            df.iloc[0,j] = i*0.001
for j in range(0,m):
    for i in range(0,n):
        if p_two[i-1,j] > 0.05 and p_two[i,j] < 0.05 or p_two[i,j] == 0.05:
            df.iloc[1,j] = i*0.001
for j in range(0,m):
    df.iloc[2,j] = df.iloc[1,j]-df.iloc[0,j]

# In[ ]:

df = np.around(df, decimals=8)
df.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\data\
                                          chi_results.csv')
df

# In[ ]:

sns.lineplot(x = test_statistic, y = (p_one[:,29]*2*0.01), color='navy', label='one
                                          -tailed, df=30')
sns.lineplot(x = test_statistic, y = (p_two[:,29]*0.01), color='blue', label='two-
                                          tailed, df=30')
sns.lineplot(x = test_statistic, y = stats.chi2.pdf(x=test_statistic, df=30))

# In[ ]:

sns.lineplot(x = test_statistic, y = (p_one[:,9]*2), color='navy', label='one-
                                          tailed, df=30')
sns.lineplot(x = test_statistic, y = p_two[:,9], color='blue', label='two-tailed,
                                          df=30')
sns.lineplot(x = test_statistic, y = stats.chi2.pdf(x=test_statistic, df=10))
```

Appendix U

Code from chapter 3.3.1.5 F-Distribution:

Interval of exploitable test statistics for the F-distribution.

```
# In[ ]:

from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import scipy.stats as stats
import sys
np.set_printoptions(threshold=sys.maxsize)

degf_n = np.arange(start=1, stop=31, step=1)
degf_d = np.arange(start=1, stop=31, step=1)
test_statistic = np.arange(start=1,stop=50,step=0.1)
length = (len(test_statistic))
N = (len(degf_n))
D = (len(degf_d))
# N = D required!
p_one = np.array([[[0.0]*length]*D]*N)
p_two = np.array([[[0.0]*length]*D]*N)

# In[ ]:

for n in range(0,N):
    for d in range(0,D):
        for i in range(0,length):
            p_one[n,d,i] = (1-stats.f.cdf(x=test_statistic[i], dfn=degf_n[n], dfd=
                                                    degf_d[d]))
            if test_statistic[i] > stats.f.ppf(q=0.5,dfn=degf_n[n], dfd=degf_d[d]):
                p_two[n,d,i] = (1-stats.f.cdf(x=test_statistic[i], dfn=degf_n[n],
                                                    dfd=degf_d[d]))*2
            else:
                p_two[n,d,i] = (stats.f.cdf(x=test_statistic[i], dfn=degf_n[n], dfd
                                                    =degf_d[d]))*2

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=test_statistic, y=p_one[0,0,:], color='orangered', label='p1
                                            , df=(1,1)')

    sns.lineplot(x=test_statistic, y=p_one[1,1,:], color='skyblue', label='p1, df=(
                                            2,2)')
    sns.lineplot(x=test_statistic, y=p_one[2,2,:], color='dodgerblue', label='p1,
                                            df=(3,3)')
```

```
    sns.lineplot(x=test_statistic, y=p_one[3,3,:], color='navy', label='p1, df=(4,4
                                            )')

    sns.lineplot(x=test_statistic, y=p_one[9,9,:], color='lightgrey', label='p1, df
                                            =(10,10)')
    sns.lineplot(x=test_statistic, y=p_one[19,19,:], color='grey', label='p1, df=(
                                            20,20)')
    sns.lineplot(x=test_statistic, y=p_one[29,29,:], color='black', label='p1, df=(
                                            30,30)')

    sns.lineplot(x=test_statistic, y=0.05, color='limegreen', label='significance
                                            level')
    g.set(xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot10.png')

# In[ ]:

sns.lineplot(x=test_statistic, y=p_two[1,1,:])
sns.lineplot(x=test_statistic, y=(p_one[1,1,:]*2))

sns.lineplot(x=test_statistic, y=p_two[12,12,:])
sns.lineplot(x=test_statistic, y=(p_one[12,12,:]*2))

sns.lineplot(x=test_statistic, y=p_two[29,29,:])
sns.lineplot(x=test_statistic, y=(p_one[29,29,:]*2))

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=test_statistic, y=p_two[0,0,:], color='orangered', label='p2
                                            , df=(1,1)')

    sns.lineplot(x=test_statistic, y=p_two[1,1,:], color='skyblue', label='p2, df=(
                                            2,2)')
    sns.lineplot(x=test_statistic, y=p_two[2,2,:], color='dodgerblue', label='p2,
                                            df=(3,3)')
    sns.lineplot(x=test_statistic, y=p_two[3,3,:], color='navy', label='p2, df=(4,4
                                            )')

    sns.lineplot(x=test_statistic, y=p_two[9,9,:], color='lightgrey', label='p2, df
                                            =(10,10)')
    sns.lineplot(x=test_statistic, y=p_two[19,19,:], color='grey', label='p2, df=(
                                            20,20)')
    sns.lineplot(x=test_statistic, y=p_two[29,29,:], color='black', label='p2, df=(
                                            30,30)')

    sns.lineplot(x=test_statistic, y=0.05, color='limegreen', label='significance
                                            level')
    g.set(xlabel='test statistic', ylabel='p-value')
```

```
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot11.png')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=test_statistic, y=p_one[1,0,:], color='salmon', label='p1,
                                            df=(2,1)')
    sns.lineplot(x=test_statistic, y=p_one[2,0,:], color='orangered', label='p1, df
                                            =(3,1)')
    sns.lineplot(x=test_statistic, y=p_one[29,0,:], color='firebrick', label='p1,
                                            df=(30,1)')

    sns.lineplot(x=test_statistic, y=p_one[0,1,:], color='skyblue', label='p1, df=(
                                            1,2)')
    sns.lineplot(x=test_statistic, y=p_one[0,2,:], color='dodgerblue', label='p1,
                                            df=(1,3)')
    sns.lineplot(x=test_statistic, y=p_one[0,29,:], color='navy', label='p1, df=(1,
                                            30)')

    sns.lineplot(x=test_statistic, y=0.05, color='limegreen',label='significance
                                            level')
    g.set(xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot12.png')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=test_statistic, y=p_two[1,0,:], color='salmon', label='p2,
                                            df=(2,1)')
    sns.lineplot(x=test_statistic, y=p_two[2,0,:], color='orangered', label='p2, df
                                            =(3,1)')
    sns.lineplot(x=test_statistic, y=p_two[29,0,:], color='firebrick', label='p2,
                                            df=(30,1)')

    sns.lineplot(x=test_statistic, y=p_two[0,1,:], color='skyblue', label='p2, df=(
                                            1,2)')
    sns.lineplot(x=test_statistic, y=p_two[0,2,:], color='dodgerblue', label='p2,
                                            df=(1,3)')
    sns.lineplot(x=test_statistic, y=p_two[0,29,:], color='navy', label='p2, df=(1,
                                            30)')

    sns.lineplot(x=test_statistic, y=0.05, color='limegreen', label='significance
                                            level')
    g.set(xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot13.png')

# In[ ]:
```

```
g = sns.lineplot(x=test_statistic, y=p_one[9,0,:], color='salmon', label='p1, df=(
                                        10,1)')
sns.lineplot(x=test_statistic, y=p_one[19,0,:], color='orangered', label='p1, df=(
                                        20,1)')
sns.lineplot(x=test_statistic, y=p_one[29,0,:], color='firebrick', label='p1, df=(
                                        30,1)')

sns.lineplot(x=test_statistic, y=p_two[0,9,:], color='skyblue', label='p2, df=(1,10
                                        )')
sns.lineplot(x=test_statistic, y=p_two[0,19,:], color='dodgerblue', label='p2, df=(
                                        1,20)')
sns.lineplot(x=test_statistic, y=p_two[0,29,:], color='navy', label='p2, df=(1,30)'
                                        )

sns.lineplot(x=test_statistic, y=0.05, color='black', label='alpha = 0.05')
g.set(xlabel='test statistic', ylabel='p-value')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=test_statistic, y=p_one[1,1,:], color='salmon', label='p1,
                                            df=(2,2)')
    sns.lineplot(x=test_statistic, y=p_one[3,1,:], color='orangered', label='p1, df
                                            =(4,2)')
    sns.lineplot(x=test_statistic, y=p_one[29,1,:], color='firebrick', label='p1,
                                            df=(30,2)')

    sns.lineplot(x=test_statistic, y=p_one[1,2,:], color='skyblue', label='p1, df=(
                                            2,3)')
    sns.lineplot(x=test_statistic, y=p_one[3,2,:], color='dodgerblue', label='p1,
                                            df=(4,3)')
    sns.lineplot(x=test_statistic, y=p_one[29,2,:], color='navy', label='p1, df=(30
                                            ,3)')

    sns.lineplot(x=test_statistic, y=p_one[1,4,:], color='lightgrey', label='p1, df
                                            =(2,5)')
    sns.lineplot(x=test_statistic, y=p_one[3,4,:], color='grey', label='p1, df=(4,5
                                            )')
    sns.lineplot(x=test_statistic, y=p_one[29,4,:], color='black', label='p1, df=(
                                            30,5)')

    sns.lineplot(x=test_statistic, y=0.05, color='limegreen',label='significance
                                            level')
    g.set(xlim=(0,15), ylim=(0.02, 0.3), xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot14.png')

# In[ ]:
```

```
g = sns.lineplot(x=test_statistic, y=p_two[1,1,:], color='orange', label='p2, df=(2
                                        ,2)')
sns.lineplot(x=test_statistic, y=p_two[3,1,:], color='red', label='p2, df=(4,2)')
sns.lineplot(x=test_statistic, y=p_two[29,1,:], color='firebrick', label='p1, df=(
                                        30,2)')

sns.lineplot(x=test_statistic, y=p_two[1,2,:], color='cyan', label='p2, df=(2,3)')
sns.lineplot(x=test_statistic, y=p_two[3,2,:], color='blue', label='p2, df=(4,3)')
sns.lineplot(x=test_statistic, y=p_two[29,2,:], color='navy', label='p2, df=(30,3)'
                                        )

sns.lineplot(x=test_statistic, y=p_two[1,4,:], color='springgreen', label='p2, df=(
                                        2,5)')
sns.lineplot(x=test_statistic, y=p_two[3,4,:], color='green', label='p2, df=(4,5)')
sns.lineplot(x=test_statistic, y=p_two[29,4,:], color='darkolivegreen', label='p2,
                                        df=(30,5)')

sns.lineplot(x=test_statistic, y=0.05, color='black',label='alpha = 0.05')

g.set(xlim=(2,30), ylim=(0.02, 0.3), xlabel='test statistic', ylabel='p-value')

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.lineplot(x=test_statistic, y=p_one[1,1,:], color='salmon', label='p1,
                                            df=(2,2)')
    sns.lineplot(x=test_statistic, y=p_one[1,3,:], color='orangered', label='p1, df
                                            =(2,4)')
    sns.lineplot(x=test_statistic, y=p_one[1,29,:], color='firebrick', label='p1,
                                            df=(2,30)')

    sns.lineplot(x=test_statistic, y=p_one[2,1,:], color='skyblue', label='p1, df=(
                                            3,2)')
    sns.lineplot(x=test_statistic, y=p_one[2,3,:], color='dodgerblue', label='p1,
                                            df=(3,4)')
    sns.lineplot(x=test_statistic, y=p_one[2,29,:], color='navy', label='p1, df=(3,
                                            30)')

    sns.lineplot(x=test_statistic, y=p_one[4,1,:], color='lightgrey', label='p1, df
                                            =(5,2)')
    sns.lineplot(x=test_statistic, y=p_one[4,3,:], color='grey', label='p1, df=(5,4
                                            )')
    sns.lineplot(x=test_statistic, y=p_one[4,29,:], color='black', label='p1, df=(5
                                            ,30)')

    sns.lineplot(x=test_statistic, y=0.05, color='limegreen',label='significance
                                            level')
    g.set(xlim=(0,15), ylim=(0.02, 0.3), xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot15.png')
```

```
# In[ ]:

g = sns.lineplot(x=test_statistic, y=p_two[1,1,:], color='orange', label='p2, df=(2
                                        ,2)')
sns.lineplot(x=test_statistic, y=p_two[1,3,:], color='red', label='p2, df=(2,4)')
sns.lineplot(x=test_statistic, y=p_two[1,29,:], color='firebrick', label='p2, df=(2
                                        ,30)')

sns.lineplot(x=test_statistic, y=p_two[2,1,:], color='cyan', label='p2, df=(3,2)')
sns.lineplot(x=test_statistic, y=p_two[2,3,:], color='blue', label='p2, df=(3,4)')
sns.lineplot(x=test_statistic, y=p_two[2,29,:], color='navy', label='p2, df=(3,30)'
                                        )

sns.lineplot(x=test_statistic, y=p_two[4,1,:], color='springgreen', label='p2, df=(
                                        5,2)')
sns.lineplot(x=test_statistic, y=p_two[4,3,:], color='lightgreen', label='p2, df=(5
                                        ,4)')
sns.lineplot(x=test_statistic, y=p_two[4,29,:], color='darkolivegreen', label='p2,
                                        df=(5,30)')

sns.lineplot(x=test_statistic, y=0.05, color='black',label='alpha = 0.05')

g.set(xlim=(2,20), ylim=(0.02, 0.3), xlabel='test statistic', ylabel='p-value')

# In[ ]:

p1array = np.array([[0.0]*D]*N)
p2array = np.array([[0.0]*D]*N)
t1array = np.array([[0.0]*D]*N)
t2array = np.array([[0.0]*D]*N)
p1p2array = np.array([[0.0]*D]*N)
p2p1array = np.array([[0.0]*D]*N)

p1array[:] = np.NaN
p2array[:] = np.NaN
t1array[:] = np.NaN
t2array[:] = np.NaN
p1p2array[:] = np.NaN
p2p1array[:] = np.NaN

p1 = np.array([0.0]*N*D)
p2 = np.array([0.0]*N*D)
t_p1 = np.array([0.0]*N*D)
t_p2 = np.array([0.0]*N*D)
p1_at_p2 = np.array([0.0]*N*D)
p2_at_p1 = np.array([0.0]*N*D)
tdiff = np.array([0.0]*N*D)
p1diff = np.array([0.0]*N*D)
p2diff = np.array([0.0]*N*D)
```

```
# In[ ]:

for n in range(0,N):
    for d in range(0,D):
        for i in range(0,length):
            if p_one[n,d,i-1,] > 0.05 and p_one[n,d,i] < 0.05 or p_one[n,d,i] == 0.
                                                                    05:
                p1array[n,d] = p_one[n,d,i]
                t1array[n,d] = i * 0.1
                p2p1array[n,d] = p_two[n,d,i]
            if p_two[n,d,i-1] > 0.05 and p_two[n,d,i] < 0.05 or p_two[n,d,i] == 0.
                                                                    05:
                p2array[n,d] = p_two[n,d,i]
                t2array[n,d] = i * 0.1
                p1p2array[n,d] = p_one[n,d,i]
# translate data into vector for dataframes
# single_df for 30x30 grid
# multiple_df for 3X3 grid
# final_df for table in appendix
for i in range(0,N):
    p1[i*N:(i+1)*N] = p1array[i,:]
    p2[i*N:(i+1)*N] = p2array[i,:]
    t_p1[i*N:(i+1)*N] = t1array[i,:]
    t_p2[i*N:(i+1)*N] = t2array[i,:]
    p1_at_p2[i*N:(i+1)*N] = p1p2array[i,:]
    p2_at_p1[i*N:(i+1)*N] = p2p1array[i,:]

# In[ ]:

tdiff = t_p2 - t_p1
p1diff = p2_at_p1 - p1
p2diff = p2 - p1_at_p2

# 2d dataframes for plots and heatmaps
df_p1 = pd.DataFrame(p1array, index=degf_n, columns=degf_d)
df_p2 = pd.DataFrame(p2array, index=degf_n, columns=degf_d)
df_p1p2 = pd.DataFrame(p1p2array, index=degf_n, columns=degf_d)
df_p2p1 = pd.DataFrame(p2p1array, index=degf_n, columns=degf_d)
df_t1 = pd.DataFrame(t1array, index=degf_n, columns=degf_d)
df_t2 = pd.DataFrame(t2array, index=degf_n, columns=degf_d)

# In[ ]:

sns.heatmap(df_p1p2)

# In[ ]:

sns.heatmap(df_p1,cmap='flare')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
```

```
                                        chapter5\plot18.png')

# In[ ]:

sns.heatmap(df_p2p1)

# In[ ]:

sns.heatmap(df_p2)

# In[ ]:

sns.heatmap(df_t2.iloc[0:10,0:10]-df_t1.iloc[0:10,0:10], cmap='flare')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot16.png')

# In[ ]:

# list all df for dataframe
df1 = np.array([0]*N**2)
df2 = np.array([0]*N)
help2 = np.array([0]*N)

for i in range(0,N):
    df2[i] = i+1
    help2[i] = i+1
for i in range(0,N):
    df2 = np.concatenate((df2,help2),axis=0)

#jumps of 10 in df, for second multi-grid plot (multiple_df)
df2_10 = np.array([0]*N)
help_10 = np.array([0]*N)
df1_10 = np.array([0]*N**2)
for i in range(0,N):
    for j in range(0,10):
        df2_10[i*10:(i+1)*10] = i*10
        help_10[i*10:(i+1)*10] = i*10
for i in range(0,10):
    df2_10[i] = 1
    help_10[i] =1
for i in range(0,N-1):
    df2_10 = np.concatenate((df2_10, help_10), axis=0)
for i in range(0,N**2):
    for j in range(0,10*N):
        df1_10[i*10*N:(i+1)*10*N] = i*10
for i in range(0,10*N):
    df1_10[i] = 1

# In[ ]:
```

```
# 1d dataframe for multi-grids
# single_df
data = pd.DataFrame([df1,df2,p1,t_p1,p2_at_p1,p2,t_p2,p1_at_p2],
                    index=['df1','df2','p1','t:p1=5%','p2:p1=5%','p2','t:p2=5%','p1
                                                                    :p2=5%'])
df = data.transpose()
# delete zeros in df1 vector
for i in range(0,N**2):
    df.iloc[(i*N):((i+1)*N),0] = i+1
#create index list, based on (df1,df2)
list1 = df.iloc[:,0].astype(int)
list2 = df.iloc[:,1].astype(int)
list_delimiter = [',']*len(df.iloc[:,0])
list_final = list1.astype(str) + list_delimiter + list2.astype(str)
df.index = list_final
single_df = df.dropna()

# In[ ]:

# multiple_df
data = pd.DataFrame([df1_10,df2_10,df1,df2,p1,t_p1,p2_at_p1,p2,t_p2,p1_at_p2],
                            index=['df1_10','df2_10','df1','df2','p1','t:p1=5%','p2:
                                                                    p1=5%','p2','t
                                                                    :p2=5%','p1:p2
                                                                    =5%'])
multiple_df = data.transpose()
# delete zeros in df1 vector
for i in range(0,N**2):
    multiple_df.iloc[(i*N):((i+1)*N),2] = i+1
multiple_df.index = list_final

# In[ ]:

# final_df
data = pd.DataFrame([df1, df2, t_p1, t_p2, tdiff, p1, p2_at_p1, p1diff, p2,
                                            p1_at_p2, p2diff],
                    index=['df1','df2','t at p1=5%', 't at p2=5%', 't interval', '
                                                                    p1', 'p2 at p1=5%', '
                                                                    delta p at p1=5%',
                            'p2', 'p1 at p2=5%', 'delta p at p2=5%'])
df = data.transpose()
# delete zeros in df1 vector
for i in range(0,N**2):
    df.iloc[(i*N):((i+1)*N),0] = i+1
df.index = list_final
first_df = df.dropna()
second_df = first_df.drop(columns=['df1','df2'])
final_df = second_df.round(decimals=4)
final_df.to_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\
                                            data\F_results.csv')
```

```
# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.FacetGrid(data=multiple_df, col='df2_10', row='df1_10')
    g.map(sns.lineplot,'t:p1=5%', 'p1', color='dodgerblue')
    g.map(sns.lineplot,'t:p1=5%', 'p2:p1=5%', color='navy')
    g.map(sns.lineplot,'t:p2=5%', 'p2', color='orangered')
    g.map(sns.lineplot,'t:p2=5%', 'p1:p2=5%', color='firebrick')
    g.set(xlabel='test statistic', ylabel='p-value')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter5\plot17.png')

# In[ ]:

IndexNames1 = single_df[single_df['df2'] > 10].index
help_df = single_df.drop(IndexNames1)
IndexNames2 = help_df[help_df['df1'] > 10].index
df_1_1 = help_df.drop(IndexNames2)
IndexNames1 = single_df[single_df['df2'] <= 10].index
help_df = single_df.drop(IndexNames1)
IndexNames2 = help_df[help_df['df1'] > 10].index
df_11to30 = help_df.drop(IndexNames2)
IndexNames1 = df_11to30[df_11to30['df2'] > 20].index
df_1_2 = df_11to30.drop(IndexNames1)
IndexNames1 = df_11to30[df_11to30['df2'] < 21].index
df_1_3 = df_11to30.drop(IndexNames1)

# In[ ]:

IndexNames = single_df[single_df['df1'] <= 10].index
help_df = single_df.drop(IndexNames)
IndexNames2 = help_df[help_df['df2'] > 10].index
df_11to30 = help_df.drop(IndexNames2)
IndexNames = df_11to30[df_11to30['df1'] > 20 ].index
df_2_1 = df_11to30.drop(IndexNames)
IndexNames = df_11to30[df_11to30['df1'] < 20 ].index
df_3_1 = df_11to30.drop(IndexNames)

# In[ ]:

with sns.axes_style('whitegrid'):
    g = sns.FacetGrid(data=df_1_1, col='df2', row='df1')
    g.map(sns.scatterplot,'t:p1=5%', 'p1', color='orangered', label='p1=5%')
    g.map(sns.scatterplot,'t:p1=5%', 'p2:p1=5%', color='navy', label='p2 at p1=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p2', color='blue', label='p2=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p1:p2=5%', color='firebrick', label='p1 at p2
                                            =5%')
    g.set(xlabel='test statistic', ylabel='p-value')
```

```
# In[ ]:

with sns.axes_style('whitegrid'):
    g = sns.FacetGrid(data=df_1_2, col='df2', row='df1')
    g.map(sns.scatterplot,'t:p1=5%', 'p1', color='orangered', label='p1=5%')
    g.map(sns.scatterplot,'t:p1=5%', 'p2:p1=5%', color='navy', label='p2 at p1=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p2', color='blue', label='p2=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p1:p2=5%', color='firebrick', label='p1 at p2
                                                =5%')
    g.set(xlabel='test statistic', ylabel='p-value')

# In[ ]:

with sns.axes_style('whitegrid'):
    g = sns.FacetGrid(data=df_1_3, col='df2', row='df1')
    g.map(sns.scatterplot,'t:p1=5%', 'p1', color='orangered', label='p1=5%')
    g.map(sns.scatterplot,'t:p1=5%', 'p2:p1=5%', color='navy', label='p2 at p1=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p2', color='blue', label='p2=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p1:p2=5%', color='firebrick', label='p1 at p2
                                                =5%')
    g.set(xlabel='test statistic', ylabel='p-value')

# In[ ]:

with sns.axes_style('whitegrid'):
    g = sns.FacetGrid(data=df_2_1, col='df2', row='df1')
    g.map(sns.scatterplot,'t:p1=5%', 'p1', color='orangered', label='p1=5%')
    g.map(sns.scatterplot,'t:p1=5%', 'p2:p1=5%', color='navy', label='p2 at p1=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p2', color='blue', label='p2=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p1:p2=5%', color='firebrick', label='p1 at p2
                                                =5%')
    g.set(xlabel='test statistic', ylabel='p-value')

# In[ ]:

with sns.axes_style('whitegrid'):
    g = sns.FacetGrid(data=df_3_1, col='df2', row='df1')
    g.map(sns.scatterplot,'t:p1=5%', 'p1', color='orangered', label='p1=5%')
    g.map(sns.scatterplot,'t:p1=5%', 'p2:p1=5%', color='navy', label='p2 at p1=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p2', color='blue', label='p2=5%')
    g.map(sns.scatterplot,'t:p2=5%', 'p1:p2=5%', color='firebrick', label='p1 at p2
                                                =5%')
    g.set(xlabel='test statistic', ylabel='p-value')

# In[ ]:

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_trisurf(single_df['df1'], single_df['df2'], single_df['t:p1=5%'], cmap='
                                            flare')
```

```
# ax.plot_trisurf(single_df['df1'], single_df['df2'], single_df['t:p2=5%'], cmap='
                                            flare')

surf = ax.plot_trisurf(single_df['df1'], single_df['df2'], single_df['t:p1=5%'],
                                            cmap='rocket')
fig.colorbar(surf, shrink=0.7)
ax.view_init(30,45)
ax.set(xlabel='df1', ylabel='df2', zlabel='t')

# In[ ]:

with sns.axes_style('whitegrid'):
    fig = plt.figure()
    ax = Axes3D(fig)
    ax.scatter(df_1_1['df1'], df_1_1['df2'], df_1_1['t:p1=5%'], color='navy', label
                                                =' t at p1 significant')
    ax.scatter(df_1_1['df1'], df_1_1['df2'], df_1_1['t:p2=5%'], color='orangered')
    ax.view_init(0,15)
    ax.set(xlabel='df1', ylabel='df2', zlabel='test statistic')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                            chapter5\plot19.white.png')
```

Appendix V

Code from chapter 3.4 P-Hacking Demonstration:

p-hacking demonstration.

```
# In[ ]:

import pandas as pd
import numpy as np
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt
from statsmodels.stats.proportion import proportions_ztest
from statsmodels.stats.weightstats import ztest
from statsmodels.stats.weightstats import ttest_ind
from statsmodels.stats.power import normal_power

# In[ ]:

data = pd.read_csv(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit
                                          \data\heart.csv')
data

# In[ ]:
# exploratory analysis
# In[ ]:

data.shape

# In[ ]:

data.nunique()

# In[ ]:

data.isnull().sum()

# In[ ]:

sns.pairplot(data)

# In[ ]:
# correlations
# In[ ]:

cor = data.corr()
sns.heatmap(cor, cmap='coolwarm')
cor

# In[ ]:
```

```
normal_data = data[['age','trestbps','chol','thalach']]
normal_cor = normal_data.corr(method='pearson')
with sns.axes_style('darkgrid'):
    sns.heatmap(normal_cor, annot=True, cmap='coolwarm')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                        chapter6\plot1.png')

# In[ ]:
# test for significance of correlations
# In[ ]:

# assumption: normal distribution
# kolmogorov smirnov test for normality
stats.kstest(rvs=normal_data.age, cdf='norm')# significant
stats.kstest(rvs=normal_data.trestbps, cdf='norm')# significant
stats.kstest(rvs=normal_data.chol, cdf='norm')# significant
stats.kstest(rvs=normal_data.thalach, cdf='norm')# significant
# shapiro-wikls test
stats.shapiro(normal_data.age)# significant
stats.shapiro(normal_data.trestbps)# significant
stats.shapiro(normal_data.chol)# significant
stats.shapiro(normal_data.thalach)# significant
# omnibus test
stats.normaltest(normal_data.age)# significant
# stats.normaltest(normal_data.trestbps)# significant
# stats.normaltest(normal_data.chol)# significant
# stats.normaltest(normal_data.thalach)# significant

# 'all three tests reject H0:normal distribution for all four variables!'

# In[ ]:

# H1: thalach and age are correated
n = len(normal_data)
df = n-2
r1 = normal_cor.thalach.age
# calculate test: Skript zur Vorlesung Analyse multivariater Daten, p.53
# H0: cor = 0
t = (r1*np.sqrt(n-2))/np.sqrt(1-r1**2) # test statistic
x = np.linspace(-10,10,1000)
sampling_dist = stats.t.pdf(x=x, df=df) # distribution under the null hypothesis
sns.lineplot(x=x, y=sampling_dist)
plt.axvline(t)
p_two = stats.t.cdf(x=t,df=df)*2
p_large = stats.t.sf(x=t, df=df)
p_small = stats.t.cdf(t, df=df)
print(f'two-tailed (H1:cor!=0): test statistic = {t}, p-value = {p_two}')
print(f'right-tailed (H1:cor>0): test statistic = {t}, p-value = {p_large}')
print(f'left-tailed (H1:cor<0): test statistic = {t}, p-value = {p_small}')
```

```
# In[ ]:

# H1: age and trestbps are correlated
r2 = normal_cor.age.trestbps
t = (r2*np.sqrt(n-2))/np.sqrt(1-r2**2) # test statistic
sns.lineplot(x=x, y=sampling_dist)
plt.axvline(t)
p_two = stats.t.sf(x=t, df=df)*2
p_large = stats.t.sf(x=t, df=df)
p_small = stats.t.cdf(t, df=df)
print(f'two-tailed (H1:cor!=0): test statistic = {t}, p-value = {p_two}')
print(f'right-tailed (H1:cor>0): test statistic = {t}, p-value = {p_large}')
print(f'left-tailed (H1:cor<0): test statistic = {t}, p-value = {p_small}')

# In[ ]:

# H1: age and chol are correlated
r3 = normal_cor.age.chol
t = (r3*np.sqrt(n-2))/np.sqrt(1-r3**2) # test statistic
sns.lineplot(x=x, y=sampling_dist)
plt.axvline(t)
p_two = stats.t.sf(x=t, df=df)*2
p_large = stats.t.sf(x=t, df=df)
p_small = stats.t.cdf(t, df=df)
print(f'two-tailed (H1:cor!=0): test statistic = {t}, p-value = {p_two}')
print(f'right-tailed (H1:cor>0): test statistic = {t}, p-value = {p_large}')
print(f'left-tailed (H1:cor<0): test statistic = {t}, p-value = {p_small}')

# In[ ]:

# H1: chol and trestbps are correlated
r4 = normal_cor.chol.trestbps
t = (r4*np.sqrt(n-2))/np.sqrt(1-r4**2) # test statistic
sns.lineplot(x=x, y=sampling_dist)
plt.axvline(t)
p_two = stats.t.sf(x=t, df=df)*2
p_large = stats.t.sf(x=t, df=df)
p_small = stats.t.cdf(t, df=df)
print(f'two-tailed (H1:cor!=0): test statistic = {t}, p-value = {p_two}')
print(f'right-tailed (H1:cor>0): test statistic = {t}, p-value = {p_large}')
print(f'left-tailed (H1:cor<0): test statistic = {t}, p-value = {p_small}')

# In[ ]:

# H1:thalach and trestbps are correlated
r5 = normal_cor.thalach.trestbps
t = (r5*np.sqrt(n-2))/np.sqrt(1-r5**2) # test statistic
sns.lineplot(x=x, y=sampling_dist)
plt.axvline(t)
```

```
p_two = stats.t.cdf(x=t, df=df)*2
p_large = stats.t.sf(x=t, df=df)
p_small = stats.t.cdf(t, df=df)
print(f'two-tailed (H1:cor!=0): test statistic = {t}, p-value = {p_two}')
print(f'right-tailed (H1:cor>0): test statistic = {t}, p-value = {p_large}')
print(f'left-tailed (H1:cor<0): test statistic = {t}, p-value = {p_small}')

# In[ ]:

# pearson r test
stats.pearsonr(x=data.thalach, y=data.age)# significant
stats.pearsonr(x=data.trestbps, y=data.age)# significant
stats.pearsonr(x=data.chol, y=data.age)# significant
stats.pearsonr(x=data.chol, y=data.trestbps)# significant
# stats.pearsonr(x=data.thalach, y=data.trestbps)# insignificant
# 'test decissions similar to result sabove'

# In[ ]:

a =  5.628106676351095e-13/5.628106676351095e-13
b = 7.762269074809919e-07/7.762269074809851e-07
c = 0.00017862864341450013/0.00017862864341449124
d = 0.032082053610872296/0.032082053610871034
a,b,c,d

# In[ ]:

# spearman rank correlation test
stats.spearmanr(a=data.thalach, b=data.age) # significant
stats.spearmanr(a=data.trestbps, b=data.age) # signficant
stats.spearmanr(a=data.chol, b=data.age) # significant
stats.spearmanr(a=data.chol, b=data.trestbps) # significant
# stats.spearmanr(a=data.thalach, b=data.trestbps) # insignificant

# In[ ]:

a = 6.024320734620622e-13/5.628106676351095e-13
b = 4.2617094650125134e-07/7.762269074809851e-07
c = 0.0006099143222853829/0.00017862864341449124
d = 0.027608539162108658/0.032082053610871034
a,b,c,d

# In[ ]:

# sort by data type
n = len(data)
print(n)
binary = data[['sex','fbs','exang','target']]
# sex: 1=male
# fbs(fasting blood sugar > 120mg/dl): 1=true
```

```
# exang (exercise induced angina): 1=yes
# target (diagnosis of heart disease): 0 = yes
numer = data[['age','trestbps','chol','thalach','oldpeak']]
# trestbps: resting blood pressure on admission to hospital
# chol: choloestorol levels
#restecg: resting electroradiographic results
# thalach: maximum heart rate acchieved
# oldpeak: ST depression induced by exercise relative to rest
# ca: number of major vessels colored by flourosopy
categ = data[['cp','restecg','slope','thal','ca']]
# cp(chest pain type): 0=typical, 1=atypical, 2=non-anginal, 3=asymptomatic
# restecg (resting electrocardiographic resutls): 0=normal, 1=ST-T wave abnormal, 2
                                        =left ventricular hypertrophy
# slope (pf peak exercise ST segment): 0=up, 1=flat, 2=down
# thal: 1=normal, 2=fixed defect, 3=reversable defect

# In[ ]:
# boxplots of numeric variables
# In[ ]:

fig, axs = plt.subplots(2,3)
axs[0,0].boxplot(x=data.age)
axs[0,1].boxplot(x=data.trestbps)
axs[0,2].boxplot(x=data.chol)
axs[1,0].boxplot(x=data.thalach)
axs[1,1].boxplot(x=data.oldpeak)
axs[1,2].boxplot(x=data.ca)
axs[0,0].set_title('age')
axs[0,1].set_title('resting bps')
axs[0,2].set_title('cholestorol levels')
axs[1,0].set_title('max heart rate')
axs[1,1].set_title('ST depression time?')
axs[1,2].set_title('# major vessels')
fig.tight_layout()

# In[ ]:
# confidence intervals of binary variables
# In[ ]:

p = binary.mean()
se_binary = np.sqrt(p*(1-p)/n)
z_score = stats.norm.ppf(0.975)

sex_ci = [p.sex-z_score*se_binary.sex,p.sex+z_score*se_binary.sex]
fbs_ci = [p.fbs-z_score*se_binary.fbs, p.fbs+z_score*se_binary.fbs]
exang_ci = [p.exang-z_score*se_binary.exang, p.exang+z_score*se_binary.exang]
target_ci = [p.target-z_score*se_binary.target, p.target+z_score*se_binary.target]

# In[ ]:
```

```
# plot 95% confidence intervals (dark)
# plt mean values (light)
with sns.axes_style('darkgrid'):
    plt.scatter(x='% male', y=sex_ci[0], color='darkblue')
    plt.scatter(x='% male', y=sex_ci[1], color='darkblue')
    plt.scatter(x='% male', y=p.sex, color='dodgerblue')
    plt.scatter(x='% fbs>120', y=fbs_ci[0], color='darkblue')
    plt.scatter(x='% fbs>120', y=fbs_ci[1], color='darkblue')
    plt.scatter(x='% fbs>120', y=p.fbs, color='dodgerblue')
    plt.scatter(x='% exercise \ninduced angina', y=exang_ci[0], color='darkblue')
    plt.scatter(x='% exercise \ninduced angina', y=exang_ci[1], color='darkblue')
    plt.scatter(x='% exercise \ninduced angina', y=p.exang, color='dodgerblue')
    plt.scatter(x='% no disease', y=target_ci[0], color='darkblue')
    plt.scatter(x='% no disease', y=target_ci[1], color='darkblue')
    plt.scatter(x='% no disease', y=p.target, color='dodgerblue')
    plt.title('95% confidende interval of binary variables')

# In[ ]:
# value count of categorical variables
# In[ ]:

categ.mode()
# cp (chest pain type): 0=typical anigna, 1=atypical anigna, 2=non-anginal pain, 3=
                                        asymptomatic
# restecg (resting electrocardiographic results): 0=normal, 1=ST-T wave abnormality
                                        , 2=left ventricular hypertrophy
# slpoe (of peak exercise ST segment): 0=up, 1=flat, 2=down
# thal: 1=normal, 2=fixed defect, 3=reversable defect

# In[ ]:

t_ang, a_ang, n_ang, asymp = 0,0,0,0
norm, anorm, ventr = 0,0,0
up, flat, down = 0,0,0
normal, fixed, revers = 0,0,0
fail_cp, fail_restecg, fail_slope, fail_thal = [],[],[],[]
for i in range(len(categ)):
    if categ.cp.iloc[i] == 0:
        t_ang += 1
    elif categ.cp.iloc[i] == 1:
        a_ang += 1
    elif categ.cp.iloc[i] == 2:
        n_ang += 1
    elif categ.cp.iloc[i] == 3:
        asymp += 1
    else:
        fail_cp.append(i)
    if categ.restecg.iloc[i] == 0:
        norm += 1
    elif categ.restecg.iloc[i] == 1:
```

```
        anorm += 1
    elif categ.restecg.iloc[i] == 2:
        ventr += 1
    else:
        fail_restecg.append(i)
    if categ.slope.iloc[i] == 0:
        up += 1
    elif categ.slope.iloc[i] == 1:
        flat += 1
    elif categ.slope.iloc[i] == 2:
        down += 1
    else:
        fail_slope.append(i)
    if categ.thal.iloc[i] == 1:
        normal += 1
    elif categ.thal.iloc[i] == 2:
        fixed += 1
    elif categ.thal.iloc[i] == 3:
        revers += 1
    else:
        fail_thal.append(i)
print(fail_cp)
print(fail_restecg)
print(fail_slope)
print(fail_thal)

# In[ ]:

plt.bar(x='typical angina', height=t_ang)
plt.bar(x='atypical angina', height=a_ang)
plt.bar(x='non-aginal pain', height=n_ang)
plt.bar(x='asymptotic', height=asymp)
print(t_ang/n)
print(asymp/n)

# In[ ]:

plt.bar(x='normal', height=norm)
plt.bar(x='ST-T wave abnormality', height=anorm)
plt.bar(x='ventricular hypertrophy', height=ventr)
print(norm/n)
print(anorm/n)
print(ventr/303)

# In[ ]:

plt.bar(x='up', height=up)
plt.bar(x='flat', height=flat)
plt.bar(x='down', height=down)
print(up/303)
```

```
print(flat/303)
print(down/303)

# In[ ]:

plt.bar(x='normal', height=normal)
plt.bar(x='fixed defect', height=fixed)
plt.bar(x='reversable defect', height=revers)
plt.bar(x='measurement error', height=len(fail_thal))
print(normal)
print(normal/301)
print(fixed/303)

# In[ ]:
# multiple tests of mean on binary variables
# In[ ]:

disease, no_disease = binary.query('target==0'), binary.query('target==1')

m, f = binary.query('sex==1'), binary.query('sex==0')
male, female = m.drop(columns=['sex']), f.drop(columns=['sex'])

y,n_= binary.query('exang==1'), binary.query('exang==0')
yes, no = y.drop(columns=['exang']), n_.drop(columns=['exang'])

fy, fn = binary.query('fbs==1'), binary.query('fbs==0')
fbs_yes, fbs_no = fy.drop(columns=['fbs']), fn.drop(columns=['fbs'])

count_dis, count_nod = disease.sum(), no_disease.sum()
count_male, count_fem = male.sum(), female.sum()
count_yes, count_no = yes.sum(), no.sum()
count_fbsy, count_fbsn = fbs_yes.sum(), fbs_no.sum()

p_male = p.drop(index=['sex'])
p_yes = p.drop(index=['exang'])
p_fbs = p.drop(index=['fbs'])

# In[ ]:

dis_basel, nod_basel, dis_nod = [],[],[]
yes_basel, no_basel, yes_no = [],[],[]
male_basel, fem_basel, male_fem = [],[],[]
fbsy_basel, fbsn_basel, fbs_yes_no = [],[],[]
design = ['smaller','two-sided','larger']
for i in range(3):
    for d in range(3):
        # search for mean differences in target
        dis_basel.append(proportions_ztest(nobs=len(disease), count=count_dis[i],
                                           value=p[i], prop_var=True,
                                           alternative=design[d])[0])
```

```
        dis_basel.append(proportions_ztest(nobs=len(disease), count=count_dis[i],
                                           value=p[i], alternative=design[d]
                                           )[1])
        nod_basel.append(proportions_ztest(nobs=len(no_disease), count=count_nod[i]
                                           , value=p[i], alternative=design[
                                           d])[0])
        nod_basel.append(proportions_ztest(nobs=len(no_disease), count=count_nod[i]
                                           , value=p[i], alternative=design[
                                           d])[1])
        dis_nod.append(proportions_ztest(nobs=len(disease), count=count_dis[i],
                                           value=(count_nod[i]/len(
                                           no_disease)), alternative=design[
                                           d])[0])
        dis_nod.append(proportions_ztest(nobs=len(disease), count=count_dis[i],
                                           value=(count_nod[i]/len(
                                           no_disease)), alternative=design[
                                           d])[1])
        #search for mean differences in exang(exercise induced angina)
        yes_basel.append(proportions_ztest(nobs=len(yes), count=count_yes[i], value
                                           =p_yes[i], prop_var=True,
                                           alternative=design[d])[0])
        yes_basel.append(proportions_ztest(nobs=len(yes), count=count_yes[i], value
                                           =p_yes[i], alternative=design[d])
                                           [1])
        no_basel.append(proportions_ztest(nobs=len(no), count=count_no[i], value=
                                           p_yes[i], alternative=design[d])[
                                           0])
        no_basel.append(proportions_ztest(nobs=len(no), count=count_no[i], value=
                                           p_yes[i], alternative=design[d])[
                                           1])
        yes_no.append(proportions_ztest(nobs=len(yes), count=count_yes[i], value=(
                                           count_no[i]/len(no)), alternative
                                           =design[d])[0])
        yes_no.append(proportions_ztest(nobs=len(yes), count=count_yes[i], value=(
                                           count_no[i]/len(no)), alternative
                                           =design[d])[1])
        # search for mean differences in gender
        male_basel.append(proportions_ztest(nobs=len(male), count=count_male[i],
                                           value=p_male[i], alternative=
                                           design[d])[0])
        male_basel.append(proportions_ztest(nobs=len(male), count=count_male[i],
                                           value=p_male[i], alternative=
                                           design[d])[1])
        fem_basel.append(proportions_ztest(nobs=len(female), count=count_fem[i],
                                           value=p_male[i], alternative=
                                           design[d])[0])
        fem_basel.append(proportions_ztest(nobs=len(female), count=count_fem[i],
                                           value=p_male[i], alternative=
                                           design[d])[1])
        male_fem.append(proportions_ztest(nobs=len(male), count=count_male[i],
```

```
                                                value=(count_fem[i]/len(female)),
                                                 alternative=design[d])[0])
        male_fem.append(proportions_ztest(nobs=len(male), count=count_male[i],
                                                value=(count_fem[i]/len(female)),
                                                 alternative=design[d])[1])
        # search for mean differences in fbs (fasting blood sugar)
        fbsy_basel.append(proportions_ztest(nobs=len(fbs_yes), count=count_fbsy[i],
                                                 value=p_fbs[i], alternative=
                                                design[d])[0])
        fbsy_basel.append(proportions_ztest(nobs=len(fbs_yes), count=count_fbsy[i],
                                                 value=p_fbs[i], alternative=
                                                design[d])[1])
        fbsn_basel.append(proportions_ztest(nobs=len(fbs_no), count=count_fbsn[i],
                                                value=p_fbs[i], alternative=
                                                design[d])[0])
        fbsn_basel.append(proportions_ztest(nobs=len(fbs_no), count=count_fbsn[i],
                                                value=p_fbs[i], alternative=
                                                design[d])[1])
        fbs_yes_no.append(proportions_ztest(nobs=len(fbs_yes), count=count_fbsy[i],
                                                 value=(count_fbsn[i]/len(fbs_no)
                                                ), alternative=design[d])[0])
        fbs_yes_no.append(proportions_ztest(nobs=len(fbs_yes), count=count_fbsy[i],
                                                 value=(count_fbsn[i]/len(fbs_no)
                                                ), alternative=design[d])[1])

target_iterables = [['sex','fbs','exang'],['left-tailed', 'two-tailed', 'right-
                                           tailed'],['statistic', 'p-value']]
sex_iterables = [['fbs','exang','target'],['left-tailed', 'two-tailed', 'right-
                                           tailed'],['statistic', 'p-value']]
exang_interables = [['sex','fbs','target'],['left-tailed', 'two-tailed', 'right-
                                           tailed'],['statistic', 'p-value']]
fbs_iterables = [['sex','exang','target'],['left-tailed', 'two-tailed', 'right-
                                           tailed'],['statistic', 'p-value']]

target_index = pd.MultiIndex.from_product(target_iterables, names=['test', 'design'
                                          , 'result'])
sex_index = pd.MultiIndex.from_product(sex_iterables, names=['test', 'design', '
                                          result'])
exang_index = pd.MultiIndex.from_product(exang_interables, names=['test', 'design',
                                           'result'])
fbs_index = pd.MultiIndex.from_product(fbs_iterables, names=['test','design','
                                          result'])

df_dis_basel, df_nod_basel, df_dis_nod = pd.DataFrame(dis_basel,index=target_index)
                                          , pd.DataFrame(nod_basel,index=
                                          target_index), pd.DataFrame(dis_nod,index
                                          =target_index)
df_male_basel, df_fem_basel, df_male_fem = pd.DataFrame(male_basel, index=sex_index
                                          ), pd.DataFrame(fem_basel, index=
                                          sex_index), pd.DataFrame(male_fem, index=
```

```
                                        sex_index)
df_yes_basel, df_no_basel, df_yes_no = pd.DataFrame(yes_basel, index=exang_index),
                                        pd.DataFrame(no_basel, index=exang_index)
                                        , pd.DataFrame(yes_no, index=exang_index)
df_fbs_yes_basel, df_fbs_no_basel, df_fbs_yes_no = pd.DataFrame(fbsy_basel, index=
                                        fbs_index), pd.DataFrame(fbsn_basel,
                                        index=fbs_index), pd.DataFrame(fbs_yes_no
                                        , index=fbs_index)

# In[ ]:

# couble check p-value calculation in stats.models.proportion -> statsmodels.stats.
                                        weighstats
def p_value (statistic):
    p_small = stats.norm.cdf(x=statistic)
    p_two = stats.norm.sf(x=statistic)*2
    p_large = stats.norm.sf(x=statistic)
    print(f'p_small={p_small}')
    print(f'p_two={p_two}')
    print(f'p_large={p_large}')

statistic=[]

for i in range(3): # number of characteristics to test
    a = proportions_ztest(nobs=len(disease), count=count_dis[i], value=(count_nod[i
                                            ]/len(no_disease)), alternative='two-
                                            sided')[0]
    statistic.append(a)
    if np.abs(a) > 1.646:
        print(f'significant finding detected in iteration {i}, statistic={a}')
        p_value(a)
        if np.abs(a) < 1.96:
            print(f'vague hypothesis detected in iteration {i}')
# results are correct!

# In[ ]:
# flexible sample size
# In[ ]:
# delete latest observation on selected non-significant z-test findings
# In[ ]:

# sex in (data, exang)
sex, e_sex = data.sex.values.tolist(), yes.sex.values.tolist()
statistic, p_val = [],[]
for i in range(len(e_sex)-2):
    count = np.sum(e_sex)
    p = np.mean(sex)
    statistic.append(proportions_ztest(nobs=len(e_sex), count=count, value=p,
                                            alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(e_sex), count=count, value=p,
```

```
                                                alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(i)
        break
    del(e_sex[len(e_sex)-1])
    del(sex[len(sex)-1])

# In[ ]:

# exang in (sex, data)
exang, m_exang = data.exang.values.tolist(), male.exang.values.tolist()
statistic, p_val = [],[]
for i in range(len(m_exang)-2):
    count = np.sum(m_exang)
    p = np.mean(exang)
    statistic.append(proportions_ztest(nobs=len(m_exang), count=count, value=p,
                                                alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(m_exang), count=count, value=p,
                                                alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(statistic[i])
        print(p_val[i])
        print(i)
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(m_exang)) - p
        se = mean/p_val[i]
        n = len(m_exang)+len(exang)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
        print(f'power={power}')
        break
    del(m_exang[len(m_exang)-1])
    del(exang[len(exang)-1])

# In[ ]:

# fbs in (data, disease)
fbs, d_fbs = data.fbs.values.tolist(), disease.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(d_fbs)-2):
    count = np.sum(d_fbs)
    p = np.mean(fbs)
    statistic.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                                alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                                alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
```

```
        print(i)
        break
    del(d_fbs[len(d_fbs)-1])
    del(fbs[len(fbs)-1])

# In[ ]:

# fbs in (disease, no disease)
n_fbs, d_fbs = no_disease.fbs.values.tolist(), disease.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(d_fbs)-2):
    count = np.sum(d_fbs)
    p = np.mean(n_fbs)
    statistic.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(i)
        break
    del(d_fbs[len(d_fbs)-1])
    del(n_fbs[len(n_fbs)-1])

# In[ ]:

# fbs in (male, female)
m_fbs, f_fbs = male.fbs.values.tolist(), female.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(d_fbs)-2):
    count = np.sum(m_fbs)
    p = np.mean(f_fbs)
    statistic.append(proportions_ztest(nobs=len(m_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(m_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        break
    del(m_fbs[len(m_fbs)-1])
    del(f_fbs[len(f_fbs)-1])

# In[ ]:

# fbs in (exang no exang)
e_fbs, n_fbs = yes.fbs.values.tolist(), no.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(d_fbs)-2):
    count = np.sum(e_fbs)
    p = np.mean(n_fbs)
```

```
    statistic.append(proportions_ztest(nobs=len(e_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(e_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(i)
        break
    del(e_fbs[len(e_fbs)-1])
    del(n_fbs[len(n_fbs)-1])

# In[ ]:
# try again with deleting outliers
# In[ ]:

# exang in (sex, data)
exang, m_exang = data.exang.values.tolist(), male.exang.values.tolist()
statistic, p_val = [],[]
for i in range(len(m_exang)-2):
    count = np.sum(m_exang)
    p = np.mean(exang)
    statistic.append(proportions_ztest(nobs=len(m_exang), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(m_exang), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(statistic[i])
        print(p_val[i])
        print(f'sample1 size={len(m_exang)}, sample2 size={len(exang)}')
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(m_exang)) - p
        se = mean/p_val[i]
        n = len(m_exang)+len(exang)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
        print(f'power={power}')
        break
    if m_exang[i] == 0:
        del(m_exang[i])
    if exang[i] == 1:
        del(exang[i])

# In[ ]:

# sex in (data, exang)
sex, e_sex = data.sex.values.tolist(), yes.sex.values.tolist()
statistic, p_val = [],[]
for i in range(len(e_sex)-2):
    count = np.sum(e_sex)
```

```
    p = np.mean(sex)
    statistic.append(proportions_ztest(nobs=len(e_sex), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(e_sex), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(statistic[i])
        print(p_val[i])
        print(f'sample1 size={len(e_sex)}, sample2 size={len(sex)}')
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(e_sex)) - p
        se = mean/p_val[i]
        n = len(e_sex)+len(sex)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
        print(f'power={power}')
        break
    if e_sex[i] == 0:
        del(e_sex[i])
    if sex[i] == 1:
        del(sex[i])

# In[ ]:

# fbs in (data, disease)
fbs, d_fbs = data.fbs.values.tolist(), disease.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(d_fbs)-2):
    count = np.sum(d_fbs)
    p = np.mean(fbs)
    statistic.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(f'sample1 size={len(d_fbs)}, sample2 size={len(fbs)}')
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(d_fbs)) - p
        se = mean/p_val[i]
        n = len(d_fbs)+len(fbs)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
        print(f'power={power}')
        break
    if d_fbs[i] == 0:
        del(d_fbs[i])
    if fbs[i] == 1:
```

```
        del(fbs[i])

# In[ ]:

# fbs in (disease, no disease)
n_fbs, d_fbs = no_disease.fbs.values.tolist(), disease.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(d_fbs)-2):
    count = np.sum(d_fbs)
    p = np.mean(n_fbs)
    statistic.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(d_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(f'sample1 size={len(d_fbs)}, sample2 size={len(n_fbs)}')
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(d_fbs)) - p
        se = mean/p_val[i]
        n = len(d_fbs)+len(n_fbs)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
        print(f'power={power}')
        break
    if d_fbs[i] == 0:
        del(d_fbs[i])
    if n_fbs[i] == 1:
        del(n_fbs[i])
print('\ntest for opposite effect direction: \n')
n_fbs, d_fbs = no_disease.fbs.values.tolist(), disease.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(n_fbs)-2):
    count = np.sum(n_fbs)
    p = np.mean(d_fbs)
    statistic.append(proportions_ztest(nobs=len(n_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(n_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(f'sample1 size={len(n_fbs)}, sample2 size={len(d_fbs)}')
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(n_fbs)) - p
        se = mean/p_val[i]
        n = len(d_fbs)+len(n_fbs)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
```

```
        print(f'power={power}')
        break
    if d_fbs[i] == 1:
        del(d_fbs[i])
    if n_fbs[i] == 0:
        del(n_fbs[i])

# In[ ]:

# fbs in (male, female)
m_fbs, f_fbs = data.fbs.values.tolist(), disease.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(m_fbs)-2):
    count = np.sum(m_fbs)
    p = np.mean(f_fbs)
    statistic.append(proportions_ztest(nobs=len(m_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(m_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(f'sample1 size={len(m_fbs)}, sample2 size={len(f_fbs)}')
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(m_fbs)) - p
        se = mean/p_val[i]
        n = len(m_fbs)+len(f_fbs)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
        print(f'power={power}')
        break
    if m_fbs[i] == 0:
        del(m_fbs[i])
    if f_fbs[i] == 1:
        del(f_fbs[i])

print('\ntest for opposite effect direction: \n')

m_fbs, f_fbs = data.fbs.values.tolist(), disease.fbs.values.tolist()
statistic, p_val = [],[]
for i in range(len(f_fbs)-2):
    count = np.sum(f_fbs)
    p = np.mean(m_fbs)
    statistic.append(proportions_ztest(nobs=len(f_fbs), count=count, value=p,
                                       alternative='larger')[0])
    p_val.append(proportions_ztest(nobs=len(f_fbs), count=count, value=p,
                                       alternative='larger')[1])
    if p_val[i] <= 0.05:
        print(p_val[i])
        print(f'sample1 size={len(f_fbs)}, sample2 size={len(m_fbs)}')
```

```
        # compute power on alpha = 5% and measured effect size
        mean = count/(len(f_fbs)) - p
        se = mean/p_val[i]
        n = len(f_fbs)+len(m_fbs)
        d = mean/se*np.sqrt(n)
        c = stats.norm.isf(q=0.05) - d*np.sqrt(n)
        power = stats.norm.sf(c)
        print(f'power={power}')
        break
    if m_fbs[i] == 1:
        del(m_fbs[i])
    if f_fbs[i] == 0:
        del(f_fbs[i])

# In[ ]:
# flexible sample size on numerical data: multiple small studies
# In[ ]:

sns.pairplot(numer)

# In[ ]:

# test for differences between disease and no disease
# sample size = 46, ngroups = 3
disease = data.query('target == 0')
nodisease = data.query('target == 1')
disease1, nodisease1 = disease.iloc[0:46], nodisease.iloc[0:55]
disease2, nodisease2 = disease.iloc[46:92], nodisease.iloc[55:110]
disease3, nodisease3 = disease.iloc[92:138], nodisease.iloc[110:165]

# In[ ]:
# age
# In[ ]:

sns.histplot(disease1.age, color='navy')
sns.histplot(nodisease1.age, color='lightseagreen')

# In[ ]:

sns.histplot(disease2.age, color='navy')
sns.histplot(nodisease2.age, color='lightseagreen')

# In[ ]:

stats.levene(disease.age, nodisease.age)# significant
stats.levene(disease1.age, nodisease1.age)# significant
stats.levene(disease2.age, nodisease2.age)# insignificant
# stats.levene(disease3.age, nodisease3.age)# significant

# stats.mannwhitneyu(x=disease.age, y=nodisease.age, alternative='larger') #
```

```
                                                significant
# stats.mannwhitneyu(x=disease1.age, y=nodisease1.age, alternative='largerd') #
                                                significant
stats.mannwhitneyu(x=disease2.age, y=nodisease2.age, alternative='greater') #
                                                significant
# stats.mannwhitneyu(x=disease3.age, y=nodisease3.age, alternative='greater') #
                                                insignificant in 'two-sided' (significant
                                                 in 'greater')

# In[ ]:

# try to reach equal variance in both groups (sub-set 1)
print(disease1.age.std()) # increase std
print(nodisease1.age.std()) # decrease std -> del outlier

# In[ ]:

group1, group2 = disease1.age.values.tolist(), nodisease1.age.values.tolist()
std = []
for i in range(len(group1)):
    t = stats.levene(group1, group2)[0]
    p = stats.levene(group1, group2)[1]
    std.append(np.std(group2))
    if p > 0.05:
        group2.remove(np.max(group2)) # pushes group2 down, better for later test
                                                    of mean difference
    else:
        print(f'statsitic = {t}, p = {p}')
        print(len(group2))
        break
    if len(group2) < 2:
        break
stats.mannwhitneyu(group1, group2, alternative='greater') # significant

# In[ ]:

# try to reach equal variance in both groups (sub-set 2)
print(disease2.age.std()) # decrease std -> del outlier
print(nodisease2.age.std()) # increase std

# In[ ]:

group1, group2 = disease2.age.values.tolist(), nodisease2.age.values.tolist()
for i in range(len(group2)):
    t = stats.levene(group1, group2)[0]
    p = stats.levene(group1, group2)[1]
    if p > 0.05:
        group1.remove(np.min(group1)) # pushes group1 up, better for later test of
                                                    mean difference
    else:
```

```
        print(f'statsitic = {t}, p = {p}')
        print(len(group1))
        break
    if len(group1) < 2:
        break
stats.mannwhitneyu(group1, group2, alternative='greater') # significant

# In[ ]:
# oldpeak differences in both group (ST-deppression from exang, relative to rest)
# In[ ]:
# assumption violated: normal distribution (see histogram above)
# -> non-parametric test for mean differences
# In[ ]:

# test assumption: equal variance in both groups
stats.levene(disease1.oldpeak, nodisease1.oldpeak)# significant
stats.levene(disease2.oldpeak, nodisease2.oldpeak)# significant
stats.levene(disease3.oldpeak, nodisease3.oldpeak)# significant
stats.levene(disease.oldpeak, nodisease.oldpeak)# significant

# In[ ]:

# -> assumption satisfied: equal variance
stats.mannwhitneyu(x=disease.oldpeak, y=nodisease.oldpeak, alternative='greater') #
                                            significant
stats.mannwhitneyu(x=disease1.oldpeak, y=nodisease1.oldpeak, alternative='greater')
                                             #significant
stats.mannwhitneyu(x=disease2.oldpeak, y=nodisease2.oldpeak, alternative='greater')
                                             #significant
stats.mannwhitneyu(x=disease3.oldpeak, y=nodisease3.oldpeak, alternative='greater')
                                             #significant

# In[ ]:

sns.histplot(data.oldpeak)

# In[ ]:

with sns.axes_style('darkgrid'):
    g = sns.histplot(data=data, x='oldpeak', hue='target', palette=['orangered','
                                             lightseagreen'])
    g.set(ylabel='absolute frequency', xlabel='ST depression induced by exercise (
                                             oldpeak) in mm')
plt.savefig(r'C:\Users\phili\OneDrive\Dokumente_One Drive\KIT\Bachelorarbeit\plots\
                                            chapter6\plot2.png')

# In[ ]:

sns.histplot(nodisease1.oldpeak, color='orange')
sns.histplot(disease1.oldpeak, color='blue')
```

```
print(nodisease1.oldpeak.std())
print(disease1.oldpeak.std())

# In[ ]:

sns.histplot(nodisease2.oldpeak, color='orange')
sns.histplot(disease2.oldpeak, color='blue')
print(nodisease2.oldpeak.std())
print(disease2.oldpeak.std())

# In[ ]:

sns.histplot(nodisease3.oldpeak, color='orange')
sns.histplot(disease3.oldpeak, color='blue')
print(nodisease3.oldpeak.std())
print(disease3.oldpeak.std())

# In[ ]:

group1 = disease1.oldpeak.values.tolist()
group2 = disease2.oldpeak.values.tolist()
group3 = disease3.oldpeak.values.tolist()
print(len(group1))
print(len(group2))
print(len(group3))
for i in range(len(data)):
    p1 = stats.levene(group1, nodisease1.oldpeak)[1]
    p2 = stats.levene(group2, nodisease2.oldpeak)[1]
    p3 = stats.levene(group3, nodisease3.oldpeak)[1]
    if p1 > 0.05:
        print('group1 is insignificant')
        print(stats.levene(group1, nodisease1.oldpeak))
        print(stats.mannwhitneyu(x=group1, y=nodisease1.oldpeak))
        print(len(group1))
    if p2 > 0.05:
        print('group2 is insignificant')
        print(stats.levene(group2, nodisease2.oldpeak))
        print(stats.mannwhitneyu(x=group2, y=nodisease2.oldpeak))
        print(len(group2))
    if p3 > 0.05:
        print('group3 is insignificant')
        print(stats.levene(group3, nodisease3.oldpeak))
        print(stats.mannwhitneyu(x=group3, y=nodisease3.oldpeak))
        print(len(group3))
    if p1 > 0.05 and p2 > 0.05 and p3 > 0.05:
        print(f'{stats.levene(group1, nodisease1.oldpeak)}, on sample size={len(
                                                group1)}')
        print(f'{stats.levene(group2, nodisease2.oldpeak)}, on sample size={len(
                                                group2)}')
        print(f'{stats.levene(group3, nodisease3.oldpeak)}, on sample size={len(
```

```
                                                    group3)}')
        break
    else:
        group1.remove(np.min(group1))
        group2.remove(np.min(group2))
        group3.remove(np.min(group3))

# In[ ]:
# examplary study results
# prove relationship between sex and chol
# (false) inference: women eat more fast food
# In[ ]:

sns.relplot(data=data, x='sex', y='chol')

# In[ ]:

data_female = data.query('sex==0')
data_male = data.query('sex==1')
mean_m = data_male.chol.mean()
std_m = data_male.chol.std()
mean_f = data_female.chol.mean()
std_f = data_female.chol.std()
n_f = len(data_female)
n_m = len(data_male)
pooled_sd = np.sqrt(((n_f-1)*std_f**2+(n_m-1)*std_m**2)/(n_f+n_m-2))
std_f, std_m # equal_va = False -> Welch's test

# In[ ]:

print([len(data_female),len(data_male)])
print([mean_f, mean_m])
ttest_ind(x1=data_female.chol,x2=data_male.chol, usevar = 'unequal', alternative='
                                            larger')

# In[ ]:

# controlling for all numeric variables
print([data_female.age.median(), data_male.age.median()])
print([data_female.age.std(),data_male.age.std()])
ttest_ind(x1=data_female.age, x2=data_male.age, alternative='two-sided')#
                                            insignificant

# In[ ]:

print([data_female.trestbps.mean(), data_male.trestbps.mean()])
print([data_female.trestbps.std(),data_male.trestbps.std()])
ttest_ind(x1=data_female.trestbps, x2=data_male.trestbps, alternative='two-sided')#
                                             insignificant
```

```
# In[ ]:

# 95% confidence interval
effect = mean_f-mean_m
se = pooled_sd/np.sqrt(n_f+n_m)
l_bound = stats.norm.ppf(loc=effect, scale=se, q=0.025)
u_bound = stats.norm.ppf(loc=effect, scale=se, q=0.975)
conf = 0.975-0.025
conf

# In[ ]:

print([l_bound, effect, u_bound])
print([effect-l_bound, effect+u_bound])

# In[ ]:
# prove relationship between exang and target
# control for age
# In[ ]:

treatment, control = data.query('exang==1'), data.query('exang==0')
mean_treatment = len(treatment.query('target==0'))
mean_control = len(control.query('target==0'))
mean_treatment, mean_control # >5

# In[ ]:

ztest(disease.age, nodisease.age)

# In[ ]:

# show relationship between angina and target
prop1 = mean_treatment/len(treatment)
prop2 = mean_control/len(control)
result = proportions_ztest(nobs=len(treatment), count=mean_treatment, value=prop2)
                                        # significant
result

# In[ ]:

# control for age
# step1: levene test for equal group variance
stats.levene(treatment.age, control.age) # significant

# In[ ]:

sns.histplot(treatment.age, color='navy')
sns.histplot(control.age, color='lightseagreen')
print(len(treatment))
print(len(control))
```

```
treatment.age.std(), control.age.std()

# In[ ]:

# delete min and max from control.age to lower std and keep mean stable
control = data.query('exang==0') # assuring no data loss!
p = stats.levene(treatment.age, control.age)[1]
print(f'p before:{p}')
while p <= 0.05:
    control = control.drop(control.age.idxmax())
    control = control.drop(control.age.idxmin())
    p = stats.levene(treatment.age, control.age)[1]
    print(p)
print(f'p after deleting {204-len(control)} outliers: {p}')

# In[ ]:

# step2: U-test for mean difference in both groups
stats.mannwhitneyu(treatment.age, control.age) # insignificant

# In[ ]:

# repeat main test on cropped data
print(len(control)) # still cropped
mean_treatment = len(treatment.query('target==0'))
mean_control = len(control.query('target==0'))
mean_treatment, mean_control # >5

# In[ ]:

prop1 = mean_treatment/len(treatment)
prop2 = mean_control/len(control)
new_result = proportions_ztest(nobs=len(treatment), count=mean_treatment, value=
                                        prop2) # significant
print(new_result)
result[1]-new_result[1]# no change in p-values

# In[ ]:
'Regional outcomes of severe acute respiratory syndrome coronavirus 2 infection on
                                        hospitalized patients with haematological
                                         malignancy'
# In[ ]:

# check age for normality
age = [1,1,
       2,2,2,2,2,2,2,2,2,
       3,3,3,3,3,3,3,3,3,3,3,
       4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,
       5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
group = [1,1,1,1,1,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,1,1,1,1,1, # treatment=1,
```

```
                                        control=0
        1,1,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,
        1,1,1,1,1,1,0,0,0,0,0,0,0,0]
sars_data = pd.DataFrame(data=[age,group], index=['age', 'group']).transpose()
stats.shapiro(age)# significant
stats.normaltest(age)# insignificant
stats.kstest(rvs=age, cdf='norm')# significant

# In[ ]:
# attempt to replicate the controlling for age
# In[ ]:
# H0: pFCR in treatment > pFCR in control
# treatment: immunoyuppressive or cytotoxic medication
# controlling for age: employed method is not stated in the study
# In[ ]:

# non-parametric test
# assumption: equal variance
stats.levene(sars_data.query('group==1').age, sars_data.query('group==0').age) #
                                        significant
# mean difference in age between treatment and control
stats.mannwhitneyu(x=sars_data.query('group==1').age, y=sars_data.query('group==0')
                                        .age) # insignificant

# In[ ]:

# parametric test
# mean difference in age between tratement and control
stats.ttest_ind(a=sars_data.query('group==1').age, b=sars_data.query('group==0').
                                        age) # insignficant
ztest(x1=sars_data.query('group==1').age, x2=sars_data.query('group==0').age) #
                                        insignificant
```

Declaration of Originality

I hereby declare that I have composed this paper by myself and without any assistance other than the sources given in my list of works cited. This paper has not been submitted in the past or is currently being submitted to any other examination institution. It has not been published. All direct quotes as well as indirect quotes which in phrasing or original idea have been taken from a different text (written or otherwise) have been marked as such clearly and in each single instance under a precise specification of the source.

Karlsruhe, 25th March 2021

.....

Philip Müller