MŰEGYETEM 1782

Budapest University of Technology and Economics

Applied Mathematics Master of Sciences Thesis

The use of Fleishman distribution in the empirical investigation of statistical tests

Author:
Tamás Ferenci

External supervisor:
Prof. Jenő Reiczigel DSc
Internal supervisor:
Dr. Márton Balázs PhD

2012

Abstract

Statistical tests are among the most important tools of modern statistics. Given a population which was only observed partly, we can not decide whether a statement H_0 is true if it pertains to the whole population. However, statistical test can give a probabilistic answer on whether H_0 stands or not, based only on the observed part of the population (called sample). Statistical tests typically impose more or less presumptions about the population; they are guaranteed to produce valid (probabilistic) answers only if these presumptions are met. Practically, the two most important property of a statistical test is robustness and power. Robustness measures how valid the test remains if its presumptions are not met, while power measures how well the test can detect that H_0 is not true, if it in fact does not stand. Practically, the most problematic task is to investigate robustness for tests which have a distribution presumption on the population (called parametric tests), and to investigate power for tests that do not have such assumption (called non-parametric tests). The common point in these (which presents their difficulty) is that both require to investigate the operation of a test on a population with non-specified distribution. (In robustness testing for the parametric case, there is a specified distribution in the presumptions, but we want to investigate the effect of deviating from it; in power testing at the non-parametric case, we stick to the presumptions of the test, but they do not specify a distribution.) For practical reasons, these non-specified distributions are usually characterized by their first four moments, so the first problem is to find a distribution family that can be parametrized to have arbitrary skewness and kurtosis. (Setting arbitrary mean and variance is trivial by linear scaling.) Common distributions do not have this property; in this thesis, we introduce – in detail – a well-known distribution family, the Fleishman distribution, that does have. In most of the cases, the analytical investigation with Fleishman distribution (or any other similar distribution) is infeasible or impossible. Thus, robustness and power are typically investigated empirically, using Monte Carlo simulation: many sample from the "non-specified" distribution (i.e. a distribution with pre-specified skewness and kurtosis) are generated, the test is applied to them, and results are recorded to determine robustness or power. With enough replications, this sufficiently converges to the true value of the investigated parameter. The method can be than re-iterated for different levels of skewness and kurtosis. This, however, has enormous computation requirement, which is infeasible even on modern personal computers. To alleviate this issue, we developed a program according to the novel principle of General Purpose GPU-computing, which harnesses the GPUs of modern video cards, as they can offer a performance that was previously only achievable with supercomputers for certain problems. More specifically: for problems that are computationally intensive (not I/O-intensive) and exhibit high data parallelism – just like Monte Carlo simulation for investigating statistical tests. We implemented a program, a simulational environment that is able to perform more than 60 million hypothesis testings per second (one-sample t-test, $n = 10$) even on a low-end video card. The environment is modular and highly flexible, so that it facilitates further extension. The program is introduced in detail in this thesis. Using this program, we systematically investigated many commonly used statistical test for robustness (parametric tests) and power (non-parametric tests). The results are presented and briefly discussed in this thesis. As a conclusion, we also briefly point out one limitation of the application of Fleishman distribution for this purpose to present a balanced discussion.

Acknowledgments

First of all, I would like to say thanks to Dr. *Balázs Kotosz*, my teacher at the Corvinus University of Budapest (CUB), with whom I started research about the robustness of statistical tests at the Department of Statistics of the CUB. He was the first to motivate me to begin investigating this fascinating topic of modern statistics. As far as CUB is concerned, I would also like to say special thanks to Prof. *László Hunyadi*, who was the reviewer of some of my early papers on this topic. He has really set a standard for me with his selfless, wholehearted support as evidenced by the amount of work he dedicated to provide helpful reviews for papers of students whom he has never even met before. I am also indebted to my present thesis advisors for their support in carrying out the research for this thesis. It was an honor working with Prof. *Jenő Reiczigel*, president of the Hungarian Clinical Biostatistical Society, who is one of the most well-known Hungarian biostatisticians. He always had another innovative idea, I have never even thought of, on how to improve my work. I am also deeply grateful to Dr. *Márton Balázs* from the Department of Stochastics at the Budapest University of Technology and Economics. Although he is primarily engaged in researching probability theory, for me, his delightful classes laid the solid foundations for statistics as well.

Contents

List of Figures

List of Tables

List of abbreviations

Abbreviation	**Meaning**
pdf	Probability density function
cdf	Cumulative distribution function
iid	Independent and identically distributed
TVD	Total variation distance
wlog	Without loss of generality
w.r.t.	With respect to
GPU	Graphical Processing Unit
GP-GPU	General Purpose GPU computing

Chapter 1

Introduction

The investigation discussed in this thesis was motivated by problems that are frequently encountered in biostatistics. Similarly to many other applied statistical fields, statistical tests are among the most basic tools that are used in biostatistics. However, especially in medical literature, they are handled somewhat carelessly: although most of these tests have presumptions, the validity of these are often not verified, despite the fact that small samples (when one can not rely on asymptotic properties) and unusual distributions are prevalent in biostatistics. This raises the question of robustness of statistical tests.

The investigation of robustness has many theoretical and practical problems – this was the direct motivation of this thesis. We have, however, widened our scope, and present results that can be applied to the investigation of more general properties of statistical tests. We will utilize an empirical approach, based on Monte Carlo simulation, which can not give analytical results, but – in exchange – can handle problems almost irrespectively of their complexity.

This Chapter is aimed to present a general overview of the preliminaries of the topic that is about to be covered in the present thesis. We will review the basic definitions from probability theory and statistics, outline the motivation for this research in more detail (including the need and impact from the statistical point of view) and outline the path to the deeper discussion of the particular questions.

First, we begin with a very compact summary of the most important definitions from probability theory in Section 1.1, just to ensure that a uniform terminology is used throughout this thesis.

Section 1.2 gives a broad overview on the theory of statistical hypothesis testing. We intentionally do not get into technical details, our aim is to provide a general picture of this crucially important area of modern statistics. After introducing the concept of statistical tests, we discuss the two most important properties of such tests in Section 1.3. These properties will be in the focus of our present work. To more specifically introduce the closer topic of this thesis, we first review (in detail) the possible ways to investigate these properties in Section 1.4.

The already existing literature on this topic is briefly review in Section 1.5. Section 1.6 concludes this chapter by describing the organization of this thesis.

1.1 Probability theory foundations

We will now briefly review a few fundamental concepts and definitions from probability theory [10]. Whenever these are referenced in this thesis, they are understood the way defined here.

We call a phenomenon $\mathcal{K}$ a *random experiment* if it has known possible outcomes, but there is no way to deterministically decide which will be the outcome when the experiment is performed.

These possible outcomes are called *elementary events* (usually denoted with ω); exactly one is realized when the random experiment is performed. The set of all elementary events is called *event space*, and is denoted with Ω. We define a σ-algebra on Ω, denoted with $\mathcal{F}$. Members of this set are called *events* (usually denoted $A, B, C, \ldots$), they are thus subsets of Ω.

We call a set function $\mathbb{P} : \mathcal{F} \to \mathbb{R}$ *probability*, if it satisfies the following properties:

1. Non-negativity: $\mathbb{P}(A) \geq 0$ for every $A \in \mathcal{F}$.

2. Normed: $\mathbb{P}(\Omega) = 1$.

3. σ-additive: if the – countably many – events $A_1, A_2, \ldots$ are mutually exclusive, i.e. $A_i \cap A_j = \emptyset$ for $i \neq j$ then
$$\mathbb{P}\left(\bigcup_i A_i\right) = \sum_i \mathbb{P}(A_i). \tag{1.1}$$

The triple $(\Omega, \mathcal{F}, \mathbb{P})$ is called *(Kolmogorov) probability space.*

We call a function $X : \Omega \to \mathbb{R}$ a *random variable* if it is measurable, i.e. if the preimage of every Borel-measurable set $\mathcal{B}$ of the real line satisfies the following: $X^{-1}(\mathcal{B}) \in \mathcal{F}$.

The function $F : \mathbb{R} \to [0, 1]$ with the definition[1] $F(x) := \mathbb{P}(X < x)$ is called the *cumulative distribution function* (or cdf, in short) of the random variable X.

If $\mathbb{P}$ is absolutely continuous with respect to (w.r.t.) the Lebesgue measure on the real line (such distributions are called simply continuous distribution) then $F(x)$ is also continuous, and has a derivate $\frac{\mathrm{d}F(x)}{\mathrm{d}x} =: f(x)$ or, to put it another way, there exists a function f such that $\int_{-\infty}^{a} f(x)\,\mathrm{d}x = F(a)$. This f function is called *probability density function* (or pdf in short).

In this thesis, we will confine ourselves to continuous distributions/variables. Consider a continuous random variable X and a – transformation – function $g : \mathbb{R} \to \mathbb{R}$ (possibly the identity function). The *expected value* of $g(X)$ is denoted with $\mathbb{E}\big(g(X)\big)$ and is defined as
$$\mathbb{E}\big(g(X)\big) := \int_{-\infty}^{\infty} g(x) f(x)\,\mathrm{d}x, \tag{1.2}$$
if this integral exists and is finite.

The so-called pth *moment* of a random variable X is denoted with μ_p, and is defined by
$$\mu_p := \mathbb{E}X^p = \int_{-\infty}^{\infty} x^p f(x)\,\mathrm{d}x, \tag{1.3}$$
if the integral exists and is finite.

The pth *central moment* of a random variable X is denoted with μ'_p, and is defined by
$$\mu'_p := \mathbb{E}\left[(X - \mu_1)^p\right] = \int_{-\infty}^{\infty} (x - \mu_1)^p f(x)\,\mathrm{d}x, \tag{1.4}$$
if the integral exists and is finite.

Specifically $\mu_1 = \mathbb{E}X$ is called the *expected value* of X, $\mu_2 - \mu_1^2 =: \mathbb{D}^2(X)$ is called the *variance* of X, $\frac{\mu'_3}{\mu_2^{3/2}}$ is called the *skewness* and $\frac{\mu'_4}{\mu_2^2}$ is called the *kurtosis* of X.

[1]Note that $\mathbb{P}(X \in A)$ is essentially just an abbreviated notation for $\mathbb{P}\Big(\{\omega : X(\omega) \in A\}\Big)$.

1.2 Statistical hypothesis testing

Inferential statistics, and *hypothesis testing* in particular, is one of the most important chapters of modern statistics. Tracing its root back to the end of the 19th and the beginning of the 20th century, it became widely used after the pioneering work of Ronald Fisher, Jerzy Neyman and Egon Pearson, among others [21]. The first applications included agrometrics, industrial quality control, followed by diverse areas of science and technology. It has given rise to an entirely new approach in biostatistics, and medical statistics, playing role in the introduction of the Evidence Based Medicine (EBM) principle [5].

Statistical tests are one of the core areas of inferential statistics. (More precisely its frequentist approach, which we will only consider in the present thesis.) First, we precisely define the concept of statistical test.

Consider a *statistical space* $(\Omega, \mathcal{F}, \mathcal{P})$ (that is: Ω is the sample space, $\mathcal{F}$ is a σ-algebra on Ω and $\mathcal{P}$ is a collection of probability measures such that its elements $\mathbb{P}$ all form a correctly defined probability space $(\Omega, \mathcal{F}, \mathbb{P})$). We will mostly consider dominated spaces, which have the property that there exists a σ-finite measure such that every $\mathbb{P} \in \mathcal{P}$ is absolutely continuous with respect to this measure. (In effect, this only states that the measures are either discrete, or continuous.) We say that the space is parametric, if $\mathcal{P}$ is parameterizable with the elements of a (not necessarily one dimensional) set, that is: $\mathcal{P} = \{\mathbb{P}_\theta : \theta \in \Theta\}$. For example, $\mathcal{P}$ might be the family of normal distributions, in this case $\Theta = \mathbb{R} \times \left[0, \infty\right)$, so every (two-dimensional) parameter $\theta = \left(\mu, \sigma^2\right) \in \Theta$ exactly defines one probability measure ($\mathcal{N}\left(\mu, \sigma^2\right)$ in this example).

Statistical hypothesis testing deals with the investigation of statements (called *hypotheses*) made on a population, of which we know only a part (called sample). To precisely define these terms: the set to which our question pertains is called the *population.* In this thesis, we will confine ourselves to univariate statistics, so the set can be considered as a set of random variables. It might be finite (when it can be represented with the realized values those random variables took, such as a set of real, but unknown numbers $(X_1, X_2, \ldots, X_N)$, where N is the size of the population) or infinite (when it can be represented with the common (background) distribution of the random variables). Either way, we presume that have been able to observe (for example measure) the values of only n elements from the population ($n < N$, if the size of the population is finite). This set $(X_1, X_2, \ldots, X_n) = \mathbf{X}$ is called the *sample.* (Mostly we presume independent sampling.) This is, of course, only true, if we are *before* the sampling, after it, we have an $(x_1, x_2, \ldots, x_n) = \mathbf{x}$ realization from the above random variables.

In parametric cases, the hypothesis can always be formulated as statements about where the θ parameter is in Θ. According to the tradition of frequentist hypothesis testing, these statements are always formulated in pairs, called *null hypothesis* (H_0) and *alternative hypothesis* (H_1) such that they are mutually exclusive statements on θ, but one of them is surely true:

$$\begin{aligned} H_0 &: \theta \in \Theta_0 \\ H_1 &: \theta \in \Theta_1, \end{aligned} \tag{1.5}$$

where $\Theta_0 \cap \Theta_1 = \emptyset$, but $\Theta_0 \cup \Theta_1 = \Theta$.

To mention a practical example, we can consider the population of healthy Hungarian males aged 20-25 (as a statistical population; this is strictly speaking a finite population, but can be considered as a population described by a background distribution for any practical purpose) where the body height is the parameter of interest. We want to answer whether it is 175 cm on average, precisely, our hypotheses are:

$$H_0 : \mu = 175 =: \mu_0 \tag{1.6}$$
$$H_1 : \mu \neq 175 = \mu_0,$$

where μ is the expected value of the background distribution. We have $n = 30$ males, who where randomly sampled from the above population. The question is: what can we say about this statement (that pertains to a population parameter!) if we only have the heights of the 30 males in the sample?

Statistical hypothesis testing aims to decide between these alternatives H_0 and H_1. (In non-parametric setting, the hypothesis can not be formed in such way (for example, they might pertain to the equality of two distributions, regardless of what their distribution is), but it does not essentially change our discussion.) Denoting the space of possible samples with $\mathcal{X}$ we can say that there is a partitioning $\mathcal{X} = \mathcal{X}_r \cup \mathcal{X}_a$ ($\mathcal{X}_r \cap \mathcal{X}_a = \emptyset$), such that for samples in $\mathcal{X}_r$ the decision is to reject the null hypothesis, i.e. accept the alternative hypothesis (rejection region), for samples in $\mathcal{X}_a$ (acceptance region), we accept the null hypothesis.

To perform this decision making, hypothesis testing utilizes a so-called *test statistic*, which is nothing else then a function (typically: real-valued function) on the sample, i.e. $T(\mathbf{X}) : \mathcal{X} \mapsto \mathbb{R}$. By using the test statistic we map the problem from $\mathcal{X}$ to $\mathbb{R}$, thus the rejection and acceptance regions are also mapped to $\mathbb{R}$. Typically, they are defined either by one (one-sided test) or two (two-sided test) cut points (partitioning $\mathbb{R}$ into an acceptance region and one or two rejection regions). These cut points are called *critical values*.

It can be immediately seen that it is not generally possible to create acceptance and rejection regions that universally guarantee correct decision about H_0. Consider our previous example: if the body height follows normal distribution in the population, then there is a non-zero chance that any sample value falls into any (true) interval on the real line, irrespectively of the expected value of the population distribution! (As the support of any normal distribution is $\mathbb{R}$.)

Hence, the best we can do is to find a T such that given H_0, the distribution of $T(X)$ (also called the *null distribution*) is known *a priori*, that is, independent of the unknown parameters[2] (using, perhaps, some presumptions on the distribution of the population) and to designate the rejection regions in such manner that there is only a small chance that the empirical value of the test statistic for a sample will fall into the region. For example, for a one-sided test, we might prescribe the rejection region as $[c_\alpha, \infty)$ (i.e. $\mathcal{X}_r = \left\{\mathbf{x} : T(\mathbf{x}) \geq c_\alpha\right\}$), where c_α is a constant depending on the parameter α (called significance level) which is simply the probability that a sample falls into the rejection region given H_0[3]. One can immediately see that this is the probability of erroneously rejecting (i.e. rejecting despite that fact that it stands) H_0 (also called *Type I Error*).

Returning to our example, we will demonstrate that

$$T(\mathbf{X}) := \frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu_0}{\sigma_0/\sqrt{n}} \tag{1.7}$$

is a proper test statistic for the hypothesis of (1.6) with iid. sampling under two assumptions:

1. The background distribution of the population is normal.

[2]Such statistic is also called a *pivot*.

[3]Precise definition would also take the fact into account that this probability might depend on the true (population) value of the parameter, and that it might not be possible in some case to construct such interval (that is, to find appropriate c_α for every α), but we only want give a broad overview of statistical hypothesis testing here, without going into technicalities.

2. The variance of this background distribution is a σ_0^2 constant that is *a priori* known.

The proof is simple: under these assumptions and H_0 $X_i \sim \mathcal{N}\left(\mu_0, \sigma_0^2\right)$, hence $\sum_{i=1}^n X_i \sim \mathcal{N}\left(n\mu_0, n\sigma_0^2\right)$ by the closedness of normal distribution under convolution, hence $\frac{1}{n}\sum_{i=1}^n X_i \sim \mathcal{N}\left(\mu_0, \sigma_0^2/n\right)$ and $\frac{1}{n}\sum_{i=1}^n X_i - \mu_0 \sim \mathcal{N}\left(0, \sigma_0^2/n\right)$ and $\frac{\frac{1}{n}\sum_{i=1}^n X_i - 175}{\sigma_0/\sqrt{n}} \sim \mathcal{N}(0,1)$ using elementary properties of expected value and variance [10], so if H_0 holds than $T(\mathbf{X}) \sim \mathcal{N}(0,1)$ – it in fact does not depend on the unknown parameter. (Its distribution, not its actual, realized value, of course.)

One can then define $c_\alpha^{(l)} := \Phi^{-1}\left(\alpha/2\right)$ and $c_\alpha^{(u)} := \Phi^{-1}\left(1-\alpha/2\right)$ lower and upper critical values and accept H_0 if $c_\alpha^{(l)} < T(\mathbf{X}) < c_\alpha^{(u)}$ (as this is a two-sided test: both high positive, and high negative values contradict the null hypothesis).

This is the so-called z-test (sometimes also referred to as the u-test).

1.3 Properties of statistical tests

The distribution of a test statistic is only known if H_0 stands *and* the presumptions of the test are met. This is often overlooked, and many student automatically presumes that H_0 can be rejected (with a given probability) if the test statistic falls into the rejection region, although strictly speaking this is only true if the presumptions are met, otherwise the distribution of the test statistic is not guaranteed to be the null distribution, even if H_0 stands. In other words, falling into the rejection region means that either H_0 is not true (with a given probability) *or* the presumptions are not met.

The two most important properties of statistical tests in which we will be interested pertain to these two events. Power (1.3.1) describes the behavior of the test if H_0 is not met, robustness (1.3.2) deals with the behavior of the test if the presumptions are not met.

1.3.1 Power of a statistical test

In this part, we will assume that the presumptions of the tests are met. Thus, if H_0 stands, the null distribution conveys every information on the behavior of the test statistic. However, the question immediately arises: what happens if H_0 is not true? This case is more complicated: the distribution of the test statistic will depend on *how* H_0 is not true. (In our example: if we assume that presumptions are still met, it means what is the true value of μ.)

Obviously, in this case we want to reject H_0. The error we can commit here is the erroneous acceptance of H_0, this is called *Type II Error*, whose probability is usually denoted with β. β is the probability that the test statistic falls into the acceptance region, given H_1. In contrast to Type I Error (with probability α) we do not have direct control on β: as outlined above, it also depends on a factor that is out of our control, namely, the true value of the investigated parameter.

Clearly α and β are not independent: if we decrease α that means we extend the acceptance region, which, in turn, automatically increases β. Hence, choosing α is a matter of trade-off between the two types of errors, and depends on how we weigh the severity of committing Type I Error relative to Type II Error.

Very often, one uses not β itself, but $1-\beta$, which is called the *power* of the statistical test. This is a natural measure of how well that test can detect a deviation from H_0 (as it measures the probability of rejecting H_0, given that it in fact does not hold).

In cases when the presumptions of the analyzed test – which are assumed to hold if we investigate power – do not include distributional presumption (i.e. what is the distribution family of the population from which the test's samples are coming), it is up to us to presume a distribution.

(These tests are usually called *non-parametric* tests, in contrast to parametric tests, where we presume a distribution family. There might be, of course, unknown parameters, but the family is presumed, i.e. we presume a function form for the distribution pdf/cdf, but perhaps with unknown parameters. For non-parametric tests, no function form is presumed at all.) In this situation, sometimes one wants to use a distribution that has known skewness and kurtosis. Although in different context, but the same question arises in Subsection 1.3.2, and will be addressed there.

1.3.2 Robustness of a statistical test

As already noted, we can only expect that a statistical test's distribution will be in fact the null distribution under H_0 if its assumptions are met, hence, one should always employ a test for a given problem, if its assumption are known to be met[4] (either from another test performed on *another* sample, or a priori).

In many practical situation, however, we can only have an approximate idea on whether the assumptions are met or not. One might – conservatively – chose to use a test with fewer assumptions, but this comes at the price of lower power, as we have already pointed it out. Hence, practical researchers often tend to still use test with more assumptions, hoping that it will not profoundly effect the null distribution (i.e. the Type I Error rate will be still close to the specified significance level). We will call this property the *robustness* of a statistical test. In our example, the question of what happens if the actual variance of the population is not σ_0 is a typical example of robustness.

A word of warning about the expression "robustness". We will use this term in the above sense, however, it should be noted that it is sometimes understood as the behavior of the test statistic if a small proportion of the sample elements takes vastly different value („gross error") [27]. As they are theoretically different concepts, there should be no confusion.

One of the most important topics within the question of robustness is the robustness with respect to non-normality. Many statistical tests, including the practically most important ones, such as Student's t-test assumes the normality of the population from which its sample(s) are coming. (Tests like this, that is, those that presume some distribution (perhaps up to one or more unknown parameters; i.e. which presume a distribution family), which is typically the normal family, are called *parametric* test.)

For sufficiently large samples, this can be neglected for tests that are based on the sum of the samples (such as t-test itself) due to the central limit theorem [10]. However, in certain areas of applied statistics, for example in biostatistics, one often finds *both* small samples *and* population distributions that are not certainly known to be normal. (This is so prevalent in medical context that one of the biostatistics textbooks has the subtitle "Statistics for Small Samples and Unusual Distributions" [47].) For this reason, the investigation of the robustness of statistical tests is especially important in biostatistics.

There is a problem that further complicates robustness investigations. For power testing, the alternative situation is usually well characterized: the true population distribution is still normal with known variance (as the presumptions are still assumed to stand) but with expected value of μ (in general, instead of 175). That is: the true population is exactly defined (perhaps with some parameter). However, in robustness testing, there is no specification for the true population distribution: when testing robustness with respect to normality, for example, the only prescription we have is that the population should not follow normal distribution.

This raises two important questions. First: how should we measure non-normality? There are

[4]But, it is also disadvantageous to employ tests that do not have assumption on what we know, as they tend to have less power.

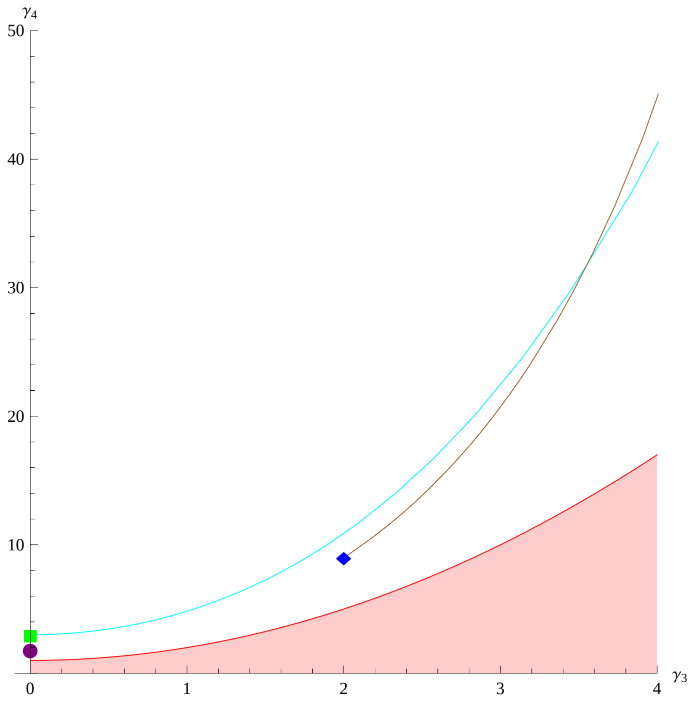


Figure 1.1: Coverage of important distributions on the skewness (γ_3)-kurtosis (γ_4) space: normal (green square), uniform (purple circle), exponential (blue diamond), lognormal (cyan line), Pareto (brown line). Shaded red area indicates the impossible region (where no distribution exists). See Section A.1.

many ways to define a metric for this (difference of certain moments from the same moment of normal, entropy, distance of the pdfs, different probability-distance measures etc., see [51]), none of which is trivially derivable from the other ones, and there is no basis for calling one of the "best". The second question: even if we agree on the metric of non-normality, there are typically infinite number of distributions for a given level of the metric. Which one should we choose for testing? (As for performing the robustness analysis, we – of course – need an exactly specified alternative.)

In this thesis, we will confine ourselves to using the *absolute moments metric* for the third and fourth moments, i.e. non-normality will be measured with the difference between the investigated distribution's skewness and kurtosis and the normal distribution's skewness and kurtosis. (The first and second moment is not interesting from this aspect, as every distribution with existing first and second moment can be transformed to have arbitrary first and second moment with trivial transformation (linear scaling).) The rationale for this will become more clear in 5.2.

After this agreement, we can turn to the second problem: there are infinite number of distributions for a given skewness-kurtosis level. Should we use this approach for robustness testing, we have to show a way to construct a distribution for a given skewness-kurtosis pair. This question is not trivial, as the distributions that are widely used in practice either have skewness-kurtosis that can not be effected through parameters (normal, uniform, exponential), or can be effected, but not independently (lognormal, Pareto), i.e. prescribing a skewness determines the kurtosis itself and vice versa [28]. This is illustrated on Figure 1.1 (see Section A.1), which shows the so-called *coverage* of several important distributions on the skewness-kurtosis space, that is, the skewness-kurtosis combinations the given distribution can take (by variation of the parameters). Distributions which fall to the first category occupy a single point, distributions from the latter list occupy a one-dimensional curve. Note that they are all of zero measure.

This Figure also illustrates another important concept: there is a region on the skewness-kurtosis space where no distribution can exist. This is due to theoretical grounds, as shown by the following theorem.

Theorem 1.3.1. *A distribution with every moment $\{\mu_i\}$ existing satisfies the inequality:*

$$\begin{vmatrix} 1 & \mu_1 & \mu_2 & \cdots & \mu_s \\ \mu_1 & \mu_2 & \mu_3 & \cdots & \mu_{s+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_s & \mu_{s+1} & \mu_{s+2} & \cdots & \mu_{2s} \end{vmatrix} \geq 0, \tag{1.8}$$

for every $s \geq 1$.

Proof. See [8] and [57]. □

Applying this for $s = 2$ (for a standardized distribution, which we might use wlog), we get that every distribution with existing first four moment has to satisfy the inequality

$$\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & \gamma_3 \\ 1 & \gamma_3 & \gamma_4 \end{vmatrix} = \gamma_4 - \gamma_3^2 \geq 0 \Rightarrow \gamma_4 \geq \gamma_3^2 + 1. \tag{1.9}$$

In other words no distribution exists, which has a kurtosis less then 1 plus its squared skewness. This is the "impossible region" shown in red on Figure 1.1.

Returning to our problem: we have to find a distribution which can be parametrized so that it can take arbitrary skewness and kurtosis. Such distribution can then be used for robustness testing (accepting the absolute moments metric as a measure for non-normality). It is also desirable that this distribution is algebraically simple, and covers the whole possible region.

One solution to this problem is the application of Fleishman distribution. This is introduced in Chapter 2.

1.4 Analytical and empirical investigation of statistical tests

The traditional way to investigate the above properties is to use an analytical approach: describe the alternative situation in exact algebraic terms (exactly specify distributions) and derive the behaviour of the statistical test.

For example, take our previous example of the z-test. Calculation of its power (with all presumptions still met) requires the analysis of the behavior of the random variable

$$\frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu_0}{\sigma_0/\sqrt{n}} \sim \mathcal{N}\left(\mu - \mu_0, 1\right), \tag{1.10}$$

where μ is the expected value of X_i (which is not μ_0 in the current setting).

Although this is clearly not following standard normal distribution if H_0 is not met, but its distribution is still known exactly for any μ, so it poses no problem to calculate the probability that it falls outside $\left[c_\alpha^{(l)}, c_\alpha^{(u)}\right]$. (Remember that power is the probability of rejection if H_0 in fact does not stand.) This function of power vs. μ (called power function) is shown on Figure 1.2 (see Section A.2).

However, there is another way to obtain this result: the power is a probability, which can be considered as the expected value of an indicator variable. (Indicating whether the test statistic falls into the rejection region under the H_1 that the true population expected value is $\mu \neq \mu_0$.) As

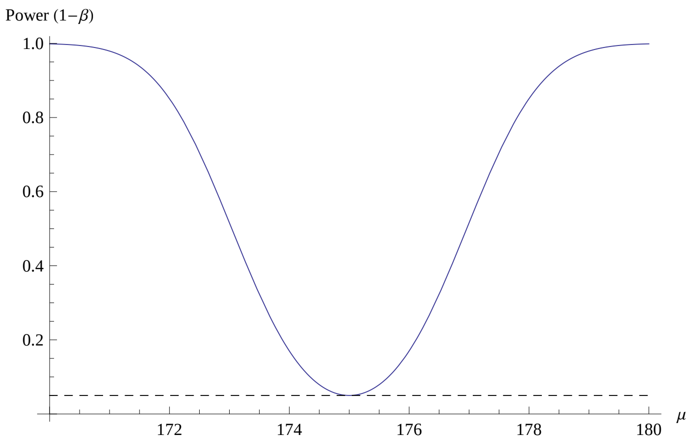


Figure 1.2: Power function of the z-test (with all presumptions met). See Section A.2.

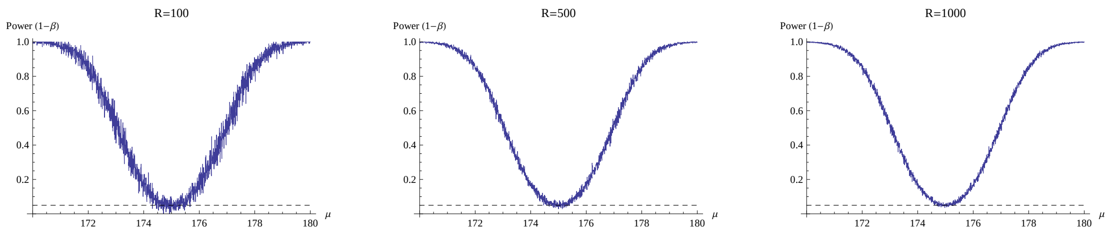


Figure 1.3: Power function of the z-test (with all presumptions met) approximated with Monte Carlo simulation with different number of replications. See Section A.2.

we know from the Law of Large Numbers [10], if we take iid samples from this indicator variable, their sum divided by the number of samples (which will be nothing else then the relative frequency of the event indicated by the variable) will converge to this probability, both stochastically and almost surely.

This points to the following procedure: generate iid random variables from the distribution specified in H_1 (in this case, $\mathcal{N}\left(\mu \neq \mu_0, \sigma_0^2\right)$) up to a specified sample size, perform the z-test on the sample, record its result, and repeat this procedure R times, where R is a sufficiently large number so that the fluctuations are acceptably small (due to the convergence discussed above). The relative frequency of rejections will approximate the power for the given μ. This procedure is called *Monte Carlo simulation* and can be considered as a way to *empirically* assess a statistical test.

Results of the Monte Carlo simulation for the power of the z-test (for different Rs) are shown on Figure 1.3 (see Section A.2).

At first glance, this does not seem to be an especially relevant addition, as the analytical method yielded an exact answer, faster, without any major theoretical complication. However, note that this simply depended on the fact that normal distribution is closed under convolution, so the sum of the samples was still normal (with easily computable parameters), even if H_0 was violated. There might be situations when this is not so.

Consider our second important property: robustness. Say, we want to test robustness with respect to the normality assumption (that is: we assume that every other presumption holds, the population's variance is still known, its expected value is still μ_0, but it is not normal). We can, for example, specify that the population follows Γ-distribution: $X_i \sim \Gamma\left(a, b\right)$. It is immediately not trivial, but one can conclude after some calculation and using induction that $\sum_{i=1}^{n} X_i \sim \Gamma\left(na, b\right)$.

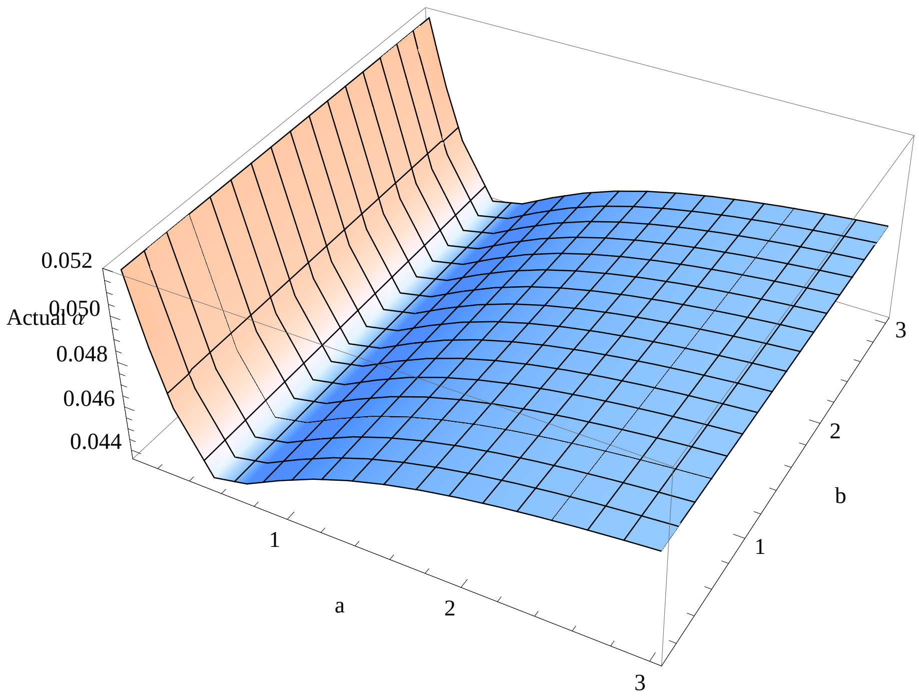


Figure 1.4: Robustness of the z-test with respect to non-normality: Γ-background distribution with parameters a and b. Vertical axis shows actual α. See Section A.3.

The distribution of the test statistic will be even more complicated:

$$\begin{aligned} T\left(\mathbf{X}\right) &= \frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu_0}{\sigma_0/\sqrt{n}} \sim \\ &\sim \Gamma\left(an, \frac{\Gamma(a)}{\sqrt{n}\sqrt{\Gamma(a)\Gamma(a+2)-\Gamma(a+1)^2}}, 1, -\frac{\sqrt{n}\Gamma(a+1)}{\sqrt{\Gamma(a)\Gamma(a+2)-\Gamma(a+1)^2}}\right), \end{aligned} \tag{1.11}$$

where $\Gamma\left(a,b,\gamma,\mu\right)$ represents the generalized Γ-distribution [28].

But we are still not finished: we have to integrate the pdf of this distribution outside $\left[c_\alpha^{(l)}, c_\alpha^{(u)}\right]$, which practically requires a computer algebra system.

After these difficulties, we obtain the behavior of the test under this H_1, i.e. its robustness. Figure 1.4 shows the actual α as a function of the background distribution's parameters a and b; Figure 1.5 shows this for $b = 2$ (See Section A.3.).

It can be clearly seen how cumbersome this procedure was. Let us now see how to solve the same problem with Monte Carlo simulation!

There is no need to calculate any distribution at all, not even the distribution of the sum of samples, and there is no need to calculate any integral. One just has to generate many samples from the specified Γ-distribution, perform the test on them, and count the number of replications where H_0 was rejected. That's it! Figure 1.6 shows the results for $b = 2$. (Note that we needed a much higher number of replications ($R = 100000$) compared to the previous example, as the function was more complicated, hence, more replication is needed to reduce the fluctuations.)

This example can be made even more dramatic by prescribing a background distribution of lognormal. This change does not effect Monte Carlo simulation *at all*: one just has to change the random number generator used, everything else remains unchanged. However, analytical tracking becomes not only much more complicated, but downright impossible: there is provably no closed-form of the pdf of the sum of lognormal densities [4].

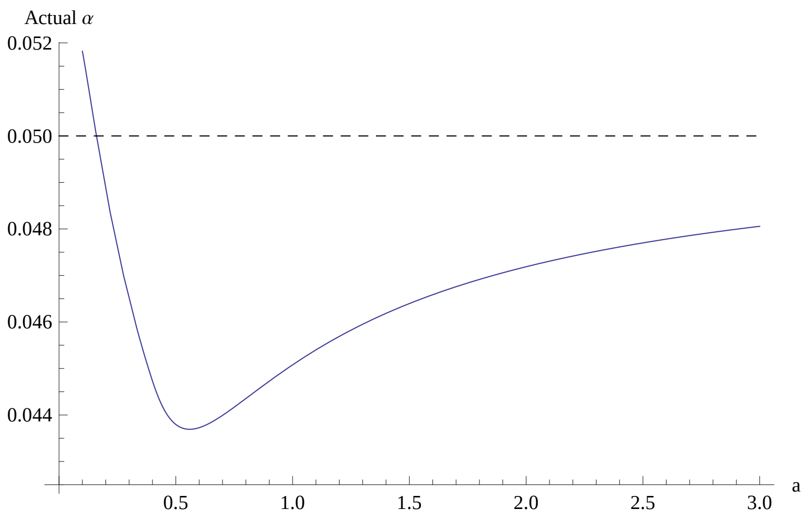


Figure 1.5: Robustness of the z-test with respect to non-normality: Γ-background distribution with parameters a and $b = 2$. Vertical axis shows actual α. See Section A.3.

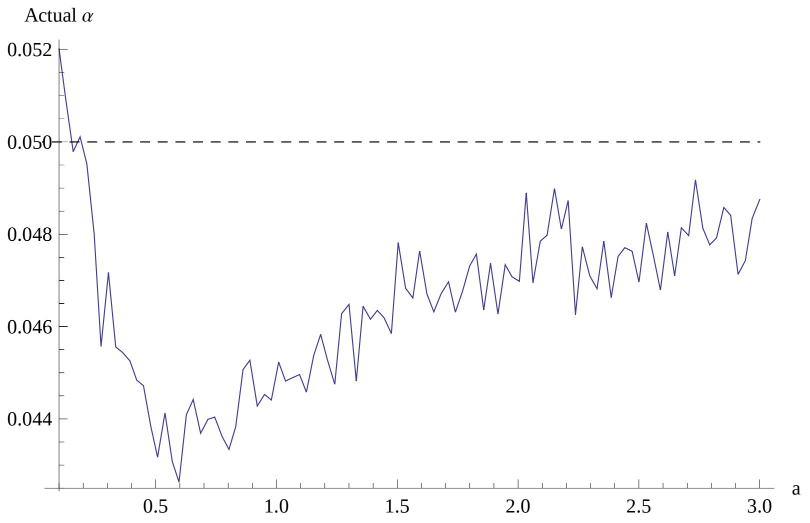


Figure 1.6: Robustness of the z-test with respect to non-normality approximated with Monte Carlo simulation ($R = 100000$): Γ-background distribution with parameters a and $b = 2$. Vertical axis shows actual α. See Section A.3.

This concept can be simply "visualized" as in Figure 1.7.

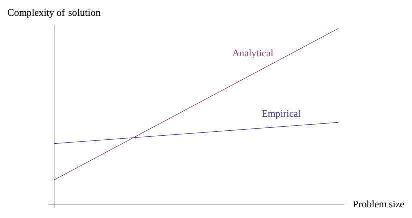


Figure 1.7: Schematic representation of the complexity of the analytical and the empirical approach as the problem size increases.

It worth noting that Monte Carlo simulation only requires that we are able to generate random numbers from a distribution, but no other information is needed about that distribution. For instance, we are able to use Monte Carlo simulation even if we have no information on the explicit pdf or cdf of the distribution (as far as we are able to generate random numbers from the distribution). This will have significance pertaining to Fleishman distribution, see Chapter 2.

The major drawback of the Monte Carlo simulation approach is that it can not provide analytical result, for example, a distribution is only reconstructible through realizations (although unlimited in number, which theoretically makes arbitrary precise reconstruction possible, but still, no analytical expression will be provided for its pdf or cdf).

To sum up, Monte Carlo simulation aims explore the behavior of a system (a statistical test in our case) in a stochastical way: through the outputs it provides as an answer to randomly given inputs [56]. This way, the complexity of the exploration does not depend on the complexity of the system, but – in exchange – only approximative, non-analytical description can be given about the system.

Monte Carlo simulation was first used in the 1940s for solving physical problems [40, 9], but traces its roots back to Buffon's famous needle dropping experiments. Physics is still amongst the major users of Monte Carlo simulation even today, but computational biology, computer graphics, and especially computational finance joined and make use of Monte Carlo simulation and Monte Carlo methods [32].

1.5 Overview of the literature

Every statistical test we will cover in this thesis has already been investigated earlier for properties, including robustness and power. As a matter of fact, there is so much literature on this question, that we can only aim to present the most important results for

The one-sample and the two-sample t-test are amongthe most widely investigated statistical tests, which includes robustness investigations (dating back to the late 1920s). Analytical handling was either done exactly for extremely small samples [54, 46, 33] or approximately through series expansions with Edgeworth- or Gram-Charlier series [3, 17, 18, 16], with Cornish-Fisher expansion [29] or with Laguerre expansion [62]. Other methods were also proposed [59].

The robustness of the ANOVA is a strongly connected question, which has also been investigated in detail, both analytically and – especially – empirically [19, 22, 36].

1.6 Organization of this thesis

The rest of this thesis is organized as follows.

As already noted, Chapter 2 introduces and discusses Fleishman distribution in detail, while Chapter 3 discusses the computational aspects of Monte Carlo simulations aimed at the empirical investigation of statistical tests.

Chapter 4 shows the results (both for power and robustness) achieved with Monte Carlo simulation employing Fleishman distribution for popular statistical tests. Chapter 5 outlines a criticism, and a possible limitation of the prior investigation. This limitation is explored in detail in Chapter 5. Finally, Chapter 6 concludes this paper with some closing remarks.

The Appendix lists every source code that was used to perform the calculations discussed in this thesis. Chapter A lists the Wolfram Mathematica codes, Chapter B contains the ”heart” of the program environment introduced in this thesis, the CUDA-codes of the Monte Carlo simulational program. Finally Chapter C shows the R-codes used for visualization.

Chapter 2

Fleishman distribution

In this Chapter, we introduce a distribution family aimed specifically at accommodating arbitrary, pre-specified moments through its parameters.

We will introduce this distribution, called Fleishman distribution in Section 2.1 and its generalized version in Section 2.2. Fleishman distributions are very simple technically, but still provide good coverage, as discussed in Section 2.3. Its advantages and disadvantages are discussed in detail in Section 2.4. Finally, in Section 2.5, we overview the open questions about Fleishman distribution and its possible extensions.

2.1 Origin and rationale of Fleishman distribution

Consider a standard normal variate $Z \sim \mathcal{N}(0,1)$. This random variable has no moment that can be set by parameters (obviously, as it has no parameters at all). Now consider the variable $a+Z$, where $a \in \mathbb{R}$ constant. This variable has a single moment (namely: the first) that can be set through parameters, a in this case. (As $a+Z \sim \mathcal{N}(a,1)$.) Now let us turn to the variable $a+bZ$, $a,b \in \mathbb{R}$ constants. This has two moments (namely: first and second) that can be adjusted through parameters a and b (as $a+bZ \sim \mathcal{N}\left(a,b^2\right)$).

What we have done in the first case can be considered to be a polynomial transformation of order zero (resulting in one adjustable moment), the second case is a polynomial transformation of order one (resulting in two adjustable moments). One might wonder if a polynomial power transformation of order two (i.e. $a+bZ+cZ^2$) results in a distribution with adjustable first three moments through parameters a, b and c. Or – and this is the more important question practically – does the distribution

$$a+bZ+cZ^2+dZ^3, \quad \text{where } Z \sim \mathcal{N}(0,1) \tag{2.1}$$

has adjustable first four moments? In line with Subsection 1.3.2, this is the critical question for us, because a positive answer would provide a distribution with arbitrary skewness and kurtosis that can be set through parameters.

It was Allan I. Fleishman who first proposed this back in 1978 [12]. Answering the question (i.e. whether the distribution indirectly defined by this transformation can have arbitrary first four moments) involves tedious linear algebra, but is straightforward.

First note that any distribution that has expected value and variance (always true for empirical distributions) can be transformed to have zero expected value and unit variance (by simple linear scaling) so we can confine ourselves to this case wlog. The expected value of the distribution defined by 2.1 is simply

$$\mathbb{E}\left(a+bZ+cZ^2+dZ^3\right)=a+b\mathbb{E}Z+c\mathbb{E}Z^2+d\mathbb{E}Z^3=a+c, \tag{2.2}$$

by the linearity of expected value and application of the following well-known lemma:

Lemma 2.1.1 (Moments of the standard normal distribution)**.** *If $Z\sim\mathcal{N}(0,1)$ is a standard normal variate, its moments are*

$$\mu_p=\mathbb{E}Z^p=\begin{cases}0 & \text{if } p \text{ is odd,}\\ (p-1)!! & \text{if } p \text{ is even}\end{cases} \tag{2.3}$$

where $n!!$ is the double factorial (i.e. $(p-1)!!=1\cdot3\cdot5\cdot\ldots\cdot(p-3)\cdot(p-1)$).

Proof. Standard normal distribution is symmetric, the power function with odd exponent is of course odd, thus, the exponentiated distribution will also be symmetric (and therefore having an expected value of zero), if the exponent is odd.

If the exponent is even, we can use Leibniz's rule (differentiation under the integral sign) by noting that $\frac{\partial^n e^{-\alpha\cdot x^2}}{\partial\alpha^n}=(-1)^n x^{2n}e^{-\alpha x^2}$ (both $e^{-\alpha\cdot x^2}$ and its derivative with respect to α is continuous), hence (set $p=2k$, as p is even) $x^{2k}e^{-x^2/2}=(-1)^k\cdot\frac{\partial^k e^{-\alpha\cdot x^2}}{\partial\alpha^k}$ for $\alpha=1/2$. Then:

$$\begin{aligned}\mu_{2k}&=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}x^{2k}e^{-x^2/2}\,\mathrm{d}x=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}(-1)^k\cdot\frac{\partial^k e^{-\alpha\cdot x^2}}{\partial\alpha^k}\,\mathrm{d}x\bigg|_{\alpha=1/2}=\\&=\frac{1}{\sqrt{2\pi}}\cdot(-1)^k\cdot\frac{\mathrm{d}^k\int_{-\infty}^{\infty}e^{-\alpha\cdot x^2}\,\mathrm{d}x}{\mathrm{d}\alpha^k}=\frac{1}{\sqrt{2\pi}}\cdot(-1)^k\cdot\frac{\mathrm{d}^k\frac{1}{\sqrt{\alpha}}\sqrt{\pi}}{\mathrm{d}\alpha^k}\bigg|_{\alpha=1/2}=\\&=\frac{1}{\sqrt{2\pi}}\cdot(-1)^k\cdot\sqrt{\pi}\cdot(-1)^k\cdot\frac{(2k-1)!!}{2^k}\cdot\alpha^{-(k+1/2)}\bigg|_{\alpha=1/2}=\\&=(2k-1)!!,\end{aligned} \tag{2.4}$$

where we used the well-known Gaussian integral [1]. □

Therefore, the condition for the first moment of the Fleishman distribution will be the following:

$$a+c=0, \tag{2.5}$$

given the already mentioned fact that we will set the first two moments zero and unit (wlog). In other words, we in effect have only three parameters, as $c=-a$ trivially.

Let us now move on to the second moment. Analytically, the second moment of the Fleishman distribution is

$$\begin{aligned}\mathbb{E}\left[\left(a+bZ+cZ^2+dZ^3\right)^2\right]&=a^2+2ab\mathbb{E}Z+2ac\mathbb{E}Z^2+2ad\mathbb{E}Z^3+b^2\mathbb{E}Z^2+2bc\mathbb{E}Z^3+2bd\mathbb{E}Z^4+\\&+c^2\mathbb{E}Z^4+2cd\mathbb{E}Z^5+d^2\mathbb{E}Z^6=a^2+2ac+b^2+6bd+3c^4+15d^2,\end{aligned} \tag{2.6}$$

using again Lemma 2.1.1 and the linearity of the expected value.

Therefore the second condition on the parameters of Fleishman distribution is

$$a^2+2ac+b^2+6bd+3c^4+15d^2=1. \tag{2.7}$$

Third and fourth moments (i.e. skewness and kurtosis itself, as we are working with standardized distributions) can be calculated in a similar way. Calculations are straightforward, but become rather cumbersome due to the long polynomial multiplications:

$$\begin{aligned}\mathbb{E}\left[\left(a+bZ+cZ^2+dZ^3\right)^3\right] &= a^3+3a^2b\mathbb{E}Z+3a^2c\mathbb{E}Z^2+3a^2d\mathbb{E}Z^3+3ab^2\mathbb{E}Z^2+6abc\mathbb{E}Z^3+\\ &+6abd\mathbb{E}Z^4+3ac^2\mathbb{E}Z^4+6acd\mathbb{E}Z^5+3ad^2\mathbb{E}Z^6+b^3\mathbb{E}Z^3+3b^2c\mathbb{E}Z^4+\\ &+3b^2d\mathbb{E}Z^5+3bc^2\mathbb{E}Z^5+6bcd\mathbb{E}Z^6+3bd^2\mathbb{E}Z^7+c^3\mathbb{E}Z^6+\\ &+3c^2d\mathbb{E}Z^7+3cd^2\mathbb{E}Z^8+d^3\mathbb{E}Z^9=\\ &=a^3+3a^2c+3ab^2+18abd+9ac^2+45ad^2+9b^2c+90bcd+15c^3+\\ &+315cd^2\end{aligned} \tag{2.8}$$

and

$$\begin{aligned}\mathbb{E}\left[\left(a+bZ+cZ^2+dZ^3\right)^4\right] &= a^4+4a^3b\mathbb{E}Z+4a^3c\mathbb{E}Z^2+4a^3d\mathbb{E}Z^3+6a^2b^2\mathbb{E}Z^2+12a^2bc\mathbb{E}Z^3+\\ &+12a^2bd\mathbb{E}Z^4+6a^2c^2\mathbb{E}Z^4+12a^2cd\mathbb{E}Z^5+6a^2d^2\mathbb{E}Z^6+4ab^3\mathbb{E}Z^3+\\ &+12ab^2c\mathbb{E}Z^4+12ab^2d\mathbb{E}Z^5+12abc^2\mathbb{E}Z^5+24abcd\mathbb{E}Z^6+\\ &+12abd^2\mathbb{E}Z^7+4ac^3\mathbb{E}Z^6+12ac^2d\mathbb{E}Z^7+12acd^2\mathbb{E}Z^8+4ad^3\mathbb{E}Z^9+\\ &+b^4\mathbb{E}Z^4+4b^3c\mathbb{E}Z^5+4b^3d\mathbb{E}Z^6+6b^2c^2\mathbb{E}Z^6+12b^2cd\mathbb{E}Z^7+\\ &+6b^2d^2\mathbb{E}Z^8+4bc^3\mathbb{E}Z^7+12bc^2d\mathbb{E}Z^8+12bcd^2\mathbb{E}Z^9+4bd^3\mathbb{E}Z^{10}+\\ &+c^4\mathbb{E}Z^8+4c^3d\mathbb{E}Z^9+6c^2d^2\mathbb{E}Z^{10}+4cd^3\mathbb{E}Z^{11}+d^4\mathbb{E}Z^{12}=\\ &=a^4+4a^3c+6a^2b^2+36a^2bd+18a^2c^2+90a^2d^2+36ab^2c+\\ &+360abcd+60ac^3+1260acd^2+3b^4+60b^3d+90b^2c^2+630b^2d^2+\\ &+1260bc^2d+3780bd^3+105c^4+5670c^2d^2+10395d^4,\end{aligned} \tag{2.9}$$

using again Lemma 2.1.1 and the linearity of the expected value.

Therefore the third condition is

$$a^3+3a^2c+3ab^2+18abd+9ac^2+45ad^2+9b^2c+90bcd+15c^3+315cd^2=\gamma_3, \tag{2.10}$$

and the fourth condition is

$$\begin{aligned}a^4+4a^3c+6a^2b^2+36a^2bd+18a^2c^2+90a^2d^2+36ab^2c+360abcd+60ac^3+1260acd^2+3b^4+60b^3d+\\ 90b^2c^2+630b^2d^2+1260bc^2d+3780bd^3+105c^4+5670c^2d^2+10395d^4=\gamma_4,\end{aligned} \tag{2.11}$$

where γ_3 and γ_4 are the third and fourth standardized central moments, respectively (i.e. skewness and kurtosis; the same as third and fourth moment in this case).

Combining the above, we obtain the following theorem:

Theorem 2.1.2 (Fitting Fleishman distribution with four moments)**.** *Let X denote a Fleishman distribution for the first four moments (i.e. $X = a + bZ + cZ^2 + dZ^3$, where $Z \sim \mathcal{N}(0, 1)$). Such distribution exists for standardized first four moments* $(0, 1, \gamma_3, \gamma_4)$ *if and only if the following* *system of equations has a solution for a, b, c and d:*

$$0 = a + c \tag{2.12}$$
$$1 = a^2 + 2ac + b^2 + 6bd + 3c^4 + 15d^2 \tag{2.13}$$
$$\gamma_3 = a^3 + 3a^2c + 3ab^2 + 18abd + 9ac^2 + 45ad^2 + 9b^2c + 90bcd + 15c^3 + 315cd^2 \tag{2.14}$$
$$\begin{aligned}\gamma_4 = {} & a^4 + 4a^3c + 6a^2b^2 + 36a^2bd + 18a^2c^2 + 90a^2d^2 + 36ab^2c + 360abcd + 60ac^3 + 1260acd^2 + \\ & + 3b^4 + 60b^3d + 90b^2c^2 + 630b^2d^2 + 1260bc^2d + 3780bd^3 + 105c^4 + 5670c^2d^2 + 10395d^4\end{aligned} \tag{2.15}$$

Proof. See above. □

2.2 Generalizing the Fleishman distribution

While having slight practical significance, it worth noting that the above logic can be extended to moments beyond the first four one. For that end, first note that we need a transforming polynomial of order $n-1$ (with n parameters) to have chance to fit the distribution to n moments (i.e. solve n equations). That is, the form of the (generalized) Fleishman-distribution for first n moments is

$$X = a_0 + a_1Z + a_2Z^2 + \ldots + a_{n-1}Z^{n-1} = \sum_{i=0}^{n-1} a_iZ^i =: p_{a_0,a_1,\ldots,a_{n-1}}(Z) \quad \text{where } Z \sim \mathcal{N}(0,1). \tag{2.16}$$

We will denote this distribution with $\mathcal{FD}(n)$. (That is: the "traditional" Fleishman distribution is $\mathcal{FD}(4)$.)

Following the logic of Section 2.1, we first determine the p-th moment of the (generalized) Fleishman distribution $\mathcal{FD}(n)$. We will need the multinomial theorem:

Lemma 2.2.1 (Moments of the (generalized) Fleishman distribution)**.** *Let $X \sim \mathcal{FD}(n)$ (i.e. $X = \sum_{i=0}^{n-1} a_iZ^i$ where $Z \sim \mathcal{N}(0,1)$), then its p-th moment is*

$$\mu_p = \mathbb{E}X^p = \sum_{\substack{k_0+k_1+\ldots+k_{n-1}=p \\ \sum_{i=0}^{n-1} i\cdot k_i \mod 2=0}} \frac{p!}{k_0!k_1!\cdots k_{n-1}!} \cdot \left(\prod_{i=0}^{n-1} a_i^{k_i}\right) \cdot \left(\sum_{i=0}^{n-1} i\cdot k_i - 1\right)!!. \tag{2.17}$$

Proof. Use the multinomial theorem:

$$\begin{aligned}\mu_p = \mathbb{E}X^p = \mathbb{E}\left[\left(\sum_{i=0}^{n-1} a_iZ^i\right)^p\right] &= \mathbb{E}\left(\sum_{k_0+k_1+\ldots+k_{n-1}=p} \frac{p!}{k_0!k_1!\cdots k_{n-1}!} \prod_{i=0}^{n-1} \left(a_iZ^i\right)^{k_i}\right) = \\ &= \sum_{k_0+k_1+\ldots+k_{n-1}=p} \frac{p!}{k_0!k_1!\cdots k_{n-1}!} \cdot \left(\prod_{i=0}^{n-1} a_i^{k_i}\right) \cdot \mathbb{E}\left(\prod_{i=0}^{n-1}\left(Z^i\right)^{k_i}\right) = \\ &= \sum_{k_0+k_1+\ldots+k_{n-1}=p} \frac{p!}{k_0!k_1!\cdots k_{n-1}!} \cdot \left(\prod_{i=0}^{n-1} a_i^{k_i}\right) \cdot \mathbb{E}\left(Z^{\sum_{i=0}^{n-1} i\cdot k_i}\right) = \\ &= \sum_{\substack{k_0+k_1+\ldots+k_{n-1}=p \\ \sum_{i=0}^{n-1} i\cdot k_i \mod 2=0}} \frac{p!}{k_0!k_1!\cdots k_{n-1}!} \cdot \left(\prod_{i=0}^{n-1} a_i^{k_i}\right) \cdot \left(\sum_{i=0}^{n-1} i\cdot k_i - 1\right)!!.\end{aligned} \tag{2.18}$$

□

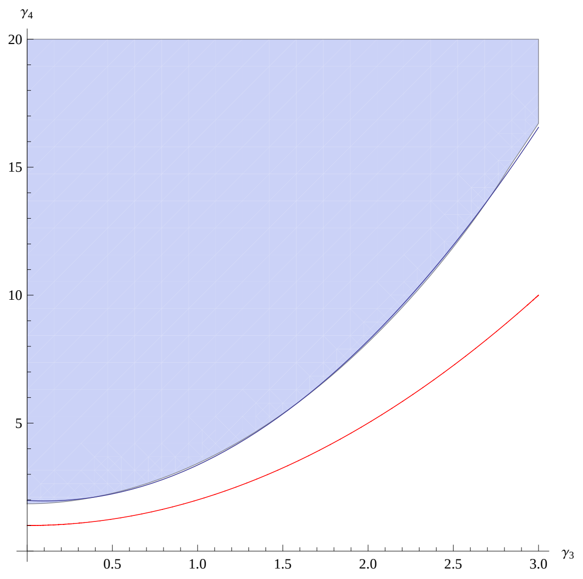


Figure 2.1: Region of existence of $\mathcal{FD}\,(4)$ Fleishman distribution (light blue shading). Border for the theoretically possible region is drawn in red; dark blue line is the quadratic approximation of Equation (2.20) for the lower bound of the region of coverage.

We can now conclude the same way as in the special case:

Theorem 2.2.2 (Fitting Fleishman distribution with n moments)**.** *Let X denote a Fleishman distribution for the first n moments (i.e. $X \sim \mathcal{FD}\,(n)$). Such distribution exists for standardized first n moments $\left(\gamma_j\right)_{j=1}^{n}$ if and only if the following system of equations has a solution for $\left(a_j\right)_{j=0}^{n-1}$:*

$$\left\{ \gamma_p = \sum_{\substack{k_0+k_1+\ldots+k_{n-1}=p \\ \sum_{i=0}^{n-1} i\cdot k_i \mod 2=0}} \frac{p!}{k_0!k_1!\cdots k_{n-1}!} \cdot \left(\prod_{j=0}^{n-1} a_i^{k_i} \right) \cdot \left(\sum_{i=0}^{n-1} i \cdot k_i - 1 \right)!! \right\}_{p=1}^{n} \tag{2.19}$$

Proof. See above. □

2.3 Solving the $\mathcal{FD}\,(4)$

Now we will return to the special (and practically important) case of using the first four moments to see for which moments is it possible to construct a Fleishman distribution.

Theorem 2.1.2 gave a necessary and sufficient condition for this. It, however, builds on the solvability of a non-linear system of equations, which has no general, closed-form solution. Therefore, we have to use numerical methods, such as Newton–Raphson method [31] or similar root-finding or optimizational method [30] to solve the equation for a given set of moments.

Thus, it is not possible to give an analytical condition for the existence of Fleishman distribution. Instead of that, we can iterate through the skewness-kurtosis space, and try to numerically solve the necessary system of equations at every point. Those points where solution exists compose the region where Fleishman distribution exists. Appendix A.4 shows an example code in Wolfram Mathematica to numerically determine this region, Figure 2.1. shows the results graphically.

One can immediately see that Fleishman distribution covers most of the theoretically possible region. The exception is a small "gap" where kurtosis is very close to the minimal possible given the skewness.

The lower limit of the coverage area of Fleishman distribution seems to be quadratic. In fact, it can be shown by numerical methods (see Appendix A.4) that the coverage region can be almost perfectly approximated as:

$$\gamma_4 > 1.738\gamma_3^2 - 0.3544\gamma_3 + 1.978. \tag{2.20}$$

This curve is also indicated on Figure A.4; one can observe the very close fit with the actual lower bound of the region of coverage. (Note that this is not in agreement with Fleishman's original publication [12]. Numerical error in that article has been already recognized.)

2.4 Advantages and disadvantages of using $\mathcal{FD}\,(4)$

Based on the above findings, it is now possible to summarize what advantages (2.4.1) and disadvantages (2.4.2) does the Fleishman distribution have for the task outlined in Subsection 1.3.2.

2.4.1 Advantages of using $\mathcal{FD}\,(4)$

The following list is a summary of the advantages of using $\mathcal{FD}\,(4)$ distribution.

- The primary advantage of $\mathcal{FD}\,(4)$ shows up if we only want to generate random numbers from this distribution. (Just as in our case with Monte Carlo simulation, see Section 1.4.) This is very easy, one only needs to calculate the coefficients, generate large number of standard normal deviates, and perform the prescribed transformation. All three steps is computationally simple, because:
  - The only random number generator needed by the simulation is one that can produce standard normal variate. This is a basic, and thus very deeply investigated problem in random number generation, so highly efficient solutions (such as the Box-Müller method, and other approaches [8]) are available.
  - If the coefficients and the random variate is given, only very simple algebraic transformations are needed to obtain the transformed variate. Even explicit exponentiation can be spared, consider the rewriting:

$$a + bZ + cZ^2 + dZ^3 = a + Z\left(b + Z\left(c + Zd\right)\right) \tag{2.21}$$

  - Finally, the coefficients a_0, a_1, a_2 and a_3 can be calculated off-line, i.e. they are the same for a given skewness-kurtosis, so they have to be calculated only once for a given skewness-kurtosis pair (this can be done prior to beginning the simulation), regardless of how many variables we generate.
- The covarage is excellent, as demonstrated in Section 2.3.

2.4.2 Disadvantages of using $\mathcal{FD}\,(4)$

The following list is a summary of the disadvantages of using $\mathcal{FD}\,(4)$ distribution.

- The coverage is, although excellent, not perfect.
- The indirect definition of Fleishman distribution makes random number generation very simple, but obtaining an explicit pdf for $\mathcal{FD}\,(4)$ very complicated in exchange. In fact, this was one of the earliest criticism against Fleishman distribution [61]. Indeed, no formula was

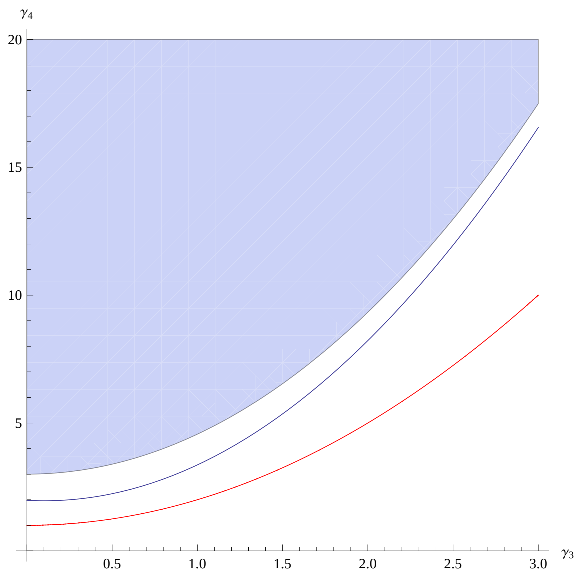


Figure 2.2: Region of existence of $\mathcal{FD}\,(4)$ Fleishman distribution restricted to transforming polynomials that are strictly monotonically increasing (light blue shading). Border for the theoretically possible region is drawn in red; dark blue line is the quadratic approximation of Equation (2.20) for the lower bound of the region of coverage for the entire $\mathcal{FD}\,(4)$ family.

given for the pdf of $\mathcal{FD}\,(4)$ until 2007 (almost three decades after the publication of the distribution). See the more detailed discussion in Subsection 2.5.1.

2.5 Further possibilities with the Fleishman distribution

In this section we investigate several aspects of Fleishman distributions that represent improvement in some respect compared to the "basic" variant introduced above. In particular, we will deal with the question of explicit cdf/pdf of Fleishman distribution (2.5.1), the usage of $\mathcal{FD}\,(n)$ for $n > 4$ (2.5.2), the usage of initial distributions other than standard normal (2.5.3), and the question of multivariate distributions (2.5.4).

State-of-the-art in these question is well summarized in [25].

2.5.1 Probability density function of Fleishman distribution

The first problem is the lack of explicit pdf: the definition $a + bZ + cZ^2 + dZ^3$ does not provide any information on the pdf of the resulting distribution. There are several ways to attack this problem.

Traditional approach

The use of traditional methods for random variable transformation leads to extremely complicated formulae. To alleviate this issue, it is worth confining ourselves to coefficients where the transforming polynomial is strictly monotonically increasing. (Figure 2.2 shows the coverage of these distributions together with the lower bound for the original $\mathcal{FD}\,(4)$ of Equation (2.20). It can be seen that there is only a minimal loss of generality by this restriction.)

The condition for this is simple: we need $p'_{a,b,c,d}\,(x) = b + 2cx + 3dx^2 > 0$ for all x. By using the quadratic formula, we obtain two conditions for this: $d > 0$ and $4c^2 - 12bd < 0$. The former is not restrictive, as the signs of b and d can be simultaniously changed without affecting the validity of Equations (2.12)-(2.15). (Observe that the sum of the exponents of b and d is even for every term.) The latter is restrictive, but its effect is minimal, as we have already discussed it.

Although this restriction is useful because such polynomials have exactly one real root (making the transformation easier, as we will see) the result is still too complicated for any practical application:

Proposition 2.5.1 (Explicit pdf of $\mathcal{FD}(4)$ with strictly monotonically increasing polynomial)**.** *The probability density function of an $\mathcal{FD}(4)$ defined by a strictly monotonically increasing polynomial is*

$$f(x) = \frac{3d\left(6bd - 2c^2 + \sqrt[3]{2}\Delta(y)^{2/3}\right)\exp\left(-\frac{\left(\frac{2\sqrt[3]{2}\left(c^2-3bd\right)}{\sqrt[3]{\Delta(y)}}-2c+2^{2/3}\sqrt[3]{\Delta(y)}\right)^2}{72d^2}\right)}{2\sqrt[6]{2}\sqrt{\pi}\sqrt[3]{\Delta(y)}\sqrt{\left(27d^2(a-y)-9bcd+2c^3\right)^2-4\left(c^2-3bd\right)^3}}, \tag{2.22}$$

where $\Delta(y) := \sqrt{\left(27d^2(a-y)-9bcd+2c^3\right)^2-4\left(c^2-3bd\right)^3}+27d^2(y-a)+9bcd-2c^3$.

Proof. First, we obtain the cdf of the transformed distribution:

$$F(x) = \mathbb{P}(X \le x) = \mathbb{P}\left(p(Z) \le x\right) = \mathbb{P}\left(Z \le p^{-1}(x)\right). \tag{2.23}$$

This is the point were we made use of the fact that $p(x)$ is strictly monotonically increasing – this way, there is a unique inverse in the above equation. (Otherwise, we might have three real roots, and the above probability would have been the standard normal's cdf at the first one, and the difference of its cdf between the second and the third one. Of course, and additional case separation would be also needed, making the result even more obfuscated.)

After we calculate the inverse (which will be quite complicated itself, as it requires solving a cubic equation), and obtain the above cdf by substitution (practically: with the application of a computer algebra system), we can simply derive the pdf (as the distributions are continuous) as:

$$f(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}. \tag{2.24}$$

Substituting $p(x) = a + bx + cx^2 + dx^3$ (with the restrictions $d > 0$ and $4c^2 - 12bd < 0$, see above), we get the formula in the Proposal. □

The above reasoning also shows that we can not expect algebraic solution for the general case of $\mathcal{FD}(n)$ when $n \ge 6$ (as per the Abel–Ruffini theorem).

Headrick's approach

Another approach to obtain a pdf is the one of Headrick et al, who claimed to be the first to derive a pdf for $\mathcal{FD}(n)$ in 2007 [26]. What they actually did was, however, not the derivation of the explicit pdf (logically, as this would not lead to any meaningful result, as shown above), but rather the conversion to an alternative (and more useful) representation of the pdf.

To understand their transformation, first note that any $\mathbb{R} \to \mathbb{R}$ function can be represented as a $\mathbb{R} \to \mathbb{R}^2$ parametric curve. For example, a pdf $f(x)$ is a mapping $x \mapsto f(x)$, but it can be also seen as a parametric curve on $\mathbb{R}^2$, i.e. a $t \mapsto \mathbb{R}^2$ in the form $\left(x(t), y(t)\right)$ (where $x(t) = t$ and $y(t) = f(t)$). We can, for example, say, that the pdf of the standard normal distribution is $x(t) = t$ and $y(t) = \frac{1}{\sqrt{2\pi}}e^{-t^2/2}$. Of course, the same can be said about the representation of a cdf.

At first glance, this transformation is nothing more than some voodoo magic aimed to represent something very simple in a very complicated form. This is, however, not the case. The primary significance of this form, which will make many calculations easier, is shown by the following Lemma. We emphasize that these can also be only applied if p is a strictly monotonically increasing transformation.

Lemma 2.5.2 (Transformation of a cdf given as parametric curve)**.** *Consider the cumulative distribution function F of a random variable X, represented as parametric curve, i.e. in a form $\big(x(t), y(t)\big) = \big(t, F(t)\big)$. Then the F' cdf of the transformed variable $p(X)$ can be represented as a parametric curve as $\big(p(t), F(t)\big)$ if the transformation is strictly monotonically increasing.*

Proof. We use the fact that p is strictly monotonically increasing – this means that inequality $a < b$ holds exactly if $p(a) < p(b)$. Therefore:

$$F(t) = \mathbb{P}(X < t) = \mathbb{P}\big(p(X) < p(t)\big) = F'\big(p(t)\big). \tag{2.25}$$

The cdf of the transformed variable can be represented as a parametric curve – in general form – as $F' : \big(t, F'(t)\big)$. This stands for every t, so specifically for $p(t)$ as well, that is, $F' : \Big(p(t), F'\big(p(t)\big)\Big)$ also holds. Now, using the equality in (2.25), we get that this latter is the same as $F' : \big(p(t), F(t)\big)$, which is just the statement in the Lemma. $\square$

The significance of this form is that it only requires the original cdf, and the transformation, but not the transformation's inverse! So, this approach can be applied even if the inverse is complicated, or does not even exist in closed form.

To show that this transformation is in fact useful for our problem as well, consider the following simple question: what is the median of a given $\mathcal{FD}(4)$ distribution? Without cdf, this can not be answered. But the above parametrisation shows a way to solve this (without producing an explicit cdf): the median is the point where $F'(t) = 1/2$. Considering the Lemma, this is characterized by $F(t) = 1/2$. But here, F is the cdf of the standard normal distribution, hence $t = 0$ so that the previous equation holds. Therefore the "x-coordinate" of the median (i.e. the median itself) will be at $p(0)$ (at a for $\mathcal{FD}(n)$), using again the above Lemma.

The same way, we can determine any quantile of the transformed distribution. (And other, practically relevant statistics as well, like trimmed mean or even mode.)

Of course, there is no way to obtain a simpler form for the pdf/cdf as shown above; even this approach is naturally bound to run into the same complicated formula, if we want to get an explicit form. To see this, consider how we would write an explicit form from the parametric one: we would express the parameter from the x coordinate, and substitute it to the y coordinate. However, expressing the parameter from the x coordinate would result in $p^{-1}(t)$, which would lead us back right to Lemma 2.5.1.

The Lemma can be formulated for pdf as well:

Lemma 2.5.3 (Transformation of a pdf given as parametric curve)**.** *Consider the probability density function f of a random variable X, represented as parametric curve, i.e. in a form $\big(x(t), y(t)\big) = \big(t, f(t)\big)$. Then the pdf of the transformed variable $p(X)$ will be $\Big(p(t), \frac{f(t)}{p'(t)}\Big)$ if the transformation is strictly monotonically increasing.*

Proof. One advantage of the parametric curve is the we can calculate the derivative coordinate-wise. Using Lemma 2.5.2 we get that the tangent vector of the cdf is

$$\big(p'(t), f(t)\big). \tag{2.26}$$

This pertains to parameter value t, that is, to the point with x coordinate $p(t)$ (see Lemma 2.5.2). The derivative at this point can be calculated by normalizing the above tangent, hence we obtain

$$\left(p(t), \frac{f(t)}{p'(t)}\right), \tag{2.27}$$

just as we stated. □

This Lemma can be used, for example, to plot the pdf of an $\mathcal{FD}(4)$ distribution – this again shows the strength of this approach, as it will be an exact plotting, despite the fact that we do not know the explicit pdf itself.

These results can also be used to derive an analytical formula for the lower boundary of the skewness-kurtosis coverage of $\mathcal{FD}(4)$ [25].

To sum up, this approach can not provide simpler explicit form for the pdf or cdf of $\mathcal{FD}(n)$ (of course), but can be used to exactly perform the plotting of the function's graph, or to exactly answer important questions (such the quantiles of the distribution). Finally, we again stress that this approach is also limited to distributions that are defined by a strictly monotonically increasing polynomial transformation (although, as we have seen, it does not present a major restriction).

2.5.2 Using $\mathcal{FD}(n)$ for $n > 4$

It is quite logical idea to improve the fit of a Fleishman distribution for a given data by using more than four moments. (That is, to enforce the equality of more than just the first four moments.) As the first four moments are still matched, one can expect at least as good fit for the data (and perhaps even better because of the utilization of higher moments). Practically, matching based on the first six moments ($\mathcal{FD}(6)$) is used, see [23] for results on using such distributions.

As for the Monte Carlo studies, this approach can be used to increase the coverage and to workaround the restriction of strictly monotonically increasing transformational polynomial (should one investigate the cdf/pdf). The reason is simple: restrictions will apply in the six dimensional space of the first six moments (when using $\mathcal{FD}(6)$), but when they are projected to the first four dimension, the covered area will be larger than the one obtained with $\mathcal{FD}(4)$ [25].

2.5.3 Using other initial distributions

Yet another idea to improve Fleishman distribution is the usage of initial distributions (i.e. what is transformed with the polynomial) other then standard normal. One obvious effect of this change will be that the support of the resulting distribution drastically changes: if the original distribution is bounded, the resulting Fleishman distribution will also be bounded.

Apart from that, we can also expect that the region of coverage (with respect to skewness-kurtosis) depends on the initial distribution. This is discussed in [25]; our result are shown on Figure 2.3. Distributions include standard normal, continuous uniform (concentrated to the interval $[0, 1]$) and standard logistic [28].

2.5.4 Multivariate distributions

Finally, one might be interested if Fleishman distributions can be extended multivariately. The question that obviously arises at this point is the prescribing of a correlation structure between the components.

The answer is positive, one can define such distributions, but this is outside the scope of the present thesis, we refer the reader to [63] and [24].

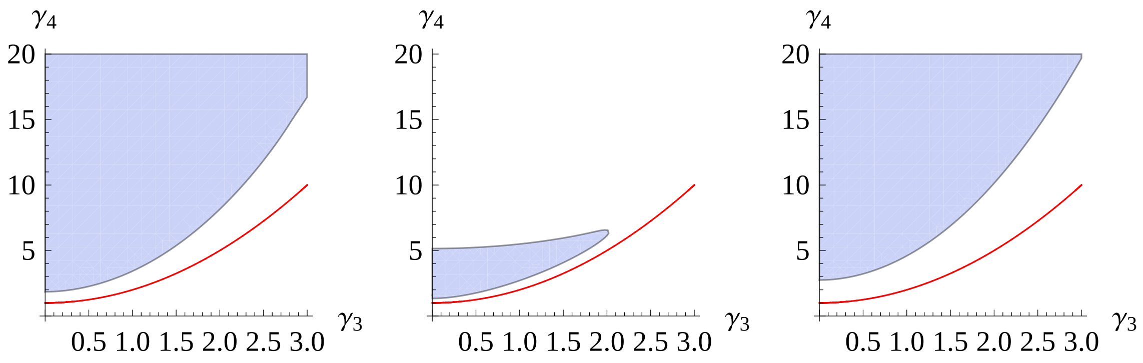


Figure 2.3: Coverage of Fleishman distribution with different initial distributions: standard normal (left), uniform (center), standard logistic (right).

Chapter 3

Computational aspects of Monte Carlo simulation

We developed a computer program, a Monte Carlo simulational environment that can be used for the empirical investigation of statistical tests. It is basically aimed to perform two kind of analyses: robustness testing with respect to distributional assumption (for parametric tests) and power testing for tests that do not have distributional assumption (non-parametric tests). These problems are practically interesting, because they require the application of a distribution with pre-specified moments, which can not be trivially solved. (We will now measure the difference of distributions by the difference of their moments, see Section 1.3. We will be mostly interested in skewness and kurtosis.) Also, these are the problems that are typically hard or impossible to handle in practice with analytical methods.

Our environment is highly flexible: it is capable to investigate a variety of tests under a variety of assumptions (all of which can be easily set by the user), and can be used to solve diverse tasks. Also, we paid pronounced attention to the modularity of the environment: it can be extended to include new tests or parameter with ease.

First, in Section 3.1 we give a broad overview on the logic of the simulational environment. It will become clear from this Section that the program requires trendemous amounts of calculations which can not be feasibly done on CPU. In Section 3.2, we present an alternative method, the so-called General Purpose GPU computing, which can be applied to speed up our calculations in an affordable way. The concrete implementation is described in Section 3.3. Finally, in Section 3.4, we briefly discuss the visualization of the results (as it was done under a separate program package).

3.1 Overview of the developed Monte Carlo simulational environment

In this thesis we present a simulational environment to empirically investigate properties (power and robustness) of statistical tests. As already discussed in Chapter 1, this environment will use Monte Carlo simulation, hence it is almost completely insensitive to the complexity of the statistical test under investigation. As introduced in Chapter 2, we will use Fleishman distribution to generate random numbers from distributions with – almost – arbitrary skewness and kurtosis as required by investigating robustness with respect to a distributional assumption (for parametric tests), or power simulation when there is no distributional assumption (non-parametric tests).

The basic mechanism of our simulational environment is shown on Figure 3.1. It is applicable for both purposes (the only difference is in the actual test employed).

The program performs investigation by scanning the skewness-kurtosis space in a grid search manner: it places an equidistantly spaced, rectangular grid on this space, and performs the analysis in every grid point. There is, however, one additional consideration here: the coverage of Fleishman distribution (but also the existence of the impossible region) would make the rectangular grid inefficient: at many points, there would be no distribution at all. To circumvene this problem, we simply used a skewness-dependent kurtosis metric, the "kurtosis above minimum" instead of the usual kurtosis, where minimum was chosen to be $2\gamma_3^2+3$. (Mostly for convenience: this comfortably avoids the impossible region, lies within the coverage of the Fleishman distribution (see (2.20)), and has an additional benefit as well: the minimum "kurtosis above minimum" for zero skewness will be exactly the kurtosis of the normal distribution. That is: in our new skewness-"kurtosis above minimum" space, the normal distribution will be just at the origin!) This is illustrated on Figure 3.2, which shows the new coordinate system with typical grid points (maximum skewness of 4.0 in 32 steps, maximum "kurtosis above minimum" of 20.0 in 32 steps). The subsequent analysis will be performed at these points. (So the skewness-kurtosis space will be explored at $32 \cdot 32 = 1024$ points.)

This analyses consist of the performing of the already introduced Monte Carlo simulation: the program generates a huge number of random variates, applies the test to them, and records the empirically found Type I Error rate. And, as outlined in the previous paragraph, does this for the whole skewness-"kurtosis above minimum" grid. The number of Monte Carlo replications has to be chosen high enough so that the fluctuations are reduced (due to the convergence). Because of the highly complex investigation, tens of millions of replications might be needed, as illustrated on Figure 3.3 on the example of robustness testing with respect to non-normality of the z-test (i.e. the analysis of the actual α for nominal $\alpha = 0.05$ under H_0 and all other presumption holding, except for the normality of the population from which the samples are coming).

This way, the program iterates through the skewness-kurtosis space and empirically investigates the behavior of the test as a function of skewness and kurtosis resulting in a surface similar to the one seen above.

3.2 The application of GP-GPU to speed up calculations

The above implementation of the program would be extremely time consuming, even on modern personal computers. As we have already seen, the magnitude of the hypothesis testings necessary at each skewness-kurtosis combination to achieve small stochastic fluctuations is ten million. Assuming a grid with a size of 30×30, this means about 10 billion hypothesis testings. Also note that even at the very small sample size of 10, this also requires the generation of 100 billion random numbers. This would represent a prohibitive computational burden, even on the best personal computers.

However, a novel computational technique, developed in the last half decade can help us in an affordable way, without the need to employ a supercomputer. To understand it, note that this problem can be solved in a parallel way: one might, for example, concurrently perform the Monte Carlo simulations for two different skewness-kurtosis combinations. (As there are no interdependences between them.) Furthermore, even the simulations for the same skewness-kurtosis combination can be done in parallel (again, the result of one does not effect the other). Actually, this task is a typical example for the so-called *embarrassingly parallel* problems, which require so little communication between the sub-tasks (to which the problem can be decomposed) that they

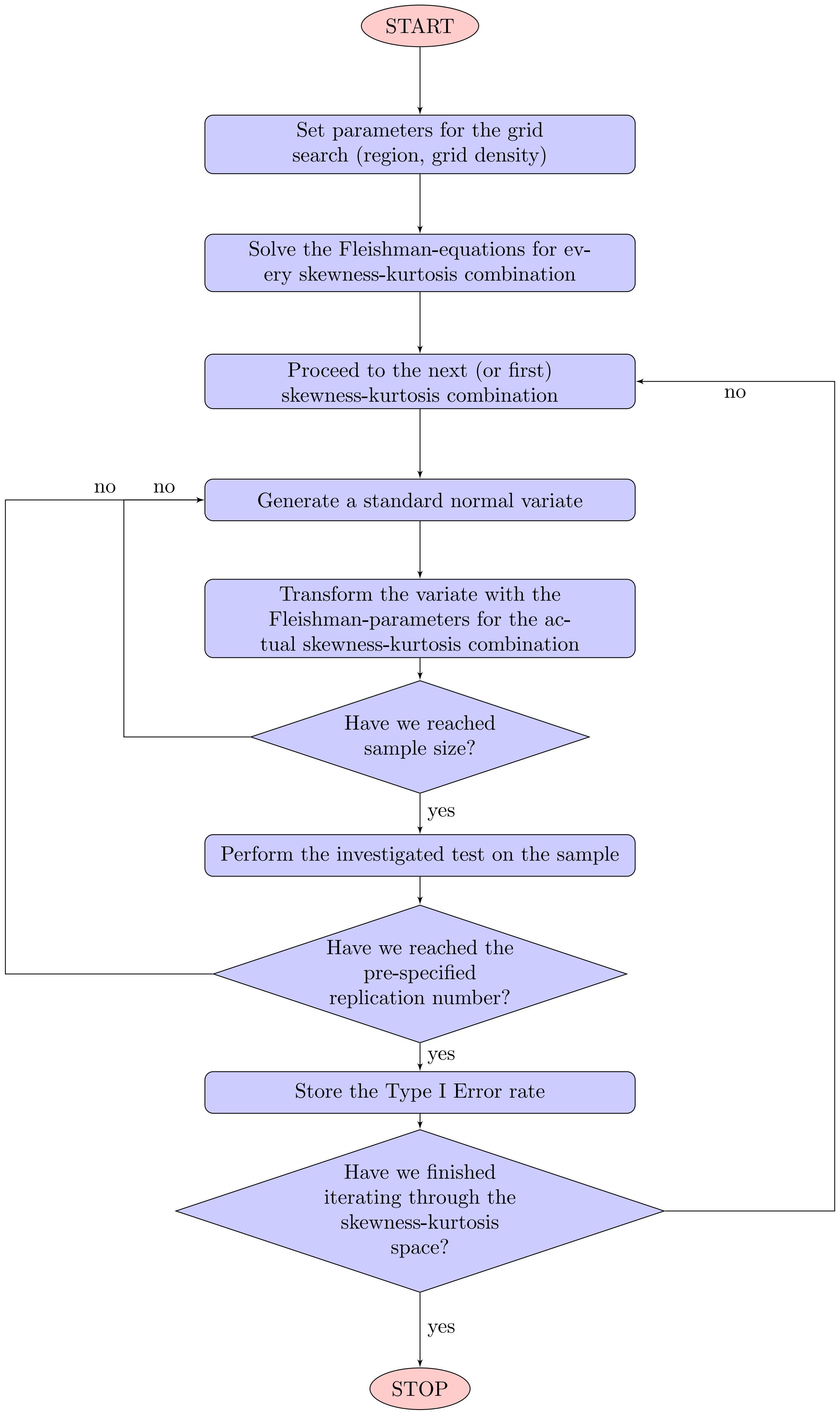

START
Set parameters for the grid search (region, grid density)
Solve the Fleishman-equations for every skewness-kurtosis combination
Proceed to the next (or first) skewness-kurtosis combination
no
Generate a standard normal variate
no
no
Transform the variate with the Fleishman-parameters for the actual skewness-kurtosis combination
Have we reached sample size?
yes
Perform the investigated test on the sample
Have we reached the pre-specified replication number?
yes
Store the Type I Error rate
Have we finished iterating through the skewness-kurtosis space?
yes
STOP

Figure 3.1: Broad overview of the mechanism of our Monte Carlo simulation environment.

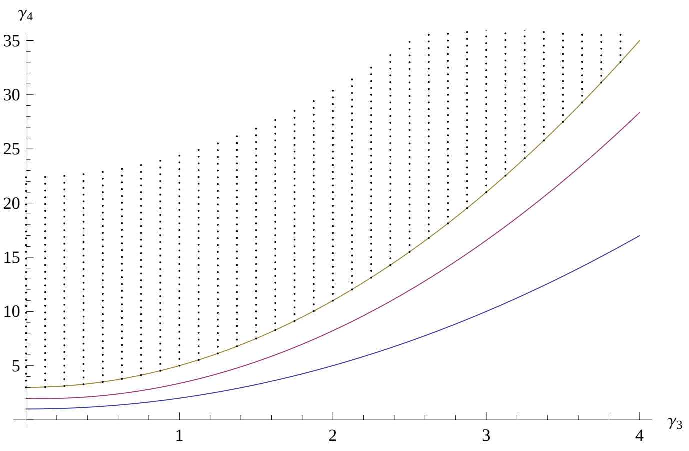
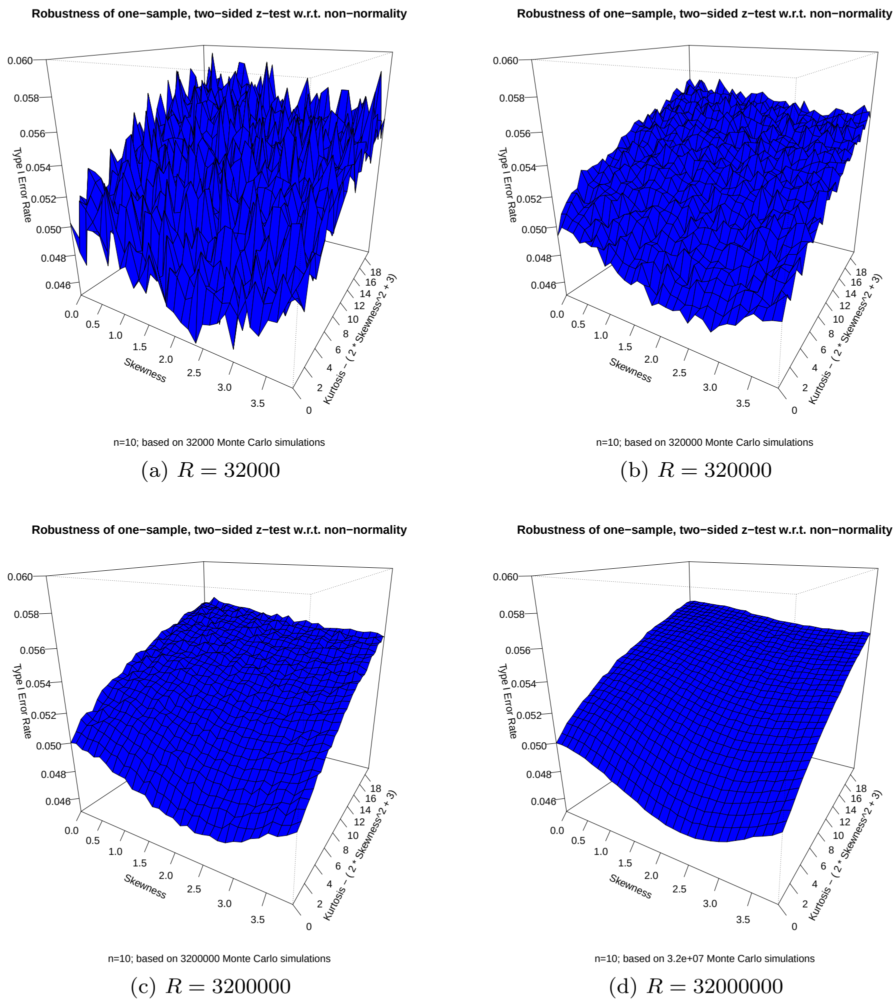


Figure 3.2: Skewness-kurtosis space with theoretical boundary for distributions (blue), lower bound of the coverage of Fleishman distribution (red), and bound of ”kurtosis above minimum” (for a given skewness) in yellow. Dots indicate a typical grid, where the properties of the statical test are analyzed.



(a) $R = 32000$

(b) $R = 320000$

(c) $R = 3200000$

(d) $R = 32000000$

Figure 3.3: Illustration for the effect of the number of Monte Carlo replications on the fluctuation of the result.

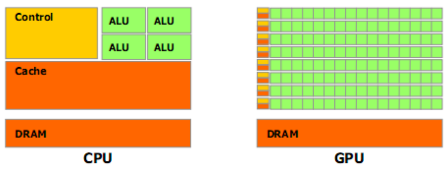


Figure 3.4: Comparison of the architecture of a CPU (left) and a GPU (right). Source: [43].

can be performed with almost unlimited parallelism [13]. (P is close to 1 in Amdahl's law[1].) As a matter of fact, the 10 billion hypothesis testings may be performed – in theory – by 10 billion processors at the very same time!

Let us now begin the story at its other end. The video cards of personal computers underwent an incredible development in the last decade. (Driven primarily by the need of 3D processing, especially by computer games.) At one point of this development, video card manufacturers began to include a chip on the cards which was dedicated to performing the computations that are necessary in 3 dimensional image generation. These chips are called Graphical Processing Units (GPUs) and represent an architecture that is specialized to the extreme: they can only perform a limited number of operations efficiently (as opposed to the CPUs of the personal computers), but in exchange they can perform these operations extremely efficiently. They are tailored exactly to the needs of such graphical operations, and the necessary, highly optimized components are placed in physically the same chip.

The most important aspect of this specialization is that the GPUs have relatively little on-chip caching and flow control, but a very high number of processing units that can operate in parallel [43]. (This fits the needs of 3 dimensional image generation, where it is typical that the very same operation has to be performed (so no advanced caching or flow control is needed), but on different data elements[2].) Thus, GPUs are best at solving tasks the can be decomposed into sub-tasks which are computation-intensive (not I/O-intensive) and can be executed in parallel (i.e not heavily interdependent on the results of the other sub-tasks). Because of this, no sophisticated flow control and caching is needed, so transistors may be rather assigned to increase parallel arithmetic capacity. This is illustrated on Figure 3.4.

To be more precise, GPUs can be seen as modified *stream processors*, representing a special SIMD (Single Instruction, Multiple Data) architecture [41]. With this specialization, GPUs can reach a performance that beats not only the more expensive CPUs, but even smaller supercomputers – for tasks that have the above characteristic... but *only* for such tasks.

As an illustration that we are not exagarating when we talk about supercomputer-like performance: an upper-level video card from NVidia's newest GeForce line as of the end of 2012 (GeForce GTX650 Ti from the GeForce 600 series) has a reference single-precision floating-point performance of 1425.4 GFLOPS with a release price of 149USD! For 499 USD, one can get 3090.4 GFLOPS (GeForce GTX680 from the GeForce 600 series [44]). As a comparison: this latter was better than the performance of the best supercomputer in the world as of 2000 June (ASCI Red of Sandia Laboratories, USA) [58]), but one could still get included in the TOP500 list of the world's most powerful supercomputers with this performance as of November 2006 [58] (500th place: Blade Cluster BL-20P of Telecommunication Company, USA)! Again: for 499 USD!

[1]Amdahl's law states [41] that if P is the proportion of a program that can be made to run in parallel, then the maximum speed-up that can be achieved by using N processors to perform that parallel part is $S(N) = \frac{1}{(1-P)+\frac{P}{N}} \xrightarrow[N\to\infty]{} \frac{1}{1-P}$.

[2]This is sometimes called *data parallelism*, which is a form of parallelism when the same operation is performed simultaneously, but on different parts of the data to be processed. This is opposed to *task parallelism* (the other end-point) where different operations are performed simultaneously, but on the same data.

It was realized very early at the history of GPUs that such fit-for-GPU tasks appear well outside the field of image generation, ranging from computational fluid dynamics to weather modeling. What if these could be executed on the GPU instead of the CPU? Many researchers began speculating that this way the performance of GPUs may be harnessed to perform a variety of non-image generation tasks. The era of GP-GPU (General Purpose GPU computing) arrived [60].

At first, programmers had to develop complex re-writes of the original code to enable its GPU-running (essentially they had to "trick the GPU into believing" that it is performing a graphical operation, in other words, real operations had to be rewritten, sometimes cumbersumbly, as graphical operations). However, shortly GPU-manufacturers also realized this potential, and started developing programming languages that support the general purpose programming of GPUs[3]. With these languages, such complicated tricks are no longer needed, GPUs can be directly programmed to general-purpose tasks. (Their special architecture, of course, still has to be respected, as already emphasized.)

Nowadays, the most widely used open-source GP-GPU computing language is OpenCL [20], while the most prevalent proprietary framework is NVidia's CUDA [42].

3.3 Implementation details

We performed the implementation of our Monte Carlo simulational environment for the empirical investigation of statistical tests based on GP-GPU – this way, we could achieve the performance necessary for such simulations in an affordable way. The plausibility of such implementation was discussed in detail in Section 3.2.

The actual implementation was done under NVidia's Compute Unified Device Architecture (CUDA) computing platform [42]. This platform enables the usage of a variety of programming languages (C, C++, Fortran) for GP-GPU computing through so-called language extensions[4]. These extensions make the expression of parallel processing possible (and also introduce the necessary restrictions).

We have used to CUDA C/C++ extension, under Microsoft Visual Studio 2008 integrated development environment [49].

A CUDA program generally consists of "ordinary" parts that run sequentially on CPU, and calls to so-called *kernels* which run in parallel on GPU. The design of a kernels has to take this parallelity into account (e.g. they can not rely on each other's results without further synchronization).

The kernels are executed by so-called *threads*. These are the basic units of the GPU processing, which run in parallel. (These threads will be mapped to the physical processing units (called streaming multiprocessor) of the GPU, but there is no need to manually specify this mapping – it is done transparently in the background by an automatic mechanism. This way, the program is automatically scaled to whatever capacity the GPU has on which it is run.)

Threads are joined together to so-called thread blocks, or shortly *blocks* in a one-, two- or three-dimensional manner. This is simply done to adapt to the real-world applications, where the task often has an intrinsic multidimensional structure (such as visualizing an array or a volume). In these cases, it is typical that the parallel parts pertain to the elements of this structure (pixel or voxel), hence the application of thread blocks reflects the true nature of the problem, making its handling more intuitive.

[3]Today, NVidia is even manufacturing "video cards" (NVidia Tesla) that do not have graphical output at all [45], i.e. they are solely useful for GP-GPU computing.

[4]There are other ways to enable GP-GPU computing for a language as well, such as the usage of CUDA-accelerated libraries or wrappers.

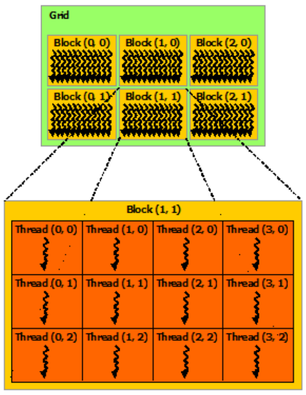


Figure 3.5: Hierarchy of threads, blocks and grids in the CUDA-model. Source: [43].

Thread blocks are then organized to *grids*, which can again be one-, two- or three-dimensional. The overall structure is shown on Figure 3.5.

In our case, it was logical to use a two-dimensional block structure, where dimensions represent skewness and kurtosis, i.e. each block represents a certain skewness-kurtosis combination. We chose to use 32 levels in both dimensions as it well fits the hardware architecture and provides enough granularity, so that the results will be smooth enough. This means in other words, that the program works in parallel in $32 \cdot 32 = 1024$ different skewness-kurtosis combinations at the same time. (Every thread and every block automatically receives a unique identification number from which it knows its own position, that is how they can do different tasks.)

Within each thread block, we specified 32 parallel threads. For architectural reasons [43] it is beneficial if the number of threads per block is a multiple of 32, so this was the minimum effective number. More could not be used because of the limitations imposed by the random number generator, as it will be immediately discussed.

To sum up, there were $32 \cdot 32 \cdot 32 = 32768$ parallel threads in our implementation. To further speed up calculations, we used another trick: one call to the kernel performs more than one (typically 100) simulational replication. (This is advantageous, because the slow part in the processing is the communication between the host (i.e. the computer) and the device (i.e. the video card). By forcing the kernel to perform more replications before it starts communicating with the host, we shift the ratio of the fast and slow parts for the former.)

Let us now see the operation of the program in detail! After initialization, it first solves the Fleishman equations for the grid points which will be scanned. For that end, we used a discrete Newton-algorithm from the `GSL` (GNU Scientific Library) library [15]. Results (which are stored as a matrix of data structures specifically declared to hold Fleishman coefficients) are then copied to the global memory of GPU. This memory is persistent across the whole running of the program, so this copying is needed only once – every time a thread needs the coefficients, it just has to look it up in the global memory. (Every threads receives the pointer to the memory area where these coefficients are stored.)

There is also a dedicated memory area in the global memory of the GPU where the results are collected after one round of simulations. (Remember that one round means performing test at *every* skewness-kurtosis combination, moreover, it means performing 32 tests.) This memory area is organized so that every block has an own space, where it has to record the number of rejections out of the 32 testings within the block. The avoid race conditions, only one thread can access this memory area within a block: there is an array in shared memory for every block with as many elements as threads (32 currently); only this is directly accessed by the threads of a block. (This is also advantageous because shared memory is much faster then global memory.) Thread #0 has the task to synchronize it with the area for the block in the global memory. This is solved by including a barrier-type synchronization instruction at the end of the kernel, which forces every thread to stop at that point until everyone is at that point (that is: every thread finished processing). At that moment Thread #0 reads out the per-block storage, aggregates the results, and uploads it to the area of the block in the global memory. As this is also a persistent storage, the next time the kernel is called, it can just increment this cell with the number of rejections in the new round.

As far as the operation of the kernel is concerned, it begins with the random number generation. Parallel Mersenne twister algorithm [48] is used to generate variables with uniform distribution which are then transformed with the Box-Müller algorithm [8] so that we obtain the standard normal variates necessary for the Fleishman-distribution.

The parameters of the Mersenne twisters are set with the `dcmt` algorithm [38]. The only drawback is that to achieve maximum performance, every thread should have an own random number generator. However, the largest parameter set created in advance contains "only" 32768 random number generators, and the creation of a larger parameter set is hopeless (practically requires a supercomputer). That was the reason why we used only 32 threads per block – this way, we just used up all random number generators.

After a random number is generated, the thread reads out the position (i.e. skewness-kurtosis) of the block in which the thread is, then reads out the corresponding element from the array of Fleishman coefficients (i.e. the coefficients that are needed to transform the standard normal variate to a variate with that skewness and kurtosis) and performs the transformation.

During the designing of this program, our key aim was *modularity*: we wanted to create an environment which can be easily extended, even by non-programmers. For that end, every single statistical test is written as a separate function. The kernel only calls the function, passes the sample as an array of float numbers, and waits for an accept/reject output (at the given significance level). The best part is that the test can be a simple C program, the whole parallelization is completely transparent – the environment can be extended with new tests without any knowledge in GPU programming!

After the kernel is called enough times (typically tens of millions are needed for smooth results which means about 10000 calls, as every block has 32 threads, and every thread performs 100 replications), the result (the matrix which contains the fraction of rejections at every skewness-kurtosis combination) is written in a serialized format, along with the most important descriptors of the simulation (number of runs, parameters of the grid etc.) to be used by the visualization program.

To have a comparison point for the performance results, Table 3.1 lists the most important parameters of the GPU we used to perform the calculations that are presented in this thesis. It worth noting when interpreting results that this is a low-end video card by today's standards…

The user interface of the program during running is shown on Figure 3.6.

After finishing the calculations, the program display a performance summary as a benchmark (Figure 3.7).

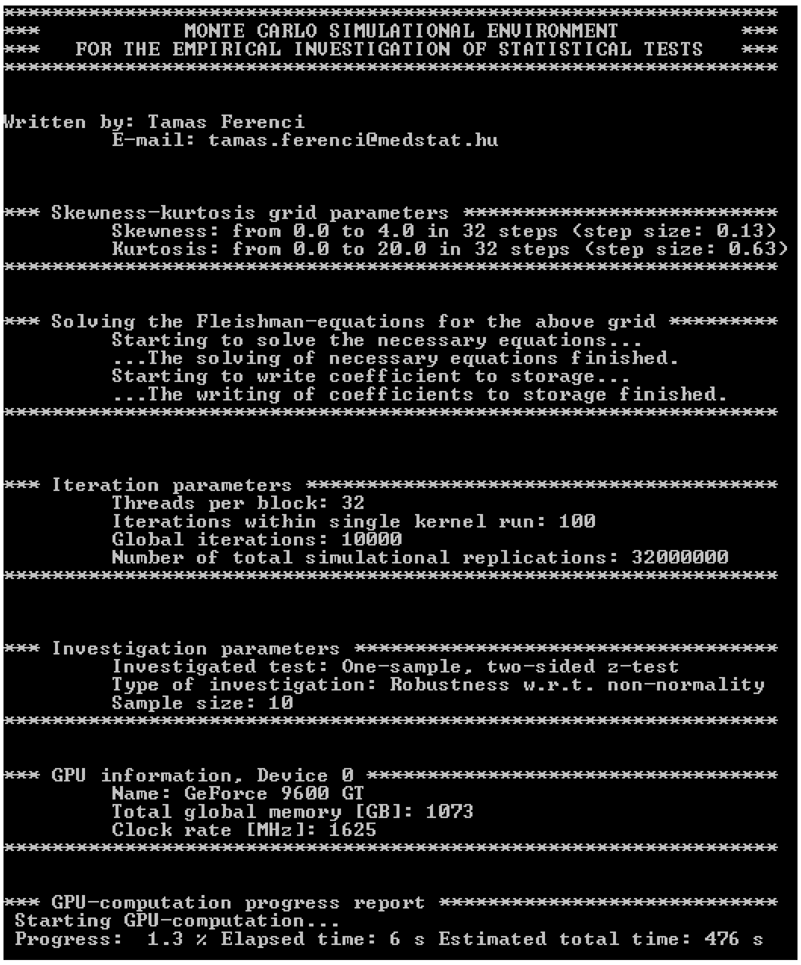
We can see that in this case (testing one-sample z-test with a sample size of 10) the average, sustained performance of our program was in excess to 60 millio hypothesis testing per second.

3.4 Visualization of the results

Our simulational environment outputs – as described in Section 3.3 – the results (actual αs or powers for different skewness-kurtosis combinations) as a serialized matrix. We have developed

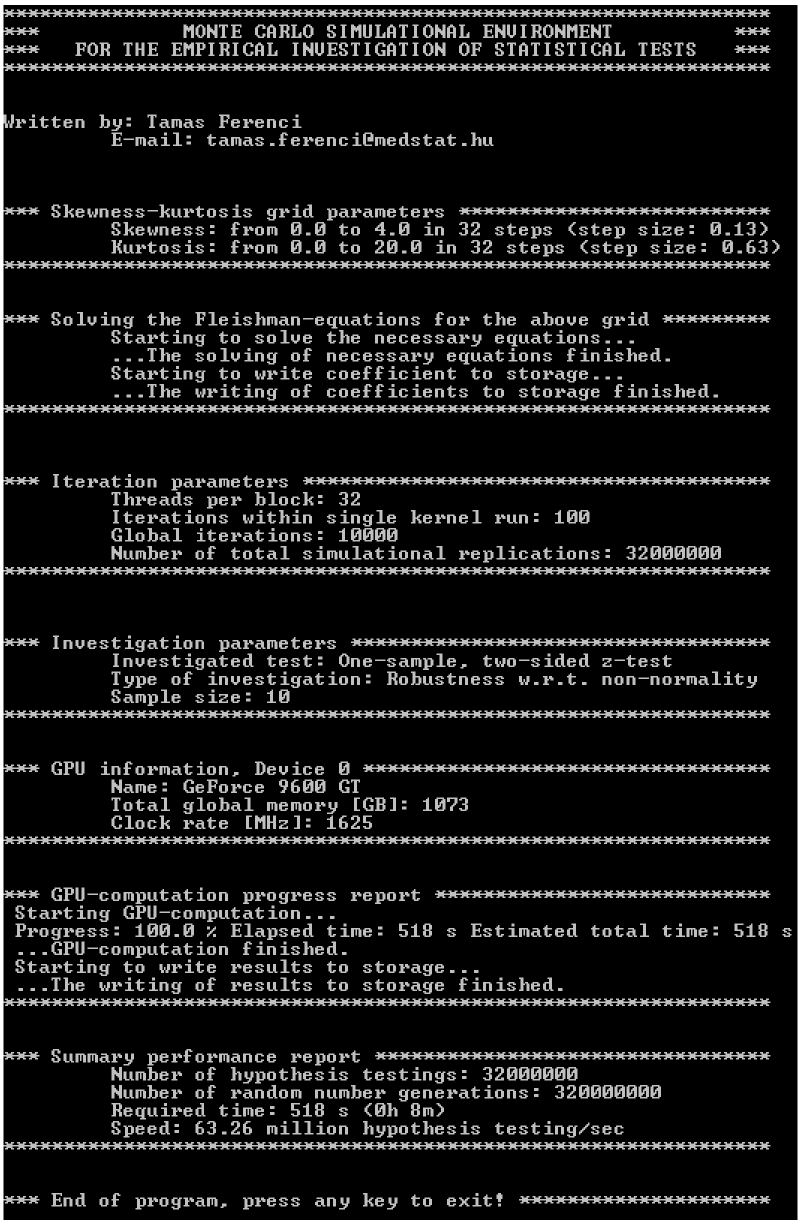
Parameter	Value
Name	NVidia GeForce 9600 GT
GPU	G94a/b
Fab	65/55 nm
Number of stream processors	64
Number of raster operation units	16
Number of texture address/texture filter units	32
Fillrate	20.8 billion texel/s
Core clock	650 Mhz
Unified shader clock	1625 MHz
Memory clock	1800 MHz
Memory	512 MB GDDR3
Memory bandwidth	57.6 GB/s
Transistor count	505 million
Floating-point performance	312 GFLOPS

Table 3.1: Parameters of the GPU used to perform the calculations that are presented in this thesis.



```
****************************************************************
***           MONTE CARLO SIMULATIONAL ENVIRONMENT           ***
***   FOR THE EMPIRICAL INVESTIGATION OF STATISTICAL TESTS   ***
****************************************************************


Written by: Tamas Ferenci
         E-mail: tamas.ferenci@medstat.hu


*** Skewness-kurtosis grid parameters **************************
         Skewness: from 0.0 to 4.0 in 32 steps (step size: 0.13)
         Kurtosis: from 0.0 to 20.0 in 32 steps (step size: 0.63)
****************************************************************


*** Solving the Fleishman-equations for the above grid *********
         Starting to solve the necessary equations...
         ...The solving of necessary equations finished.
         Starting to write coefficient to storage...
         ...The writing of coefficients to storage finished.
****************************************************************



*** Iteration parameters ***************************************
         Threads per block: 32
         Iterations within single kernel run: 100
         Global iterations: 10000
         Number of total simulational replications: 32000000
****************************************************************



*** Investigation parameters ***********************************
         Investigated test: One-sample, two-sided z-test
         Type of investigation: Robustness w.r.t. non-normality
         Sample size: 10
****************************************************************


*** GPU information, Device 0 **********************************
         Name: GeForce 9600 GT
         Total global memory [GB]: 1073
         Clock rate [MHz]: 1625
****************************************************************


*** GPU-computation progress report ****************************
 Starting GPU-computation...
 Progress:  1.3 % Elapsed time: 6 s Estimated total time: 476 s
```

Figure 3.6: User interface of the developed simulational environment during running.



```
****************************************************************
***            MONTE CARLO SIMULATIONAL ENVIRONMENT          ***
***   FOR THE EMPIRICAL INVESTIGATION OF STATISTICAL TESTS   ***
****************************************************************


Written by: Tamas Ferenci
         E-mail: tamas.ferenci@medstat.hu



*** Skewness-kurtosis grid parameters **************************
         Skewness: from 0.0 to 4.0 in 32 steps (step size: 0.13)
         Kurtosis: from 0.0 to 20.0 in 32 steps (step size: 0.63)
****************************************************************


*** Solving the Fleishman-equations for the above grid *********
         Starting to solve the necessary equations...
         ...The solving of necessary equations finished.
         Starting to write coefficient to storage...
         ...The writing of coefficients to storage finished.
****************************************************************



*** Iteration parameters ***************************************
         Threads per block: 32
         Iterations within single kernel run: 100
         Global iterations: 10000
         Number of total simulational replications: 32000000
****************************************************************



*** Investigation parameters ***********************************
         Investigated test: One-sample, two-sided z-test
         Type of investigation: Robustness w.r.t. non-normality
         Sample size: 10
****************************************************************


*** GPU information, Device 0 **********************************
         Name: GeForce 9600 GT
         Total global memory [GB]: 1073
         Clock rate [MHz]: 1625
****************************************************************


*** GPU-computation progress report ****************************
 Starting GPU-computation...
 Progress: 100.0 % Elapsed time: 518 s Estimated total time: 518 s
 ...GPU-computation finished.
 Starting to write results to storage...
 ...The writing of results to storage finished.
****************************************************************


*** Summary performance report *********************************
         Number of hypothesis testings: 32000000
         Number of random number generations: 320000000
         Required time: 518 s (0h 8m)
         Speed: 63.26 million hypothesis testing/sec
****************************************************************


*** End of program, press any key to exit! *********************
```

Figure 3.7: User interface of the developed simulational environment after completing simulations.

a program under the R statistical program package [50] (version 2.15.1) to perform an effective visualization of such matrices (see Appendix C.1.)

The program first reconstructs the matrix and performs the visualization afterwards; either with perspective plot or with contour plot. The former is more spectacular, and gives an overall impression more quickly, but the latter is more precise, as no part of the visualization can be lost by being in shadow [6].

The visualization program also interprets the supplementary information provided by the CUDA program (number of runs, parameters of the grid etc.) to automatically label the plots accordingly.

Chapter 4

Empirical investigation of statistical tests with Fleishman distribution

In this Chapter, we will synthetise the concepts already introduced: we will use Fleishman distribution (Chapter 2) to solve the problems (robustness and power investigation) outlined in Chapter 1 with the method (Monte Carlo simulation with GP-GPU) discussed in Chapter 3.

We will go through the practically most important tests and present both the results for the empirical investigation of their properties, and the discussion of these results, which address the most important findings.

We will first present the results for the robustness of parametric test w.r.t. their distributional assumption in Section 4.1, and then, we will turn to power testing of non-parametric tests in Section 4.2.

4.1 Robustness testing of parametric tests

In this Section, we review robustness of parametric tests (i.e. tests that assume a distribution family for the population) with respect to their distributional assumption. (Which will be normality in every case.)

We will review one-sample, two-sample and K-sample tests as well, but we will confine ourselves to two-sided tests (to make the discussion more compact).

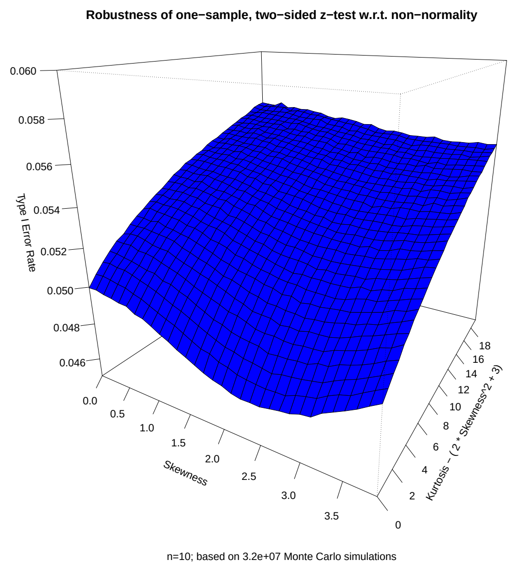
As already noted, the length of discussion will be varying; we will focus on the interesting aspects of the tests.

4.1.1 One-sample, two-sided z-test

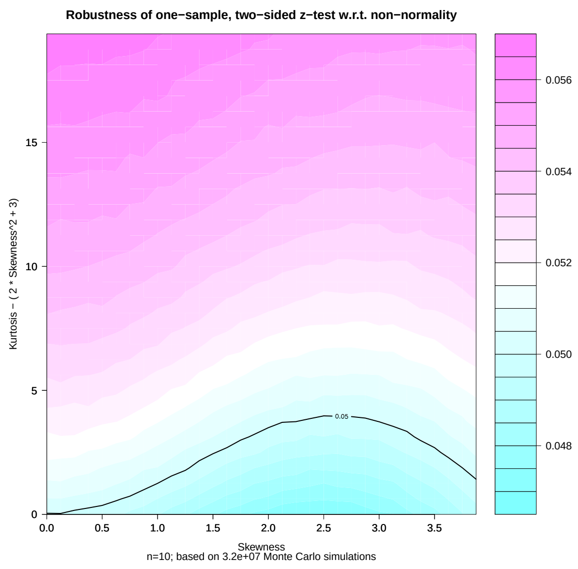
One-sample, two-sided z-test is a test for the hypothesis pair

$$\begin{aligned} H_0 &: \mu = \mu_0 \\ H_1 &: \mu \neq \mu_0, \end{aligned} \tag{4.1}$$

for $\mu_0 \in \mathbb{R}$ and μ being the expected value of the population from which the sample is coming. It has the following assumptions:



(a) Perspective plot



(b) Contour plot

Figure 4.1: Robustness of the one-sample, two-sided z-test with respect to non-normality with Fleishman distribution (nominal $\alpha = 0.05$)

- The distribution of the population is normal.
- The variance of the population is σ_0^2 constant known *a priori*.
- The sampling is iid.

The test statistic and its null distribution is:

$$Z(\mathbf{X}) = \frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu_0}{\sigma_0/\sqrt{n}} \overset{H_0}{\sim} z. \tag{4.2}$$

One-sample z-test is rarely used in practice because the *a priori* known population variance is irrealistic in almost every practical setting. For this reason, we will only shortly discuss this test.

Results for the robustness of this test with respect to non-normality with Fleishman distribution are shown on Figure 4.1 for $n = 10$.

It can be seen that the test is *very highly robust* with respect to non-normality: the smallest actual α was 0.04698 on the whole testing grid, the largest was 0.05690. This can be attributed to the *a priori* known variance.

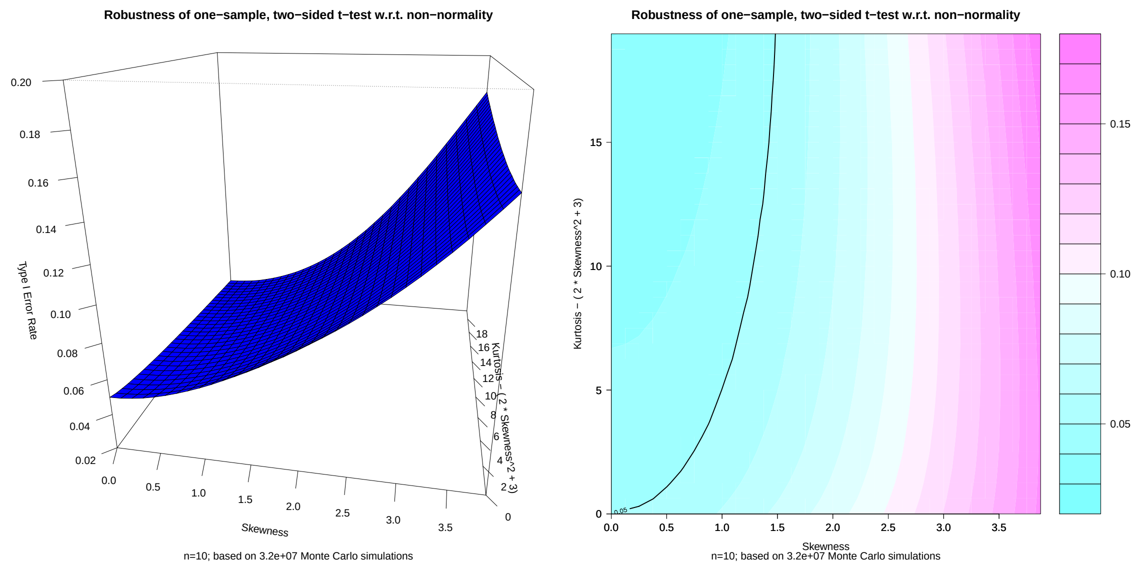
4.1.2 One-sample, two-sided t-test

One-sample, two-sided t-test is a test for the hypothesis pair

$$\begin{aligned} H_0 &: \mu = \mu_0 \\ H_1 &: \mu \neq \mu_0, \end{aligned} \tag{4.3}$$

for $\mu_0 \in \mathbb{R}$ and μ being the expected value of the population from which the sample is coming. It has the following assumptions:

- The distribution of the population is normal.
- The sampling is iid.



(a) Perspective plot (b) Contour plot

Figure 4.2: Robustness of the one-sample, two-sided t-test with respect to non-normality with Fleishman distribution (nominal $\alpha = 0.05$)

The test statistic and its null distribution is:

$$t\left(\mathbf{X}\right) = \frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu_0}{s^*/\sqrt{n}} \overset{H_0}{\sim} t\left(n-1\right). \tag{4.4}$$

It can be seen that this test is very similar to the z-test, the only exception being that it does not presume known population variance. As a consequence it has less power (it also has to estimate the variance from the sample), but in practice this variance is almost never known *a priori*, so t-test is the one that is almost exclusively used instead of the z-test. For this reason we will discuss our results in more detail.

Results for the robustness of this test with respect to non-normality with Fleishman distribution are shown on Figure 4.2.

This point is perhaps the best to elaborate on another issue (that affects every result presented in this format). The problem is with the scaling of the kurtosis axis. Note that it is not "true" kurtosis itself, but rather kurtosis minus $2\gamma_3^2+3$. As already explained, this is necessary to keep the scanning in the feasible region (and, in addition, within the coverage of the Fleishman distribution). Strictly speaking, there is still no problem: we might still plot skewness and true kurtosis for presenting the results. However, this would be a "graphically" very unfortunate choice: most of the plot would be empty, with the part pertaining to high values of skewness almost completely shifted compared to those that belong the low skewness (due to the rapid growth of the $2\gamma_3^2 + 3$ function). This would be especially apparent for perspective plots.

For this reason, we used the "kurtosis above minimum" axis instead of the usual kurtosis. This, however, introduces a distortion: points are displaced based on their skewness. This is illustrated on Figure 4.3. (See also Figure 3.2 for this discussion!) This figure shows different slices of the surface shown on Figure 4.2, along the skewness axis (i.e. Type I Error rate vs. skewness for different levels of kurtosis, Subfigures 4.3a and 4.3b) and along the kurtosis axis (i.e. Type I Error rate vs. kurtosis for different levels of skewness, Subfigures 4.3c and 4.3d). The figure shows the effect of using the transformed kurtosis (Subfigures 4.3a and 4.3c) vs. using the "true" kurtosis

(Subfigures 4.3b and 4.3d).

Subfigure 4.3b, which required slicing the surface parallel to the skewness axis (including points which were possible not sampled at the original investigation) was created by bilinear interpolation using the `R` library `fields` [14].

For the reasons described above we will still use figures with "kurtosis above minimum" axes, but these considerations should be kept in mind when interpretating them.

Overall, the test is not robust to non-normality: it gets too liberal, with actual α going up to almost 0.2 in our scenarios. It is especially sensitive to skewness, but kurtosis also affects its behavior. This conclusion is in concordance with the literature findings (Section 1.5).

There is one final phenomenon we want to demonstrate on this test: the effect of sample size. Fiugre 4.4 shows the same plots as in Figure 4.2, but with varying sample sizes this time ($n = 10, 20, 50, 100$).

One can wonderfully observe the effect of the Central Limit Theorem [10].

4.1.3 Two independent sample, two-sided t-test

Two independent sample, two-sided t-test is a test for the hypothesis pair

$$\begin{aligned} H_0 &: \mu_1 = \mu_2 \\ H_1 &: \mu_1 \neq \mu_2, \end{aligned} \tag{4.5}$$

with μ_1 and μ_2 being the expected values of the populations from which the samples are coming. It has the following assumptions:

- The distributions of the populations are normal.
- The variances of the populations are equal (but not necessarily known).
- The sampling is iid.

The test statistic and its null distribution is:

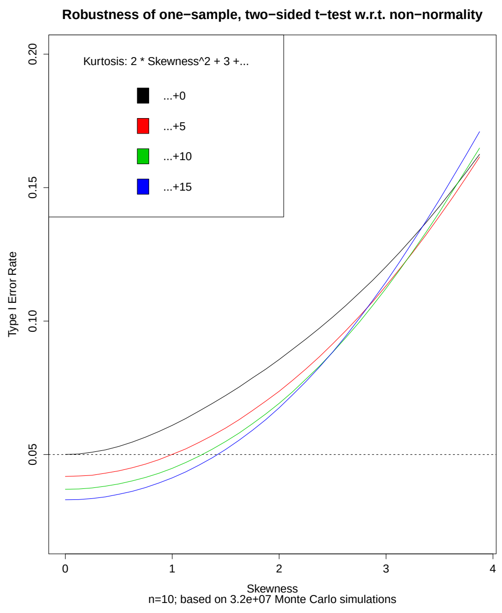
$$t(\mathbf{X}) = \frac{\frac{1}{n}\sum_{i=1}^{n} X_{1,i} - \frac{1}{n}\sum_{i=1}^{n} X_{2,i}}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}} \cdot \sqrt{\frac{(n_1-1)s_1^{*2} + (n_2-1)s_2^{*2}}{n_1+n_2-2}}} \overset{H_0}{\sim} t(n_1 + n_2 - 2). \tag{4.6}$$

Two-sample t-test is perhaps the most widely used test in many areas of applied statistics.

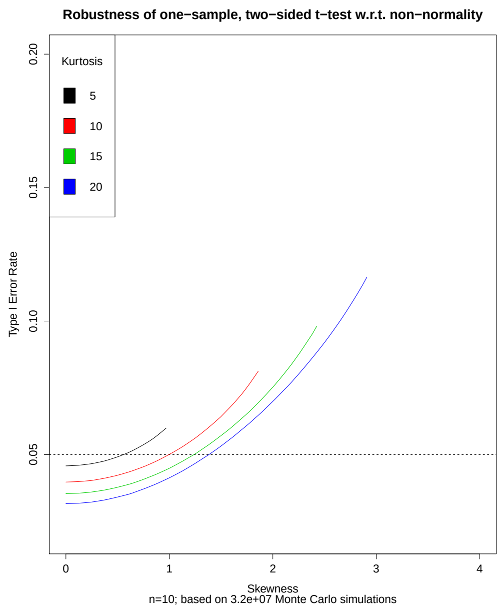
Results for the robustness of this test with respect to non-normality with Fleishman distribution are shown on Figure 4.5 for $n = 10$ (in both groups).

It can be seen that the test is again not especially robust, but – in contrast to the one-sample variant – it gets too conservative (not too liberal) for non-normal populations, with actual α going down to about 0.02 in our test. This conclusion is in concordance with the literature findings (Section 1.5).

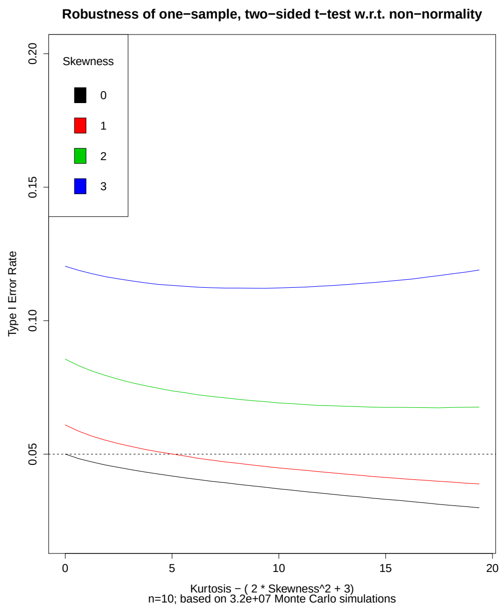
Note that this investigation presumed that both population has the same – non-normal – distribution. The investigation can be made deeper by including the possibility of different background distributions for the two populations – at the cost of more dimensionality, which makes the illustration, understooding of the results more complicated.



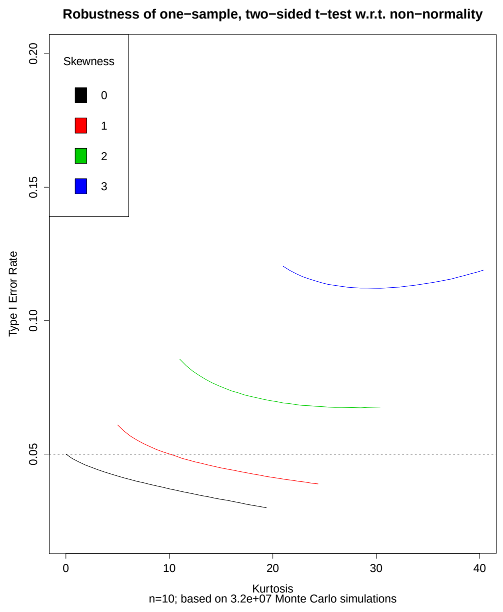
(a) Slicing parallel to the skewness axis, using transformed kurtosis



(b) Slicing parallel to the skewness axis, using original kurtosis

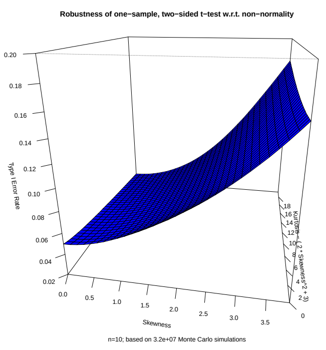


(c) Slicing parallel to the kurtosis axis, using transformed kurtosis

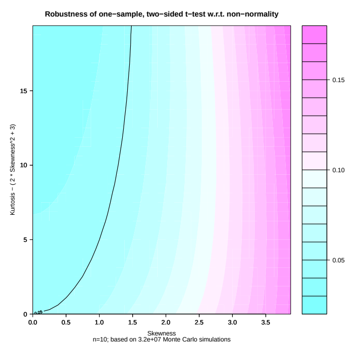


(d) Slicing parallel to the kurtosis axis, using original kurtosis

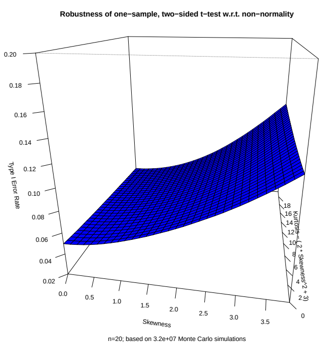
Figure 4.3: Illustration of the distortion effect of using "kurtosis above minimum" instead of kurtosis (see also Figure 3.2).



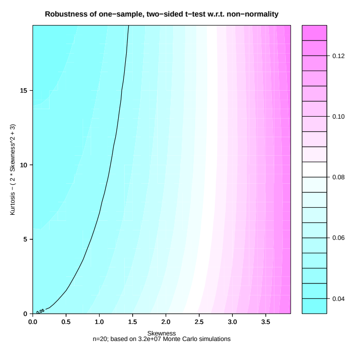
(a) Perspective plot, $n = 10$



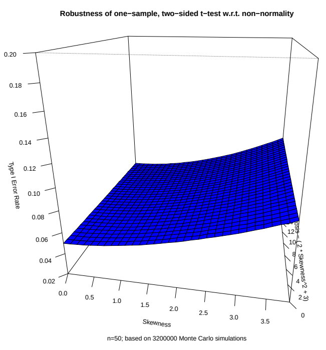
(b) Contour plot, $n = 10$



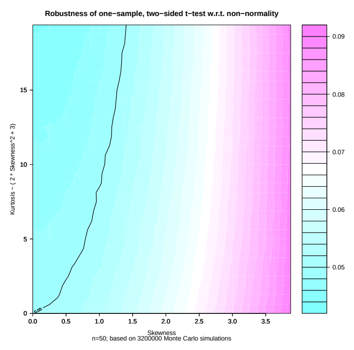
(c) Perspective plot, $n = 20$



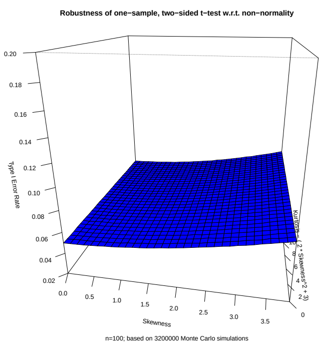
(d) Contour plot, $n = 20$



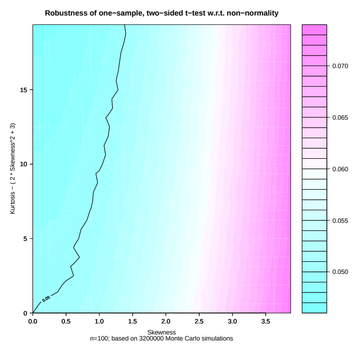
(e) Perspective plot, $n = 50$



(f) Contour plot, $n = 50$

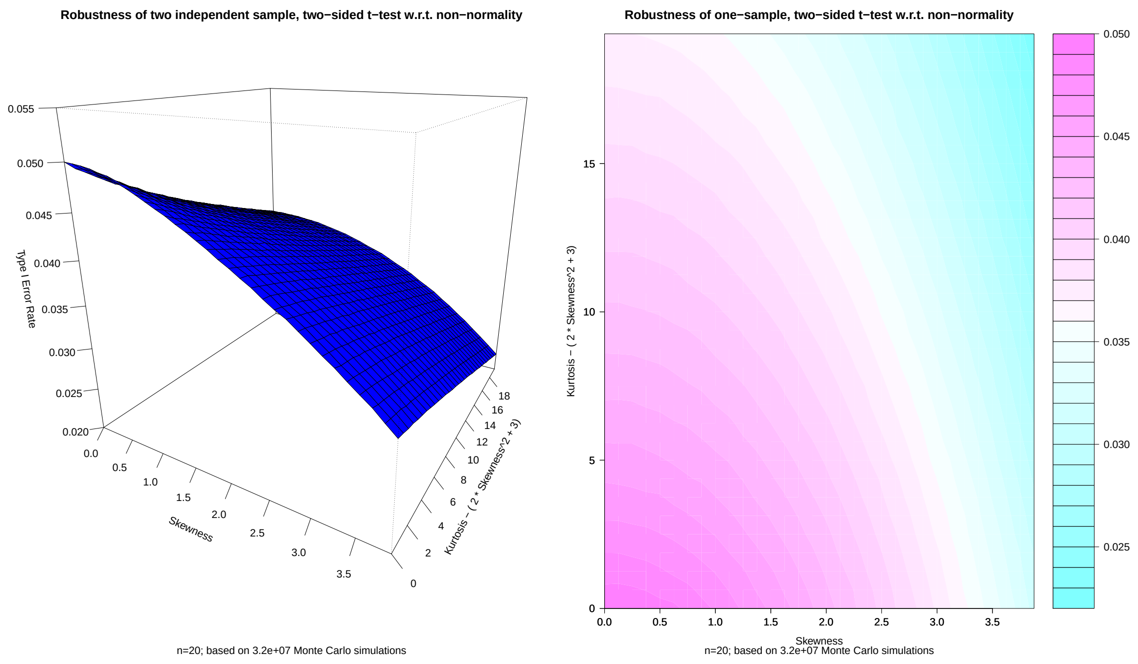


(g) Perspective plot, $n = 100$



(h) Contour plot, $n = 100$

Figure 4.4: Impact of sample size on the robustness of the one-sample, two-sided t-test (nominal $\alpha = 0.05$).



(a) Perspective plot (b) Contour plot

Figure 4.5: Robustness of the two independent sample, two-sided t-test with respect to non-normality with Fleishman distribution (nominal $\alpha = 0.05$)

4.1.4 K independent sample ANOVA

K independent sample ANOVA is a test for the hypothesis pair

$$\begin{aligned} H_0 &: \mu_1 = \mu_2 = \ldots = \mu_k \equiv \mu \\ H_1 &: \exists 1 \le j \le k : \mu_j \neq \mu, \end{aligned} \tag{4.7}$$

with μ_i being the expected value of population i. It has the following assumptions:

- The distributions of the populations are normal.
- The variances of the populations are equal (but not necessarily known).
- The sampling is iid.

The test statistic and its null distribution is:

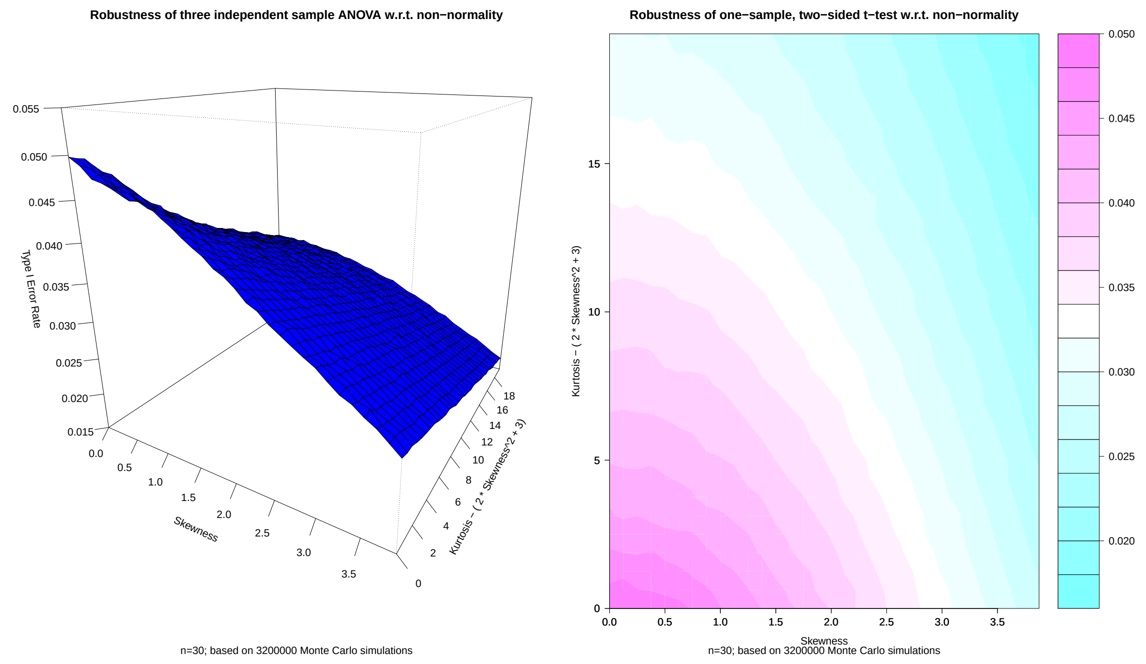
$$F(\mathbf{X}) = \frac{\sum_{j=1}^{K} \sum_{i=1}^{n_j} \left(X_{ij} - \overline{X}_j \right)^2 / (K-1)}{\sum_{j=1}^{K} n_j \left(\overline{X}_j - \overline{X} \right)^2 / (n-K)} \overset{H_0}{\sim} \mathcal{F}(K-1, n-K). \tag{4.8}$$

Results for the robustness of this test for $K = 3$ with respect to non-normality with Fleishman distribution are shown on Figure 4.6 for $n = 10$ (in all groups).

It can be seen that the test's behavior is very similar to the two-sample t-test.

4.2 Power testing of non-parametric tests

In this Section, we will turn our attention to the power of non-parametric tests (i.e. tests that do not assume a distribution family for the population).



(a) Perspective plot (b) Contour plot

Figure 4.6: Robustness of the K independent sample ANOVA ($K = 3$) with respect to non-normality with Fleishman distribution (nominal $\alpha = 0.05$)

In power testing, we will always assume that presumptions hold (these, however, do not specify a distribution for the population – this is the point where the application of Fleishman distribution, or similar distribution becomes necessary), but the H_0 does not. Because of this, we will intentionally use populations that violate the condition in H_0.

4.2.1 Two-sample, two-sided Mann-Whitney U-test

The hypotheses of the two-sample, two-sided Mann-Whitney U-test can be formulated in many (mostly equivalent) ways. One possibility is to use the so-called stochastic equality

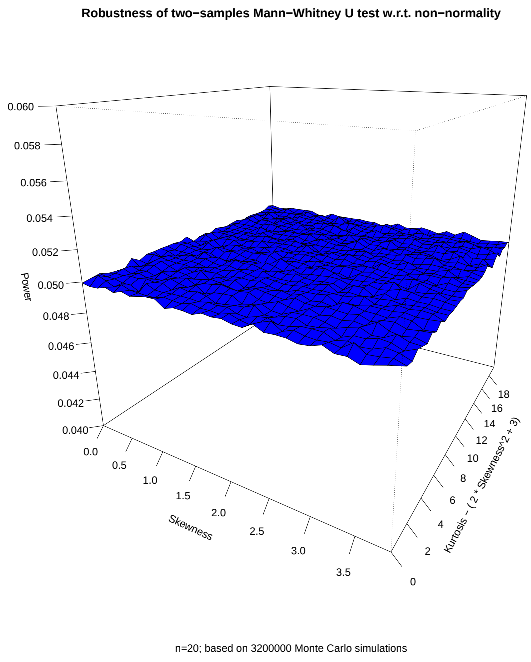
$$\begin{aligned} H_0 &: \mathbb{P}(X > Y) = \mathbb{P}(X < Y) \\ H_1 &: \mathbb{P}(X > Y) \neq \mathbb{P}(X < Y), \end{aligned} \tag{4.9}$$

It can be seen that it in fact imposes no presumption on the population of X and Y. It, however, has other constraints:

- It assumes a shift-model, i.e. that Y has a same distribution as X in shape, but, perhaps shifted.
- The sampling is iid.

First, let us "check" whether the test in fact lives up to expectations! We will perform a robustness testing (which theoretically should not be effected at all by different skewness-kurtosis levels). Indeed,this happens, see Figure 4.7.

Turning to our true question, let us perform a power analysis! For that, we have to presume how we want to violate H_0 – now we introduced a $\delta = +0.5$ shift in one of the samples. The results are shown on Figure 4.8.


(a) Perspective plot

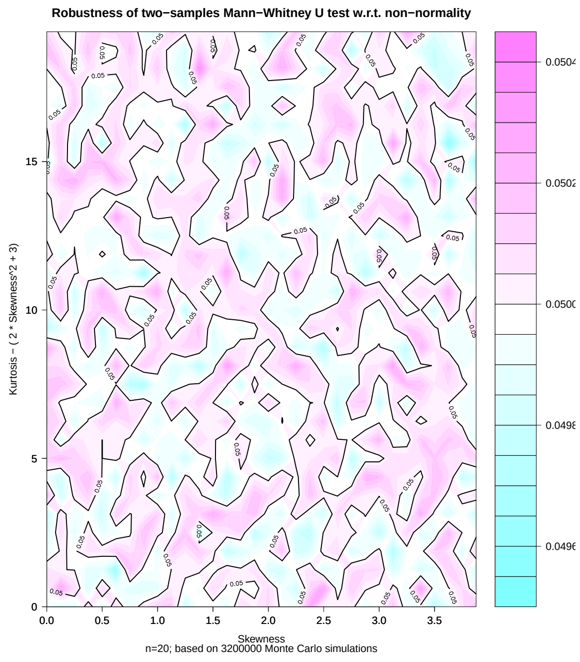
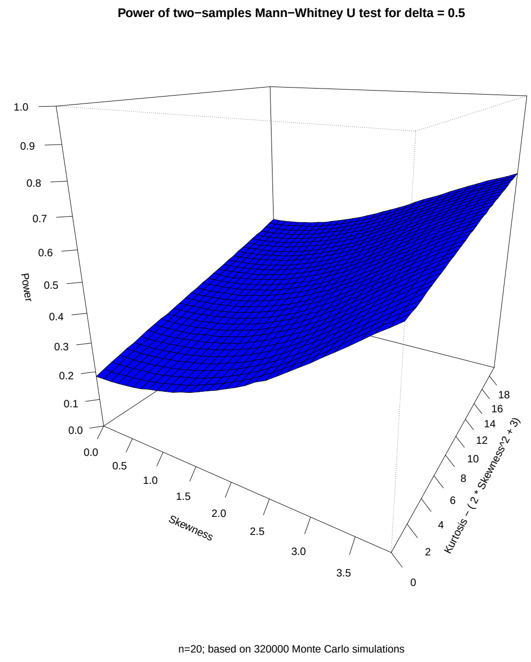

(b) Contour plot

Figure 4.7: Robustness of the two-sample, two-sided Mann-Whitney U-test with respect to non-normality with Fleishman distribution (nominal $\alpha = 0.05$)


(a) Perspective plot

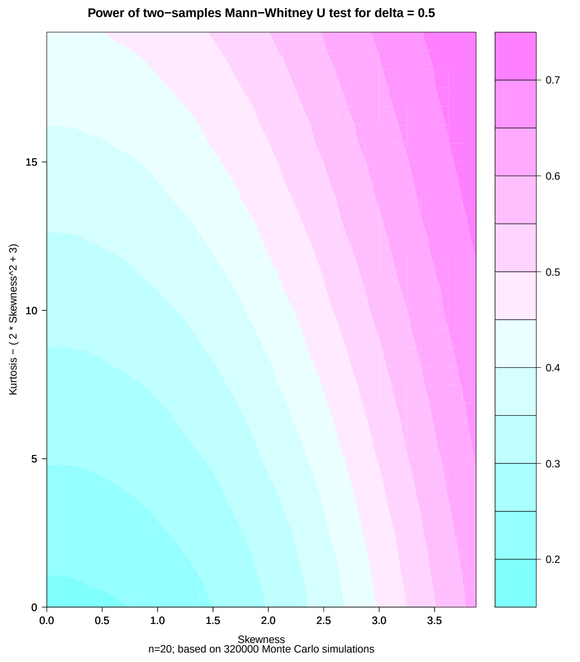

(b) Contour plot

Figure 4.8: Power analysis of the two-sample, two-sided Mann-Whitney U-test for a shift of $\delta = +0.5$.

We can see that the background distribution in fact has a major effect on the power of the test.

Chapter 5

Limitations of Fleishman distribution

In this Chapter, we present one limitation of our approach that is induced by the application of a distribution that is selected based on its first four moments.

In Section 5.1, we introduce the problem itself, and in Section 5.2 we refer the most important results that pertain to this limitation.

5.1 A problem statement

The whole approach that utilizes Fleishman distributions is based on the implicit assumption that a distribution is well-characterized by its first four moments. To make the statement "well-characterized" more precise: this approach implicitly assumes that the test's behavior for a given population can be adequately grabbed by the population's first four moments.

However, examples can be shown that might make us uncertain about this assumption: it is not true that distributions are necessarily similar if they have identical first four moments. Figure 5.1 shows the pdf of the mixture normal distribution $\frac{3}{4}\mathcal{N}\left(0,1\right)+\frac{1}{4}\mathcal{N}\left(2,2^2\right)$ and the Fleishman distribution with identical first four moments.

The result is embarrassing: the pdfs are *visibly* different. While this does not necessarily invalidate our analysis (despite the difference of the pdfs, the tests might show similar behavior, for example, if they are dominantly dependent on the first four moments, and not on the information about the distribution beyond its first four moments), this is still not comforting, at best.

What if we want to investigate the robustness of a test for an applied statistical field where such mixture distributions are typical? Using Fleishman distribution for the empirical investigation would then imply that we base our investigation on a distribution that has visibly different pdf than the typical distributions encountered on that field. This makes the question of how representative are the previous results in Chapter 4 unavoidable. (This is not an academical question: such mixture-type distributions are in fact often encountered for example in biostatistics: often, healthy and sick subjects have similar distribution for a given marker, but with shifted mean. Another typical example is the distribution of parasites in quantitative parasitology [55].)

We will address the question of how similar the behavior of the tests are if the first four moments are identical, i.e what limits can be given if we have information on the equality of the first for moments (but nothing more) in Chapter 5. In the meantime, let us examine what differences are possible between the pdfs of two distributions if they have a certain number of identical moments!

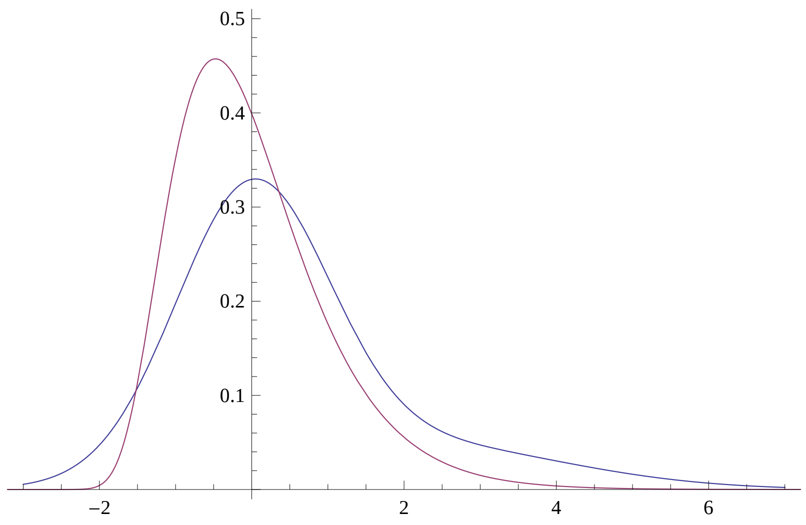


Figure 5.1: Pdf of the mixture normal distribution $\frac{3}{4}\mathcal{N}\left(0,1\right)+\frac{1}{4}\mathcal{N}\left(2,2^2\right)$ (blue) and the Fleishman distribution with identical first four moments (red).

First, let us agree that we will measure the difference between two pdfs with total variation distance (TVD), that is, essentially with the L_1 distance between the pdfs. (We will confine ourselves to continuous distributions, so this definition is satisfactory.)

First note that increasing the number of matching moments does not necessarily help in this sense (in extreme case, it can even make the TVD larger).

At first glance, this might be surprising, but actually it is understandable by considering what higher moments mean: they carry information on the tail of the distribution [35] (the higher moment we consider, the more distant part of the distribution is important), so including more moments might make the approximation of the tails better, but has almost nothing to do with the central part of the distribution. There, the pdf might even be more different (as far as the restriction on the smaller moments are fulfilled), because the restriction imposed through higher moments does not effect the distribution's central part, which can still arbitrarily contribute to TVD. This is in concordance with the findings of McCullagh [39], who demonstrated that two distributions with visibly different pdf (note, that McCullagh also used TVD to measure this) can have virtually identical moment-generating function. With a slight modification of his example, we can create an almost perfect demonstration for our point: consider the pdfs $f_1\left(x\right)=\phi\left(x\right)$ and $f_2\left(x\right)=\phi\left(x\right)\cdot\left(1+\cos\left(2\pi x\right)\right)\cdot\frac{1}{1+e^{-2\pi^2}}$, where $\phi\left(x\right)=\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ is the standard normal density. These pdfs are shown on Figure 5.2. They are visible different, their TVD (through numerical integration) is 0.6366. In spite of this, their moments are virtually equal, with the largest difference relative to $f_1\left(x\right)$'s moments being 0.0075% among the first 50 moments! Note however, that this "virtual equality" only stands in relative terms; the absolute difference might still be very high, as the numerical value of moments increases extremely fast [64, 37]. This poses problems when the moments are numerically matched.

The central result here is that of Akhiezer [2], who has shown that the L_∞ distance of the cdfs (but not the TVD) can be limited based on the moments of the distribution, using the so-called moment matrix ([34]) utilizing an appropriately defined series of orthogonal polynomials. Unfortunately this bound, however, can not be sharp, of course, in terms of pdfs, as derivatives (value of pdf) in a point can be arbitrarily different even if the cdf takes the exact same value. Other limits for the difference of cdfs has been also given [52].

We are not aware, however, of any non-trivial limit for the difference of pdfs as a function of

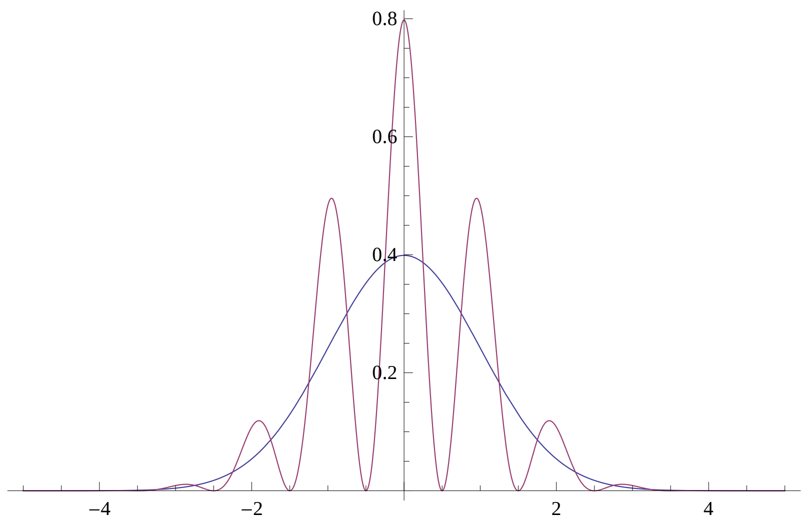


Figure 5.2: Probability density functions $f_1(x)$ and $f_2(x)$.

identical moments.

One might wonder if the phenomenon that the TVD of the approximation of a distribution does not seem to converge to 0 is not a contradiction if the distribution has moment-generating function, hence it is surely uniquely determined by its moments [10, 7]. This contradiction, however, is apparent. While it is true that an infinite number of moments uniquely determines the distribution, this is not a "convergent series expansion", i.e. there is no limit whatsoever on the error if we truncate the series at some point. Also, the matching of the moments can only be performed numerically in practice (and, for $k > 4$, even in theory) which leads us back to the problems exposed in connection with the McCullagh-example.

5.2 Comparion with mixture normal distributions

Therefore, the question is simply whether a given test's properties depend mostly only on the first four moments of the population (which would mean that the different pdfs do not pose a problem), or these properties are also effected by information in the distribution beyond its first four moments.

Let us notice that this question can be again investigated empirically! We can choose a "reference" distribution (from some known family) and perform a usual robustness-analysis using this family. Then, we can generate Fleishman distribution with the same first four moments as this reference distribution, and repeat the analysis. By comparing the results, we can decide whether the inclusion of the first four moments "was enough".

Due to the reasons outlined above (i.e. their significance in biostatistics), we chose mixture normal distributions as a reference distribution. We have in fact done this investigation for many mixture normal distributions – the results are not shown here in detail to space considerations, but are available at [53, 11]. These results suggest that for simple cases, like the robustness analysis of the parametric t test, there is no major difference between the results obtained with mixture normal and moment-matched Fleishman distribution, except for extremely small sample sizes. However, the contrary is true for the power testing of the non-parametric Mann-Whitney U-test: there can be major differences in the calculated power between the two approaches.

The explanation for this phenomenon is quickly understandable if we think of the most important approach to analytical robustness investigation of the t-test: the different series expansions.

They are all based on some approximation of the distribution which includes only the first few moments. Should they provide a good approximation with only four moments (taking the operation of the t-test into account), we can expect that the behavior of the t-test is also basically determined by these moments, hence it will similar for even visible different distributions, provided that they have identical first four moments.

This is, however, not true for the power-analysis of the Mann-Whitney U-test for which no such series expansion is available. Hence, it is no surprise that the results can be vastly different between mixture normal and Fleishman, even though they have identical first four moments.

These warn us to be careful when we want to generalize the results obtained with Fleishman distribution: while they might be very useful *in general*, their representativeness might also be questioned, for example in biostatistics. In such situations the best (if possible) is perhaps to use distributions that actually came from that field (previous studies etc.).

Chapter 6

Conclusion

The investigation of the properties of statistical tests is of crucial importance in modern statistics. This is underlined by the significance of these tests on the application areas, and the myriad of situations when their properties become important. (Especially as this is often overlooked by casual users of statistics.) Can we apply the t-test even if we are not completely sure about population normality? What power can we expect from the Mann Whitney U-test for a given background distribution? Such question often crop up in many areas of applied statistics, such as biostatistics.

Testing the robustness of parametric tests, and the power of non-parametric tests poses a special problem, as they require the application of a *per definitionem* non-specified distribution. Even if we agree that this means a distribution with given skewness and kurtosis, the problem is far from being solved.

We introduced a distribution called Fleishman distribution in this thesis that is able to meet this expectation: it can take (almost) arbitrary skewness and kurtosis with proper parameterizing. We discussed many aspects of this distribution in detail.

Using the Fleishman distribution one can perform an empirical investigation of a statistical test through Monte Carlo simulation. However, its computational burden still poses a problem even nowadays. To solve this issue, we successfully applied a novel computational method, called General Purpose GPU computing to this problem. This approach harnesses the incredible capacity of modern video cards that can be used for highly parallel, computation-intensive problems like this empirical investigation of statistical tests. We developed a computer environment that is capable to perform more than 70 million t-tests per second ($n = 10$) even on low-end video cards. The environment is modular and highly flexible, so that it facilitates further extension.

As a demonstration, we performed investigations about many routinely used statistical tests with this environment. We also pointed out one limitation of this approach, we should warn us to use the results of such investigation with care.

Overall, this thesis presented a novel, flexible, high-performance solution to empirically investigate properties of statistical tests based on the Fleishman distribution and demonstrated its applicability in practice.

There are, of course, many ways on which this work can be further developed. While the tests currently included in the environment are the practically most important ones, there are dozens, if not hundreds, that could be still implemented. This way, a more general framework for the research of statistical tests could be developed.

In addition to that, many properties are not yet investigated: the effect of homogeneity of variances for the t-test, the effect of different population distributions for the two-sample and k-sample tests, and so on. By including these in the environment, it would gain further power for such studies. It might as well facilitate the development of new statistical tests, or the correction of the already used ones.

Finally, our environment may also be used in education, as it can spectacularly illustrate basic statistical principles, like the effect of Central Limit Theorem in practice.

Appendix A

Wolfram Mathematica source codes

A.1 Coverage of important distributions

```
Skewness[LogNormalDistribution[\[Mu], \[Sigma]]]
Kurtosis[LogNormalDistribution[\[Mu], \[Sigma]]]
LN = ParametricPlot[{Sqrt[-1 + E^\[Sigma]^2] (2 + E^\[Sigma]^2), -3 +
     3 E^(2 \[Sigma]^2) + 2 E^(3 \[Sigma]^2) + E^(
     4 \[Sigma]^2)}, {\[Sigma], 0.00001, 2}, AxesOrigin -> {0, 0},
  PlotRange -> {{-0.1, 4}, {0, 50}}, AspectRatio -> 1,
  PlotStyle -> Cyan]
Skewness[ParetoDistribution[k, \[Alpha]]]
Kurtosis[ParetoDistribution[k, \[Alpha]]]
Par = ParametricPlot[{(
    2 Sqrt[(-2 + \[Alpha])/\[Alpha]] (1 + \[Alpha]))/(-3 + \[Alpha]), (
    3 (-2 + \[Alpha]) (2 + \[Alpha] +
       3 \[Alpha]^2))/((-4 + \[Alpha]) (-3 + \[Alpha]) \[Alpha])}, {\
\[Alpha], 4.01, 1000}, AxesOrigin -> {0, 0},
  PlotRange -> {{-0.1, 4}, {0, 50}}, PlotStyle -> Brown,
  AspectRatio -> 1]
Show[{LN, Par,
  ListPlot[{{{Skewness[UniformDistribution[]],
       Kurtosis[UniformDistribution[]]}}, {{Skewness[
        NormalDistribution[0, 1]],
       Kurtosis[NormalDistribution[0, 1]]}}, {{Skewness[
        ExponentialDistribution[1]],
       Kurtosis[ExponentialDistribution[1]]}}},
   PlotMarkers -> {Automatic, Medium},
   PlotStyle -> {Purple, Green, Blue}],
  Plot[x^2 + 1, {x, 0, 4}, PlotStyle -> Red, Filling -> Bottom,
   FillingStyle -> Directive[Opacity[0.2], Red]]}, AspectRatio -> 1,
 AxesLabel -> {"\!\(\*SubscriptBox[\(\[Gamma]\), \(3\)]\)",
   "\!\(\*SubscriptBox[\(\[Gamma]\), \(4\)]\)"}]
```

A.2 Power of the z-test

```
TestPower[\[Mu]_, \[Alpha]_] :=
 CDF[NormalDistribution[\[Mu] - 175, 1],
   Quantile[NormalDistribution[0, 1], \[Alpha]/2]] + (1 -
    CDF[NormalDistribution[\[Mu] - 175, 1],
     Quantile[NormalDistribution[0, 1], 1 - \[Alpha]/2]])
Plot[TestPower[\[Mu], 0.05], {\[Mu], 170, 180},
 Epilog -> {Dashed, Line[{{170, 0.05}, {180, 0.05}}]},
 AxesLabel -> {"\[Mu]", "Power (1-\[Beta])"}]
TestPowerMC[\[Mu]_, \[Alpha]_,
  n_] := (n -
    Count[RandomVariate[NormalDistribution[\[Mu] - 175, 1], n],
      x_ /; x > Quantile[NormalDistribution[0, 1], \[Alpha]/2] &&
        x < Quantile[NormalDistribution[0, 1], 1 - \[Alpha]/2]])/n
p100 = Plot[TestPowerMC[\[Mu], 0.05, 100], {\[Mu], 170, 180},
  Epilog -> {Dashed, Line[{{170, 0.05}, {180, 0.05}}]},
  AxesLabel -> {"\[Mu]", "Power (1-\[Beta])"}, PlotPoints -> 2000,
  MaxRecursion -> 0, PlotLabel -> "R=100"]
```

A.3 Robustness of the z-test

```
distNorm = NormalDistribution[\[Mu]0, \[Sigma]0]
distsumNorm = NormalDistribution[n*\[Mu]0, Sqrt[n]*\[Sigma]0]
distGamma = GammaDistribution[a, b, 1, 0]
distsumGamma = GammaDistribution[n*a, b, 1, 0]
TestStatNorm =
 TransformedDistribution[(1/n*X -
     Mean[distNorm])/(StandardDeviation[distNorm]/Sqrt[n]),
  X \[Distributed] distsumNorm]
TestStatGamma =
 TransformedDistribution[(1/n*X -
     Mean[distGamma])/(StandardDeviation[distGamma]/Sqrt[n]),
  X \[Distributed] distsumGamma]
ActualAlpha[dist_, \[Alpha]_] :=
 CDF[dist,
   Quantile[NormalDistribution[0, 1], \[Alpha]/2]] + (1 -
    CDF[dist, Quantile[NormalDistribution[0, 1], 1 - \[Alpha]/2]])
ActualAlpha[TestStatNorm, \[Alpha]]
ActualAlpha[TestStatGamma, \[Alpha]]
Plot3D[ActualAlpha[TestStatGamma, \[Alpha]] /. n -> 10 /. \[Alpha] ->
    0.05, {a, 0.1, 3}, {b, 0.1, 3},
 AxesLabel -> {"a", "b", "Actual \[Alpha]"}]
Plot[ActualAlpha[TestStatGamma, \[Alpha]] /. n -> 10 /. \[Alpha] ->
    0.05 /. b -> 2, {a, 0.1, 3},
 AxesLabel -> {"a", "Actual \[Alpha]"},
 Epilog -> {Dashed, Line[{{0, 0.05}, {3, 0.05}}]},
 AxesOrigin -> {0, 0.0425}]
RobustnessGammaMC[a_] :=
 Count[Table[(1/10*Total[RandomVariate[GammaDistribution[a, 2], 10]] -
        Mean[GammaDistribution[a, 2]])/(StandardDeviation[
        GammaDistribution[a, 2]]/Sqrt[10]), {10000}],
    x_ /; x < Quantile[NormalDistribution[0, 1], 0.05/2] ||
      x > Quantile[NormalDistribution[0, 1], 1 - 0.05/2]]/10000
Plot[RobustnessGammaMC[a], {a, 0.1, 3}, PlotPoints -> 100,
 MaxRecursion -> 0, AxesLabel -> {"a", "Actual \[Alpha]"},
 Epilog -> {Dashed, Line[{{0, 0.05}, {3, 0.05}}]}]
```

A.4 Coverage of $\mathcal{FD}(4)$

```
FD4[a_, b_, c_, d_] := TransformedDistribution[a + b*Z + c*Z^2 + d*Z^3, Z \[Distributed]
    NormalDistribution[0, 1]]
Table[Moment[FD4[a, b, c, d], p], {p, 1, 4}]
FD4Coefficients[g3_, g4_] := (sol = NSolve[Table[Moment[FD4[a, b, c, d], p], {p, 1, 4}]
    == {0, 1, g3, g4}, {a, b, c, d}, Reals]; If[Length[sol] == 0, {}, sol[[1]]])
FleishmanCoverage = RegionPlot[FD4Coefficients[g3, g4] != {}, {g3, 0, 3}, {g4, 0, 20}]
Show[{FleishmanCoverage, Plot[g3^2 + 1, {g3, 0, 3}]}]
```

Appendix B

CUDA source codes

B.1 Declarations

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>

#include <stdlib.h>
#include <time.h>
#include <math.h>
#include "gsl_vector.h"
#include "gsl_multiroots.h"

#define SAMPLESIZE 10
#define THREADSPERBLOCK 32
#define ITERATIONSPERRUN 100

#define  DCMT_SEED 4172
#define  MT_RNG_PERIOD 607

typedef struct {
    unsigned int matrix_a;
    unsigned int mask_b;
    unsigned int mask_c;
    unsigned int seed;
} mt_struct_stripped;

#define PI 3.14159265358979f
#define MT_RNG_COUNT 32768
#define          MT_MM 9
#define          MT_NN 19
#define       MT_WMASK 0xFFFFFFFFU
#define       MT_UMASK 0xFFFFFFFEU
#define       MT_LMASK 0x1U
#define      MT_SHIFT0 12
#define      MT_SHIFTB 7
#define      MT_SHIFTC 15
#define      MT_SHIFT1 18

__device__ static mt_struct_stripped ds_MT[MT_RNG_COUNT];
static mt_struct_stripped h_MT[MT_RNG_COUNT];
//const unsigned int SEED = 777; //kiiktatva ld. void seedMTGPU()

struct fgvparams
{
        float skew;
        float kurt;
};

struct transformparams
```

```
{
        float a;
        float b;
        float c;
        float d;
};
```

B.2 Mersenne twister random number generation

```
void loadMTGPU(const char *fname){
    FILE *fd = fopen(fname, "rb");
    if(!fd){
        printf("initMTGPU(): failed to open %s\n", fname);
        printf("TEST FAILED\n");
        exit(0);
    }
    if( !fread(h_MT, sizeof(h_MT), 1, fd) ){
        printf("initMTGPU(): failed to load %s\n", fname);
        printf("TEST FAILED\n");
        exit(0);
    }
    fclose(fd);
}

unsigned long hash( time_t t, clock_t c )
{
        // Get a uint32 from t and c
        // Better than uint32(x) in case x is floating point in [0,1]
        // Based on code by Lawrence Kirby (fred@genesis.demon.co.uk)

        static unsigned long differ = 0;  // guarantee time-based seeds will change

        unsigned long h1 = 0;
        unsigned char *p = (unsigned char *) &t;
        for( size_t i = 0; i < sizeof(t); ++i )
        {
                h1 *= UCHAR_MAX + 2U;
                h1 += p[i];
        }
        unsigned long h2 = 0;
        p = (unsigned char *) &c;
        for( size_t j = 0; j < sizeof(c); ++j )
        {
                h2 *= UCHAR_MAX + 2U;
                h2 += p[j];
        }
        return ( h1 + differ++ ) ^ h2;
}

void seedMTGPU() {
    int i;
    //Need to be thread-safe
        mt_struct_stripped *MT = (mt_struct_stripped *)malloc(MT_RNG_COUNT * sizeof(
            mt_struct_stripped));

    for(i = 0; i < MT_RNG_COUNT; i++){
        MT[i]       = h_MT[i];
        MT[i].seed =  hash( time(NULL), clock() );
        //MT[i].seed =  777; //csak teszteleshez, hogy ugyanazok legyenek a generalt
            szamok
    }
        cudaMemcpyToSymbol(ds_MT, MT, sizeof(h_MT));

    free(MT);
}
```

B.3 Statistical tests

```
__device__ int OneSampleZTest(float* Sample)
{
        float sum=0;
        for(int i=0;i<SAMPLESIZE;i++)
                sum+=Sample[i];
        if (abs((sum/SAMPLESIZE)*sqrt((float)SAMPLESIZE))>1.95996)
                return 1;
        else
                return 0;
}
```

B.4 Kernel

```
__global__ void RobustnessKernel(float* d_a, struct transformparams* params)
{
        const int UniqueBlockIndex = blockIdx.y * gridDim.x + blockIdx.x;
        const int UniqueThreadIndex = UniqueBlockIndex * blockDim.x + threadIdx.x;

        __shared__ int BlockSum[THREADSPERBLOCK];
        if ( threadIdx.x == 0 ) {
                for(int i=0;i<THREADSPERBLOCK;i++)
                        BlockSum[i]=0;
        }
        __syncthreads();

        float Sample[SAMPLESIZE];

        int iState, iState1, iStateM;
        unsigned int mti, mti1, mtiM, x;
        float x1,x2;
        unsigned int mt[MT_NN];

        float r,phi;

        //Load bit-vector Mersenne Twister parameters
        mt_struct_stripped config = ds_MT[UniqueThreadIndex];

        //Initialize current state
        mt[0] = config.seed;
        for(iState = 1; iState < MT_NN; iState++)
                mt[iState] = (1812433253U * (mt[iState - 1] ^ (mt[iState - 1] >> 30)) +
                    iState) & MT_WMASK;

        iState = 0;
        mti1 = mt[0];

        for(int iter=0;iter<ITERATIONSPERRUN;iter++) {
                for(int i=0;i<SAMPLESIZE/2;i++) {
                        iState1 = iState + 1;
                        iStateM = iState + MT_MM;
                        if(iState1 >= MT_NN) iState1 -= MT_NN;
                        if(iStateM >= MT_NN) iStateM -= MT_NN;
                        mti  = mti1;
                        mti1 = mt[iState1];
                        mtiM = mt[iStateM];

                        x    = (mti & MT_UMASK) | (mti1 & MT_LMASK);
                        x    =  mtiM ^ (x >> 1) ^ ((x & 1) ? config.matrix_a : 0);
                        mt[iState] = x;
                        iState = iState1;

                        //Tempering transformation
                        x ^= (x >> MT_SHIFT0);
                        x ^= (x << MT_SHIFTB) & config.mask_b;
                        x ^= (x << MT_SHIFTC) & config.mask_c;
                        x ^= (x >> MT_SHIFT1);

                        x1=((float)x + 1.0f) / 4294967296.0f;

                        iState1 = iState + 1;
                        iStateM = iState + MT_MM;
                        if(iState1 >= MT_NN) iState1 -= MT_NN;
                        if(iStateM >= MT_NN) iStateM -= MT_NN;
                        mti  = mti1;
                        mti1 = mt[iState1];
                        mtiM = mt[iStateM];

                        x    = (mti & MT_UMASK) | (mti1 & MT_LMASK);
                        x    =  mtiM ^ (x >> 1) ^ ((x & 1) ? config.matrix_a : 0);
                        mt[iState] = x;
                        iState = iState1;

                        //Tempering transformation
                        x ^= (x >> MT_SHIFT0);
                        x ^= (x << MT_SHIFTB) & config.mask_b;
                        x ^= (x << MT_SHIFTC) & config.mask_c;
                        x ^= (x >> MT_SHIFT1);

                        x2=((float)x + 1.0f) / 4294967296.0f;

                        r = sqrtf(-2.0f * logf(x1));
                        phi = 2 * PI * x2;
                        x1 = r * __cosf(phi);
                        x2 = r * __sinf(phi);

                        x1=params[UniqueBlockIndex].a+params[UniqueBlockIndex].b*x1+
                            params[UniqueBlockIndex].c*x1*x1+params[UniqueBlockIndex].d*
                            x1*x1*x1;
                        x2=params[UniqueBlockIndex].a+params[UniqueBlockIndex].b*x2+
                            params[UniqueBlockIndex].c*x2*x2+params[UniqueBlockIndex].d*
                            x2*x2*x2;

                        Sample[i*2]=x1;
                        Sample[i*2+1]=x2;
                }
                BlockSum[threadIdx.x]+=OneSampleZTest(Sample);
        }

        __syncthreads();
        if ( threadIdx.x == 0 )
                for(int i=0;i<THREADSPERBLOCK;i++)
                        d_a[UniqueBlockIndex]+=BlockSum[i];
}
```

B.5 Solving the Fleishman equations

```
int egyenletek(const gsl_vector* x, void* params, gsl_vector* f)
{
        const double skew=((struct fgvparams*) params) -> skew;
        const double kurt=((struct fgvparams*) params) -> kurt;
        const double a = gsl_vector_get(x,0);
        const double b = gsl_vector_get(x,1);
        const double c = gsl_vector_get(x,2);
        const double d = gsl_vector_get(x,3);
        //printf("szamok: %f, %f, %f, %f",a,b,c,d);

        const double y0=a+c;
        const double y1=b*b+6*b*d+15*d*d+2*c*c-1;
        const double y2=2*c*(b*b+24*b*d+105*d*d+2)-skew;
        const double y3=24*(b*d+c*c*(1+b*b+28*b*d)+d*d*(12+48*b*d+141*c*c+225*d*d))-(kurt
            -3);

        gsl_vector_set(f,0,y0);
        gsl_vector_set(f,1,y1);
        gsl_vector_set(f,2,y2);
        gsl_vector_set(f,3,y3);

        return GSL_SUCCESS;
}

struct transformparams solveeqns(struct fgvparams params)
{
        const gsl_multiroot_fsolver_type* T;
    gsl_multiroot_fsolver* s;
    size_t n=4;
    size_t iter=0;
    int state;
    gsl_vector* x=gsl_vector_alloc(n);
    gsl_vector_set(x,0,0.0);
    gsl_vector_set(x,1,1.0);
    gsl_vector_set(x,2,0.5);
    gsl_vector_set(x,3,0.5);
    gsl_multiroot_function F;

    F.f=&egyenletek;
    F.n=4;
    F.params=&params;
    T=gsl_multiroot_fsolver_dnewton;
    s=gsl_multiroot_fsolver_alloc(T,n);
    gsl_multiroot_fsolver_set(s,&F,x);

    do {
                iter++;
                state=gsl_multiroot_fsolver_iterate(s);

                if (state)
                        break;

                state=gsl_multiroot_test_residual(s->f,1e-5);
        } while (state==GSL_CONTINUE&&iter<1000);

        gsl_vector* root=gsl_multiroot_fsolver_root(s);
        transformparams sol;
        sol.a=gsl_vector_get(root,0);
        sol.b=gsl_vector_get(root,1);
        sol.c=gsl_vector_get(root,2);
        sol.d=gsl_vector_get(root,3);

        //sol.d=sol.d/sol.c;
        //sol.c=sol.c/sol.b;

        gsl_multiroot_fsolver_free(s);
        gsl_vector_free(x);

        if (iter==1000) {
                sol.a=0;
                sol.b=1;
                sol.c=0;
                sol.d=0;
                //printf("baj van!\n");
                return sol;
        }

        return sol;
}
```

B.6 Main program

```
int main()
{
    float *h_a;
    float *d_a;

    FILE* out = fopen( "output.txt", "w" );
    FILE* ParamsOutFile = fopen( "paramsoutfile.txt", "w" );

    clock_t timePre,timePost;
```

```

const int SkewGridIterations = 32;
    const float SkewGridMax = 4.0;
const float SkewGridStepSize = SkewGridMax / SkewGridIterations;
const int KurtGridIterations = 32;
    const float KurtGridMax = 20.0;
const float KurtGridStepSize = KurtGridMax / KurtGridIterations;
    const long MCIterations = 10;

struct fgvparams params;
    struct transformparams *sol,*d_sol;

    float skewness;

    sol=(struct transformparams*)malloc(SkewGridIterations*KurtGridIterations*sizeof(struct transformparams));

    printf("***************************************************************\n");
    printf("***␣␣␣␣␣␣␣␣␣␣␣MONTE␣CARLO␣SIMULATIONAL␣ENVIRONMENT␣␣␣␣␣␣␣␣␣␣***\n");
    printf("***␣␣␣FOR␣THE␣EMPIRICAL␣INVESTIGATION␣OF␣STATISTICAL␣TESTS␣␣␣***\n");
    printf("***************************************************************\n");
    printf("\n\n");
    printf("Written␣by:␣Tamas␣Ferenci\n");
    printf("\t␣E−mail:␣tamas.ferenci@medstat.hu\n");

    printf("\n\n");
    printf("\n***␣Skewness−kurtosis␣grid␣parameters␣*************************\n");
    printf("\t␣Skewness:␣from␣%2.1f␣to␣%2.1f␣in␣%i␣steps␣(step␣size:␣%2.2f)\n",0.0,SkewGridMax,SkewGridIterations,SkewGridStepSize);
    printf("\t␣Kurtosis:␣from␣%2.1f␣to␣%2.1f␣in␣%i␣steps␣(step␣size:␣%2.2f)\n",0.0,KurtGridMax,KurtGridIterations,KurtGridStepSize);
    printf("***************************************************************\n");


    printf("\n\n");
    printf("***␣Solving␣the␣Fleishman−equations␣for␣the␣above␣grid␣*********\n");
printf("\t␣Starting␣to␣solve␣the␣necessary␣equations...\n");
    for(int i = 0; i < SkewGridIterations; i++) {
        for(int j = 0; j < KurtGridIterations; j++)
        {
            skewness = i * SkewGridStepSize;
            params.skew = skewness;
            params.kurt = ( 2 * skewness * skewness + 3 ) + j * KurtGridStepSize;
            sol[ i * KurtGridIterations + j ] = solveeqns( params );
        }
    }
    printf("\t␣...The␣solving␣of␣necessary␣equations␣finished.\n");
    printf("\t␣Starting␣to␣write␣coefficient␣to␣storage...\n");
    for(int i=0;i<SkewGridIterations;i++)
        for(int j=0;j<KurtGridIterations;j++)
            fprintf(ParamsOutFile,"%f;%f;%f;%f\n",sol[i*KurtGridIterations+j].a,sol[i*KurtGridIterations+j].b,sol[i*KurtGridIterations+j].c,sol[i*KurtGridIterations+j].d);
    fclose(ParamsOutFile);
    printf("\t␣...The␣writing␣of␣coefficients␣to␣storage␣finished.\n");
    cudaMalloc((void**)&d_sol,SkewGridIterations*KurtGridIterations*sizeof(struct transformparams));
    cudaMemcpy(d_sol,sol,SkewGridIterations*KurtGridIterations*sizeof(struct transformparams),cudaMemcpyHostToDevice);
    printf("***************************************************************\n");

    loadMTGPU("MersenneTwister.dat");
seedMTGPU();

cudaThreadSynchronize();

size_t memSize = SkewGridIterations * KurtGridIterations * sizeof(long);
h_a = (float *) malloc(memSize);
cudaMalloc( (void **) &d_a, memSize );
```

```
cudaMemset(d_a,0,memSize);

    printf("\n\n");
    printf("\n***␣Iteration␣parameters␣****************************************\n");
    printf("\t␣Threads␣per␣block:␣%i\n", THREADSPERBLOCK);
    printf("\t␣Iterations␣within␣single␣kernel␣run:␣%i\n", ITERATIONSPERRUN);
    printf("\t␣Global␣iterations:␣%i\n", MCIterations);
    printf("\t␣Number␣of␣total␣simulational␣replications:␣%i\n", THREADSPERBLOCK * ITERATIONSPERRUN * MCIterations);
    printf("****************************************************************\n");

    printf("\n\n");
    printf("\n***␣Investigation␣parameters␣************************************\n");
    printf("\t␣Investigated␣test:␣One-sample,␣two-sided␣z-test\n");
    printf("\t␣Type␣of␣investigation:␣Robustness␣w.r.t.␣non-normality\n");
    printf("\t␣Sample␣size:␣%i\n", SAMPLESIZE);
    printf("****************************************************************\n");

    printf("\n\n");
    printf("***␣GPU␣information,␣Device␣0␣************************************\n");
    cudaDeviceProp deviceProp;
cudaGetDeviceProperties(&deviceProp, 0);
printf("\t␣Name:␣%s\n", deviceProp.name);
    printf("\t␣Total␣global␣memory␣[GB]:␣%i\n", (int) deviceProp.totalGlobalMem/1000000);
    printf("\t␣Clock␣rate␣[MHz]:␣%i\n", deviceProp.clockRate/1000);
    printf("****************************************************************\n");

    printf("\n\n");
printf("***␣GPU-computation␣progress␣report␣****************************\n");
    printf("␣Starting␣GPU-computation...\n");
    dim3 numBlocks(SkewGridIterations,KurtGridIterations);
    dim3 threadsPerBlock(THREADSPERBLOCK);
    timePre = clock();
    for(int i=0;i<MCIterations;i++) {
        fflush(stdout);
        printf("\r␣Progress:␣%4.1f␣%%", (float)(i)/MCIterations*100);
        printf("␣Elapsed␣time:␣%i␣s",(clock()-timePre)/CLOCKS_PER_SEC);
        if (i>0)
            printf("␣Estimated␣total␣time:␣%i␣s",(clock()-timePre)/CLOCKS_PER_SEC*MCIterations/i);
        seedMTGPU();
        RobustnessKernel<<< numBlocks, threadsPerBlock >>>( d_a, d_sol );
    }
    timePost = clock();
    printf("\n");
printf("␣...GPU-computation␣finished.\n");
    cudaMemcpy( h_a, d_a, memSize, cudaMemcpyDeviceToHost );
    printf("␣Starting␣to␣write␣results␣to␣storage...\n");
    fprintf(out, "%i\n", THREADSPERBLOCK * ITERATIONSPERRUN * MCIterations );
    fprintf(out, "%i\n", SkewGridIterations );
    fprintf(out, "%i\n", KurtGridIterations );
    fprintf(out, "%f\n", SkewGridMax );
    fprintf(out, "%f\n", KurtGridMax );
    fprintf(out, "%f\n", SkewGridStepSize );
    fprintf(out, "%f\n", KurtGridStepSize );
    fprintf(out, "%i\n", SAMPLESIZE );
    for (long i = 0; i < SkewGridIterations * KurtGridIterations; i++)
        fprintf(out,"%f\n",h_a[i] / ( THREADSPERBLOCK * ITERATIONSPERRUN * MCIterations ) );
    fclose(out);
    printf("␣...The␣writing␣of␣results␣to␣storage␣finished.\n");
    printf("****************************************************************\n");

cudaFree(d_a);
cudaFree(d_sol);
free(h_a);
free(sol);

    printf("\n\n");
    printf("*** Summary performance report **********************************\n");
        printf("\t Number of hypothesis testings: %i\n",  THREADSPERBLOCK *
            ITERATIONSPERRUN * MCIterations);
        printf("\t Number of random number generations: %i\n", THREADSPERBLOCK *
            ITERATIONSPERRUN * MCIterations * SAMPLESIZE);
        printf("\t Required time: %i s (%ih %im)\n",(timePost-timePre)/CLOCKS_PER_SEC, (
            int) (timePost-timePre)/CLOCKS_PER_SEC / 3600, (int) ((timePost-timePre)/
            CLOCKS_PER_SEC) % 3600 / 60 );
        printf("\t Speed: %.2f million hypothesis testing/sec\n", (float) THREADSPERBLOCK
             * ITERATIONSPERRUN * MCIterations * SkewGridIterations * KurtGridIterations
            /((timePost-timePre)/CLOCKS_PER_SEC) / 1000000);
        printf("****************************************************************\n");

        printf("\n\n");
        printf("*** End of program, press any key to exit! *********************");
        getchar();

        cudaDeviceReset();

    return 0;
}
```

Appendix C

R source codes

C.1 Visualization of the results

```r
RawResult <- read.table( "output.txt", dec = ".", sep = "," )
TotalReplications <- RawResult[ 1, ]
SkewGridIterations <- RawResult[ 2, ]
KurtGridIterations <- RawResult[ 3, ]
SkewGridMax <- RawResult[ 4, ]
KurtGridMax <- RawResult[ 5, ]
SkewGridStepSize <- RawResult[ 6, ]
KurtGridStepSize <- RawResult[ 7, ]
SampleSize <- RawResult[ 8, ]
RawResult <- RawResult[ -c( 1:8 ), ]

Result <- matrix( nrow = SkewGridIterations, ncol = KurtGridIterations )

for ( i in 0:( SkewGridIterations * KurtGridIterations - 1 ) )
        Result[ i %/% SkewGridIterations + 1, i %% KurtGridIterations + 1 ] <- RawResult[ i + 1 ]

persp( x = seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize),
       y = seq( from = 0, to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize),
       z = Result, xlab = "Skewness", ylab = "Kurtosis - ( 2 * Skewness^2 + 3)",
       zlab = "Type I Error Rate", main="Robustness of one-sample, two-sided t-test w.r.t. non-normality",
       sub = paste("n=",SampleSize,"; based on ",TotalReplications," Monte Carlo simulations", sep = "" ),
       col="blue", zlim = c(0.02,0.2),  phi = 20, theta = 10, axes = TRUE, ticktype = "detailed", nticks = 10)

filled.contour( x = seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize),
                y = seq( from = 0, to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize),
                z = Result, xlab = "Skewness", ylab = "Kurtosis - ( 2 * Skewness^2 + 3)",
                main = "Robustness of one-sample, two-sided t-test w.r.t. non-normality",
                color.palette = cm.colors, sub = paste("n=",SampleSize,"; based on ",TotalReplications,
                                                       " Monte Carlo simulations", sep = "" ),
                plot.axis = { axis( 1 ); axis( 2 );
                              contour(x = seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize),
                                      y = seq( from = 0,to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize),
                                      z = Result, add=TRUE, levels = 0.05, lwd = 2 ) } )

SkewGridForTrueKurt <- seq( from = 0, to = 4, length = 100 )
T1ErrorRateForTrueKurt <- function( TrueKurt ) {
  return( interp.surface( list( x = seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize),
                                y = seq( from = 0, to = KurtGridMax - KurtGridStepSize,
                                    by = KurtGridStepSize ),
                                z = Result ), loc = matrix( c( SkewGridForTrueKurt,
                                                              TrueKurt - 2 *
                                                                  SkewGridForTrueKurt^2
                                                                  - 3 ),
                                                           nc = 2, byrow = FALSE ) ) )
}

plot( seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize ), Result
    [ , 1 ], "l",
      xlab = "Skewness", ylab = "Type I Error Rate", ylim = c( 0.02, 0.2 ), col = 1,
      main = "Robustness of one-sample, two-sided t-test w.r.t. non-normality",
      sub = paste("n=",SampleSize,"; based on ",TotalReplications," Monte Carlo 
          simulations", sep = "" ) )
abline( h = 0.05, lty = "dashed" )
lines( seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize ),
    Result[ , 9 ], col = 2 )
lines( seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize ),
    Result[ , 17 ], col = 3 )
lines( seq( from = 0, to = SkewGridMax - SkewGridStepSize, by = SkewGridStepSize ),
    Result[ , 25 ], col = 4 )
legend( "topleft", legend = c( "...+0", "...+5", "...+10", "...+15" ), fill = 1:4,
        title = "Kurtosis: 2 * Skewness^2 + 3 + ..." )

plot( SkewGridForTrueKurt, T1ErrorRateForTrueKurt( 5 ),
      "l", xlab = "Skewness", ylab = "Type I Error Rate",
      main = "Robustness of one-sample, two-sided t-test w.r.t. non-normality", ylim = c(
           0.02, 0.2 ), col = 1,
      sub = paste("n=",SampleSize,"; based on ",TotalReplications," Monte Carlo 
          simulations", sep = "" ) )
abline( h = 0.05, lty = "dashed" )
lines( SkewGridForTrueKurt, T1ErrorRateForTrueKurt( 10 ), col = 2 )
lines( SkewGridForTrueKurt, T1ErrorRateForTrueKurt( 15 ), col = 3 )
lines( SkewGridForTrueKurt, T1ErrorRateForTrueKurt( 20 ), col = 4 )
legend( "topleft", legend = c( "5", "10", "15", "20" ), fill = 1:4, title = "Kurtosis" )

plot( seq( from = 0, to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize ), Result
    [ 1 , ], "l",
      xlab = "Kurtosis - ( 2 * Skewness^2 + 3)", ylab = "Type I Error Rate", ylim = c(
          0.02, 0.2 ), col = 1,
      main = "Robustness of one-sample, two-sided t-test w.r.t. non-normality",
      sub = paste("n=",SampleSize,"; based on ",TotalReplications," Monte Carlo 
          simulations", sep = "" ) )
abline( h = 0.05, lty = "dashed" )
lines( seq( from = 0, to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize ),
    Result[ 9 , ], col = 2 )
lines( seq( from = 0, to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize ),
    Result[ 17 , ], col = 3 )
lines( seq( from = 0, to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize ),
    Result[ 25 , ], col = 4 )
legend( "topleft", legend = c( "0", "1", "2", "3" ), fill = 1:4, title = "Skewness" )

plot( seq( from = 0, to = KurtGridMax - KurtGridStepSize, by = KurtGridStepSize ), Result
    [ 1 , ], "l",
      xlab = "Kurtosis", ylab = "Type I Error Rate", ylim = c( 0.02, 0.2 ), col = 1, xlim
           = c( 0, 40 ),
      main = "Robustness of one-sample, two-sided t-test w.r.t. non-normality",
      sub = paste("n=",SampleSize,"; based on ",TotalReplications," Monte Carlo 
          simulations", sep = "" ) )
abline( h = 0.05, lty = "dashed" )
lines( seq( from = 2 * 1^2 + 3, to = 2 * 1^2 + 3 + KurtGridMax - KurtGridStepSize, by =
    KurtGridStepSize ),
        Result[ 9 , ], col = 2 )
lines( seq( from = 2 * 2^2 + 3, to = 2 * 2^2 + 3 + KurtGridMax - KurtGridStepSize, by =
    KurtGridStepSize ),
        Result[ 17 , ], col = 3 )
lines( seq( from = 2 * 3^2 + 3, to = 2 * 3^2 + 3 + KurtGridMax - KurtGridStepSize, by =
    KurtGridStepSize ),
        Result[ 25 , ], col = 4 )
```

```r
legend( "topleft", legend = c( "0", "1", "2", "3" ), fill = 1:4, title = "Skewness" )
```

Bibliography

[1] M Abramowitz and I Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, New York, 1964.

[2] NI Akhiezer. *The classical moment problem: and some related questions in analysis.* Olivery & Boyd, 1965.

[3] MS Bartlett. The effect of non-normality on the t distribution. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31:223–231, 1935.

[4] NC Beaulieu and F Rajwani. Highly accurate simple closed-form approximations to lognormal sum distributions and densities. *Communications Letters, IEEE*, 8(12):709–711, 2004.

[5] JA Claridge and TC Fabian. History and development of evidence-based medicine. *World Journal of Surgery*, 29:547–553, 2005.

[6] WS Cleveland. *The elements of graphing data.* AT&T Bell Laboratories, 1994.

[7] CA Coelho, RP Alberto, and LM Grilo. When do the moments uniquely identify a distribution. Technical report, New University of Lisbon, 2005.

[8] L Devroye. *Non-Uniform Random Variate Generation.* Springer, 1st edition, 1986.

[9] R Eckhardt. Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science*, 15:131–136, 1987.

[10] W Feller. *An Introduction to Probability Theory and Its Applications.* Wiley, 3rd edition, 1968.

[11] T Ferenci. The Fleishman family of distributions and its application in the investigations of tests relevant in biostatistics [in Hungarian]. Presentation at Hungarian Clinical Biostatistical Society, 24 February 2012, 2012.

[12] A Fleishman. A method for simulating non-normal distributions. *Psychometrika*, 43(4):521–532, 1978.

[13] I Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[14] R Furrer, D Nychka, and S Sain. *fields: Tools for spatial data*, 2012. R package version 6.7.

[15] M Galassi, J Davies, J Theiler, B Gough, G Jungman, M Booth, and F Rossi. *Gnu Scientific Library: Reference Manual.* Network Theory Ltd., 2003.

[16] AK Gayen. The dsitribution of 'student's' t in random samples of any size drawn from non-normal universes. *Biometrika*, 36(3-4):353–369, 1949.

[17] RC Geary. The distribution of "student's" ratio for non-normal samples. *Supplement to the Journal of the Royal Statistical Society*, 3(2):178–184, 1936.

[18] RC Geary. Testing for normality. *Biometrika*, 34(3-4):209–242, 1947.

[19] GV Glass, PD Peckham, and JR Sanders. Consequences of failure to meet assumptions underlying the fixed effects analyses of variance and covariance. *Review of Educational Research*, 42(3):237–288, 1972.

[20] Khronos Group. Opencl - the open standard for parallel programming of heterogeneous systems. `http://www.khronos.org/opencl/`, 2012.

[21] A Hald. *A history of mathematical statistics from 1750 to 1930.* Wiley, 1998.

[22] MR Harwell, EN Rubinstein, WS Hayes, and CC Olds. Summarizing monte carlo results in methodological research: The one- and two-factor fixed effects anova cases. *Journal of Educational Statistics*, 17(4):315–339, 1992.

[23] T Headrick. Fast fifth-order polynomial transforms for generating univariate and multivariate nonnormal distributions. *Computational Statistics & Data Analysis*, 40(4):685–711, 2002.

[24] T Headrick. Simulating multivariate nonnormal distributions. *Journal of Modern Applied Statistical Methods*, 3:65–71, 2004.

[25] T Headrick. *Statistical Simulation: Power Method Polynomials and Other Transformations.* CRC Press, 1st edition, 2009.

[26] T Headrick and R Kowalchuk. The power method transformation: Its probability density function, distribution function, and its further use for fitting data. *Journal of Statistical Computation and Simulation*, 77(3):229–249, 2007.

[27] PJ Huber. *Robust Statistics.* Wiley Series in Probability and Statistics. Wiley-Interscience, 1981.

[28] N Johnson, S Kotz, and N Balakrishnan. *Continuous Univariate Distributions.* Wiley, 2nd edition, 1994.

[29] NJ Johnson. Modified t tests and confidence intervals for asymmetrical populations. *Journal of the American Statistical Association*, 73(363):536–544, 1978.

[30] C Kelley. *Iterative Methods for Linear and Nonlinear Equations.* Society for Industrial and Applied Mathematics, 1st edition, 1987.

[31] C Kelley. *Solving Nonlinear Equations with Newton's method.* Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[32] DP Kroese, T Taimre, and ZI Botev. *Handbook of Monte Carlo Methods (Wiley Series in Probability and Statistics).* Wiley, 1st edition, 2011.

[33] J Laderman. The distribution of "student's" ratio for samples of two items drawn from non-normal universes. *The Annals of Mathematical Statistics*, 10(4):376–379, 1939.

[34] BG Lindsay. On the determinants of moment matrices. *The Annals of Statistics*, 17(2):711–721, 1989.

[35] BG Lindsay and P Basak. Moments determine the tail of a distribution (but not much else). *The American Statistician*, 54(4):248–251, 2000.

[36] LM Lix, JC Keselman, and HJ Keselman. Consequences of assumption violations revisited: A quantitative review of alternatives to the one-way analysis of variance "f" test. *Review of Educational Research*, 66(4):579–619, 1996.

[37] A Luceno. Further evidence supporting the numerical usefulness of characteristic functions. *The American Statistician*, 51(3):233–234, 1997.

[38] M Matsumoto and T Nishimura. Dynamic creation of pseudorandom number generators. In *Proceedings of the Third International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 56–69, 1998.

[39] P McCullagh. Does the moment-generating function characterize a distribution? *The American Statistician*, 48(3):208–208, 1994.

[40] N Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, Special Issue, 1987.

[41] L Null and J Lobur. *The Essentials of Computer Organization And Architecture*. Jones and Bartlett Publishers, Inc., USA, 2012.

[42] NVIDIA. Cuda. `http://www.nvidia.com/object/cuda_home_new.html`, 2012.

[43] NVIDIA. *CUDA C Programming Guide*. NVIDIA Corporation, Santa Clara, CA, USA, 2012.

[44] NVIDIA. *NVIDIA GeForce GTX 680 Whitepaper*, 2012.

[45] NVIDIA. *Tesla Kepler Family Product Overview*, 2012.

[46] V Perlo. On the distribution of student's ratio for samples of three drawn from a rectangular distribution. *Biometrika*, 25(1-2):203–204, 193.

[47] MA Pett. *Nonparametric Statistics in Health Care Research: Statistics for Small Samples and Unusual Distributions*. SAGE, 1997.

[48] V Podlozhnyuk. *Parallel Mersenne Twister*. NVIDIA Corporation, 2007.

[49] L Powers and M Snell. *Microsoft Visual Studio 2008 Unleashed*. SAMS, Carmel, IN, USA, 1st edition, 2008.

[50] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[51] ST Rachev, SV Stoyanov, and FJ Fabozzi. *A Probability Metrics Approach to Financial Risk Measures*. Wiley-Blackwell, 2011.

[52] S Rácz, Á Tari, and M Telek. A moments based distribution bounding method. *Mathematical and Computer Modelling*, 43(1112):1367–1382, 2006.

[53] J Reiczigel. On the validity of power simulation based on Fleishman distributions. Poster at the 33rd Annual Conference of the International Society for Clinical Biostatistics, Bergen, Norway, 19-23 August 2012, 2012.

[54] PR Rider. On the distribution of the ratio of mean to standard deviation in small samples from non-normal universes. *Biometrika*, 21(1-4):124–143, 1929.

[55] L Rozsa, J Reiczigel, and G Majoros. Quantifying parasites in sample of hosts. *Journal of Parasitology*, 86(2):228–232, 2000.

[56] SS Sawilowsky. You think youve got trivials? *Journal of Modern Applied Statistical Methods*, 2(1):218–225, 2003.

[57] JA Shohat and JD Tamarkin. *The Problem of Moments*. American Mathematical Society, 1970.

[58] TOP500 Supercomputer Sites. Statistics / sublist generator. `http://www.top500.org/statistics/sublist/`, 2012.

[59] CD Sutton. Computer-intensive methods for tests about the mean of an asymmetrical distribution. *Journal of the American Statistical Association*, 88(423):802–810, 1993.

[60] L Szirmay-Kalos and L Szécsi. *General Purpose Computing on Graphics Processing Units*, pages 1451–1495. mondAt Kiadó, 2011.

[61] P Tadikamalla. On simulating non-normal distributions. *Psychometrika*, 45:273–279, 1980.

[62] ML Tiku and WY Tan. *Sampling distributions in terms of Laguerre polynomials with applications*. New Age International, New Delhi, 1999.

[63] C Vale and V Maurelli. Simulating multivariate nonnormal distributions. *Psychometrika*, 48:465–471, 1983.

[64] L Waller. Does the characteristic function numerically distinguish distribution? *The American Statistician*, 49(2):150–152, 1995.