Karlsruher Institut für Technologie

Bachelor Thesis

On data-dependent selections of the tuning parameter in certain goodness-of-fit tests for normality

Minh Hieu Nguyen
Matr.-Nr. 1917404

Date of submission

Supervisor: Dr. Bruno Ebner

Department of Mathematics

Karlsruhe Institute of Technology

Abstract

This bachelor thesis deals with goodness of fit tests for normal distribution and the automatic selection of the tuning parameter. Two certain classes of tests for normality based on the moment generating function are presented, both depending on a tuning parameter, in order to improve the performance of these location-scale invariant test statistics, the automatic selecting of the tuning parameter when the set of the tuning parameter is finite will be studied. These data-dependent methods will be applied on the proposed classes of tests for normal distribution. For the implementation, the statistical analysis and modeling are implemented in the statistical software package `R`.

Contents

1 Introduction

The normal distribution is well-known as the central probability distribution in statistics and many phenomena in nature have approximately normal distributions. Statistical research processes often require the data to be considered normally distributed. It is therefore routinely essential to determine whether an observed data set is suitable for the assumption of normality. The goodness of fit of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question.

Starting with some theoretical basics, the necessary mathematical foundations and notations of the normal distribution as well as the goodness-of-fit test are provided in section 2. With these at hand, in section 3 the Henze-Visagie test for normality in any dimension based on a partial differential equation involving the moment generating function and the Zghoul test for normality based on the empirical moment generating function are introduced. In section 4, a method for generating the critical values for these tests is discussed and its simulation is presented.

These tests rely on a specific parameter, known as the "tuning parameter", which is certainly the key reason why its selection is vital to acquire an efficient performing test procedure. Section 5 is devoted to the problem of data-dependent choice of the tuning parameter of these certain test classes, which is also applied for the Henze-Visagie and Zghoul tests. In section 6 the finite-sample power behavior of all mentioned tests for univariate normality will be simulated and the result will be briefly discussed.

2 Background

2.1 Normal distribution

A continuous random variable X has a normal (or Gaussian) distribution with parameters μ and $\sigma^2(\mu \in \mathbb{R}, \sigma > 0)$, denoted by $X \sim \mathcal{N}(\mu, \sigma^2)$, if its probability density function (PDF) is given by

$$\phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

The parameter μ is the "center" (or mean) of the distribution and σ is the "spread" (or standard deviation) of the distribution. Normal distribution is also known as the Bell curve for its external appearance. The Bell curve is a perfectly symmetrical distribution with skewness of zero and relatively short tails with kurtosis of three. We say that X has a standard normal distribution if $\mu = 0$ and $\sigma = 1$. The moment generating function(MGF), or Laplace transform, of X is defined by

$$M_X(t) = E(\exp(t^T X))$$

with $t \in \mathbb{R}$ for which the expectation exists. We have the MGF of standard normal distribution

$$\begin{aligned} M_x(t) = E(e^{tx}) &= \int_{\mathbb{R}} e^{tx} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \mathrm{d}x = \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}(x^2-2tx)} \mathrm{d}x \\ &= \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}((x-t)^2-t^2)} \mathrm{d}x = e^{\frac{t^2}{2}} \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}} e^{\frac{-(x-t)^2}{2}} \mathrm{d}x \\ &\text{let } u = x - t \text{ with } d_x = d_u \\ &= e^{\frac{t^2}{2}} \underbrace{\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}} e^{\frac{-u^2}{2}} \mathrm{d}u}_{=1} \text{ (PDF of } \mathcal{N}(0,1)) \\ &= e^{\frac{t^2}{2}}. \end{aligned}$$

A location–scale family is a family of probability distributions parametrized by a location parameter and a non-negative scale parameter. Let $f(x)$ be any pdf, then for any $\mu(-\infty < \mu < \infty)$ and any $\sigma > 0$, the family of pdfs

$$\frac{1}{\sigma} f(\frac{x-\mu}{\sigma}),$$

indexed by the parameter (μ, σ), is called the location-scale family with standard pdf $f(x)$; μ is called the location parameter and σ is called the scale parameter.

That means Y is a random variable with pdf $f_Y(y) = f(y|\mu, \sigma)$ (the location-scale family member) if and only if there exists another random variable X with $f_x(x) = f(x)$ (the standard member) such that $Y = \sigma X + \mu$ is a linear (location-scale) transformation of a standard random variable X. Therefore normal distribution is a location–scale families distribution, that means for $X \sim \mathcal{N}(0,1)$ and $Y = \mu X + \sigma$ with $\mu \in \mathbb{R}, \sigma > 0$, we have $Y \sim \mathcal{N}(\mu, \sigma)$.

2.2 Goodness-of-fit test

The goodness-of-fit test is usually used when judging whether or not the underlying population distribution, from which a sample is drawn, differs from a specific distribution. The method can be used for testing any specified distributions. In the present thesis, the problem of testing whether a population distribution is an normal distribution is discussed. Goodness-of-fit tests typically summarize the difference between observed values and expected values in the given model.

Given a sample $X = [X_1, ..., X_n]^T$ of independent and identically distributed real-valued random variables from a distribution function F, assume that

$$T_{n,\alpha} = T_{n,\alpha}(X_1, ..., X_n)$$

for $\alpha \in \Lambda$, is a finite family of statistic for testing the hypothesis

$$H_0 : F \in \mathcal{F}$$

against a general alternative hypothesis. In this thesis, we only study $\mathcal{F}$, a location-scale family

$$\mathcal{F} = \{F_0((. - \mu)/\sigma) : \mu \in \mathbb{R}, \sigma > 0\}$$

with F_0 a known distribution function on $\mathbb{R}$.

3 The certain classes of goodness-of-fit tests for univariate normal distribution

Let $X, X_1, X_2, ...$ be real-valued independent and identically distributed (iid.) random variables defined on an underlying probability space $(\Omega, \mathcal{A}, \mathbb{P})$. The problem of interest is to test the hypothesis

$$H_0 : \mathbb{P}^X \in \mathcal{N} = \{\mathcal{N}(\mu, \sigma^2) \mid (\mu, \sigma^2) \in \mathbb{R} \times (0, \infty)\},$$

against general alternatives.

3.1 The Henze Visagie test

The Henze Visagie test (N. Henze, J. Visagie, 2019) is based on a certain partial different equation class of test for multivariate normal distribution, but in this thesis only the case for univariate normal distribution is reviewed.

Suppose X is a centered random variable and assume that the moment generating function (MGF) $m(t) = \mathbb{E}[\exp(tX)]$ exists for each $t \in \mathbb{R}$ and satisfies the system of ordinary differential equations

$$\begin{cases} \frac{\partial}{\partial t} m(t) = t m(t) \quad \text{for} \quad t \in \mathbb{R}. \\ m(0) = 1. \end{cases} \tag{3.1}$$

The unique solution of this initial value problem is $m(t) = \exp(t^2/2), t \in \mathbb{R}$, which is the MGF of the standard normal distribution. To model the standardization assumption leading to (3.1), we consider the scaled residuals

$$Y_{n,j} = \frac{X_j - \overline{X_n}}{S_n} \text{ for } j = 1, ..., n,$$

where $\overline{X_n} = \frac{1}{n}\sum_{j=1}^n X_j$ is the mean and $S_n^2 = \frac{1}{n}\sum_{j=1}^n (X_j - \overline{X_n})$ is the sample variance. Denoting the estimation of the MGF, which can be accomplished using the empirical MGF by $M_n(t) = \frac{1}{n}\sum_{j=1}^n e^{tY_{n,j}}, t \in \mathbb{R}$ of $Y_{n,1}, ..., Y_{n,n}$ and to employ the weighted $\mathcal{L}^2$ -statistic, we have the test statistic

$$T_{n,\gamma} = n \int_{\mathbb{R}} (M_n'(t) - t M_n(t))^2 w_\gamma(t) \mathrm{d}t \tag{3.2}$$

where

$$w_\gamma(t) = \exp\left(-\gamma t^2\right),\ t \in \mathbb{R}.$$

Rejection of H_0 is for large values of $T_{n,\gamma}$ and γ is known as tuning parameter, the role of $\gamma > 0$ will be discussed in the next section.

We will implement the test statistic (3.2) in the statistical software `R` later. Therefore, it is necessary to have a computational form of the test statistic. Inserting these expressions

$$M_n'(t) = \frac{1}{n}\sum_{j=1}^{n} Y_{n,j} e^{tY_{n,j}}$$

and

$$\begin{aligned}\left(M_n'(t) - tM_n(t)\right)^2 &= \frac{1}{n^2}\left(\sum_{j=1}^{n} e^{tY_{n,j}}(Y_{n,j} - t)\right)^2 \\ &= \frac{1}{n^2}\sum_{j,k=1}^{n} e^{t(Y_{n,j}+Y_{n,k})}(Y_{n,j}Y_{n,k} - t(Y_{n,j} + Y_{n,k}) + t^2)\end{aligned}$$

into the test statistic (2.6), we obtain

$$T_{n,\gamma} := \frac{1}{n}(S_1 - S_2 + S_3) \tag{3.3}$$

where

$$\begin{aligned}S_1 &= \sum_{j,k=1}^{n} \int_{\mathbb{R}} e^{t(Y_{n,j}+Y_{n,k})} Y_{n,j}Y_{n,k} w_\gamma(t)\mathrm{d}t \\ S_2 &= \sum_{j,k=1}^{n} \int_{\mathbb{R}} e^{t(Y_{n,j}+Y_{n,k})} t(Y_{n,j} + Y_{n,k}) w_\gamma(t)\mathrm{d}t \\ S_3 &= \sum_{j,k=1}^{n} \int_{\mathbb{R}} e^{t(Y_{n,j}+Y_{n,k})} t^2 w_\gamma(t)\mathrm{d}t.\end{aligned}$$

The Gaussian integral

$$\int_{\mathbb{R}} e^{-x^2}\mathrm{d}x = \sqrt{\pi}$$

is used to compute these sums. In the flowing, we write $Y_{n,j,k}^{+} = Y_{n,j} + Y_{n,k}$, with this we obtain the expressions of S_1, S_2 and S_3, which are

$$S_1 = \sum_{j,k=1}^{n} Y_{n,j}Y_{n,k} \int_{\mathbb{R}} e^{tY_{n,j,k}^{+}} w_\gamma(t)\mathrm{d}t \text{ where } I_1 = \int_{\mathbb{R}} e^{tY_{n,j,k}^{+}} w_\gamma(t)\mathrm{d}t \tag{3.4}$$

let $u = \frac{2\gamma t - Y^+_{n,j,k}}{2\sqrt{\gamma}}$ we have $d_t = \frac{1}{\sqrt{\gamma}} d_u$

$$I_1 = \int_{\mathbb{R}} \exp\left(tY^+_{n,j,k} - \gamma t^2\right)\mathrm{d}t = \int_{\mathbb{R}} \exp\left(\frac{{Y^+_{n,j,k}}^2}{4\gamma} - (\sqrt{\gamma}t - \frac{Y^+_{n,j,k}}{2\sqrt{\gamma}})^2\right)\mathrm{d}t$$
$$= \frac{\sqrt{\pi} e^{{Y^+_{n,j,k}}^2/4\gamma}}{2\sqrt{\gamma}} \underbrace{\int_{\mathbb{R}} \frac{2e^{-u^2}}{\sqrt{\pi}} \mathrm{d}u}_{\text{the Gaussian integral}} = \sqrt{\frac{\pi}{\gamma}} \exp\left({Y^+_{n,j,k}}^2/4\gamma\right), \ (I)$$

$$S_2 = \sum_{j,k=1}^{n} \int_{\mathbb{R}} e^{tY^+_{n,j,k}} t Y^+_{n,j,k} w_\gamma(t)\mathrm{d}t \tag{3.5}$$

where

$$I_2 = \int_{\mathbb{R}} e^{tY^+_{n,j,k}} t Y^+_{n,j,k} w_\gamma(t)\mathrm{d}t = \int_{\mathbb{R}} \exp(tY^+_{n,j,k} - \gamma t^2) t Y^+_{n,j,k}\mathrm{d}t$$
$$= \frac{Y^+_{n,j,k}}{2\gamma} \underbrace{\int_{\mathbb{R}} (2\gamma t - Y^+_{n,j,k}) e^{Y^+_{n,j,k}t - \gamma t^2}\mathrm{d}t}_{=\int_{\mathbb{R}} e^{Y^+_{n,j,k}t-\gamma t^2}\mathrm{d}(2\gamma t - Y^+_{n,j,k}) = 0} + \frac{Y^+_{n,j,k}}{2\gamma} \int_{\mathbb{R}} e^{Y^+_{n,j,k}t - \gamma t^2}\mathrm{d}t_i$$
$$= \frac{\sqrt{\pi} {Y^+_{n,j,k}}^2 \exp\left({Y^+_{n,j,k}}^2/4\gamma\right)}{2\gamma^{3/2}}, \ (II)$$

$$S_3 = \sum_{j,k=1}^{n} \int_{\mathbb{R}} \exp(tY^+_{n,j,k} - \gamma t^2) t^2 \mathrm{d}t \tag{3.6}$$

where

$$I_3 = \int_{\mathbb{R}} \exp(tY^+_{n,j,k} - \gamma t^2) t^2 \mathrm{d}t = \int_{\mathbb{R}} t^2 [\exp\frac{{Y^+_{n,j,k}}^2}{4\gamma} - (\sqrt{\gamma}t - \frac{Y^+_{n,j,k}}{2\sqrt{\gamma}})^2]\mathrm{d}t$$
$$= e^{{Y^+_{n,j,k}}^2/4\gamma} \int_{\mathbb{R}} t^2 (\exp\left(-(\sqrt{\gamma}t) - \frac{Y^+_{n,j,k}}{2\sqrt{\gamma}}\right)^2)\mathrm{d}t$$
$$\text{let } u = 2\gamma t - Y^+_{n,j,k} \text{ with } t^2 = \frac{(u + Y^+_{n,j,k})^2}{4\gamma^2} \text{ and } d_x = \frac{d_u}{2\gamma}$$
$$= e^{{Y^+_{n,j,k}}^2/(4\gamma)} \int_{\mathbb{R}} \frac{1}{8\gamma^3} (u + Y^+_{n,j,k})^2 e^{\frac{-u^2}{4\gamma}}\mathrm{d}u$$
$$= \frac{e^{\frac{{Y^+_{n,j,k}}^2}{4\gamma}}}{8\gamma^3} [\underbrace{\int_{\mathbb{R}} (u^2 e^{\frac{-u^2}{4\gamma}})\mathrm{d}u}_{A} + \underbrace{\int_{\mathbb{R}} 2uY^+_{n,j,k} e^{\frac{-u^2}{4\gamma}}\mathrm{d}u}_{B} + \underbrace{\int_{\mathbb{R}} {Y^+_{n,j,k}}^2 e^{\frac{-u^2}{4\gamma}}\mathrm{d}u}_{C}]$$

$$A = \underbrace{(-2\gamma u e^{\frac{-u^2}{4\gamma}})|_{\mathbb{R}}}_{=0} + \int_{\mathbb{R}} 2\gamma e^{\frac{-u^2}{4\gamma}} \mathrm{d}u \text{ with } v = \frac{u}{2\gamma} \text{ we have } d_u = 2\sqrt{\gamma} d_v$$

$$= 2\sqrt{\pi}\gamma^{3/2} \underbrace{\int_{\mathbb{R}} \frac{2e^{-v^2}}{\sqrt{\pi}} \mathrm{d}v}_{\text{the Gaussian integral}} = 4\sqrt{\pi}\gamma^{3/2}$$

$$B = -4Y_{n,j,k}^{+}\gamma \int_{\mathbb{R}} e^{\frac{-u^2}{4\gamma}} \mathrm{d}(\frac{-u^2}{4\gamma}) = 0$$

$$C = {Y_{n,j,k}^{+}}^2 \sqrt{\gamma}\sqrt{\pi} \underbrace{\int_{\mathbb{R}} \frac{2e^{-v^2}}{\sqrt{\pi}} \mathrm{d}v}_{\text{the Gaussian integral}} \text{ with } v = \frac{u}{2\gamma} \text{ we have } d_u = 2\sqrt{\gamma} d_v$$

$$= 2\sqrt{\pi} {Y_{n,j,k}^{+}}^2 \sqrt{\gamma},$$

from (A)(B)(C) we have

$$I_3 = \frac{e^{{Y_{n,j,k}^{+}}^2/(4\gamma)}}{8\gamma^3}(4\sqrt{\pi}\gamma^{3/2} + 2\sqrt{\pi}{Y_{n,j,k}^{+}}^2\sqrt{\pi}) = \frac{\sqrt{\pi}(2\gamma + {Y_{n,j,k}^{+}}^2) e^{\frac{{Y_{n,j,k}^{+}}^2}{4\gamma}}}{4\gamma^{5/2}} \quad (III).$$

Inserting (I) (II) (III) into (3.4) (3.5) (3.6) respectively, we obtain the computational form of the test statistic

$$HV_{n,\gamma} = \frac{1}{n}(\frac{\pi}{\gamma})^{1/2} \sum_{j,k=1}^{n} \exp\left(\frac{(Y_{n,j,k}^{+})^2}{4\gamma}\right)\left(Y_{n,j}Y_{n,k} - \frac{(Y_{n,j,k}^{+})^2}{2\gamma} + \frac{1}{2\gamma} + \frac{(Y_{n,j,k}^{+})^2}{4\gamma^2}\right). \quad (3.7)$$

3.2 The Zghoul test of univariate normality

The test of Zghoul is based on the empirical moment generating function, which is a weighted $\mathcal{L}^2$ -statistic, given by

$$Z_{n,\gamma} = n \int_{\mathbb{R}} (M_n(t) - m(t))^2 w_\gamma(t) \mathrm{d}t, \quad (3.8)$$

where

$$w_\gamma(t) = \exp(-\gamma t^2),\ t \in \mathbb{R}.$$

Rejection of H_0 is for large values of $Z_{n,\gamma}$, see A. A. Zghoul (2010) and γ is also known as tuning parameter and the role of $\gamma > 0$ will be discussed in the next section. We will implement the test statistic $Z_{n,\gamma}$ in the statistical software `R` later. Therefore, it is necessary to have a computational form of the test statistic, which can be calculated as

$$Z_{n,\gamma} = n \int_{\mathbb{R}} (M_n(t) - m(t))^2 w_\gamma(t) \mathrm{d}t$$

$$= \underbrace{n \int_{\mathbb{R}} M_n^2(t) e^{-\gamma t^2} \mathrm{d}t}_{S_1} \underbrace{- 2n \int_{\mathbb{R}} M_n(t) m(t) e^{-\gamma t^2} \mathrm{d}t}_{S_2} + \underbrace{n \int_{\mathbb{R}} m^2(t) e^{-\gamma t^2} \mathrm{d}t}_{S_3}.$$

Using the expression (I) with $Y^+_{n,i,j} = Y_{n,i} + Y_{n,j}$

$$\int_{\mathbb{R}} \exp\left(tY^+_{n,j,k} - \gamma t^2\right)\mathrm{d}t = \sqrt{\frac{\pi}{\gamma}} \exp\left({Y^+_{n,j,k}}^2/4\gamma\right), t \in \mathbb{R},$$

we have

$$\begin{aligned}
S_1 &= n\int_{\mathbb{R}} M_n^2(t)e^{-\gamma t^2}\mathrm{d}t = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\int_{\mathbb{R}} \exp\left(tY^+_{n,j,k} - \gamma t^2\right)\mathrm{d}t \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{\frac{\pi}{\gamma}} \exp\left({Y^+_{n,j,k}}^2/4\gamma\right) \\
S_2 &= -2\sum_{i=1}^{n}\int_{\mathbb{R}} \exp\left(tY_{n,i} - (\gamma - \frac{1}{2})\right)\mathrm{d}t = -2\sum_{i=1}^{n}\sqrt{\frac{\pi}{(\frac{1}{2}-\gamma)}} \exp\frac{Y^2_{n,i}}{4(\gamma - \frac{1}{2})} \\
S_3 &= n\int_{\mathbb{R}} \exp\left(1-\gamma\right)t^2\mathrm{d}t = n\sqrt{\frac{\pi}{(\gamma-1)}}.
\end{aligned}$$

Inserting these results in the integral of the test statistic Zghoul, we obtain the computational form

$$Z_{n,\gamma} = \sqrt{\pi}\Big[\frac{n}{\sqrt{\gamma-1}} - \frac{2}{\sqrt{\gamma-\frac{1}{2}}}\sum_{i=1}^{n}\exp\frac{Y^2_{n,i}}{4(\gamma-\frac{1}{2})} + \frac{1}{n\sqrt{\gamma}}\sum_{i,j=1}^{n}\exp\left(Y^+_{n,i,j}\right)^2/4\gamma)\Big], \quad (3.9)$$

this test rejects normality for large values of the test statistic.

4 A method for generating critical values

As we have mentioned, $T_{n,\gamma}$ for $\gamma \in \Lambda$ the test statistic for the hypothesis H_0, whose large values are considered significant. In this case, the level of significance, which refers to the degree of significance in which we accept or reject the null-hypothesis, is α and the critical value $c_{n,\gamma}(\alpha)$ of our statistical test is the boundary of the acceptance region of the test. For our particular tests, rejection of H_0 is for large values of $T_{n,\gamma}$

$$T_{n,\gamma}(X_1, ..., X_n) > c_{n,\gamma}(\alpha)$$

where the significance level is

$$\alpha = P(Type\ I\ Error) = P(Reject\ H_0|H_0\ True) = P^{T_{n,\gamma}}(T_{n,\gamma}(X_1, ..., X_n) > c_{n,\gamma}(\alpha)),$$

so we have

$$c_{n,\gamma}(\alpha) = inf\{c \geq 0 : P^{T_{n,\gamma}}(T_{n,\gamma}(X_1, ..., X_n) > c\ |H_0\ True) \geq 1 - \alpha\} \quad (4.1)$$

which is the quantiles of order $(1-\alpha)$ of $T_{n,\gamma}$ under H_0.

Writing $F^{T_{n,\gamma}}$ for the distribution function of the test statistic $T_{n,\gamma}$ (continuous from the right, as usual) under H_0. We have $c_{n,\gamma}(\alpha)$ is the unique solution of the inequation

$$F^{T_{n,\gamma}}(t-) \leqslant 1-\alpha \leqslant F^{T_{n,\gamma}}(t) \ for\ t \in \mathbb{R}. \tag{4.2}$$

Under the null hypothesis, we generate many samples $(X_1^i, ..., X_n^i)$ for $i = 1, ..., m$ and $m \in \mathbb{N}$ is the considered number of Monte Carlo simulation, is used to estimate the upper percentiles of the distribution of $T_{n,\gamma}$ for the sample sizes n.

Corresponding to an ascending ordered sample $\{T_{n,\gamma}(X_1^j, ..., X_n^i), ..., T_{n,\gamma}(X_1^m, ..., X_n^m)\}$ of observations on $F^{T_{n,\gamma}}$. Let $0 < p < 1$, the sample p-th quantile is defined as the p-th quantile ξ_p of the sample distribution function $F^{T_{n,\gamma}}$. Regarding the sample p-th quantile as an estimator of ξ_p , we denote it by $\xi_{p,m}$

$$\xi_{p,m} := \begin{cases} \frac{1}{2}(T_{n,\gamma[mp]} + T_{n,\gamma[mp+1]}) \text{ for } mp \in \mathbb{N} \\ T_{n,\gamma[mp+1]} \text{ for } mp \notin \mathbb{N} \end{cases} \tag{4.3}$$

Theorem. Let $0 < p < 1$. If ξ_p is the unique solution x of $F^{T_{n,\gamma}}(x-) \leqslant p \leqslant F^{T_{n,\gamma}}(x)$ then $\xi_{p,m} \to \xi_p \ for\ m \to \infty$ almost surely. *Proof:* See R. J. Serfling (1980, S.75).

That means, the quantiles of order $(1-\alpha)$ of $T_{n,\gamma}$ under H_0 is the unique solution of the inequation, which is the critical value denoted by $c_{n,\gamma}(\alpha) = F^{-1}_{T_{n,\gamma}(1-\alpha)}$, can be approximated by performing Monte Carlo experiments under the null hypothesis.

$$\xi_{(1-\alpha),m} \to \xi_{(1-\alpha)} \text{ almost surely for } m \to \infty$$

The numerical results below were obtained using the software package `R`. Monte Carlo simulation was used in order to estimate the upper percentiles of the distribution of $HV_{n,\gamma}$ and $Z_{n,\gamma}$ for the sample sizes $n = 10, 20, 50$. Table 1 provides the empirical 95-percentiles of $(16\gamma^{2+1/2}/\pi^{1/2})HV_{n,\gamma}$ obtained by simulating 100.000 samples from $\mathcal{N}(0,1)$ distribution using $\gamma_{HV} = 2.5, 3, 4, 5, 7, 10$. Table 2 provides the empirical 95-percentiles of $1000Z_{n,\gamma}$ obtained by simulating 100.000 samples from $\mathcal{N}(0,1)$ distribution using $\gamma_Z = 1.5, 2, 3, 5, 9, 15$.

	γ					
n	2.5	3	4	5	7	10
10	59.850	58.220	53.591	50.083	47.745	44.979
20	119.115	104.389	88.294	79.485	70.131	65.184
50	219.382	173.079	131.732	110.610	92.291	82.927

Table 1: the empirical 95-percentiles of $(16\gamma^{2+1/2}/\pi^{1/2})HV_{n,\gamma}$

	γ					
n	1.5	2	3	5	9	15
10	1885.307	406.606	62.165	7.165	0.739	0.112
20	4882.133	895.159	110.043	11.207	1.051	0.155
50	12134.918	1675.366	170.144	14.987	1.335	0.189

Table 2: the empirical 95-percentiles of $1000Z_{n,\gamma}$

5 Data-dependent choice of the tuning parameter

As we see the tests statistics form (3.7) and (3.9) the parameter γ, is known as tuning parameter, plays a crucial role. In the last section, the method is to evaluate the test power performance for γ varying in a finite set of tuning parameter $\Lambda = \{\gamma_i : i = 1, ..., s\}$, and then suggesting a selection of γ that produces a test with a reasonable power against a wide range of alternative distributions. However, this strategy of taking a fixed tuning parameter does not prevent us from obtaining a test that achieves a very low power against some of the considered alternative distributions. In other words, without data-dependent choice for this parameter, a fixed value for the parameter is chosen which can perform well for some alternatives, but poorly for others.

Hence, in this section, the data-dependent methods for determining optimal tuning parameters will be studied to obtain a performing test procedure when the set of tuning parameter is finite.

5.1 An improvement of the classical Bonferroni multiple test procedure

For our particular tests, we assign a preset probability α of a type-1 error. Our experiment involves performing $|\Lambda| = s$ tests, we expect αs to be declared as significant if we use a value of α for each. This is the problem of multiple comparisons, in that we would like to control the false positive rate not just for any single test but also for the entire collection (or family) of tests that makes up our experiment. In this case, we perform $|\Lambda| = s$ independent tests, each with a preset type one error of α, so that the number of false positives follows from the Binomial distribution $Bin(s, \alpha)$, with α the probability of a "success" (a false positive) and s the number of trails. Hence, the probability of k such false positives is

$$P(k\ false\ positive) = \frac{s!}{(s-k)!k!}(1-\alpha)^{s-k}\alpha^k$$

for s large and α small, this is closely approximated by the Poisson distribution $Pois(s\alpha)$

$$\lim_{s\to\infty} P(k\ false\ positive) = \frac{(s\alpha)^k e^{-s\alpha}}{k!}$$

with the expected value (the expected number of false positives) is $s\alpha$. Bonferroni corrections (and their relatives) are the standard approach for controlling the experiment-wide false positive value α by specifying what u values should be used for each individual test (i.e., we declare a test to be significant if the $p - value \leq u$). The probability of not making any type I (false positive) errors in s independent tests, each of level u, is $(1-u)^s$. Hence, the probability of at least one false positive is just one minus this,

$$\alpha = 1 - (1-u)^s.$$

If we wish an experiment-wide false positive rate of α (i.e., the probability of one, or more, false positives over the entire set of tests is u), solving for the u value required for each test is

$$u = 1 - (1-\alpha)^{1/s}.$$

Noting that $(1-u)^s \approx 1 - su$, we obtain the Bonferroni method, taking

$$u = \alpha/s.$$

Our test is combining, based on the same available data and the test procedures associated with different values of the tuning parameter γ into a single test procedure that could show a good power performance against a wide range of alternative distribution. This was the case of the multiple test approach, which can be viewed as an improvement of the classical Bonferroni multiple test procedure. The proposed test leads to the rejection of the null hypothesis if one of the statistics $T_{n,\gamma}$ for $\gamma \in \Lambda$, is larger than its $(1-u)$ quantile under the null hypothesis, the level u being calibrated so that the resulting multiple test has a level of significance at most equal to α. Thus, the associated critical region is given by

$$\{\max_{\gamma \in \Lambda}(T_{n,\gamma} - c_{n,\gamma}(u))\} \tag{5.1}$$

for some $u \in (0,1)$.This testing procedure is closely related to the single step $minP$ multiple testing procedure based on minima of unadjusted p-values, see (D. S, van der Lann MJ (2008)). Unlike classical Bonferroni multiple, that can be obtained by taking $u = \alpha/s$, where $|\Lambda| = s$ denotes the cardinality of Λ, the previous rejection region takes into account the dependence structure among the test statistics $T_{n,\gamma}\ for\ \gamma \in \Lambda$. As the previous critical region can be written as $\{T_{n,\overline{\gamma}} > c_{n,\overline{\gamma}}(u)\}$ where

$$\overline{\gamma_u} = \overline{\gamma_u}(X_1, ..., X_n) = \arg\max_{\gamma \in \Lambda} \left(T_{n,\gamma} - c_{n,\gamma}(u)\right). \tag{5.2}$$

The (5.2) multiple test procedure based on a data-dependent method for selecting the tuning parameter γ: for a given sample of size n, one selects the value $\gamma \in \Lambda$ for which the test statistic $T_{n,\gamma}$ shows stronger evidence, at level u, against the null hypothesis.

Monte Carlo simulation was used in order to estimate the $(1-u)$ quantile under the null hypothesis $c_{n,\gamma}$ of $HV_{n,\gamma}$ and Z_n for the sample sizes $n = 10, 20, 50$. Tables 3, 4

below provide the empirical $(100-\alpha/s)$-percentiles of $(16\gamma^{2+d/2}/\pi^{d/2})HV_{n,\gamma}$ and $1000Z_{n,\gamma}$ obtained by simulating 100.000 samples from $\mathcal{N}(0,1)$ distribution. According to the suggestion of authors of these tests, the sets of tuning parameter $\gamma_{HV} = 2.5, 3, 4, 5, 7, 10$, $\gamma_Z = 1.5, 2, 3, 5, 9, 15$ and $s = 6$ are used for the simulation.

The results are denoted as $c_{n,\gamma}(u)$ in (5.2) and applied for the Henze-Visagie, Zghoul tests which we denote as HV^B and Z^B. We will discuss some properties of the powers of the tests based on this data-dependent choice of γ as well as the tests taking a fixed tuning parameter in section 6.

	γ					
n	2.5	3	4	5	7	10
10	160.863	147.659	129.298	116.101	103.774	95.361
20	479.722	374.011	261.226	221.368	172.910	156.583
50	1027.325	712.184	421.371	323.026	235.081	192.133

Table 3: the empirical $(100 - 0.05/s)$-percentiles of $(16\gamma^{2+d/2}/\pi^{d/2})HV_{n,\gamma}$

	γ					
n	1.5	2	3	5	9	15
10	5881.684	1170.158	156.544	16.197	1.571	0.226
20	27734.385	3613.259	347.660	30.118	2.510	0.332
50	101681.229	9047.307	587.411	40.690	3.063	0.409

Table 4: the empirical $(100 - 0.05/s)$-percentiles of $1000Z_{n,\gamma}$

5.2 An alternative data-dependent method, based on the bootstrap

The bootstrap is a method for estimating the distribution of an estimator or test statistic by resampling one's data or a model estimated from the data. In many situations, the bootstrap can be used to perform hypothesis tests that are more reliable in finite samples than tests based on asymptotic theory. The bootstrap method introduced in B. Efron (1979) is a very general resampling procedure for estimating the distributions of statistics based on independent observations.

Consider the problem of estimating variability of location estimates by the Bootstrap method. We view the observations $Y_{n,1}, ..., Y_{n,n}$, the scaled residuals of $(X_1, ..., X_n)$ as realizations of independent random variables with common distribution function F, it is appropriate to investigate the variability and sampling distribution of a location estimate calculated from a sample of size n. The test statistic $T_{n,\gamma}$ is a function of the random variables $Y_{n,1}, ..., Y_{n,n}$ and hence it has a probability distribution, its sampling distribution,which is determined by n and F.

We generate many samples $Y^*_{k,1}, ..., Y^*_{k,n}, (k = 1, ..., B)$, say $B \in \mathbb{N}$ is the considered number of bootstrap samples, of size n from F; from each sample we calculate the value of $\overline{T_{n,\gamma}}$. The empirical distribution of the resulting values $\overline{T_{n,\gamma}}(Y^*_{k,1}, ..., Y^*_{k,n})$ for $k = 1, ..., B$ is an approximation to the distribution function of $T_{n,\gamma}$, which is good if B is very large.
The test with critical region $\{T_{n,\gamma^*} > c_{n,\gamma^*}(\alpha)\}$ is considered, where $\gamma^* = \gamma^*(X_1, ..., X_n)$ is obtained by maximizing the bootstrap power, that is,

$$\gamma^* = \gamma^*((X_1, ..., X_n)) = \arg\max_{\gamma\in\Lambda} \frac{1}{B} \sum_{k=1}^{B} I\left(\overline{T_{n,\gamma}}(Y^*_{k,1}, ..., Y^*_{k,n}) > c_{n,\gamma}(\alpha)\right) \tag{5.3}$$

where $Y^*_{k,1}, ..., Y^*_{k,n}$ is a bootstrap random sample of size n drawn with replacement from the empirical distribution function of the transformed sample $Y_{n,1}, ..., Y_{n,n}$ and $I(A)$ denotes the indicator function of the set A, see Allison.J.S, Santana.L (2015).

The (5.3) methods are based on a data-dependent procedure for selecting the tuning parameter $\gamma \in \Lambda$, for a given sample of size n. We apply this method for the Henze-Visagie and Zghoul test to obtain $HV^{A.S}$ and $Z^{A.S}$ as the tests based on the data- dependent choice of γ, the powers of these tests will be presented in section 6.

Recently there has been a critical review of this method, Tenreiro (2019). Unfortunately, the proposed method presents two important drawbacks. Firstly, the suggested bootstrap procedure, as based on a bootstrap random sample drawn from the empirical distribution function of the transformed sample $Y_{n,j} = \frac{X_j - \overline{X_n}}{S_n}$ for $j = 1, ..., n$ and not from the empirical distribution function of the original sample, does not always produce a good approximation for the power associated with the distribution of the observations. Secondly, by using the quantiles of order $(1 - \alpha)$ of each one of the test statistics $T_{n,\gamma}$ to define the critical region, the proposed test is not correctly calibrated and may reach a level of significance much bigger than α.

To provide a perspective for testing the validity of this method, practical conclusions are drawn from the discussion of the Monte Carlo results in Section 6 by looking at the performance of these tests in comparison to each other, using both the data-dependent choice of γ and a fixed tuning parameter.

6 Power Results

In the power study presented in this section we simulate from several univariate distributions. The critical values presented in Table $1, 2, 3, 4$ are used in order to calculate the power results for all mentioned test. The power estimates presented below are based on 10000 random samples. Power results are reported for a sample size of $n = 10, 20, 50$ and a nominal significance level of $\alpha = 0.05$ is used throughout. The power against the standard normal distribution shows that the nominal level is maintained very closely.

As for the alternative distributions, $NMix(0.5, 1, 4)$ and $NMix(0.75, 1, 4)$ denote mixtures of the normal distributions $\mathcal{N}(0, 1)$ and $\mathcal{N}(0, 4)$. The mixture $NMix(0.5, 1, 4)$ gives equal weight to these distributions, while $NMix(0.75, 1, 4)$ is obtained when the probability of sampling from the standard normal distribution is 0.75.

The remaining alternative distributions considered are t-distributions with $3, 5, 10$ degrees of freedom, the lognormal distributions with parameters $(0, 1/2)$ and $(0, 1/4)$ (denoted by $LN(.)$), the χ^2-distributions with 5 and 15 degrees of freedom, the standard logistic distribution, the Weibull distribution $W(1, 0.5)$ and $W(2, 1)$, the Pearson type VII distributions with 5 and 10 degrees of freedom (denoted by $P_{VII}(.)$), and the skew-normal law with skewness parameters 3 and 5 (denoted by $SN(3)$), the uniform distribution $\mathcal{U}(-\sqrt{3}, \sqrt{3})$, the beta distribution $Beta(1, 4)$ and $Beta(2, 5)$, and the gamma distribution $\Gamma(1, 5)$ and $\Gamma(5, 1)$, the Gumbel distribution $Gum(1, 2)$ with location parameter 1 and scale parameter 2, the exponential distribution $Exp(1)$, the Cauchy distribution $Cauchy(0, 1)$, the Laplace distribution $Laplace(0, 1)$.

The results presented in Table 5-10 show that the power against the standard normal distribution of $HV^{A.S}$ and $Z^{A.S}$ as well as HV^B and Z^B are maintained very closely the level of significance α, moreover for many cases the Monte Carlo power of the tests using the data-dependent choice compare favourably to the maximum achievable power for the tests calculated over the chosen grid of values of the tuning parameter. However, for those alternatives where they do differ, the data-dependent choice shows a dramatic improvement over the fixed choice (as with the χ^2_5 alternative for HV for $n = 20, 50$). We cannot claim that these methods are uniformly more powerful than the method employing a fixed choice, but we do note that, even in cases where the fixed choice does well (as with the $Exp(1)$ alternative for HV), the loss in power of using the test based on the data-dependent choice of the parameter is only slightly less than the power using the fixed choice.

Although HV and Z tests show strong power performance especially against $Exp(1)$, χ^2_5, $Cauchy(0, 1)$, $\Gamma(1, 5)$, t_3 very interesting are their behaviors for the uniform distribution $\mathcal{U}(-\sqrt{3}, \sqrt{3})$, where they fail to detect the alternative for any value of γ for any n.

However, the considered tests based on the data-dependent tuning parameter selectors $\overline{\gamma_u}$ and γ^* have shown to be serious competitors for the recommended tests based on a fixed tuning parameter, and perhaps should be used – especially $\overline{\gamma_u}$ because is less time-consuming than γ^*.

	$HV_{2.5}$	HV_3	HV_4	HV_5	HV_7	HV_{10}	HV^B	$HV^{A.S}$
$\mathcal{N}(0,1)$	5.09	4.55	4.90	4.81	5.07	4.96	4.74	4.98
$NMix(0.5,1,4)$	10.61	10.54	10.51	10.40	10.20	10.89	10.25	11.37
$NMix(0.75,1,4)$	11.71	11.37	11.88	11.53	11.49	11.63	11.23	12.71
t_3	21.72	21.28	21.92	21.99	22.04	21.54	21.31	22.15
t_5	13.11	13.07	13.26	13.27	13.23	13.65	13.74	13.91
t_{10}	9.06	8.53	8.75	8.15	8.00	8.23	8.33	8.68
$LN(0,1)$	22.68	22.30	22.43	23.33	22.69	23.70	22.98	22.74
$LN(0,1/4)$	9.98	10.35	9.94	10.56	10.38	10.05	9.98	10.16
χ^2_5	17.01	17.35	17.71	18.08	18.66	18.67	18.00	17.59
χ^2_{15}	9.45	9.69	9.76	9.54	9.60	9.31	9.52	9.99
$Logistic(0,1)$	10.08	9.62	9.85	9.84	9.35	9.21	9.24	9.83
$W(1,0.5)$	9.17	8.63	8.55	8.88	9.12	9.02	9.04	8.87
$W(2,1)$	11.94	12.30	11.33	11.95	11.91	11.85	11.38	11.97
$P_{VII}(5)$	12.84	13.41	13.50	13.57	12.85	13.02	12.77	13.68
$P_{VII}(10)$	8.87	8.36	8.07	8.58	8.67	8.47	8.30	8.75
$SN(3)$	9.11	9.10	9.24	9.39	8.92	9.66	8.71	9.44
$SN(5)$	11.42	11.97	11.65	11.95	11.61	11.87	11.97	10.91
$\mathcal{U}(-\sqrt{3},\sqrt{3})$	1.47	1.60	1.52	1.64	1.74	1.84	1.61	1.68
$Beta(1,4)$	16.03	15.88	17.06	17.24	17.68	17.99	18.32	17.83
$Beta(2,5)$	6.86	6.61	6.40	7.26	7.65	7.39	7.33	7.35
$\Gamma(1,5)$	31.28	31.53	32.39	33.59	34.52	34.07	33.88	33.63
$\Gamma(5,1)$	12.03	11.69	11.00	11.79	11.97	12.25	11.87	12.14
$Gum(1,2)$	14.88	14.89	15.28	15.29	15.38	15.44	14.97	15.34
$Exp(1)$	57.39	56.26	57.92	57.88	57.72	57.54	57.95	58.87
$Cauchy(0,1)$	32.25	31.93	31.61	33.14	34.01	34.64	34.48	34.01
$Laplace(0,1)$	18.14	17.91	17.64	18.38	18.06	17.70	18.26	18.42

Table 5: Monte Carlo power estimates of the Henz-Visagie test case for $n = 10$

	$HV_{2.5}$	HV_3	HV_4	HV_5	HV_7	HV_{10}	HV^B	$HV^{A.S}$
$\mathcal{N}(0,1)$	5.19	4.79	5.08	5.28	4.95	4.82	5.18	5.12
$NMix(0.5,1,4)$	16.31	15.40	15.13	15.11	15.11	14.74	14.62	15.70
$NMix(0.75,1,4)$	19.59	19.93	19.59	19.76	18.87	18.54	18.50	19.49
t_3	37.10	37.16	37.52	36.64	36.37	36.65	36.76	37.01
t_5	23.03	21.83	22.31	21.79	22.64	21.34	21.54	22.96
t_{10}	12.37	12.13	12.02	12.45	12.78	12.24	11.87	12.08
$LN(0,1)$	41.37	42.22	43.30	45.54	46.45	46.68	47.72	46.19
$LN(0,1/4)$	17.75	17.15	18.48	18.60	18.93	19.01	19.31	18.82
χ^2_5	32.41	33.08	34.68	35.69	36.50	37.95	38.48	35.96
χ^2_{15}	15.90	16.66	16.24	17.40	17.26	17.97	17.32	16.99
$Logistic(0,1)$	14.53	14.14	14.56	14.09	14.21	14.43	14.26	13.80
$W(1,0.5)$	14.49	14.20	14.60	14.69	15.67	15.97	15.19	15.14
$W(2,1)$	20.28	21.10	21.68	22.07	22.07	22.89	22.51	21.80
$P_{VII}(5)$	23.08	22.33	22.63	22.28	22.16	21.24	21.68	22.27
$P_{VII}(10)$	12.68	11.70	12.42	11.86	12.50	11.64	11.39	12.56
$SN(3)$	14.06	14.62	15.10	15.68	16.18	16.33	15.76	15.89
$SN(5)$	18.81	19.50	20.27	21.84	22.47	22.42	23.64	21.85
$\mathcal{U}(-\sqrt{3},\sqrt{3})$	0.12	0.16	0.18	0.27	0.35	0.33	0.32	0.34
$Beta(1,4)$	27.68	28.43	32.20	33.00	35.36	36.84	36.06	34.35
$Beta(2,5)$	9.40	9.86	10.43	10.50	11.49	11.40	10.98	11.09
$\Gamma(1,5)$	55.11	58.11	60.75	62.97	65.47	66.09	66.20	64.25
$\Gamma(5,1)$	19.86	19.93	20.73	21.79	22.48	23.40	22.90	21.86
$Gum(1,2)$	27.88	28.16	28.93	29.45	31.54	30.74	30.54	29.28
$Exp(1)$	83.02	83.67	83.18	82.33	81.78	80.72	81.59	83.15
$Cauchy(0,1)$	55.88	58.55	60.89	63.29	64.32	65.57	66.83	63.88
$Laplace(0,1)$	27.48	28.41	28.16	28.01	27.34	26.85	26.40	27.74

Table 6: Monte Carlo power estimates of the Henz-Visagie test case for $n = 20$

	$HV_{2.5}$	HV_3	HV_4	HV_5	HV_7	HV_{10}	HV^B	$HV^{A.S}$
$\mathcal{N}(0,1)$	5.15	4.65	4.68	5.08	5.11	4.64	4.79	4.84
$NMix(0.5,1,4)$	22.95	24.47	23.54	22.36	21.94	20.31	20.29	22.38
$NMix(0.75,1,4)$	35.13	34.85	33.20	31.89	31.62	30.04	30.55	31.49
t_3	65.57	65.64	64.86	63.67	61.70	59.36	60.20	61.92
t_5	40.43	40.34	39.59	38.93	38.11	35.59	36.38	38.53
t_{10}	19.83	20.11	19.80	19.06	17.81	17.93	17.29	18.76
$LN(0,1)$	74.68	78.75	82.04	84.28	86.56	87.55	87.86	86.72
$LN(0,1/4)$	33.12	36.37	37.62	40.87	42.38	43.21	43.39	43.09
χ^2_5	61.84	66.04	71.43	73.88	77.86	79.27	79.57	78.19
χ^2_{15}	29.90	32.50	34.59	37.40	39.52	40.84	40.99	39.56
$Logistic(0,1)$	23.52	24.17	23.67	23.36	22.61	21.84	21.43	21.58
$W(1,0.5)$	26.19	27.82	29.07	31.72	34.29	33.69	35.69	33.79
$W(2,1)$	39.33	42.42	45.04	47.91	50.88	51.48	51.70	50.57
$P_{VII}(5)$	39.96	40.82	38.77	38.86	37.16	35.26	36.06	38.11
$P_{VII}(10)$	19.80	20.11	19.57	19.59	18.74	17.64	18.28	18.29
$SN(3)$	25.61	27.69	30.95	32.45	36.54	37.20	36.05	36.53
$SN(5)$	36.56	39.95	44.57	48.64	53.15	55.21	55.37	53.58
$\mathcal{U}(-\sqrt{3},\sqrt{3})$	0.00	0.00	0.01	0.01	0.04	0.08	0.07	0.06
$Beta(1,4)$	50.91	59.02	67.02	73.57	78.82	81.63	80.89	79.64
$Beta(2,5)$	13.51	15.51	19.68	23.51	27.00	28.73	29.61	27.72
$\Gamma(1,5)$	89.81	92.77	95.07	96.47	97.46	98.22	98.10	97.56
$\Gamma(5,1)$	40.12	42.87	45.82	49.80	52.77	54.14	54.36	52.57
$Gum(1,2)$	53.51	56.49	59.90	62.70	65.69	66.96	67.31	66.61
$Exp(1)$	98.90	99.02	98.88	99.01	98.42	98.08	98.26	98.37
$Cauchy(0,1)$	89.04	92.22	95.33	96.70	97.28	98.14	98.12	97.65
$Laplace(0,1)$	48.42	48.45	47.18	47.24	44.18	42.00	42.57	44.02

Table 7: Monte Carlo power estimates of the Henz-Visagie test case for $n = 50$

	$Z_{1.5}$	Z_2	Z_3	Z_5	Z_9	Z_{10}	Z^B	$Z^{A.S}$
$\mathcal{N}(0,1)$	5.11	4.79	4.99	5.12	4.76	5.28	5.09	5.49
$NMix(0.5,1,4)$	10.08	10.57	10.99	10.83	10.91	10.11	9.97	10.37
$NMix(0.75,1,4)$	11.73	12.31	11.32	12.14	12.05	11.54	12.15	11.41
t_3	20.11	22.01	21.61	22.13	21.15	21.74	22.08	20.63
t_5	13.44	13.96	13.71	13.87	13.56	12.83	13.92	13.57
t_{10}	8.53	8.67	8.85	8.83	8.55	8.46	8.60	8.14
$LN(0,1)$	22.25	22.19	22.54	23.48	23.61	24.23	23.61	20.91
$LN(0,1/4)$	9.84	10.30	10.42	11.08	10.58	10.42	10.41	9.38
χ^2_5	16.73	17.45	17.54	18.46	18.45	17.93	18.90	16.34
χ^2_{15}	9.32	10.03	10.13	10.33	9.88	10.26	9.66	8.81
$Logistic(0,1)$	9.24	9.53	9.59	9.74	10.07	9.77	9.64	9.53
$W(1,0.5)$	9.03	8.79	8.05	9.10	8.93	8.93	8.29	7.91
$W(2,1)$	11.73	11.83	11.79	11.50	12.88	11.76	11.90	10.86
$P_{VII}(5)$	13.84	12.85	13.97	13.51	13.10	12.92	13.68	12.36
$P_{VII}(10)$	8.80	8.26	8.13	8.68	8.52	8.43	8.01	8.18
$SN(3)$	8.62	9.16	9.38	9.60	9.20	9.56	9.34	8.63
$SN(5)$	11.09	11.62	11.82	12.06	11.70	12.12	11.96	10.25
$\mathcal{U}(-\sqrt{3},\sqrt{3})$	1.53	1.45	1.39	1.90	1.83	1.71	1.84	1.75
$Beta(1,4)$	15.70	16.40	16.55	18.57	18.49	18.24	18.42	14.90
$Beta(2,5)$	6.62	7.54	7.08	7.37	7.27	6.89	7.42	6.45
$\Gamma(1,5)$	30.23	30.84	32.63	33.57	34.47	35.26	35.09	29.63
$\Gamma(5,1)$	11.18	11.72	11.92	12.52	12.17	11.74	12.19	11.01
$Gum(1,2)$	14.78	14.71	15.05	15.57	15.98	15.04	15.33	13.89
$Exp(1)$	55.87	56.55	56.73	57.53	57.06	57.14	57.79	56.89
$Cauchy(0,1)$	29.62	31.20	32.34	33.95	35.41	35.48	35.40	28.68
$Laplace(0,1)$	17.80	17.46	18.55	18.14	17.81	17.63	18.11	16.65

Table 8: Monte Carlo power estimates of the Zghoul test case for $n = 10$

	$Z_{1.5}$	Z_2	Z_3	Z_5	Z_9	Z_{10}	Z^B	$Z^{A.S}$
$\mathcal{N}(0,1)$	5.01	4.98	4.98	4.86	5.19	5.17	4.96	5.15
$NMix(0.5,1,4)$	15.99	15.53	14.88	14.91	15.16	14.05	14.78	14.27
$NMix(0.75,1,4)$	19.44	19.77	19.24	19.09	18.50	18.57	17.79	17.58
t_3	37.27	37.29	37.55	36.02	36.14	35.01	36.70	33.39
t_5	22.60	22.35	22.87	22.24	21.47	21.26	22.34	19.76
t_{10}	11.81	11.99	12.13	12.03	11.42	11.24	11.47	11.29
$LN(0,1)$	39.81	40.55	42.57	46.22	48.54	48.86	48.45	37.06
$LN(0,1/4)$	16.35	16.33	16.79	18.23	18.91	19.23	19.72	15.79
χ^2_5	28.41	30.34	33.67	35.62	38.26	38.49	40.07	29.67
χ^2_{15}	14.70	14.99	15.73	16.58	17.79	17.58	18.34	14.98
$Logistic(0,1)$	14.65	14.14	14.46	14.57	13.97	13.80	13.78	13.48
$W(1,0.5)$	13.25	13.18	13.79	14.99	15.53	15.20	15.86	12.75
$W(2,1)$	18.59	19.59	21.56	22.23	22.78	22.56	23.10	18.96
$P_{VII}(5)$	22.38	22.51	22.04	22.01	21.57	21.92	20.86	20.49
$P_{VII}(10)$	12.41	11.82	12.06	11.95	12.04	12.00	11.81	10.80
$SN(3)$	12.99	12.93	14.49	15.85	16.30	16.41	17.00	13.63
$SN(5)$	17.22	18.35	19.50	21.88	23.50	23.52	23.94	17.89
$\mathcal{U}(-\sqrt{3},\sqrt{3})$	0.10	0.09	0.28	0.27	0.51	0.40	0.53	0.63
$Beta(1,4)$	24.13	26.17	29.75	33.83	37.55	37.73	39.16	25.61
$Beta(2,5)$	8.15	8.68	9.57	10.85	12.13	11.19	12.21	10.08
$\Gamma(1,5)$	50.65	53.81	59.00	63.77	67.83	67.59	68.75	50.12
$\Gamma(5,1)$	19.06	19.21	21.55	21.78	23.21	23.55	23.02	19.40
$Gum(1,2)$	25.53	25.73	27.69	30.20	31.67	30.93	31.30	25.00
$Exp(1)$	83.59	83.02	83.07	81.53	80.21	79.41	80.45	79.76
$Cauchy(0,1)$	50.17	54.26	58.75	64.36	66.86	67.91	68.28	50.63
$Laplace(0,1)$	28.74	27.82	28.05	28.16	26.84	27.17	26.42	24.75

Table 9: Monte Carlo power estimates of the Zghoul test case for $n = 20$

	$Z_{1.5}$	Z_2	Z_3	Z_5	Z_9	Z_{10}	Z^B	$Z^{A.S}$
$\mathcal{N}(0,1)$	4.89	5.06	5.09	4.90	4.89	5.08	4.93	4.91
$NMix(0.5,1,4)$	23.98	24.18	22.59	21.97	20.64	19.42	20.57	17.00
$NMix(0.75,1,4)$	35.65	34.97	34.40	32.55	29.05	27.79	28.82	24.49
t_3	65.11	65.36	64.72	62.23	58.29	57.35	58.88	54.58
t_5	40.42	41.28	39.52	39.40	36.32	34.12	35.20	31.46
t_{10}	20.01	20.63	19.09	18.40	17.98	17.15	17.39	15.22
$LN(0,1)$	68.38	73.43	80.35	85.47	87.87	88.81	89.15	69.93
$LN(0,1/4)$	31.31	32.38	37.84	40.61	43.79	45.13	45.72	29.99
χ^2_5	53.60	60.05	67.54	76.05	79.88	81.19	81.11	56.71
χ^2_{15}	26.53	29.77	33.62	38.92	41.50	42.51	42.55	27.50
$Logistic(0,1)$	24.84	24.87	23.67	23.22	21.22	20.78	20.93	17.77
$W(1,0.5)$	23.00	25.65	28.26	32.91	34.46	35.85	37.12	22.07
$W(2,1)$	35.10	38.83	44.47	48.86	52.18	52.79	53.37	35.32
$P_{VII}(5)$	40.69	41.24	40.05	38.74	36.36	34.20	35.37	30.94
$P_{VII}(10)$	19.98	20.26	19.52	19.98	17.68	16.96	16.97	15.02
$SN(3)$	20.99	25.21	28.84	34.42	37.28	38.08	40.02	23.97
$SN(5)$	28.75	34.88	42.87	51.48	55.91	57.27	57.80	33.34
$\mathcal{U}(-\sqrt{3},\sqrt{3})$	0.00	0.00	0.00	0.01	0.10	0.12	0.19	0.31
$Beta(1,4)$	39.04	48.04	62.90	76.04	82.02	83.79	84.25	49.07
$Beta(2,5)$	10.28	12.74	18.21	25.40	29.50	31.31	33.03	17.30
$\Gamma(1,5)$	81.46	87.59	94.23	97.07	98.05	98.36	98.47	86.59
$\Gamma(5,1)$	35.14	38.36	44.83	50.38	53.78	55.68	57.26	35.84
$Gum(1,2)$	47.38	51.67	58.09	63.79	67.16	68.51	68.59	47.34
$Exp(1)$	98.83	98.99	98.96	98.66	98.11	97.25	97.65	97.10
$Cauchy(0,1)$	81.24	87.60	94.01	97.19	97.90	98.19	98.53	86.76
$Laplace(0,1)$	47.51	48.02	48.21	45.90	41.98	39.32	41.05	36.20

Table 10: Monte Carlo power estimates of the Zghoul test case for $n = 50$

References

Allison.J.S , Santana.L . On a data-dependent choice of the tuning parameter appearing in certain goodness-of-fit tests. // Journal of Statistical Computation and Simulation. 2015.

Efron B. Another Look at the Jackknife // The Annals of Statistics, Vol. 7, No. 1, (Jan., 1979), pp. 1-26. 1979.

Henze Norbert, Visagie Jaco. Testing for normality in any dimension based on a partial differential equation involving the moment generating function // Ann Inst Stat Math. 2019.

S Dudoit, Lann MJ van der. Multiple testing procedures with applications to genomics. New York: Springer, 2008.

Serfling Robert J. Approximation theorems of mathematical statistics. New York: Wiley, 1980.

Tenreiro Carlos. On the automatic selection of the tuning parameter appearing in certain families of goodness-of-fit tests // Journal of Statistical Computation and Simulation. 2019.

Zghoul Ahmad A. A goodness of fit test for normality based on the empirical moment generating function. // Communications in Statistics - Simulation and Computation. 2010.

Anhang 1: install package

```
install.packages("foreach")
install.packages("doParallel")
install.packages("PearsonDS")
install.packages("sn")
install.packages("reliaR")
```

Anhang 2: Test Statitstic

```
# scaled_residuals
scaled_residuals <- function(n,X){
  emp_mean = 1/n * (sum(X))
  emp_covar = 0
  for (i in 1: n) {
    emp_covar = emp_covar + (1/n) * (X[i,] - emp_mean)^2
  }
  Y = numeric(n)
  for (i in 1:n) {
    Y[i] = (X[i,] - emp_mean)/ sqrt(emp_covar)
  }
  return(Y)
}
# Henz-Visagie test
Henz_Visagie_test = function(n,g,X){
  Y = scaled_residuals(n,X)
  temp_test_statistic = matrix( 0 , nrow = n, ncol = n)
  for (j in 1:n ){
    for (k in 1:n){
      temp_test_statistic[j,k] =  exp((Y[j] +
         Y[k])^2/(4*g)) * ( (Y[j]*Y[k]) - (Y[j] +
         Y[k])^2/(2*g) + 1/(2*g) + (Y[j] +
         Y[k])^2/(4*(g^2)))
    }
  }
  return( 16 * ( g^(2+(1/2)))/ (pi^(1/2)) * (1/n) *
     ((pi/g)^(1/2)) * sum(temp_test_statistic))
}

# Zghoul test
Zghoul_test = function(n,g,X){
  Y = scaled_residuals(n,X)
  A = matrix(0, ncol = n, nrow = n)
  for (i in 1:n){
    for (j in 1:n) {
```

```
      A[i,j] = ( 1 / (n*sqrt(g)) ) * exp( (Y[i] + Y[j])^2 /
         (4*g) )
    }
  }
  B = c(0, rep(n))
  for (i in 1:n) {
    B[i] = 2/(sqrt(g - 0.5)) * exp( (Y[i]^2) / (4*g-2) )
  }
  return( 1000* sqrt(pi) * ( ( n/sqrt(g-1) ) - sum(B) +
     sum(A) ) )
}
```

Anhang 3: critical value estimation

```
library(foreach)
library(doParallel)
Montecarlo_critical_value_test = function( FUN, n, g,
   alpha, no_cores, no_simu){
  registerDoParallel(no_cores)  # use multicore, set to the
     number of our cores
  temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
    FUN(n,g, matrix(rnorm(n), nrow = n, ncol = 1) )
  }
  stopImplicitCluster()
  return( quantile(temp, probs= 1- alpha ) ) # u = alpha/
     cardility von g
}

Montecarlo_critical_value_table = function(FUN, n, g, alpha
   ,no_cores , no_simu){
  temp_critical_value = matrix(0, nrow = length(n), ncol =
     length(g))
  for (j in 1 : length(n)){
    for (k in 1: length(g)){
      temp_critical_value [j,k] =
         Montecarlo_critical_value_test(FUN,
         n[j],g[k],alpha, no_cores, no_simu)
    }
  }
  return(temp_critical_value)
}

# for Benferroni, Bootstrap based,  test
Montecarlo_critical_value_data_dependent_test = function(
```

```
   FUN, n, alpha, no_cores, no_simu){
  registerDoParallel(no_cores)  # use multicore, set to the
     number of our cores
  temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
    FUN(n, matrix(rnorm(n), nrow = n, ncol = 1) )
  }
  stopImplicitCluster()
  return( quantile(temp, probs= 1- alpha ) ) # u = alpha/
     cardility von g
}

Montecarlo_critical_value_data_dependent_table =
   function(FUN, n, alpha ,no_cores , no_simu){
  temp_critical_value = numeric(length(n))
  for (j in 1 : length(n)){
      temp_critical_value [j] =
         Montecarlo_critical_value_Benferroni_test(FUN,
         n[j],alpha, no_cores, no_simu)
  }
  return(temp_critical_value)
}
```

Anhang 4: Bonferroni Test Statitstic

```
library(doParallel)

# The first method is based on minima of unadjusted p-values
minima_of_unadjusted_p_values = function (FUN
   ,Bonferroni_critical_value,n,g,x){
  tuning_para = numeric(length(g))
  for (i in 1 : length(g)){
    tuning_para[i] = FUN(n,g[i],x) -
       Bonferroni_critical_value[i]
  }
   #erg = c(g[which.max(tuning_para)],
      which.max(tuning_para))
  return(g[which.max(tuning_para)]) # tuning para meter and
     the position in the set
}
# Henze Visagie Bonferroni test
HV_bonferroni_test <- function(n,x){
  Hv_bonferroni_cri_10_20_50 = matrix(c(160.862, 147.659,
     129.298, 116.101, 103.774, 95.361,
                                        479.722, 374.011,
```

```
                                          261.226, 221.368,
                                          172.910, 156.583,
                                        1027.325,
                                          712.183,421.370,
                                          323.026, 235.081,
                                          192.133),
                                      byrow = T , nrow = 3,
                                        ncol = 6)
  g_hv = c(2.5, 3, 4, 5, 7, 10)
  if(n ==10){ Hv_bonferroni_cri =
     Hv_bonferroni_cri_10_20_50[1,]}
  if(n ==20){ Hv_bonferroni_cri =
     Hv_bonferroni_cri_10_20_50[2,]}
  if(n ==50){ Hv_bonferroni_cri =
     Hv_bonferroni_cri_10_20_50[3,]}
  tunning_para=
     minima_of_unadjusted_p_values(Henz_Visagie_test,
     Hv_bonferroni_cri, n, g_hv, x)
  return(Henz_Visagie_test(n, tunning_para, x))
}

# Zghoul Bonferroni test
Z_bonferroni_test <- function(n,x){
  Z_bonferroni_cri_10_20_50 =
     matrix(c(5881.684,1170.158,156.544,16.197, 1.571,0.226,
                                          27734.385,
                                            3613.259,
                                            347.660, 30.118,
                                            2.510, 0.332,
                                          101681.229,
                                            9047.307,
                                            587.411, 40.690,
                                            3.063, 0.409),
                                       byrow = T , nrow = 3,
                                         ncol = 6)
  g_Z = c(1.5, 2, 3, 5, 9, 15)
  if(n ==10){ Z_bonferroni_cri =
     Z_bonferroni_cri_10_20_50[1,]}
  if(n ==20){ Z_bonferroni_cri =
     Z_bonferroni_cri_10_20_50[2,]}
  if(n ==50){ Z_bonferroni_cri =
     Z_bonferroni_cri_10_20_50[3,]}
  tunning_para= minima_of_unadjusted_p_values(Zghoul_test,
     Z_bonferroni_cri, n, g_Z, x)
```

```
  return(Zghoul_test(n, tunning_para, x))
}
```

Anhang 5: bootstrap based test Statitstic

```
# scaled_residuals
scaled_residuals <- function(n,X){
  emp_mean = 1/n * (sum(X))
  emp_covar = 0
  for (i in 1: n) {
    emp_covar = emp_covar + (1/n) * (X[i,] - emp_mean)^2
  }
  Y = numeric(n)
  for (i in 1:n) {
    Y[i] = (X[i,] - emp_mean)/ sqrt(emp_covar)
  }
  return(Y)
}
# test with scaled residual Y als input
HV_test <- function(n,g,Y){
  temp_test_statistic = matrix( 0 , nrow = n, ncol = n)
  for (j in 1:n ){
    for (k in 1:n){
      temp_test_statistic[j,k] =  exp((Y[j] +
         Y[k])^2/(4*g)) * ( (Y[j]*Y[k]) - (Y[j] +
         Y[k])^2/(2*g) + 1/(2*g) + (Y[j] +
         Y[k])^2/(4*(g^2)))
    }
  }
  return( 16 * ( g^(2+(1/2)))/ (pi^(1/2)) * (1/n) *
     ((pi/g)^(1/2)) * sum(temp_test_statistic))
}
Z_test <- function(n,g,Y){
  A = matrix(0, ncol = n, nrow = n)
  for (i in 1:n){
    for (j in 1:n) {
      A[i,j] = ( 1 / (n*sqrt(g)) ) * exp( (Y[i] + Y[j])^2 /
         (4*g) )
    }
  }
  B = c(0, rep(n))
  for (i in 1:n) {
    B[i] = 2/(sqrt(g - 0.5)) * exp( (Y[i]^2) / (4*g-2) )
  }
```

```
  return( 1000* sqrt(pi) * ( ( n/sqrt(g-1) ) - sum(B) +
     sum(A) ) )
}

# Bootstrap based method for scaled residual
bootstrap_for_Y <- function(n,Y){
  Y_bootstrap = numeric(n)
  for (j in 1: n ){ #sample size
    Y_bootstrap[j] = Y[sample(n, 1)]
  }
  return(Y_bootstrap)
}
# bootstrap based
bootstrap_based_para <- function(FUN,critical_value,n,g,X) {
  tuning_para = numeric(length(g))
  Y = scaled_residuals(n,X)
  for (i in 1: length(g)){
    I = 0 # the indicatot function
    for (k in 1: 100) { #Bootstrap number
      Y_bootstrap = bootstrap_for_Y(n,Y)
      I = I + (1/100)*( FUN(n,g[i], Y_bootstrap) >
         critical_value[i] )
    }
    tuning_para[i] = I
  }
  erg = g[which.max(tuning_para)]
  return(erg) # tuning para meter and the position in the
     set
}
HV_bootstrap_test <- function(n,X){
  Hv_test_cri_10_20_50 = matrix(c(59.850,58.220, 53.591,
     50.083, 47.745, 44.979,
                                  119.115, 104.389, 88.294,
                                     79.485, 70.131, 65.184,
                                  219.382, 173.079,
                                     131.732, 110.610,
                                     92.291, 82.927),
                                  byrow = T , nrow = 3,
                                     ncol = 6)
  g_hv = c(2.5, 3, 4, 5, 7, 10)
  if(n ==10){ Hv_cri = Hv_test_cri_10_20_50[1,]}
  if(n ==20){ Hv_cri = Hv_test_cri_10_20_50[2,]}
  if(n ==50){ Hv_cri = Hv_test_cri_10_20_50[3,]}
  tunning_para = bootstrap_based_para(HV_test, Hv_cri, n,
```

```
    g_hv, X)
  return(Henz_Visagie_test(n, tunning_para, X))
}
Z_bootstrap_test <- function(n,X){
  Z_test_cri_10_20_50 = matrix(c(1885.307, 406.606, 62.165,
    7.165, 0.739, 0.112,
                                 4882.133, 895.159,
                                   110.043, 11.207, 1.051,
                                   0.155,
                                 12134.918, 1675.366,
                                   170.144, 14.987, 1.335,
                                   0.189),
                               byrow = T , nrow = 3, ncol =
                                 6)
  g_Z = c(1.5, 2, 3, 5, 9, 15)
  if(n ==10){ Z_cri = Z_test_cri_10_20_50[1,]}
  if(n ==20){ Z_cri = Z_test_cri_10_20_50[2,]}
  if(n ==50){ Z_cri = Z_test_cri_10_20_50[3,]}
  tunning_para = bootstrap_based_para(Z_test, Z_cri, n,
    g_Z, X)
  return(Zghoul_test(n, tunning_para, X))
}
# test with normal distribution
X = matrix(rnorm(10), nrow = 10, ncol = 1)
HV_bootstrap_test(20, matrix(rnorm(20), nrow = 20, ncol =
  1) )
Z_bootstrap_test(20, matrix(rnorm(20), nrow = 20, ncol = 1)
  )
```

Anhang 6: Henz Visagie power result

```
# used function
test_decision_Henz_Visage <- function(n,g, cri_wert,
  no_cores, no_simu){
  # power result of Henz_Visagie
  test_erg <- numeric(26) # number of distribution samples
  for (i in 1: 26) {
    registerDoParallel(no_cores)  # use multicore, set to
      the number of our cores
    temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
      Henz_Visagie_test(n,g, distribution_samples(i,n)) >
        cri_wert
    }
    test_erg[i] = sum(temp)*100 / no_simu
```

```
    stopImplicitCluster()
  }
  return(test_erg)
}
test_decision_Henz_Visage_tab <- function(n,g, cri_wert_d1,
   no_cores, no_simu){
  test_decision_Henz_Visage_table = matrix(0, ncol =
     length(g), nrow =  26) # nrow is the number of
     distribution samples
  for (i in 1: length(g)){
    test_decision_Henz_Visage_table[ ,i] =
       test_decision_Henz_Visage(n,g[i], cri_wert_d1[i],
       no_cores, no_simu) # cause it returns a vector
  }
  colnames(test_decision_Henz_Visage_table) <- g
  #c(2.5, 3, 4, 5, 7, 10)
  return(test_decision_Henz_Visage_table)
  #(print(xtable(as.data.frame(test_decision_Henz_Visage_table)),
     include.colnames = TRUE,include.rownames=FALSE) )
}

# result present

g_hv = c(2.5, 3, 4, 5, 7, 10)
g_zghoul = c( 1.5, 2, 3, 5, 9, 15)
n = c(10,20,50)
no_cores = 8
no_simulation = 10000
#Henz_Visagie_critical_value =
   read.csv("Henz_Visagie_critical_value.csv ")
# create the power result table for henz_visagie test
start.time <- Sys.time()
#test_decision_Henz_Visage(n[1], g_hv[1],
   Henz_Visagie_critical_value[1,1] , 1, 10000)
Henz_visagie_power_result_50 =
   test_decision_Henz_Visage_tab(n[3], g_hv,
   Henz_Visagie_critical_value[3,] , no_cores,
   no_simulation)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
colnames(Henz_visagie_power_result_50) = g_hv
write.csv(Henz_visagie_power_result_50, file =
   "Henz_visagie_power_result_50.csv␣" , row.names = FALSE)
```

```
# tuning para test

#Bonferoni
Henz_Visagie_Benferoni_critical_value <-
   HV_Zghoul_benferoni_critical_value[1,]

test_decision_Henz_Visage_bonferroni <- function(n,
   cri_wert, no_cores, no_simu){
  # power result of Henz_Visagie
  test_erg <- numeric(26) # number of distribution samples
  for (i in 1: 26) {
    registerDoParallel(no_cores)  # use multicore, set to
       the number of our cores
    temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
      #(n, distribution_samples(i,n)) > cri_wert
      HV_bonferroni_test(n, distribution_samples(i,n))>
         cri_wert
    }
    test_erg[i] = sum(temp)*100 / no_simu
    stopImplicitCluster()
  }
  return(test_erg)
}

#test_decision_Henz_Visage_bonferroni(n[1],
   Henz_Visagie_Benferoni_critical_value[1], 1, 1000)

n = c(10,20,50)
no_cores = 8
no_simulation = 10000

start.time <- Sys.time()
test_decision_Henz_Visage_bonferroni_table <- matrix(0,
   nrow = 26, ncol = 3)
for (i in 1:3){
  test_decision_Henz_Visage_bonferroni_table[,i] =
     test_decision_Henz_Visage_bonferroni(n[i],
     Henz_Visagie_Benferoni_critical_value[i], no_cores,
     no_simulation)
}
end.time <- Sys.time()
```

```
time.taken <- end.time - start.time
time.taken
colnames(test_decision_Henz_Visage_bonferroni_table) = n
write.csv(test_decision_Henz_Visage_bonferroni_table, file
   = "Henz_Visage_bonferroni_power_result_table.csv␣" ,
   row.names = FALSE)

# Bootstrap based method

Henz_Visagie_Bootstrap_critical_value <-
   HV_Zghoul_Bootstrap_critical_value[1,]

test_decision_Henz_Visage_bootstrap <- function(n,
   cri_wert, no_cores, no_simu){
  test_erg <- numeric(26) # number of distribution samples
  for (i in 1: 26) {
    registerDoParallel(no_cores)  # use multicore, set to
       the number of our cores
    temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
      HV_bootstrap_test(n, distribution_samples(i,n))>
         cri_wert
    }
    test_erg[i] = sum(temp)*100 / no_simu
    stopImplicitCluster()
  }
  return(test_erg)
}
start.time <- Sys.time()
test_decision_Henz_Visage_bootstrap_table <- matrix(0, nrow
   = 26, ncol = 3)
for (i in 1:3){
  test_decision_Henz_Visage_bootstrap_table[,i] =
     test_decision_Henz_Visage_bootstrap(n[i],
     Henz_Visagie_Bootstrap_critical_value[i], no_cores,
     no_simulation)
}
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
colnames( test_decision_Henz_Visage_bootstrap_table) = n
write.csv( test_decision_Henz_Visage_bootstrap_table, file
   = "Henz_Visage_bootstrap_power_result_table.csv␣" ,
   row.names = FALSE)
```

Anhang 7: Zghoul power result

```
# used function
test_decision_Zghoul <- function(n,g, cri_wert, no_cores,
   no_simu){
  # power result of Zghoul
  test_erg <- numeric(26) # number of distribution samples
  for (i in 1: 26) {
    registerDoParallel(no_cores)  # use multicore, set to
       the number of our cores
    temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
      Zghoul_test(n,g, distribution_samples(i,n)) > cri_wert
    }
    test_erg[i] = sum(temp)*100 / no_simu
    stopImplicitCluster()
  }
  return(test_erg)
}

test_decision_Zghoul_tab <- function(n,g, cri_wert_d1,
   no_cores, no_simu){
  test_decision_Zghoul_table = matrix(0, ncol = length(g),
     nrow =  26) # nrow is the number of distribution
     samples
  for (i in 1: length(g)){
    test_decision_Zghoul_table[ ,i] =
       test_decision_Zghoul(n,g[i], cri_wert_d1[i],
       no_cores, no_simu) # cause it returns a vector
  }
  colnames(test_decision_Zghoul_table) <- g
  #c(2.5, 3, 4, 5, 7, 10)
  return(test_decision_Zghoul_table)
  #(print(xtable(as.data.frame(test_decision_Henz_Visage_table)),
     include.colnames = TRUE,include.rownames=FALSE) )
}

# result present
g_zghoul = c( 1.5, 2, 3, 5, 9, 15)
n = c(10,20,50)
no_cores = 8
no_simulation = 100
Zghoul_critical_value = read.csv("Zghoul_critical_value.csv␣
   ")
# create the power result table for henz_visagie test
```

```
start.time <- Sys.time()
Zghoul_power_result_10 = test_decision_Zghoul_tab(n[1],
   g_zghoul, Zghoul_critical_value[1,] , no_cores,
   no_simulation)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken

colnames(Zghoul_power_result_10) = g_zghoul
write.csv(Zghoul_power_result_10, file =
   "Zghoul_power_result_10.csv␣" , row.names = FALSE)



# tuning para test

#Bonferoni

test_decision_Zghoul_bonferroni <- function(n, cri_wert,
   no_cores, no_simu){
  # power result of Zghoul
  test_erg <- numeric(26) # number of distribution samples
  for (i in 1: 26) {
    registerDoParallel(no_cores)  # use multicore, set to
       the number of our cores
    temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
      #(n, distribution_samples(i,n)) > cri_wert
      Z_bonferroni_test(n, distribution_samples(i,n))>
         cri_wert
    }
    test_erg[i] = sum(temp)*100 / no_simu
    stopImplicitCluster()
  }
  return(test_erg)
}

Zghoul_Benferoni_critical_value <-
   data.matrix(HV_Zghoul_benferoni_critical_value)[2,]

start.time <- Sys.time()
test_decision_Zghoul_bonferroni_table <- matrix(0, nrow =
   26, ncol = 3)
for (i in 1:3){
```

```
  test_decision_Zghoul_bonferroni_table[,i] =
     test_decision_Zghoul_bonferroni(n[i],
     Zghoul_Benferoni_critical_value[i], no_cores,
     no_simulation)
}
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
colnames(test_decision_Zghoul_bonferroni_table) = n
write.csv(test_decision_Zghoul_bonferroni_table, file =
   "Zghoul_bonferroni_power_result_table.csv␣" , row.names
   = FALSE)

# Bootstrap based method

test_decision_Zghoul_bootstrap <- function(n, cri_wert,
   no_cores, no_simu){
  test_erg <- numeric(26) # number of distribution samples
  for (i in 1: 26) {
    registerDoParallel(no_cores)  # use multicore, set to
       the number of our cores
    temp = foreach(j = 1 : no_simu, .combine = c) %dopar%{
      Z_bootstrap_test(n, distribution_samples(i,n))>
        cri_wert
    }
    test_erg[i] = sum(temp)*100 / no_simu
    stopImplicitCluster()
  }
  return(test_erg)
}

Zghoul_Bootstrap_critical_value <-
   data.matrix(HV_Zghoul_Bootstrap_critical_value)[2,]

start.time <- Sys.time()
test_decision_Zghoul_bootstrap_table <- matrix(0, nrow =
   26, ncol = 3)
for (i in 1:3){
  test_decision_Zghoul_bootstrap_table[,i] =
     test_decision_Zghoul_bootstrap(n[i],
     Zghoul_Bootstrap_critical_value[i], no_cores,
     no_simulation)
}
end.time <- Sys.time()
```

```
time.taken <- end.time - start.time
time.taken
colnames( test_decision_Zghoul_bootstrap_table) = n
write.csv( test_decision_Zghoul_bootstrap_table, file =
   "Zghoul_bootstrap_power_result_table.csv␣" , row.names =
   FALSE)
```

Declaration

I hereby declare that this piece of written work is the result of my own independent scholarly work, and that in all cases material from the work of others (in books, articles, essays, dissertations, and on the internet) is acknowledged, and quotations and paraphrases are clearly indicated. No material other than that listed has been used. This written work has not previously been used as examination material at this or any other university. This written work has not yet been published.
Place, date