TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DIPLOMARBEIT

Logarithmic Concave Density Estimation and its Application in a Use Case in the Semiconductor Industry

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Masterstudium Statistik-Wirtschaftsmathematik
E 066 395

eingereicht von

Martin Hofbauer
Matrikelnummer 01325242

ausgeführt am
Institut für Stochastik und Wirtschaftsmathematik
der Fakultät für Mathematik und Geoinformation der Technischen Universität Wien

Betreuer: Associate Prof. Diplom-Statistiker Klaus Nordhausen, PhD

Wien, August 18, 2020

Unterschrift Verfasser

Unterschrift Betreuer

Abstract

The estimation of a probability density function of a random variable X, given a sample $x_1, \ldots, x_n$, is a task with major interest for statisticians. There are several approaches to achieve this task, resulting in different possible techniques for practitioners. A big shortcoming of many of these techniques is the dependency on parameters that determine the actual estimation. Logarithmic concave density estimation has the potential to free practitioners from such parametric requirements and thus opens the door to automatic density estimation. In this thesis we will give an overview of logarithmic concave densities, and make use of this theory to solve a specific problem in a use case in the semiconductor industry. To handle the tasks in this use case we will mainly focus on two applications of logarithmic concave density estimators: the replacement of density estimators in tail index estimation with the smoother log-concave density estimator and a statistical test that has the detection of the presence of mixtures as target. For the latter application, we will suggest an additional parameter, which improves the performance of the algorithm under the H_0 with only little additional computation time. Furthermore we will implement those applications in a procedure that has the rating of densities from testing-parameters in the semiconductor industry as target. We will test this procedure using real data, conclude our findings and give an outlook about possible further development.

Acknowledgments

I want to thank my supervisor, Prof. Nordhausen for always helping me with useful suggestions and quick responses to upcoming questions. He has been very supportive in the conception of the topic and of course the creation of this thesis.

A big thanks to Guenther Walther from the Department of Statistics at Stanford University for the provisioning of his Matlab-code of the algorithm suggested in [43], which helped me to implement the corresponding procedure in R.

Furthermore I want to thank my colleagues from the product engineering team of Infineon Graz for giving me the opportunity to develop my thesis within the team and providing me proper data for testing the developed algorithm. Special thanks to my primary supervisor, Dipl.-Ing. Christopher Moswitzer, who supported me directly with innovative suggestions and honest feedback of my work.

Contents

List of Symbols

$\mathbb{N}$	...	natural numbers (including 0)
$\mathbb{N}^*$	...	natural numbers without 0
$\mathbb{R}^+$	...	positive real numbers (> 0)
$\mathbb{R}_0^+$	...	non-negative real numbers (≥ 0)
$\lfloor \cdot \rfloor$	...	floor function: $\lfloor x \rfloor := \max_{y \in \mathbb{N}: y \leq x} \ y$
$\lceil \cdot \rceil$	...	ceiling function: $\lceil x \rceil := \min_{y \in \mathbb{N}: y \geq x} \ y$
$\lim_{z \to x^+}$	...	one-sided limit (z decreases to x from above)
$\lim_{z \to x^-}$	...	one-sided limit (z increases to x from below)
$\boldsymbol{v}$	...	vectors can mostly be identified as lower-case, bold symbols
$\boldsymbol{M}$	...	matrices can mostly be identified as upper-case, bold symbols
X	...	random variable (upper-case characters)
$x_1, \ldots, x_n$	...	sample (realizations) of the random variable X
$x_{(1)}, \ldots, x_{(n)}$	...	order statistics of the sample $x_1, \ldots, x_n$
$\mathcal{F}_d$	...	class of log-concave densities on $\mathbb{R}^d$
f, g	...	density functions ("PDF" - typically lower-case symbols)
F, G	...	distribution functions ("CDF" - typically upper-case symbols)
$\hat{f}_n$	...	log-concave density estimator
$\hat{\psi}_n$	...	log of the log-concave density estimator ($\log \hat{f}_n = \hat{\psi}_n$)
$\hat{F}_n$	...	log-concave distribution estimator ($\hat{F}_n(x) = \int_{-\infty}^{x} \hat{f}_n(y) \mathrm{d}y$)
$\mathbb{F}_n$	...	empirical distribution function of a sample $x_1, \ldots, x_n$
$f * g$	$\cdots$	convolution of two functions $(f * g)(x) := \int_{-\infty}^{\infty} f(y) \cdot g(x - y) \mathrm{d}y$
$\operatorname{supp}(f)$	$\cdots$	support of a function: $\{x \in D \mid f(x) \neq 0\}$

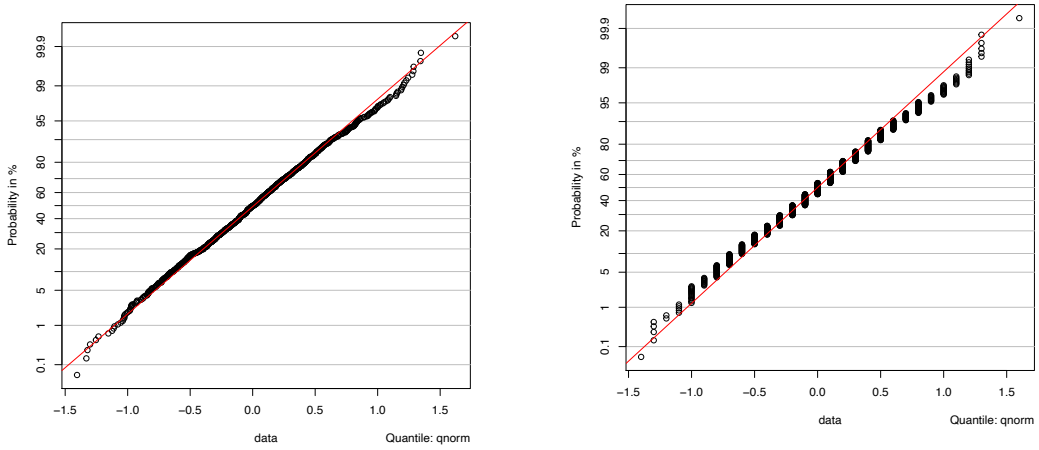
1. Introduction

In the development processes of high quality products in nearly every industry segment, frequent testing of these products is necessary. Consequently the arising amount of data is tremendous. The detailed analysis of such data can be done by highly skilled engineers but this comes with an obvious disadvantage: the rating of several hundred or even thousands of data-samples from testing-parameters can be very monotone and thus error-prone. Therefore it is self-evident that an automatic procedure, that covers such tasks, would help. Even a tool that needs manual reevaluation for some parameters after the automatic rating may help, since a lot of working time could be saved by such a digital assistant doing routine checks of newly available datasets. A reevaluation by an engineer would then only be necessary if some deviation from the expected outcome occurs. Obviously, such a transfer of monotone work from highly skilled engineers to digital tools, that assist them, gives an economic motivation for companies. The main target of this thesis is the development of such a tool.

This thesis is developed in cooperation with Infineon Austria [21], more precisely the product engineering team of the business line "Power Integration & Supply" within the division "Automotive", located at the development center in Graz, Austria. Employees of the team have the duty to assess large amounts of data and thus it would be advantageous to have the possibility to get an overview of newly available datasets quickly. Therefore every part of the procedure we are developing should be executable in an acceptable amount of time on common personal computer systems. This fact motivates us to keep an eye on the necessary computation time of the algorithms we will use.

The target of the overall procedure is the rating of univariate densities that come from testing-processes of semiconductor products in different stages of development. Thus we are not able to make a lot of assumptions on the properties of the data, that work as input for the algorithm. At least we can exclude some data from further analysis since it is not in the scope of interest for the task: some datasets contain information about properties of the testing-machines or categorical data, which we can skip and leave unrated. The data we are focusing on is, as mentioned always univariate, and mainly continuous, even if some actually discrete samples

can nevertheless occur due to e.g. the resolution of the saved values[1]. We will assume such samples as continuous as long as a minimum of unique values (can be specified by the operator) appears. To give a visual idea of how such samples can look like, we simulate an example in Figure 1.1.



(a) Data with precise values.

(b) Data with rounded values (probably due to machine-inaccuracy).

Figure 1.1.: Probability plots of an artificial example and its rounded version.

The target distribution for all samples is the normal distribution, $N(\mu, \sigma^2)$ with sometimes known μ and unknown σ. The location μ is sometimes known since such technical parameters are distributed around some target value and should only differ from it due to natural inaccuracies of the components that appear in the production process. Nevertheless we will not make use of this additional information if it is available since we want to create a procedure that works generally the same way, no matter if μ is available or not.

A possible violation of the target distribution is a mixture distribution, that probably results in a multimodal distribution in the overall sample. This can for example happen when the process of testing a large amount of technical devices is allocated to two (or more) separate testing-machines, on which the measured values differ in general (precisely, the values would also differ for the exact same device). We will try to identify such violations by performing a statistical test, that we will describe and analyze in Chapter 4.

[1]Of course no machine is perfectly accurate but for some actually accurate enough measurements, the values are rounded or truncated after a specific decimal place when saving them.

For unimodal densities, and thus most of our samples, we are mainly interested in the tailing behavior. This has a specific reason: most of the measured testing-parameters have some lower or upper limit that is necessary for the performance of the product or quality restrictions by the customer. When some tested device violates such a limit, it has to be discarded. It is possible that this happens only for a few devices in early samples but when the development of this product goes on and production-quantities increase, more devices might get discarded and the yield-loss can become severe. Such problems can be discovered early by the detection of heavy tails of a parameter's distribution. Therefore it is self-evident that such violations are worth being detected as early as possible. Since it is possible that there is only a lower or an upper limit, we are interested in a separate evaluation of the lower and upper tail's behavior. We will tackle this task with tail index estimation in Chapter 5.

Note that even if we will sometimes call the rating process "classification", the task is not in the scope of mathematical classification in a strict sense such that every sample has an unknown class and the assignment to a class by the algorithm can either be correct or not. It is about assigning classes to data that give an overview of the data and detect severe violations of the assumptions but there is no predefined set of classes that necessarily has to be assigned. Thus it is not a standard problem that can easily be solved by one single straight-forward method, which is the reason why we develop a completely new tool and procedure.

The algorithm will be called "NN 2.0" since a machine learning algorithm, which is called "NN"[1], with a similar target is currently used by some members of the team. In the development of this predecessor of NN 2.0, different approaches were taken: naive Bayes, k-nearest-neighbor (kNN) and random forest algorithms were used and reached accuracies between 48 % and 87 % of correctly classified densities in some testing-dataset. Finally a random forest algorithm was chosen and fed with approximately 1000 densities to train the algorithm. Nevertheless such an algorithm works as a black-box model and is always dependent on the quality and variety of the training-dataset. Moreover it always has a subjective component as it depends on a manually classified training-dataset. This gives motivation to tackle the task of automatic density classification once more and try other approaches with new possible procedures and algorithms.

It is self-evident that we prefer nonparametric procedures for a task that should work as automatic as possible. We decide to investigate the possibilities of using

[1]This name comes from the fact that workpackages are assigned to a dummy variable "NN - nomen nominandum" in a work management system when too heavy workload does not allow allocation to a real employee. Since such tools should save working time and let skilled employees do other tasks, the team decided to choose this name for such a digital assistant.

the logarithmic concave density estimator to achieve automatic density estimation and consequently classification of the given data with respect to shape and violations from the normal distribution. Logarithmic concave density estimation is a topic that gathered a lot of attention in the past few years and is thus a reasonable choice for us to be the theoretical origin of this thesis. Besides of theoretical properties that were under investigation e.g. in [6], [23] and [31], we will focus on possible applications and mainly use techniques proposed in [43] and [25].

In the following two chapters we will introduce the term of logarithmic concave densities and the logarithmic concave density estimator, including their properties and some examples. These chapters will be focused on theory, whilst we will discuss two applications, that will be the main part of the overall procedure in Chapter 4 and Chapter 5. The final algorithm that is developed will be presented in Chapter 6, which includes the application to real datasets to test the procedure. Finally we will conclude our findings and discuss shortcomings and possible extensions of the algorithm in Chapter 7.

2. Logarithmic Concave Densities

In this chapter, we introduce some definitions and theorems in the field of logarithmic concave densities. Not every result that is presented in this section has direct impact on the content of later chapters, but some can be seen as essential parts of the field of logarithmic concave densities (e.g. Theorem 2.8) and can therefore be seen as mandatory to get an overview of the topic. Other results will be used in later chapters and therefore might even be seen as fundamental for this thesis (e.g. Theorem 2.7 for Chapter 5 and Theorem 2.12 for Chapter 4). An analogous review of logarithmic concave densities can for example be seen in [35], from where we will mainly use the notation.

First of all, we want to start with the most obvious definitions one can think of when considering the title of this chapter.

Definition 2.1. *A function $\phi : \mathbb{R}^d \to \mathbb{R}$ is called concave if*

$$\phi\left(\alpha x + (1-\alpha)y\right) \geq \alpha\phi\left(x\right) + (1-\alpha)\phi\left(y\right)$$

holds for every $\alpha \in [0;1]$ and $x, y \in \mathbb{R}^d$.

Note that we could also claim *alpha* $\in (0;1)$, because for $\alpha = 0$ or $\alpha = 1$, the left- and right-hand-side are obviously equal.

Definition 2.2. *A density function $f : \mathbb{R}^d \to [0;\infty)$ is called logarithmic concave (log-concave) if it is of the form*

$$f(x) = \exp\left(\phi(x)\right) \tag{2.1}$$

for some concave function $\phi : \mathbb{R}^d \to [-\infty;\infty)$.
Equivalently f can be described as log-concave through $f(x) = \exp\left(-\psi(x)\right)$ for some convex function ψ (for example as in [36]).
The convention $\log 0 := -\infty$ is used and therefore $\exp\left(-\infty\right) = 0$ is well-defined.

A different definition of log-concavity is provided through the following corollary.

Corollary 2.3. *A density function $f : \mathbb{R}^d \to [0;\infty)$ is log-concave if and only if*

$$f(\alpha x_1 + (1-\alpha)x_2) \geq [f(x_1)]^{\alpha} [f(x_2)]^{1-\alpha} \tag{2.2}$$

holds for every $x_1, x_2 \in \mathbb{R}^d$ and every $\alpha \in [0;1]$.

Proof. The correctness of the corollary follows from the concavity of the function $\phi = \log f$.
Let $x_1, x_2 \in \mathbb{R}^d$ and $\alpha \in [0;1]$.

$$\begin{aligned} & \log(f(\alpha x_1 + (1-\alpha)x_2)) \quad \geq \quad \alpha \log(f(x_1)) + (1-\alpha)\log(f(x_2)) \\ \Leftrightarrow\ & \log(f(\alpha x_1 + (1-\alpha)x_2)) \quad \geq \quad \log\left((f(x_1))^{\alpha} \cdot (f(x_2))^{1-\alpha}\right) \end{aligned}$$

Due to the monotonicity of the function $\exp(\cdot)$, the inequality is fulfilled. ■

Definition 2.4. *The class of upper semi-continuous log-concave densities on $\mathbb{R}^d$ with respect to the d-dimensional Lebesgue measure is denoted by $\mathcal{F}_d$, $d \in \mathbb{N}^*$.*

Upper semi-continuity for $\mathcal{F}_d$ is not claimed throughout the whole literature. But it makes things easier as mentioned by [35]: we do not need to worry about densities that differ on a set of zero Lebesgue measure. This is also reasonable for our task, since we are basically assuming samples to come from standard probability distributions.

2.1. Distribution Families in $\mathcal{F}_1$

For the unskilled reader, the property of log-concavity might seem quite unnecessary in first. Therefore we want to discuss the natural appearance of log-concavity by proving it in several common probability distributions. Due to the application on univariate densities, which we are focusing on in this thesis, we will only consider the case of univariate random variables. Nevertheless, we want to mention that analogous results do exist for several multivariate distributions as well. For other overviews, see for example [2] or [36].

Concavity is easily provable for sufficiently smooth functions using the second derivative. We will recap the corresponding basic theorem but need the following lemma in advance to prove it afterwards.

Lemma 2.5. *A real function $f : D \to \mathbb{R}$ ($D \subseteq \mathbb{R}$) is concave if and only if*

$$\frac{f(z) - f(x)}{z - x} \geq \frac{f(y) - f(z)}{y - z}$$

for every $x < z < y \in D$.

Proof. Let $\alpha \in (0;1)$, $x < y \in D$ and set $z := \alpha x + (1-\alpha)y$.
Then the following inequalities are equivalent:

$$\begin{aligned}
f(\underbrace{\alpha x + (1-\alpha)y}_{=z}) &\geq \alpha f(x) + (1-\alpha)f(y) \\
f(z) &\geq \alpha f(x) + (1-\alpha)f(y) \\
(y-x)f(z) &\geq \alpha(y-x)f(x) + (1-\alpha)(y-x)f(y) \\
(y-x)f(z) &\geq (\alpha y - \alpha x)\, f(x) + [(1-\alpha)y - (1-\alpha)x]\, f(y) \\
(y-x)f(z) &\geq \left[y - \underbrace{(\alpha x + (1-\alpha)y)}_{=z}\right] f(x) + \left[\underbrace{\alpha x + (1-\alpha)y}_{=z} - x\right] f(y) \\
0 &\geq (y-z)f(x) + (z-x)f(y) - (y \underbrace{-z+z}_{=0} - x)f(z) \\
0 &\geq (z-x)(f(y)-f(z)) - (y-z)(f(z)-f(x)) \\
\frac{f(z)-f(x)}{z-x} &\geq \frac{f(y)-f(z)}{y-z}.
\end{aligned}$$

Since the first inequality is the definition of concavity of f, the equivalency of the lemma is proven. ■

We are now able to introduce the mentioned theorem that provides an easy way to prove concavity of smooth functions.

Theorem 2.6. *A real, twice differentiable function $f : D \to \mathbb{R}$ $(D \subseteq \mathbb{R})$ is concave if and only if $f''(x) \leq 0$ for every $x \in D$.*

Proof.
"⇒:"
Let $f : D \to \mathbb{R}$ be concave and twice differentiable and choose $x < y \in D$.
Then $f'(x) = \lim\limits_{z \to x^+} \frac{f(z)-f(x)}{z-x}$ and $f'(y) = \lim\limits_{z \to y^-} \frac{f(y)-f(z)}{y-z}$. But from Lemma 2.5, we know that

$$\frac{f(z)-f(x)}{z-x} \geq \frac{f(y)-f(z)}{y-z}$$

holds for every $z \in (x;y)$. Therefore we know that $x < y \Rightarrow f'(x) \geq f'(y)$, which means that the function f' is monotonically decreasing $\Leftrightarrow f'' \leq 0$.
"⇐"
Choose $x < z < y \in D$. Because of the mean value theorem, there exist $\xi_1 \in (x;z)$ and $\xi_2 \in (z;y)$, such that

$$\frac{f(z)-f(x)}{z-x} = f'(\xi_1) \text{ and } \frac{f(y)-f(z)}{y-z} = f'(\xi_2).$$

We know that $f''(x) \leq 0$ and therefore f' is a monotonically decreasing function. Since $\xi_1 < \xi_2$ by construction, we know that $f'(\xi_1) \geq f'(\xi_2)$, which leads to the concavity of f through Lemma 2.5. ■

Now we are able to analyze several interesting standard distribution families with respect to log-concavity. The list below is by far not complete, but we will refer to these distributions later, especially in Chapter 5.

Theorem 2.7. *Let f be the density function of a given probability distribution.*

(a) Normal Distribution $N(\mu, \sigma^2)$:
$f : \mathbb{R} \to \mathbb{R}_0^+, x \mapsto \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ *($\mu \in \mathbb{R}$ and $\sigma^2 \in \mathbb{R}^+$) is log-concave.*

(b) Generalized Pareto Distribution:
$f : [\mu; \infty) \to \mathbb{R}_0^+,\ x \mapsto \frac{1}{\sigma}\left(1 + \frac{\gamma(x-\mu)}{\sigma}\right)^{-(1+\frac{1}{\gamma})}$ *if $\gamma > 0$,*
$f : \left[\mu; \mu - \frac{\sigma}{\gamma}\right) \to \mathbb{R}_0^+,\ x \mapsto \frac{1}{\sigma}\left(1 + \frac{\gamma(x-\mu)}{\sigma}\right)^{-(1+\frac{1}{\gamma})}$ *if $\gamma < 0$ and*
$f : [\mu; \infty) \to \mathbb{R}_0^+,\ x \mapsto \frac{1}{\sigma} e^{-\frac{x-\mu}{\sigma}}$ *if $\gamma = 0$*
($\mu \in \mathbb{R}$, $\sigma \in \mathbb{R}^+$)
is log-concave for $\gamma \in [-1; 0]$.

(c) Gumbel Distribution:
$f : \mathbb{R} \to \mathbb{R}_0^+, x \mapsto \frac{1}{\sigma} \exp\left(-\frac{x-\mu}{\sigma} - e^{-\frac{x-\mu}{\sigma}}\right)$ *($\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$) is log-concave.*

(d) Frechet Distribution:
$$f : \mathbb{R} \to \mathbb{R}_0^+, x \mapsto \begin{cases} \frac{\alpha}{\sigma}\left(\frac{x-m}{\sigma}\right)^{-\alpha-1} e^{-\left(\frac{x-m}{\sigma}\right)^{-\alpha}} & x > m \\ 0 & x \leq m \end{cases}$$
($m \in \mathbb{R}, \sigma, \alpha \in \mathbb{R}^+$)
is not log-concave.

(e) Reversed Weibull Distribution:
$$f : \mathbb{R} \to \mathbb{R}_0^+, x \mapsto \begin{cases} \frac{\alpha}{\sigma}\left(\frac{x-m}{\sigma}\right)^{\alpha-1} e^{-\left(\frac{x-m}{\sigma}\right)^{\alpha}} & x > m \\ 0 & x \leq m \end{cases}$$
($m \in \mathbb{R}, \sigma \in \mathbb{R}^+ \alpha \in \mathbb{R}^+$)
is log-concave for $\alpha \geq 1$.

Proof. To prove the statements, we make use of Theorem 2.6 by checking the second derivative of the function $\log f$.

(a) $[\log f(x)]'' = -\frac{1}{\sigma^2} \leq 0 \Rightarrow \log f$ is concave $\Rightarrow f$ is log-concave.

(b) If $\gamma \neq 0$, then

$$[\log f(x)]'' = \underbrace{\gamma(\gamma+1)}_{\leq 0 \text{ for } \gamma \in [-1;0]} \underbrace{\frac{1}{\sigma^2\left(1+\frac{\gamma x}{\sigma}\right)^2}}_{>0}.$$

If $\gamma = 0$, then

$$[\log f(x)]'' = 0.$$

Therefore, f is log-concave for $\gamma \in [-1; 0]$.

(c) $[\log f(x)]'' = -\frac{1}{\sigma^2} \cdot e^{-\frac{x-\mu}{\sigma}} \leq 0 \Rightarrow \log f$ is concave $\Rightarrow f$ is log-concave.

(d) For $x > m$:

$$\begin{aligned}
[\log f(x)]'' &= (x-m)^{-2} \cdot (1+\alpha) - \frac{\alpha}{\sigma^2} \cdot (1+\alpha) \cdot \left(\frac{x-m}{\sigma}\right)^{-\alpha-2} \\
&= \underbrace{(1+\alpha) \cdot (x-m)^{-2}}_{>0} \cdot \left(1 - \alpha \cdot \left(\frac{x-m}{\sigma}\right)^{-\alpha}\right).
\end{aligned}$$

To check whether $[\log f(x)]'' \leq 0$ or not, we analyze the sign of

$$1 - \alpha \cdot \left(\frac{x-m}{\sigma}\right)^{-\alpha}. \tag{2.3}$$

Actually (2.3) is non-positive only for some x:

$$\begin{aligned}
\alpha \left(\frac{x-m}{\sigma}\right)^{-\alpha} &\geq 1 \\
\Leftrightarrow \quad x &\leq m + (\alpha)^{\frac{1}{\alpha}} \cdot \sigma.
\end{aligned}$$

Therefore, the density is only log-concave on $\left(m; m + \sigma \cdot (\alpha)^{\frac{1}{\alpha}}\right]$ and thus f is not log-concave in general.

(e) Analogous to the Frechet-distribution:

$$[\log f(x)]'' = \underbrace{(1-\alpha)}_{\leq 0 \text{ for } \alpha \geq 1} \cdot \underbrace{(x-m)^{-2}}_{>0} \cdot \underbrace{\left(1 + \alpha \cdot \left(\frac{x-m}{\sigma}\right)^{\alpha}\right)}_{>0}.$$

This time, the sign of the last expression is clear for every $x > m$ since $\alpha \cdot \left(\frac{x-m}{\sigma}\right)^{\alpha} > 0$.

This leads to the desired result: f is log-concave if $\alpha \geq 1$. ■

Of course, the densities mentioned in Theorem 2.7 are by far not the complete set of densities in $\mathcal{F}_1$. Some other exemplary elements of $\mathcal{F}_1$, according to [35], are the logistic density, $\beta(a, b)$-densities with $a, b \geq 1$ and $\Gamma(\alpha, \lambda)$-densities with $\alpha \geq 1$. These and many other examples can easily be verified analogous to Theorem 2.7.

To give an idea of how graphs of log-concave distributions look like, we present two of the above densities in Figure 2.1. The normal distribution is log-concave for every parametrization, which can easily be verified by the obvious concavity of the graph of $\log f$. For the Frechet distribution, we deduced an area of log-concavity for some x in the proof of Theorem 2.7 (d). For our exemplary parametrization $(m = 1, \sigma = 2, \alpha = \frac{1}{2})$, we know that local log-concavity occurs for $x \leq m + (\alpha)^{\frac{1}{\alpha}} \cdot \sigma = 1 + \left(\frac{1}{2}\right)^2 \cdot 2 = 1.5$. This fact can be seen in Figure 2.1d.

Note that Theorem 2.7 (b) and (c) to (e) give results for the two distribution families used in extreme value theory, since the Generalized Extreme Value Distribution (GEVD) is a combination of the Gumbel-, the Frechet- and the Reversed Weibull-distribution. We will discuss this fact in detail in Chapter 5.

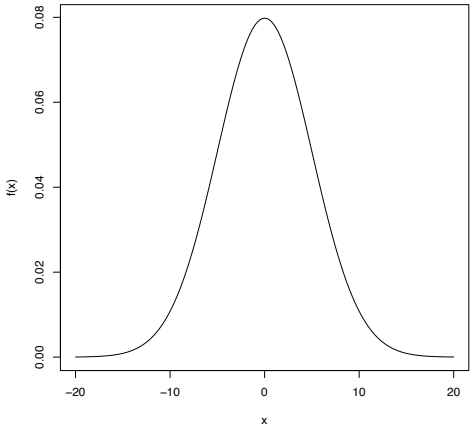
2.2. Properties of $\mathcal{F}_1$

As already mentioned earlier, we will focus on univariate densities and therefore the class $\mathcal{F}_1$. Nevertheless, we will present some results which hold for the higher dimensional case as well to give an insight in the wide range of desirable properties of densities in $\mathcal{F}_d$. For some key results in the literature we will present a theorem and just refer to the original proof when we do not need the result in later sections. For results where a deeper insight in the proof is advantageous, we will present the proof after the theorem.

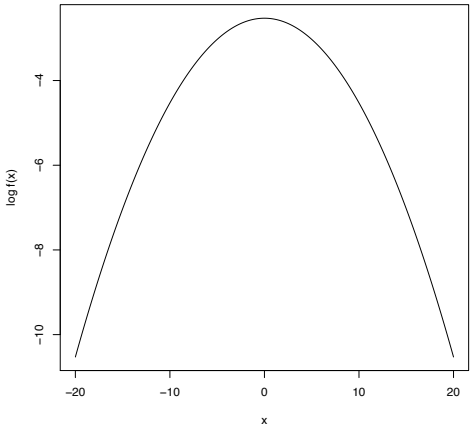
The following statement, first published in [20] can be seen as an alternative characterization of log-concave densities on $\mathbb{R}$. Due to this characterization, according to [44], "log-concave densities are sometimes referred to as strongly unimodal."

Theorem 2.8. *Let f be a density function on $\mathbb{R}$. Then $f \in \mathcal{F}_1$ if and only if the convolution $f * g$ is unimodal for every unimodal density g.*

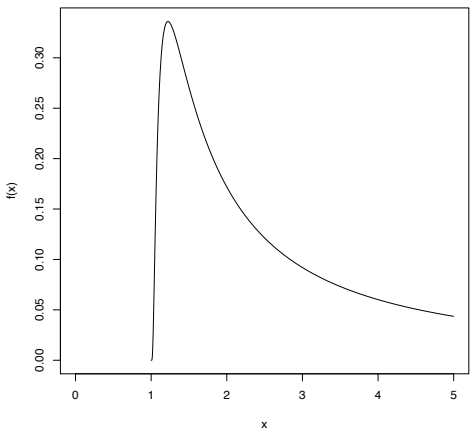
Proof. See [20]. ■



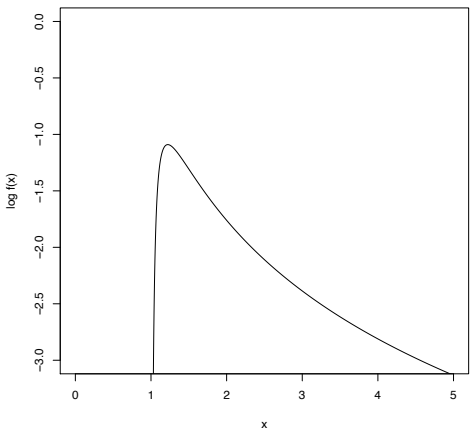
(a) Normal Distribution with $\mu = 0$ and $\sigma = 5$: density function.



(b) Normal Distribution with $\mu = 0$ and $\sigma = 5$: log density function.



(c) Frechet Distribution with $m = 1$, $\sigma = 2$ and $\alpha = \frac{1}{2}$: density function.



(d) Frechet Distribution with $m = 1$, $\sigma = 2$ and $\alpha = \frac{1}{2}$: log density function.

Figure 2.1.: Two exemplary distributions to illustrate the graphs of log-concave densities.

For the multivariate case, a similar characterization is available. According to [35], it is "convenient to think of log-concave densities as unimodal[1] densities with exponentially decaying tails." The property of exponentially decaying tails is justified by the following theorem.

[1]Unimodality is meant in the sense of the upper level sets being convex.

Theorem 2.9. *For any $f \in \mathcal{F}_d$, $d \in \mathbb{N}^*$, there exists $\alpha > 0$, $\beta \in \mathbb{R}$ such that $f(x) \leq e^{-\alpha||x||+\beta}$ for all $x \in \mathbb{R}^d$*

Proof. See [6]. ■

The class of log-concave distributions is closed under convolution. The following theorem by [27] can be used to easily see this fact.

Theorem 2.10. *Let $d = d_1 + d_2$ for $d_1, d_2 \in \mathbb{N}$, and let $f : \mathbb{R}^d \to [0; \infty)$ be log-concave. Then*

$$x \mapsto \int_{\mathbb{R}^{d_2}} f(x, y)\, \mathrm{d}y$$

is log-concave on $\mathbb{R}^{d_1}$.

Proof. See [27]. ■

Of course, this result can be used in different ways but it is also worth mentioning it by itself. Nevertheless we are interested in the behavior of log-concave densities under convolution because it plays a key role in the smoothing of log-concave densities by ensuring that a log-concave function is still log-concave after applying possible smoothing procedures.

Corollary 2.11. *If f and g are log-concave densities on $\mathbb{R}^d$ with $d \in \mathbb{N}$, then their convolution $f * g$ is a log-concave density on $\mathbb{R}^d$.*

Proof. Since f and g are log-concave, it is easy to show that the function $(x, y) \mapsto f(x - y)\, g(y)$ is also log-concave. Applying Theorem 2.10 leads to the desired result. ■

The next theorem was published in [43], where it is fundamental for a statistical test, that has the detection of the presence of mixtures as target. We will introduce this test, implement it in R and apply it to several artificial and real datasets in Chapter 4.

Theorem 2.12. *Let f_i, $i \in \{1, \ldots, m\}$ be log-concave densities on $\mathbb{R}^d$ and $0 \leq p_i \leq 1$ for $i = 1, \ldots, m$ with $\sum_{i=1}^m p_i = 1$. Then on any compact set $G \subseteq \cap_{i=1}^m supp\,(f_i)$ the representation*

$$f(x) := \sum_{i=1}^{m} p_i f_i(x) = \exp\left(\phi(x) + c\,||x||^2\right) \tag{2.4}$$

holds for a concave function ϕ on $\mathbb{R}^d$ and a constant $c \geq 0$.

Proof. (from [43]) Let $\phi_i = \log f_i$ for any $i \in \{1, \dots m\}$ and $M := \max_G e^{\phi_i}$. Set $x_\alpha := \alpha x_1 + (1-\alpha)x_2$ and $\phi_{i,\alpha} := \alpha\phi_i(x_1) + (1-\alpha)\phi_i(x_2)$ for any $x_1, x_2 \in G$ and $\alpha \in (0;1)$. Note that $\phi_i(x_\alpha) \neq \phi_{i,\alpha}$ for non-linear ϕ.

Next, we define $F(t) := e^t - M\frac{t^2}{2}$ on $(-\infty; \log M]$. Since $F''(t) = e^t - M \leq 0$ on $(-\infty; \log M]$, the function $D(\cdot)$ is concave by Theorem 2.6.

By assumption, f_i is log-concave and therefore ϕ is concave, thus

$$\begin{aligned} f_i(x_\alpha) &\geq e^{\phi_{i,\alpha}} \\ &= F(\phi_{i,\alpha}) + M\frac{\phi_{i,\alpha}}{2} \\ &\geq \alpha F(\phi_i(x_1)) + (1-\alpha)F(\phi_i(x_2)) + M\frac{\phi_{i,\alpha}^2}{2} \\ &= \alpha f_i(x_1) + (1-\alpha)f_i(x_2) - M\alpha(1-\alpha)\frac{(\phi_i(x_1) - \phi_i(x_2))^2}{2}. \end{aligned}$$

By theorem 41D in [30], ϕ_i is Lipschitz with some constant L_i. Therefore we can continue the above inequality chain:

$$\begin{aligned} &\geq \alpha f_i(x_1) + (1-\alpha)f_i(x_2) - L_i^2 M\alpha(1-\alpha)\frac{||x_1 - x_2||^2}{2} \\ &= \alpha\left(f_i(x_1) - L_i^2 M\frac{||x_1||^2}{2}\right) + (1-\alpha)\left(f_i(x_2) - L_i^2 M\frac{||x_2||^2}{2}\right) + L_i^2 M\frac{||x_\alpha||^2}{2} \end{aligned}$$

$$\Leftrightarrow f_i(x_\alpha) - L_i^2 M\frac{||x_\alpha||^2}{2} \geq \alpha\left(f_i(x_1) - L_i^2 M\frac{||x_1||^2}{2}\right) + (1-\alpha)\left(f_i(x_2) - L_i^2 M\frac{||x_2||^2}{2}\right).$$

Thus we proved that $f_i(x) = \psi_i(x) + b_i||x||^2$ for a concave function $\psi_i = f_i - L_i^2 M\frac{||\cdot||^2}{2}$ on G and $b_i = \frac{L_i^2 M}{2} \geq 0$. Since this calculations hold for any $i \in \{1, \dots m\}$, also $\sum_{i=1}^m p_i f_i(x) = \psi(x) + b||x||^2$ on G for a concave function $\psi = \sum_{i=1}^m p_i\psi_i$ and $b = \sum_{i=1}^m p_i b_i \geq 0$.

Now we just need to show that

$$\log\left(\psi(x) + b||x||^2\right) - c||x||^2 =: \phi(x)$$

is concave for some $c \geq 0$. The concavity of ψ implies the existence of $0 \leq D \leq b\left(||x||^2\right)_\alpha - b||x_\alpha||^2$ with $\psi(x_\alpha) + b||x_\alpha||^2 \geq \psi_\alpha + b\left(||x||^2\right)_\alpha - D \geq \min_G\left(\psi(x) + b||x||^2\right)$, where $\left(||x||^2\right)_\alpha := \alpha||x_1||^2 + (1-\alpha)||x_2||^2$. Thus

$$\begin{aligned} \log\left(\psi(x_\alpha) + b||x_\alpha||^2\right) &\geq \log\left(\psi_\alpha + b\left(||x||^2\right)_\alpha - D\right) \\ &\geq \log\left(\psi_\alpha + b\left(||x||^2\right)_\alpha\right) - \frac{D}{\min_G\left(\psi(x) + b||x||^2\right)} \\ &\geq \left(\log\left(\psi(x) + b||x||^2\right)\right)_\alpha - c\left(||x||^2\right)_\alpha + c||x_\alpha||^2, \end{aligned}$$

where $c := \frac{b}{\min_G\left(\psi(x)+b\|x\|^2\right)} < \infty$. The second-to-last inequality follows from the fact that $\log(y-D) \geq \log(y) - \frac{D}{m}$ for $y \geq 0$ and $y - D \geq m > 0$.

The resulting inequality describes the concavity of the function ϕ and thus the proof is complete. ■

2.3. Statistical Tests on Logarithmic Concavity

There are several tests for log-concavity proposed in the literature. We will sketch one of these concepts as described in [1]. This concept focuses on positive-valued, univariate random variables but analogous tests based on multivariate samples are available as well, for example in [7]. Note that samples in our use case are not necessarily positive so we will use a different way of testing, which we will describe in the end of this section. The following framework is nevertheless worth to be mentioned and is presented in analogy to some of the theorems in the previous section, which were also presented but will not all be used in the overall algorithm that is developed in Chapter 6.

Note that random variables in this chapter are assumed to be positive-valued. Therefore we assume $F(0) = 0$ for distribution functions in this section. As this concept comes from reliability theory, we recap the following additional definitions:

Definition 2.13. *Let $X_1, \ldots, X_n$ be independent identically distributed (i.i.d.) random variables with density function f on $\mathbb{R}$ with support $supp(f) = [a;b] \subset \mathbb{R}$ and F its distribution function (= CDF - Cumulative Density Function). Moreover, let $X_{(1)}, \ldots, X_{(n)}$ be the order statistics of $X_1, \ldots, X_n$.*

- $S : [a;b] \to [0;1], x \mapsto 1 - F(x)$ *is called the* ***survival function****.*
- $h : [a;b) \to \mathbb{R}_0^+, x \mapsto \frac{f(x)}{S(x)}$ *is called the* ***hazard function****.*
- $D_i := (n-i+1)(X_{(i)} - X_{(i-1)}),\ i = 1, \ldots, n$ *(take $X_{(0)} \equiv 0$) are the* ***normalized spacings of the order statistics****.*

The following theorem provides the main idea of testing on log-concavity in reliability theory:

Theorem 2.14. *For a density function f and the corresponding survival function S and hazard function h, the following statements hold:*

$$f \text{ is log-concave} \Rightarrow h \text{ is non-decreasing} \Leftrightarrow S \text{ is log-concave}$$

Proof. "f is log-concave $\Rightarrow$ h is non-decreasing"
Choose two points $x_1, x_2 \in supp(f)$ with $x_1 < x_2$. Note that by definition

$$S(x) = 1 - \int_0^x f(y)\mathrm{d}y = \int_x^\infty f(y)\mathrm{d}y = \int_0^\infty f(x+u)\mathrm{d}u$$

and thus

$$f(x_2)S(x_1) - f(x_1)S(x_2) = \int_0^\infty \left[\underbrace{f(x_1+u)f(x_2) - f(x_1)f(x_2+u)}_{\geq 0}\right] \mathrm{d}u \geq 0. \tag{2.5}$$

The inequality $f(x_1+u)f(x_2) - f(x_1)f(x_2+u) \geq 0$ is true since $f(x_1+u)f(x_2) \geq f(x_1)f(x_2+u)$ due to Corollary 2.3: we can apply the corollary twice to the two outer points x_1 and x_2+u. When we choose $\alpha = \frac{x_1-x_2}{x_1-x_2-u}$, we get

$$f\left(x_1+u\right) \geq \left(f(x_1)\right)^{\frac{x_1-x_2}{x_1-x_2-u}} \cdot \left(f(x_2+u)^{\frac{-u}{x_1-x_2-u}}\right.$$

but when we choose $\alpha = \frac{-u}{x_1-x_2-u}$, we get

$$f\left(x_2\right) \geq \left(f(x_1)\right)^{\frac{-u}{x_1-x_2-u}} \cdot \left(f(x_2+u)\right)^{\frac{x_1-x_2}{x_1-x_2-u}}.$$

Multiplying the two equations, leads to the desired result.

The inequality in (2.5) gives

$$\begin{aligned} f(x_2)S(x_1) - f(x_1)S(x_2) &\geq 0 \\ f(x_2)S(x_1) &\geq f(x_1)S(x_2) \\ \frac{f(x_2)}{S(x_2)} &\geq \frac{f(x_1)}{S(x_1)} \\ h(x_2) &\geq h(x_1), \end{aligned}$$

which describes the property of h being non-decreasing since $x_1 < x_2$.

"h is non-decreasing $\Leftrightarrow$ S is log-concave"
Note that

$$[\log S(x)]' = -\frac{f(x)}{S(x)} = -h(x).$$

The function h being non-decreasing is equivalent to $-h'(x) \leq 0$ and thus equivalent to $[\log S(x)]'' \leq 0$. ■

Theorem 2.14 provides an idea of testing on log-concavity of a density function. Since log-concavity is a necessary condition for h being non-decreasing and thus S being log-concave, it is reasonable to construct tests on increasing hazard rates or concavity of $\log S$.

Such tests for instance are based on the normalized spacings of the order statistics as described in Definition 2.13. This is due to the downward trend that these values follow under the assumption of log-concavity.

Theorem 2.15. *Let $D_1, \ldots, D_n$ be the normalized spacing of the order statistics of $X_1, \ldots, X_n$ from a distribution with density function f.*

Under the assumption of f being log-concave,

$$P(D_i \leq D_j) < \frac{1}{2} < P(D_i > D_j)$$

holds for all $1 \leq i < j \leq n$.

Proof. (from [1]) The main trick, first suggested by [28], is to transform X to the random variable $Z = s(X) := -\log S(X)$, where S denotes the survival function of X. By construction, $s(x)$ is differentiable and by Theorem 2.14 convex and strictly increasing ($h(x) \leq 0$ by definition and thus $(-\log S)' = h(x) \geq 0$). Due to the monotonicity of the logarithm, $Z_{(i)} = s(X_{(i)})$ for all $i \in \{1, \ldots, n\}$.

First, we prove that $Z \sim Exp(\lambda = 1)$ (Z is obviously i.i.d. as well as X). Therefore we formulate the distribution function of the random variable Z:

$$\begin{aligned} F_Z(z) &= P(Z \leq z) \\ &= P\left(-\log S(X) \leq z\right) \\ &= P\left(S(X) \geq e^{-z}\right) \\ &= P\left(1 - F(X) \geq e^{-z}\right) \\ &= P\left(X \leq F^{-1}(1 - e^{-z})\right) \\ &= 1 - e^{-z}. \end{aligned}$$

Thus Z is exponentially distributed with shape parameter $\lambda = 1$.

Second, we prove that for the normalized spacings of the order statistics $B_1, \ldots, B_n$ of the random variable Z, $B \sim Exp(\lambda = 1)$ holds as well. Therefore, recap that the joint density function of the order statistics of the random variable Z with density function g can be expressed by

$$g_O(z_1^{(O)}, \ldots, z_n^{(O)}) = n! \prod_{i=1}^{n} g(z_i^{(O)}) \mathbb{1}_{\left[0 \leq z_1^{(O)} \leq \cdots \leq z_n^{(O)}\right]}(z_1^{(O)}, \ldots, z_n^{(O)}).$$

Using the transformation

$$\begin{aligned} z_1^{(O)} &= \frac{b_1}{n} \\ z_2^{(O)} &= \frac{b_1}{n} + \frac{b_2}{n-1} \\ &\vdots \\ z_n^{(O)} &= \frac{b_1}{n} + \frac{b_2}{n-1} + \cdots + d_n \end{aligned}$$

gives the joint density function of B:

$$g_B(b_1, \ldots, b_n) = e^{-\sum_{i=1}^n b_i} \prod_{i=1}^n \mathbb{1}_{[b_i \geq 0]}(b_1, \ldots, b_n)$$

and thus, $B \sim Exp(\lambda = 1)$.

Now we have all we need to prove the theorem. Since $B \sim Exp(\lambda = 1)$, it follows from the memorylessness of the exponential distribution that for any $i < j$, $P(B_i > B_j) = \frac{1}{2} = P(B_i < B_j)$. But since $s(x)$ is convex, for any $i < j$ with $B_i > B_j$ implies that $D_i > D_j$ but not vice versa. Therefore $P(D_i > D_j) > P(B_i > B_j) = \frac{1}{2}$. ■

So we can summarize that the downward trend of the D_i's as described in Theorem 2.15 is in favor of log-concavity of the density function f. On this fact, one can set up many different test statistics, which catch the behavior of the D_i's in terms of monotonicity as mentioned above. We will mention three test statistics, that were also mentioned in [1].

1. Proschan and Pyke [28]
$$P_n = \sum_{i<j} \mathbb{1}_{(D_i > D_j)},$$

2. Epstein [15]
$$E_n = \sum_{i=1}^{n-1} \sum_{j=1}^{i} \frac{D_j}{\sum_k D_k}, \text{ and}$$

3. Bickel and Doksum [10]
$$B_n = \sum_{i=1}^{n} i \cdot \log\left(1 - \frac{R_i}{n+1}\right),$$
where R_i denotes the rank of D_i among all $D_1, \ldots, D_n$.

The previous framework of testing is an interesting approach, but it is not generally applicable for the scope of this thesis, since we are not only focusing on positive-valued random variables. Thus, we will choose a more intuitive approach to test if procedures with the assumption of log-concavity are applicable. The main idea of this test comes from [7]. For a given sample $x_1, \ldots, x_n$ from any density f, we can easily compute the log-concave density estimator, as we will present it in the next chapter. Sampling from this estimated density gives us the sample $y_1, \ldots, y_n$. The basic idea is now, that if f is log-concave, $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ should "look similar". If f is significantly not log-concave, the samples should also differ significantly. To test, whether the samples come from the same distribution or not, we apply the two-sample Kolmogorov-Smirnov test.

3. Logarithmic Concave Density Estimation

In continuation to the last chapter, where we introduced the class $\mathcal{F}_1$ itself, we will now focus on the estimation of density functions, from which an independent identically distributed (i.i.d.) sample $x_1, \ldots, x_n$ of size n is available. The task of density estimation can be seen as one of major interest for statisticians. For example the book "Density Estimation for Statistics and Data Analysis" [39] can be seen as "an ideal reference for practitioners" [9] and is cited over 25000 times (July, 2020) according to Google Scholar [18]. This fact should give a feeling for the importance of the overall topic of density estimation in general.

First we will give an overview of parametric density estimation techniques, and then introduce the log-concave density estimator. Afterwards we will discuss the computation of the log-concave density estimator and introduce a smoothing procedure suggested in [11]. Finally we will compare this non-parametric density estimator with the ones defined in the beginning to motivate its use in the next chapters. We will mainly use the same notation and approach as in [35] but combine it with theorems from [33], [11] and [32].

3.1. Parametric Density Estimation Techniques

One of the most "straight-forward" and intuitive ways to estimate densities, is to simply illustrate the available observations graphically. Probably the most used illustration technique in this sense are histograms and they generally work as follows: the observations are assigned to bins and the weights of these bins are illustrated as bars over the range of assigned observations to this bin. The height of the bins can either be defined by the absolute or relative number of observations in this bin or in a way that the area under such a bar is determined by the relative appearance of observations in this bin throughout the whole set of observations. Since we want to estimate densities, we focus on the second alternative because

then the area of all bars sums up to 1, in analogy to densities having the property $\int_{\mathbb{R}} f(x)\mathrm{d}x = 1$.

Even if histograms are widely used to get a first impression of observed data, they come with a variety of shortcomings. First of all, despite of their low level of complexity, they are parametric. To create a histogram, one has to determine the properties of the bins, which are precisely the number of bins and the size of each bin. Assuming the bins to be of equal width, one needs to determine two parameters:

(i) the number of bins, m and

(ii) the width of each bin, h.

To determine these two parameters, a wide class of rules can be applied. We will mention just two general cases. First, a basic "rule of thumb" is **Sturges' rule** [41]:

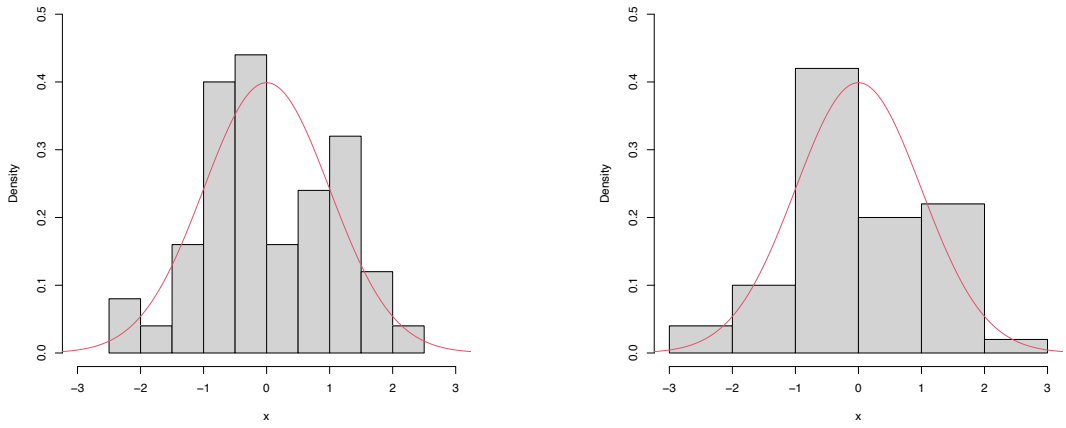
- Choose the number of bins as $m = 1 + \lceil \log_2(n) \rceil$ with n being the number of observations and $\lceil \cdot \rceil$ the ceiling function.
- Afterwards, choose the width of the bins equidistantly between the smallest and the largest observation: $h = \frac{x_{(n)} - x_{(1)}}{m}$.

Contrary to the above rule, that is focused on the choice of m, another alternative is to determine the width of the bins first and afterwards the number of bins accordingly. This concept is applied in **Scott's rule** [38]:

- Choose the width of each bin as $h = \frac{3.5 \cdot s}{\sqrt[3]{n}}$ with s being the empirical standard deviation of the sample.
- Then the number of bins is determined by $m \approx \frac{x_{(n)} - x_{(1)}}{h}$.

To show the impact of the choice of the parameters, we draw a $N(0, 1)$-sample of size 50, apply the two rules above and compare the results. In Figure 3.1, the resulting histograms are shown. We see that the histograms are quite different since the first one even lets suggest the presence of two modes.

In addition to the shortcoming of being a parametric procedure, we can detect another disadvantage of histograms in Figure 3.1. The discrete bars of a histogram are not very likely to give satisfying results (because of its discreteness) when one needs to use the estimated density as a function in some further analysis, e.g. see Chapter 5.



(a) Histogram using Sturges' rule.

(b) Histogram using Scott's rule.

Figure 3.1.: Comparison of two different rules for determining the parameters of a histogram for a $N(0,1)$-sample of size 50 and the theoretical density (red).

To fix this problem we obviously need techniques that fit a continuous density function to a given sample. A widely used but nevertheless parametric procedure is kernel density estimation.

Definition 3.1. *Given a sample $x_1, \ldots, x_n$ of n i.i.d. observations from a continuous probability distribution with density function f, we call*

$$\hat{f}_n^K(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

*the **kernel density estimator** with bandwidth $h \in \mathbb{R}^+$ and kernel function $K : \mathbb{R} \to \mathbb{R}_0^+$.*

For the kernel function K,

$$\int_{-\infty}^{\infty} K(x)\mathrm{d}x = 1 \text{ and } \int_{-\infty}^{\infty} K^2(x)\mathrm{d}x < \infty$$

must hold.

The definition itself already gives a feeling how the estimator works. When we want to estimate the value of the density function at a specific point x_0, other points should have an impact on this estimation according to their distance from

x_0. We could say that we add n kernel functions, that have their centers at each observation and a radius where they are significantly large enough. This radius of the kernel function is determined by the bandwidth h.

One of the most frequently used kernel functions is the "Gaussian kernel"

$$K_G(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}},$$

which is also the default kernel in R [29] when using the function `density()`.

It is easy to see, that the bandwidth h is in place of the standard deviation of a $N(\mu, \sigma^2)$-density when K_G is used in Definition 3.1. Therefore we can think of the bandwidth as the standard deviation of every kernel function, that is assigned to the observations. Naturally the value of h strongly correlates with the number of peaks that f_n^K has. Thus, even if we get a smooth and continuous density estimation, we are still highly dependent on the choice of the kernel function K and especially the bandwidth h.

For the bandwidth, there are also some "rules of thumb" available. One of them is "Scott's rule" [37]:

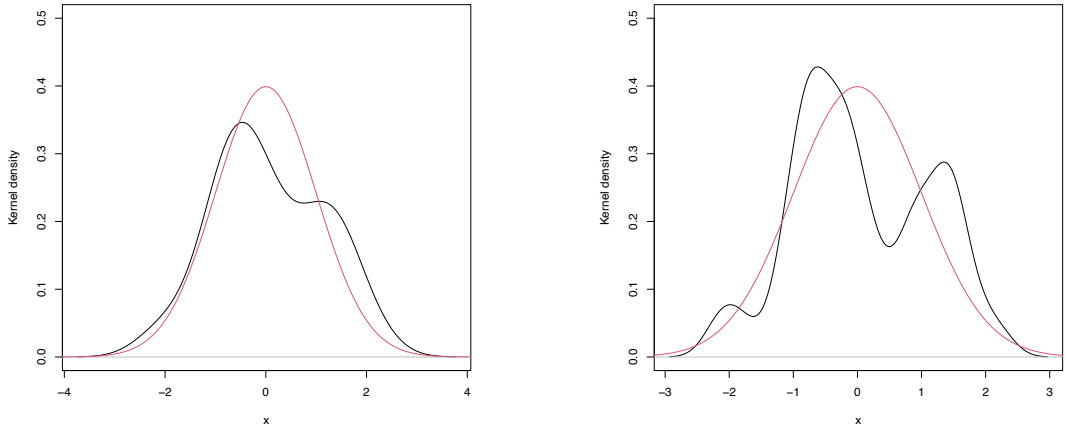
$$h = \left(\frac{4}{3}\right)^{\frac{1}{5}} \cdot s \cdot n^{-\frac{1}{5}}$$

where s is the empirical standard deviation of the sample.

Figure 3.2 and Figure 3.3 show the kernel density estimators with different kernels and bandwidths for the same data as in Figure 3.1. We see that the dependency from parameters can lead to massive failure in estimation. For example mulimodality can be detected misleadingly when one reviews such an estimation or the other way around, modes could become invisible due to a too large bandwidth. All of this should motivate the introduction of a nonparametric density estimation technique, which we are focusing on in this thesis.

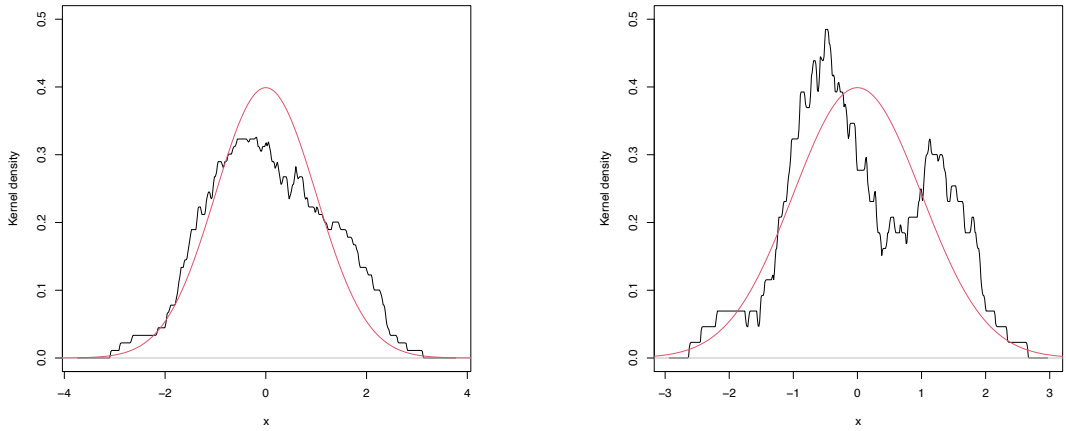
Note: the kernel density estimator does not necessarily need to be as smooth as in Figure 3.2. In Figure 3.3, kernel density estimation is applied to the same data with the same bandwidths but using a non-smooth kernel: the "rectangular kernel":

$$K_R(x) = \begin{cases} \frac{1}{2} & \text{for } |x| \leq 1 \\ 0 & \text{else} \end{cases}.$$



(a) Bandwidth: $h \approx 0.512$ (Scott's rule). (b) Bandwidth: $h = 0.25$.

Figure 3.2.: Kernel density estimators with Gaussian kernel and different bandwidths (true density in red).



(a) Bandwidth: $h \approx 0.512$ (Scott's rule). (b) Bandwidth: $h = 0.25$.

Figure 3.3.: Kernel density estimators with rectangular kernel and different bandwidths (true density in red).

3.2. The Logarithmic Concave Maximum Likelihood Estimator

"To free practitioners from restrictive parametric [...] assumptions" [35], we now

want to introduce a non-parametric method as a solution to the problems, that occurred with parametric procedures in the last section. Thus, we now want to consider a sample of i.i.d. observations $x_1, \dots, x_n$ from a distribution with log-concave density f. As we saw in Chapter 2, the assumption of f being log-concave is not a tremendous shortcoming. Theorem 2.7 gave an insight in the variety of the class $\mathcal{F}_1$, which of course consists of even more probability distribution families than the ones we mentioned.

Mainly, we want to make use of the log-concavity in the context of maximum likelihood estimation. In general, we want to maximize the log-likelihood function

$$\int \log f \mathrm{d}\mathbb{F}_n = \int \phi \mathrm{d}\mathbb{F}_n \tag{3.1}$$

where $\mathbb{F}_n$ is the empirical distribution function of the sample and ϕ a concave function as described in Definition 2.2. A big shortcoming of the maximization problem "max (3.1)" subject to $f \in \mathcal{F}_1$, is the restriction of f being a density ($\int f = 1$).

According to [11] and first introduced in [40], Theorem 3.1, we add a Lagrange term to (3.1), leading to the following log-likelihood function:

$$L(\phi) := \int \phi \, \mathrm{d}\mathbb{F}_n - \int e^{\phi(x)} \, \mathrm{d}x$$

An intuitive explanation, that the maximization of $L(\cdot)$ actually results in a function, satisfying $\int e^{\phi(x)} \mathrm{d}x = 1$ is given in the proof of Theorem 3.5. Thus the maximization problem is free of the restriction of f being a density function.

We are now able to formulate the estimator as solution of a maximization problem over the (convex) set of all concave functions.

Definition 3.2. *Consider a sample of n i.i.d. observations $x_1, \dots, x_n$ from a distribution with log-concave density function. Then we call*

$$\hat{f}_n := e^{\hat{\psi}_n}$$

with

$$\hat{\psi}_n := \underset{\phi \text{ is concave}}{\operatorname{argmax}} \int \phi \, \mathrm{d}\mathbb{F}_n - \int e^{\phi(x)} \, \mathrm{d}x \tag{3.2}$$

*the **log-concave density estimator** of the sample $x_1, \dots, x_n$.*

Note: the log-concave density estimator $\hat{f}_n$ is sometimes referred to as "log-concave projection", see [35].

The most obvious questions in terms of user-friendliness and overall usability of this density estimation technique are the questions of existence and uniqueness of the maximizer $\hat{\psi}_n$. The following two theorems, taken from [33] (including their proofs) give answers to these questions. Moreover, Theorem 3.3 provides a handy characterization of how (unsmoothed) log-concave density estimators look like.

Theorem 3.3. *Let $\hat{\psi}_n$ be a solution of (3.2) and $x_{(1)}, \dots, x_{(n)}$ the order statistics of the underlying sample $x_1, \dots, x_n$. Then*

$\hat{\psi}_n$ is continuous and piecewise linear on $\big[x_{(1)}; x_{(n)}\big]$ and

$\hat{\psi}_n(x) = -\infty$ for $x \in \mathbb{R} \setminus \big(x_{(1)}; x_{(n)}\big)$.

Proof. (from [33]) Fix any concave function $\phi : \mathbb{R} \to [-\infty; \infty)$ with $L(\phi) < \infty$. Now define a piecewise linear function $\bar{\phi} : \mathbb{R} \to [-\infty; \infty)$, such that $\forall i \in \{1, \dots, n\} : \bar{\phi}(x_i) = \phi(x_i)$. Moreover, $\bar{\phi}$ should be linear between successive observations and $\bar{\phi} \equiv -\infty$ on $\mathbb{R} \setminus \big(x_{(1)}; x_{(n)}\big)$. By definition of concavity, $\phi(x) \geq \bar{\phi}(x)$ and thus

$$L(\phi) \leq L(\bar{\phi})$$

holds for all $x \in \mathbb{R}$ with strict inequality unless $\phi \equiv \hat{\psi}_n$. Therefore, maximizers of L must have a form such as $\bar{\phi}$. ■

We will use the following representation of such maximizers $\hat{\psi}_n$, which is a direct implication of the last representation theorem.

Corollary 3.4. *Let $\hat{\psi}_n$ be a solution of (3.2). Then, $\hat{\psi}_n$ is fully represented by the vector $\hat{\boldsymbol{\psi}}_{\boldsymbol{n}} = \Big(\hat{\psi}_n(x_i)\Big)_{i=1}^{n}$.*

Proof. Obviously the representation holds due to the piecewise linearity described in Theorem 3.3. ■

Now we are able to formulate and prove the following theorem that justifies the framework of log-concave density estimation and the concept of log-concave projections.

Theorem 3.5. *A solution of (3.2) exists and is unique.*

Proof. (from [33]) Due to Theorem 3.3, we are able to only consider such concave functions as mentioned in the theorem for proving the existence of a solution. Moreover, it suffices to only consider functions with $\int e^{\phi(x)}\mathrm{d}x = 1$: if $\phi = \phi_0 + t$ with $\exp\phi_0$ being a probability density ($\int e^{\phi_0} = 1$) and some number $t \neq 0$, it follows from the definition of L:

$$L(\phi) = \underbrace{\int \phi_0 \, \mathrm{d}\mathbb{F}_n - \int e^{\phi_0(x)} \, \mathrm{d}x}_{=L(\phi_0)} + \underbrace{\int t \, \mathrm{d}\mathbb{F}_n}_{=t} + (1 - e^t) \underbrace{\int e^{\phi_0(x)} \, \mathrm{d}x}_{=1} < L(\phi_0).$$

The last inequality holds since $t + 1 - e^t < 0 \Leftrightarrow t + 1 < e^t$, which is true (think of $t + 1$ and e^t graphically) for all $t \neq 0$. Thus, a maximizer of $L(\cdot)$ must fulfill $\int e^{\phi(x)}\mathrm{d}x = 1$.

We represent such functions with vectors as described in Corollary 3.4. Thus, since $L(\cdot)$ is continuous for the existence of a maximizer it suffices to show that

$$L(\phi) \to -\infty \left(\Leftrightarrow \int \phi \, \mathrm{d}\mathbb{F}_n \to -\infty \right)$$

whenever $||\boldsymbol{\phi}||_2 \to \infty$. $||\cdot||_2$ represents the L_2-norm on $\mathbb{R}^n$: $||\boldsymbol{x}||_2 := \sqrt{\sum_{i=1}^n x_i^2}$.

So let $\left(\boldsymbol{\phi}^{(k)}\right)_{k=1}^{\infty}$ be a sequence of such vectors with $\left|\left|\boldsymbol{\phi}^{(k)}\right|\right|_2 \to \infty$ and $\phi_i^{(k)} \to \gamma_i \in [-\infty; \infty], \forall i = 1, \ldots, n$ as $k \to \infty$.

- Suppose first that $\forall i \in \{1, \ldots, n\} : \gamma_i < \infty$. Then $\exists i \in \{1, \ldots, n\} : \gamma_i = -\infty$ since $\left|\left|\boldsymbol{\phi}^{(k)}\right|\right|_2 \to \infty$. Then $L(\boldsymbol{\phi}^{(k)}) = \frac{1}{n}\sum_{i=1}^n \phi_i^{(k)} - 1 \to -\infty$ as $k \to \infty$.
- Suppose now that $\exists j \in \{1, \ldots, n\} : \gamma_j = \infty$. Keep this index now fixed: $\gamma_j = \infty$ and $j > 1$ (for the case $j = 1$, the proof works analogously but we claim $j > 1$ due to easier indexing). Due to the piecewise linearity of ϕ

$$\begin{aligned} 1 &\geq \int_{x_{j-1}}^{x_j} \exp\left(\phi^{(k)}(x)\right) \mathrm{d}x \\ &= (x_j - x_{j-1}) \exp(\phi_j^{(k)}) \frac{1 - \exp(-\delta_k)}{\delta_k} \\ &\geq (x_j - x_{j-1}) \exp(\phi_j^{(k)})(1 + \delta_k)^{-1}, \end{aligned}$$

  where $\delta_k := \phi_j^{(k)} - \phi_{j-1}^{(k)}$. The latter inequality is a consequence of

$$\frac{1 - e^{-x}}{x} \geq \frac{1}{1 + x} \text{ for } x > 0.$$

Thus δ_k is bounded from below by $(x_j - x_{j-1})\exp(\phi_j^{(k)}) - 1$. Consequently γ_j entails that

$$\begin{aligned} -\phi_j^{(k)} - \phi_{j-1}^{(k)} &= -2\phi_j^{(k)} + \delta_k \\ &\geq -2\phi_j^{(k)} + (x_j - x_{j-1})\exp(\phi_j^{(k)}) - 1 \\ &\to \infty. \end{aligned}$$

The last expression tends to infinity since $\exp(x) - 2x \to \infty$ for $x \to \infty$. (Analogously, if $j = 1$, then $-\phi_1^{(k)} - \phi_2^{(k)}$ tends to infinity.)

These considerations show that $L(\phi^{(k)}) \to -\infty$.

Thus, we proved the existence of a solution but not its uniqueness. Therefore, observe that $L(\cdot)$ is strictly convex in the sense that

$$L(\alpha\phi_1 + (1-\alpha)\phi_2) < \alpha L(\phi_1) + (1-\alpha)L(\phi_2)$$

for any $\alpha \in (0;1)$ and two different (on an area with non-zero Lebesgue measure) concave functions $\phi_1, \phi_2 : \mathbb{R} \to [-\infty, \infty)$ such that $\int \exp\phi_i < \infty$. This is a consequence of the strict convexity of the exponential function $\exp(\cdot)$ ■

Another interesting question is the goodness of fit of the log-concave density estimator compared to the actual density. Therefore we present a theorem from [14], that should give an idea, that the log-concave density estimator actually estimates log-concave densities in a satisfying way.

Theorem 3.6. *Let f_0 be a density function on $\mathbb{R}$ with $\int_{-\infty}^{\infty} f_0(x)|\log(f_0(x))|\mathrm{d}x < \infty$ and $x_1, \ldots, x_n$ a sample from f_0. Then the log-concave density estimator $\hat{f}_n \in \mathcal{F}_1$ minimizes the Kullback-Leibler divergence*

$$d_{KL}^2(f_0, f) := \int_{-\infty}^{\infty} f_0(x)\log\left(\frac{f_0(x)}{f(x)}\right)\mathrm{d}x$$

over all $f \in \mathcal{F}_1$.

Proof. See [14]. ■

Note, that for log-concave densities f_0 the theorem provides that the Kullback-Leibler divergence (which is positive by definition) can even converge to zero as the sample size increases. This statement also justifies the approach for testing on log-concavity of a density we made at the end of Section 2.3, because samples from log-concave densities result in comparably good log-concave density estimators.

A wide variety of results on the consistency and asymptotic behavior of the log-concave density estimator exists, including results for the fit of a non-log-concave density. For further results, see for example [14], [6] or [3].

3.3. Computation of Log-Concave Density Estimators

Now we will focus on the actual computation of the log-concave density estimator in practice. First, we want to reformulate the optimization problem (3.2) to classify the optimization problem and to make clear which kind of problem has to be solved.

According to Theorem 3.3 and Corollary 3.4 we can use the transformed objective function

$$\tilde{L}(\underbrace{\psi_1, \ldots, \psi_n}_{\boldsymbol{\psi}}) := \frac{1}{n}\sum_{i=1}^{n}\psi_i - \sum_{k=1}^{n-1}\left[\left(x_{(k+1)} - x_{(k)}\right)\int_0^1 e^{(1-t)\psi_k + t\psi_{k+1}}\mathrm{d}t\right] \tag{3.3}$$

instead of the original log-likelihood function (3.2). This function needs to be maximized over the set of concave functions. Since the new, transformed likelihood function (3.3) is now defined for vectors, we need to reformulate the condition of some ψ being concave to a condition for the corresponding vector $\boldsymbol{\psi}$. The property of concavity for a piecewise linear function as ours is, is easy to imagine: the slope of the function from $x_{(1)}$ to $x_{(2)}$ needs to be non-negative and the slopes of the following sections must not be larger than each one before. The non-negativity of the first slope is clear since $\psi(x) = -\infty$ for $x \leq x_{(1)}$. In fact the restrictions to the slopes mean that

$$\begin{aligned} \frac{\psi_3 - \psi_2}{x_{(3)} - x_{(2)}} &\leq \frac{\psi_2 - \psi_1}{x_{(2)} - x_{(1)}} \\ &\vdots \\ \frac{\psi_n - \psi_{n-1}}{x_{(n)} - x_{(n-1)}} &\leq \frac{\psi_{n-1} - \psi_{n-2}}{x_{(n-1)} - x_{(n-2)}} \end{aligned}$$

are the constraints for our maximization problem.

To present the maximization problem in a brief way, we define

$$\delta_k := x_{(k+1)} - x_{(k)}, k = 1 \ldots, n-1$$

and

$$\boldsymbol{V} := \begin{pmatrix} a_1 & b_1 & c_1 & 0 & \cdots & 0 \\ 0 & a_2 & b_2 & c_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n-2} & b_{n-2} & c_{n-2} \end{pmatrix} \in \mathbb{R}^{(n-2)\times n}.$$

with $a_k := \frac{1}{\delta_k}$, $b_k := -\frac{1}{\delta_{k+1}} - \frac{1}{\delta_k}$ and $c_k := \frac{1}{\delta_{k+1}}$ for $k = 1, \ldots, n-2$.

Note that $\delta_k > 0$ is assumed for every $k \in \{1, \ldots, n-1\}$ for simplicity and therefore $\frac{1}{\delta_k}$ is well-defined[1].

Now we can define the optimization problem, one has to solve for calculating the log-concave density estimator, in a proper way:

Corollary 3.7. *The function $\hat{\psi}_n$ is a solution to the maximization problem (3.2) if and only if the corresponding vector $\hat{\boldsymbol{\psi}}_{\boldsymbol{n}}$ (according to corollary 3.4) is a solution of the following optimization problem:*

$$\begin{aligned} &\max_{\boldsymbol{\psi}\in\mathbb{R}^n} \tilde{L}(\boldsymbol{\psi}) \\ &s.t.\ \boldsymbol{V}\boldsymbol{\psi} \leq 0 \end{aligned} \tag{3.4}$$

Proof. Follows directly from the derivation in this section. ■

There are different approaches in the literature to solve the optimization problem in (3.4). An overview and comparison of possible algorithms is presented in [32], where the main part is about the "Iterative Convex Minorant Algorithm" (ICMA), which is first described in detail in [22]. We will use the ICMA in Chapter 4, but for now we want to sketch the "Active Set Algorithm", as described in [12]. This algorithm is also used in the R-package "logcondens" [13], which we will use to compute the log-concave density estimator[2].

The main component of the algorithm is the set of "active"[3] constraints of a vector $\boldsymbol{\psi} \in \mathbb{R}^n$:

$$A(\boldsymbol{\psi}) := \{j \in \{1, \ldots, n-2\} \,|\, \boldsymbol{V}_{j,\cdot} \cdot \boldsymbol{\psi} \geq 0\}.$$

[1] If $\delta_k = 0 \Leftrightarrow x_j = x_{j+1}$ for some $j \in \{1, \ldots, n-1\}$ would occur in the sample, one could simply give the observations that appear more frequently a higher weight in (3.3). This is possible since there is an implicit weighting in the factor $\frac{1}{n}$ in the first part of the function and 1 for every summand in the second part.

[2] This package is only capable of computing the log-concave density estimator for univariate densities. For multivariate densities the package "LogConcDEAD" [5] is available.

[3] Obviously eponymous for the algorithm.

Thus for feasible $\boldsymbol{\psi} \in \mathbb{R}^n$ ($\boldsymbol{V}\boldsymbol{\psi} \leq 0$) the "inactive" constraints correspond to the knots where the function changes slope. Using Newton methods, it is then easy to compute a maximizer of $\tilde{L}(\cdot)$ subject to $\boldsymbol{V}_{j,\cdot}\boldsymbol{\psi} = 0$ for all $j \in A$ for some fixed $A = A\left(\tilde{\psi}\right)$.

The algorithm works iterative. We start with a feasible vector $\boldsymbol{\psi} \in \mathbb{R}^n$ as first candidate and then maximize $\tilde{L}(\cdot)$ as described above. This maximizer is then the new candidate function if it is feasible. If not, we change the candidate function as far as possible along the line segment joining our current feasible point to the maximizer while still remaining feasible. Since this new point has a larger active set than the last one, it is reasonable to use the active set of this new vector for the optimization in the next iteration.

3.4. Smoothing Procedure

The nice property of log-concave density estimators, mentioned in Theorem 3.3, can probably be seen as unnatural for continuous densities due to the sharp peaks at the knots where the logarithm of the density changes slope. When one is asked to think of a density function, it is very likely that the upcoming function is smoother compared to the ones we get via log-concave density estimation. Thus in [11] a smoothing procedure for log-concave density estimators is proposed.

Definition 3.8. *Let $\hat{f}_n$ be the log-concave density estimator of a sample as described in Definition 3.2. Then*

$$\hat{f}_n^*(x) := \int_{-\infty}^{\infty} \phi_\gamma(x-y)\,\hat{f}_n(y)\,\mathrm{d}y$$

is called the ***smoothed log-concave density estimator****. The function ϕ_γ is the density function of a normal distribution $N(\mu, \sigma^2)$ with $\mu = 0$ and $\sigma = \gamma$.*

Note that the smoothed log-concave density estimator can be written as convolution $\phi_\gamma * \hat{f}_n$. Since both functions are log-concave ($\hat{f}_n$ per definition and ϕ_γ according to Theorem 2.7) and the class of log-concave densities is closed under convolution as described in Corollary 2.11, the smoothed log-concave density estimator is log-concave as well.

One could see the requirement of the parameter γ as a problem. In fact, γ plays a similar role as the bandwidth h for the kernel density estimator. In Section 4 of [13], a rule for the choice of the optimal bandwidth γ is proposed. The automatic

application of this choice, nevertheless makes the computation of the smoothed log-concave density estimator a non-parametric procedure.

Figure 3.4 shows the resulting log-concave density estimator and its smoothed version for the data from Section 3.1. The dashed vertical lines indicate the observations where the slope of $\log \hat{f}_n$ changes.

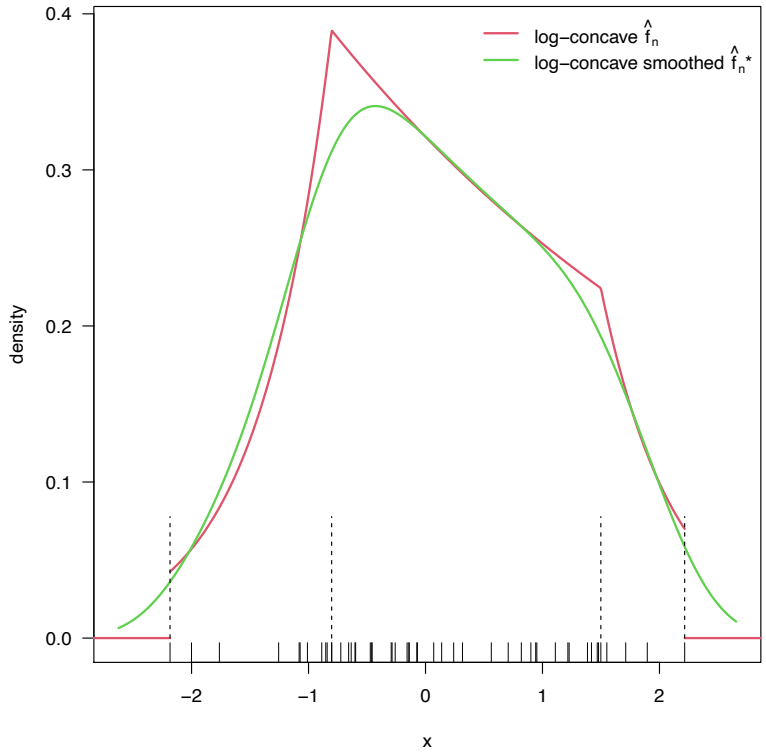


Figure 3.4.: Log-concave density estimator: raw and smoothed version.

3.5. Comparison to other Density Estimation Techniques

First of all we want to recall the example and the parametric density estimation techniques in Section 3.1. There we considered a small sample ($n = 50$) of a standard normal distribution and applied parametric procedures to estimate the density of the sample. We now want to compare the histogram and the kernel density estimator (using Scott's rule) with the log-concave density estimator and

its smoothed version. All four estimators and the real density are plotted for comparison in Figure 3.5. It is by no means clear which of these estimators describes the sample best. We just want to point out , that the log-concave density estimator is the only fully automatic one and leads to a satisfying result but comes with the shortcoming, that unimodality has to be guaranteed or at least has to be assumable.

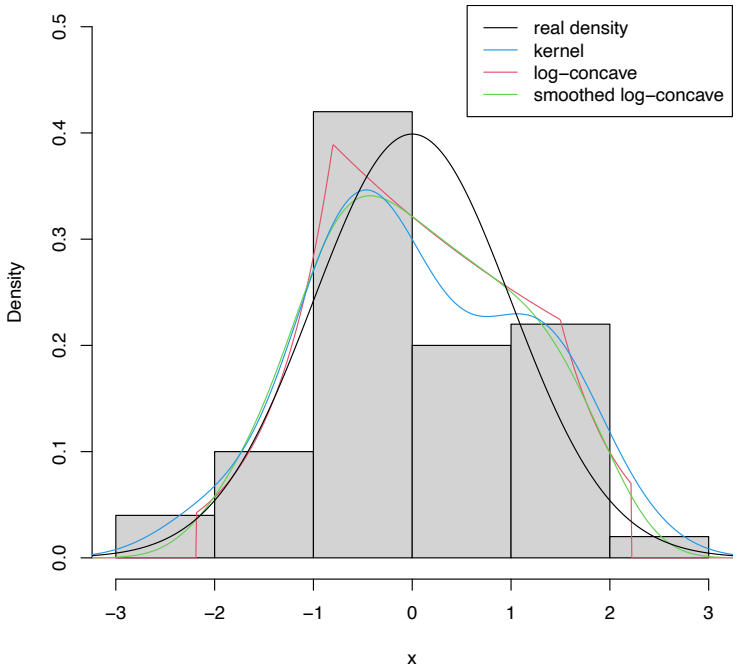
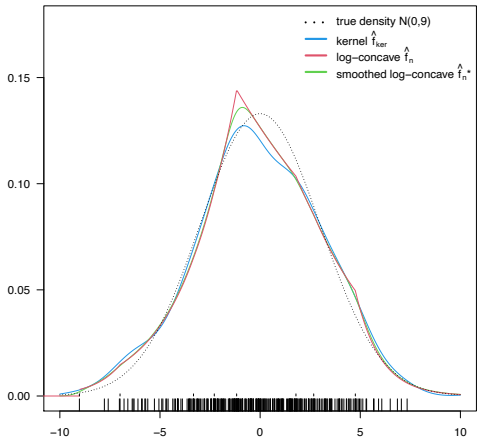
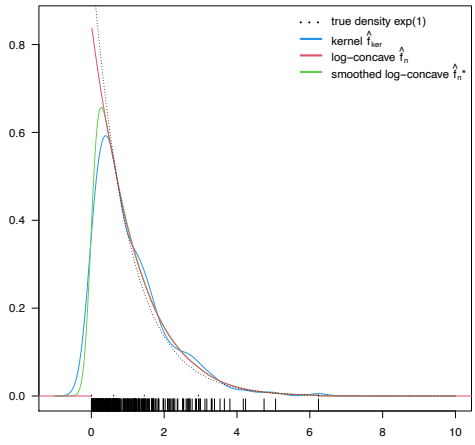


Figure 3.5.: Comparison of different density estimators.

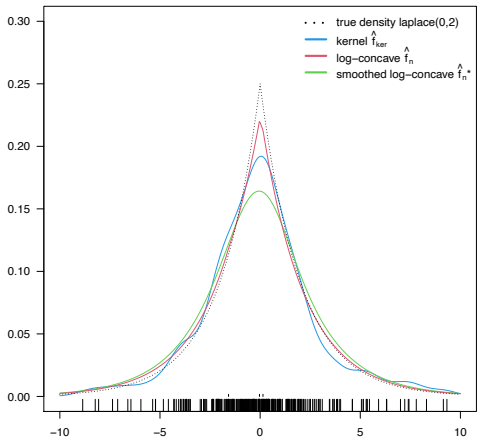
In the abstract of [19], the log-concave density estimator was described as "the most efficient [...], at least for large samples". Therefore, we want to apply some density estimation techniques to larger samples ($n = 300$) from different distributions. In Figure 3.6 we see the results for the two variants (smoothed and unsmoothed) of the log-concave density estimator and the kernel density estimator (Scott's rule). We can conclude that both, the log-concave density estimator and the kernel density estimator work quite well in these exemplary cases. Nevertheless we see the sometimes bumpy behavior of the kernel density estimator for example on the upper tail of the $\Gamma(1, 2)$-sample. Another interesting fact that can be seen in Figure 3.6, is that distributions with sharp peaks as the exponential distribution are estimated very well with the (unsmoothed) log-concave density estimator.



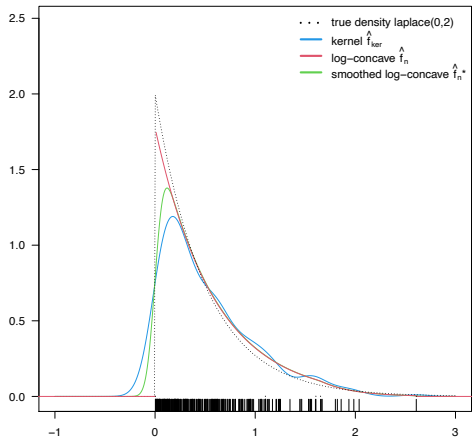
(a) $X \sim N(0, 9)$



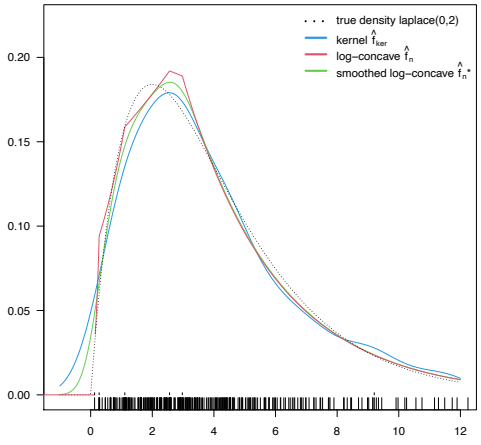
(b) $X \sim \exp(1)$



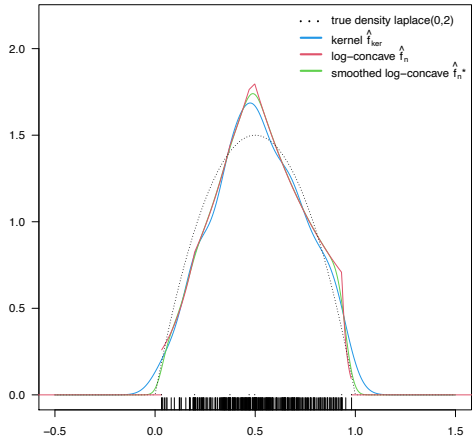
(c) $X \sim Laplace(0, 2)$



(d) $X \sim \Gamma(1, 2)$



(e) $X \sim \chi^2(4)$



(f) $X \sim \beta(2, 2)$

Figure 3.6.: Density estimation for samples (sample sizes $n = 300$ each) from different distributions.

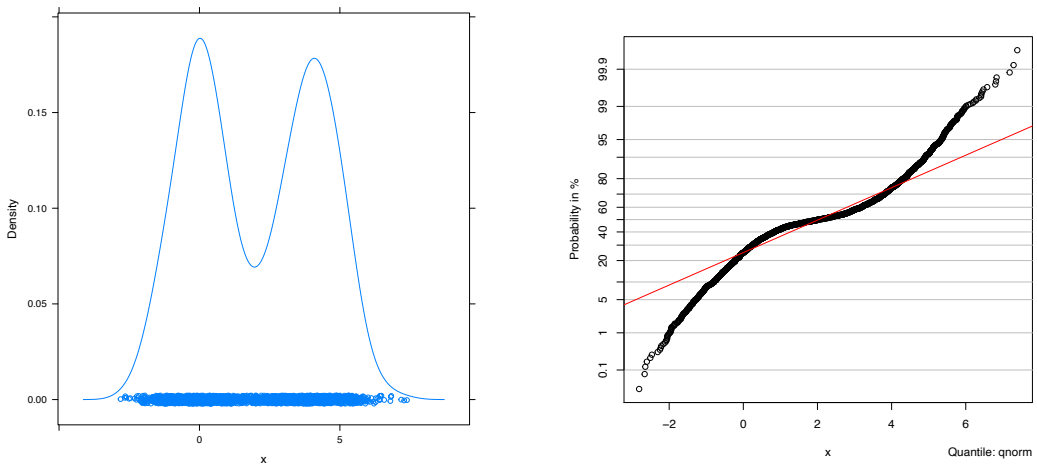
4. Detecting the Presence of Mixtures

The goal of this chapter is to fix a problem that can easily occur in the type of data we observe. One can think of a machine that tests electrical components and due to time-efficiency it does not test each device after the other, but it tests some devices simultaneously. It is possible that the results on each of the sockets, where the devices are tested, follow a clean normal distribution, but the sockets themselves differ from each other. Thus, when we look at measured data of an electrical test, it is possible that a mixture of two or more distributions occurs in the overall distribution. In this chapter we focus on a test first introduced in [45] that has the detection of the presence of such mixtures as objective.

Note that the presence of mixtures is strongly correlated with the number of modes. For the human eye, the best chance to detect mixtures of distributions, is when the mixture components are different enough, that the overall distribution obviously has two or more modes. For example a sample, drawn from the mixture $\frac{1}{2}N(0,1)+\frac{1}{2}N(4,1)$ is very likely to result in a density with two modes. Figure 4.1 shows the (kernel-)density and the probability plot of such a sample. For our task (see Chapter 1) it seems very likely that multimodality would be detected manually in this case and thus should also be detected by an automatic procedure.

4.1. An Overview of the Test

The description of the algorithm, we want to use, can be seen in detail in [43] and our implementation in R is presented in Section A.2. Nevertheless we want to sketch the procedure briefly in this section. The origin of the test is described in Theorem 2.12, which provides a characterization of a function that consists of a mixture of m log-concave densities. Obviously a single log-concave density (no mixtures) corresponds with a density of the form (2.4) with $c = 0$. Thus our objective is to test whether $c = 0$ (the null-model) or $c > 0$ (presence of mixtures).



(a) Density plot.

(b) Normal probability plot.

Figure 4.1.: Exemplary sample drawn from the mixture $\frac{1}{2}N(0,1) + \frac{1}{2}N(4,1)$.

Similar to the log-concave density estimator, which we discussed in Chapter 3, we are now interested in the solution of the optimization problem

$$\max_{\phi \text{ is concave}} \frac{1}{n}\sum_{i=1}^{n} \phi(x_i) - \int_{x_{(1)}}^{x_{(n)}} \exp\left(\phi(x) + c \cdot x^2\right) \mathrm{d}x \tag{4.1}$$

where $c \geq 0$ is a fixed number.

In fact in [43] it is shown, that problem (4.1) has a solution $\hat{\phi}_n$, for which $\hat{f}_n := \exp\left(\hat{\phi}_n + c \cdot x^2\right)$ is exactly the maximum likelihood estimator of (2.4). This fact is obviously very helpful[1] but nevertheless brings out the difficulty for the construction of such a test: we are interested in the value of the parameter c but we can compute the function in which we are interested in only for a fixed value of $c \geq 0$[2]. We want to handle this problem by using a grid of different equidistant values of c. Afterwards we compare the goodness of fit of the different estimations in comparison to the null-model ($c = 0$).

For fixed c, the function $x \mapsto \hat{\phi}_n(x) + c \cdot x^2$ is piecewise parabolic with parabolas of curvature c. For large values of c the function will therefore have deep dips between the observations (knots), whereas for small values of c the function will only have shallow dips. For the null-model ($c = 0$) the function is perfectly concave. The

[1] Additionally it is even shown that the solution $\hat{\phi}_n$ is piecewise linear in analogy to Corollary 3.4.

[2] The estimated functions are computed using the Iterative Minorant Algorithm, see [22].

idea is now to measure the distance of the piecewise derivative of $\hat{\phi}_n(x) + c \cdot x^2$ from the class of monotonically decreasing functions (remember Chapter 3, where we reformulated the optimization problem using only the slopes of the piecewise defined function). The target function $T(\cdot)$ to measure the distance of any function g from the class of decreasing functions Mon will be $d_\omega(d, Mon) := \inf_{m \in Mon} ||(g - m)\omega||_\infty$[1]. The function ω is a weight function, where we will use the estimation of the null-model $\hat{f}_n^0$. The reference to assess the level of significance to the evaluated target function is then obtained via Monte Carlo sampling of new samples from the null-model.

To give an idea, how c effects the outcome of the maximum likelihood estimation, examples of estimations for different values of c are given in Figure 4.2 for an underlying unimodal density and in Figure 4.3 for multimodal cases.

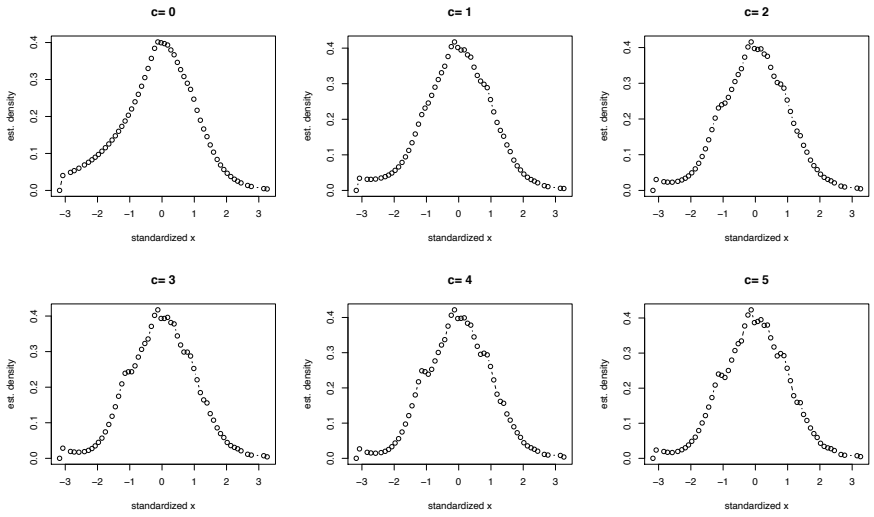
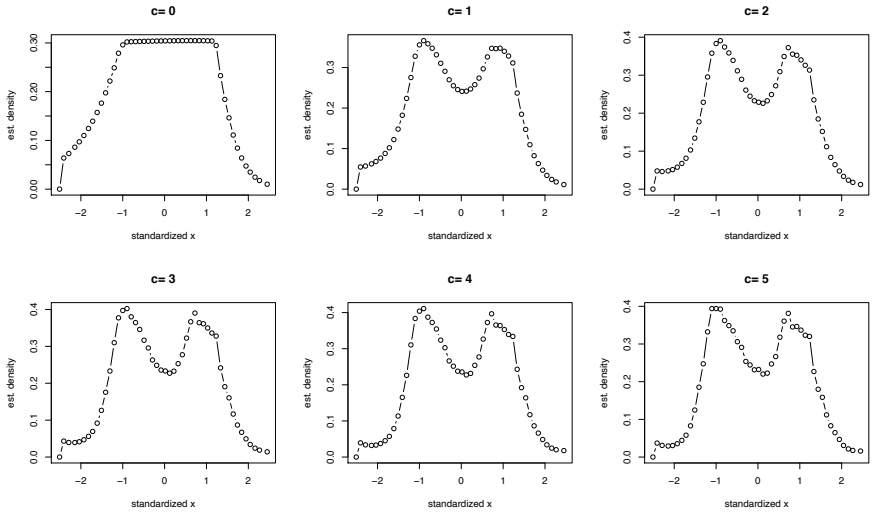


Figure 4.2.: Impact of $c \in \mathcal{C}$ for a sample from a unimodal density ($N(0,1)$) with $\mathcal{C} = \{0; 1; 2; 3; 4; 5\}$

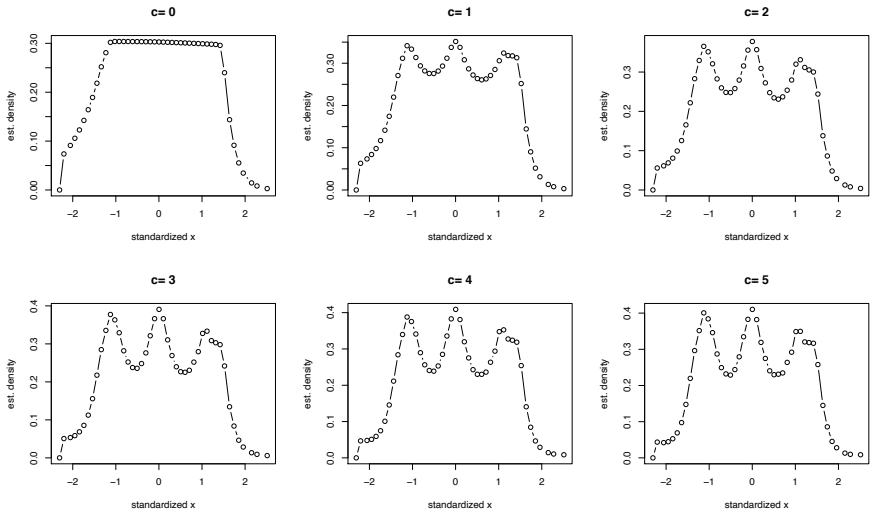
The procedure was originally implemented in Matlab, as described in [43]. The code is available from the author upon request and is in our case only used to compare the results of our own implementation with the original Matlab code. We implement the procedure in R to achieve compatibility with other tools and functions that we will use, see Chapter 6. The implementation in R can be seen in Section A.2.

In the following sections, we will discuss some properties of the procedure and

[1] $||\cdot||_\infty$ is the supremum norm with $||f||_\infty := \sup_{x \in D} |f(x)|$.



(a) Impact of $c \in \mathcal{C}$ for a $\frac{1}{2}N(0,1) + \frac{1}{2}N(3,1)$ sample with $\mathcal{C} = \{0;1;2;3;4;5\}$



(b) Impact of $c \in \mathcal{C}$ for a $\frac{1}{3}N(0,1) + \frac{1}{3}N(3,1) + \frac{1}{3}N(6,1)$ sample with $\mathcal{C} = \{0;1;2;3;4;5\}$

Figure 4.3.: Impact of c for samples from multimodal densities.

introduce the parameters that have to be chosen by the user and their impact on the result. The following parameters are under investigation:

- B describes the number of samples that were taken into account for determining the significance of the result through Monte Carlo resampling. See Section 4.2.
- *BIN.DIST* describes the distance between two neighboring bins in the bin-

ning procedure that is made in advance of the actual algorithm for simplification. See Section 4.3.

- $\mathcal{C}$ describes the grid for the values of c where the density is estimated. It consists of n_c values from 0 to c_{max}. See Section 4.4.

4.2. The Monte Carlo Sample Size B

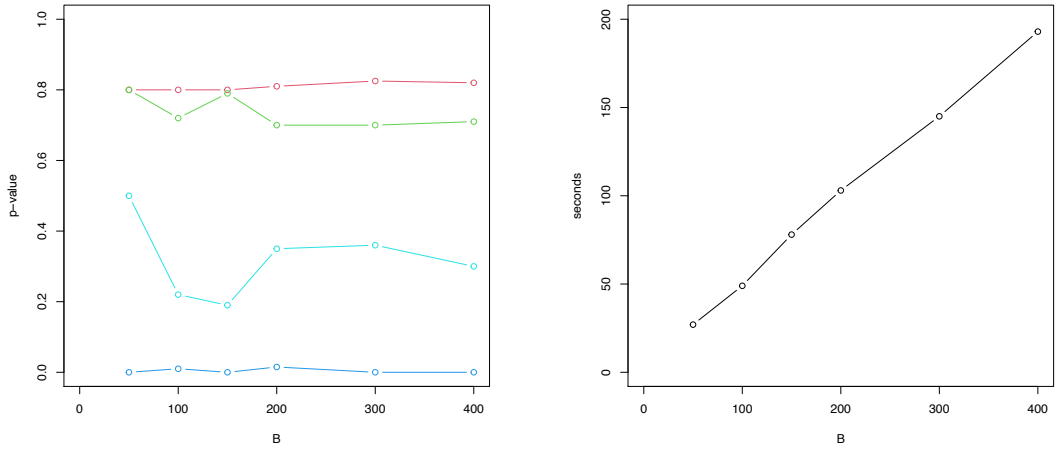
The parameter B determines the number of repetitions in the algorithm to simulate the (not analytically calculable) p-value. To save computation time when running the code a few times in a row, we want to choose the value for B as small as possible, but large enough to get satisfying results. To get an insight in the impact of the choice of the parameter B on the result, we generate four exemplary sets of data and test with different values of B.

Figure 4.4a shows the results of simulation runs for 4 exemplary samples with different values of B[1]. The results show that the choice of B does not seem to affect the result very much. Even small numbers of B lead to a satisfying result. We toggle $B = 99$ (results in 100 simulation runs, including the original sample) to be the absolute minimum because of the way how the result is calculated. As the p-value is calculated by simple division, a smaller number for B leads to a more discrete p-value. $B = 99$ is a quite natural number to be the minimal choice for the parameter, as it results in an integer percentage. Nevertheless, we choose $B = 199$ to be our standard level for this parameter. With respect to the result in Figure 4.4a, this choice seems to be quite intuitive, as the results do not change that much for higher values of B.

Figure 4.4b shows the influence of the parameter B on the computation time. The result is not very surprising as the relationship is linear due to the fact that a higher value of B results in more iterations of the same sub-procedure[2].

[1]The other parameters are kept on the level suggested by [43] ($C.NR = 11$, $C.MAX = 3$ and $BIN.DIST = 0.1$).

[2]The computation time is evaluated within R with the function *microbenchmark()* and 10 repetitions of the algorithm. The values in Figure 4.4b show the mean of the different results from all 10 repetitions.



(a) Influence of B on the result of the algorithm for four exemplary samples.

(b) Influence of B on the computation time in seconds.

Figure 4.4.: A few simulation runs with different values of B.

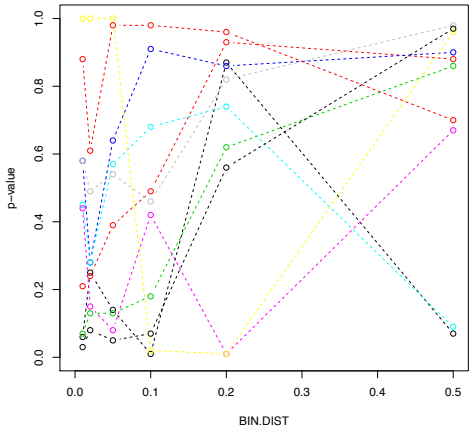
4.3. Binning Procedure

To keep computation time low, the raw data is simplified in advance to the actual algorithm: the observations are assigned to bins, that are arranged equidistantly on the range of the data. The parameter *BIN.DIST* that has to be chosen, is the distance in terms of empirical standard deviations that the bins should have from each other. In Section 4 of [43], the author determines this constraining binning procedure to have a low impact on the quality of the outcome of the algorithm. We will try to justify this proposition as we run simulations for several cases of provided data and parameters for the essential parameter *BIN.DIST*. A value of 0.1 for *BIN.DIST* is suggested in [43]. Therefore we try to get an insight of the dependencies between the outcome, the parameter *BIN.DIST* and the sample size of the raw data[1] by trying larger and smaller values for *BIN.DIST* than the suggested value.

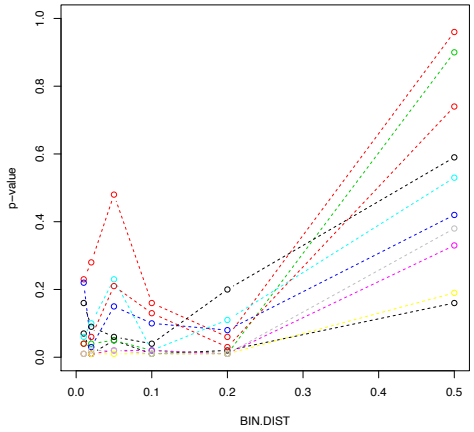
The results of these simulations are given in Figure 4.5. All simulation runs are made with the suggested choices for *C.NR* ($= 11$), *C.MAX* ($= 3$) and B ($= 199$). Each plot contains 10 repetitions with newly sampled data, evaluated for a predefined set of values for *BIN.DIST*: $\{0.01; 0.02; 0.05; 0.1; 0.2; 0.5\}$. The four

[1]The idea of a dependency between the sample size of the raw data and the outcome of the procedure came up while other computations were made and is something that should be taken into account when the algorithm is applied to real data with high sample sizes.

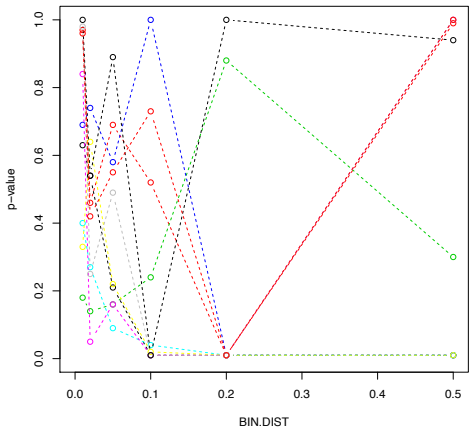
different plots show the result for different types of samples. In Figure 4.5a and 4.5b, we simulate samples of size $n = 500$. In Figure 4.5a the sample is drawn from a standard normal distribution ($N(0, 1)$). In Figure 4.5b a mixture of two normal distributions is used ($\frac{1}{2}N(0, 1) + \frac{1}{2}N(3, 1)$). In Figure 4.5c and 4.5d, we simulate samples of larger sizes ($n = 2000$) from the same distributions.



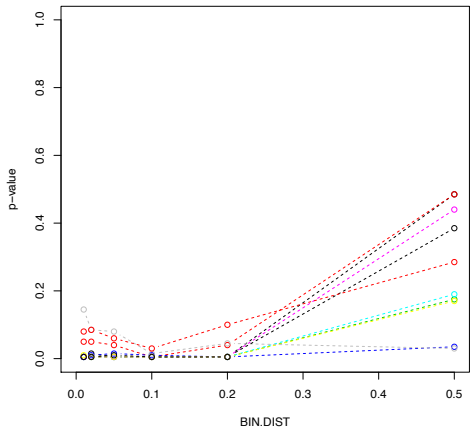
(a) 10 simulation runs with 500 N(0,1) - samples each.



(b) 10 simulation runs with 500 ($\frac{1}{2}$N(0,1) + $\frac{1}{2}$N(3,1)) - samples each.



(c) 10 simulation runs with 2000 N(0,1) - samples each.



(d) 10 simulation runs with 2000 ($\frac{1}{2}$N(0,1) + $\frac{1}{2}$N(3,1)) - samples each.

Figure 4.5.: A few simulation runs with different values of *BIN.DIST*.

The results show that the calculated p-values are influenced by the choice of the parameter *BIN.DIST*, especially in case of a sample from a single distribution, see Figures 4.5a and 4.5c. In [43], this parameter "was not found to affect the solution much". This statement suggests that the resulting p-value stays the same for different values of the parameter. But as we can see, this is not precisely correct. Nevertheless we see the expected result when we compare the plots for the unimodal and the bimodal underlying data. Thus we will use the binning procedure in advance, because it is necessary to shrink computation time at least a little bit[1].

Under the assumption of the null hypothesis, the numerically non-biased p-value should follow a uniform distribution: $p \sim U(0,1)$. To test if the calculated p-value approximately does so, we run the test on two different levels of *BIN.DIST* a thousand times each. The results are presented as percentiles for each of the two settings in Table 4.1. For every single test, a sample of size 2000 from a standard normal distribution is used[2].

BIN.DIST	Percentiles								
	1%	5%	10%	25%	50%	75%	90%	95%	99%
0.1	0.005	0.005	0.005	0.005	0.01	0.195	0.57	0.695	0.82
0.05	0.005	0.005	0.005	0.04	0.21	0.525	0.79	0.865	0.895

Table 4.1.: Quantitative overview of resulted p-values for 1000 simulation runs with $N(0,1)$-samples of size 2000 each.

A major shortcoming of the procedure in its current form can be seen in Table 4.1. The p-value does not seem to follow a uniform distribution. A look into the origin of these computations, show that the bad results are caused by the multiscale manner of the code: the discreteness of the $\mathcal{C}$-grid causes highly biased maximization over the parameter c in every of the $B+1$ Monte Carlo samples. In the following section we will present an extension of the algorithm that should improve the non-satisfying results (p-value$\nsim U(0,1)$) from Table 4.1.

[1] Simulation runs with a sufficiently fine grid (*BIN.DIST*$\approx$*0.0001*) to keep the calculated p-value stable resulted in an increase of computation time by a factor between 10 and 100.

[2] For each computation $C.NR = 11$, $C.MAX = 3$ and $B = 199$ is chosen.

4.4. Determining the Optimal $\mathcal{C}$-grid

As described in Section 4.1, the true value of c is not known and therefore the calculations are done for several values of c. The algorithm provides the possibility to choose the number of tested values n_c (equivalent to `C.NR` in the R-code in Section A.2) and the largest of these values c_{max} (equivalent to $C.MAX$ in the R-code in Section A.2). The set $\mathcal{C}$ is then created as a equidistant grid from $c = 0$ to $c = c_{max}$. Of course a finer and larger grid would lead to a better result as the possibility to hit a value near to c_{true} is higher, but n_c has a direct impact on the computation time and should therefore be kept low.

We want to observe what "kind of grid" is suitable for satisfying results and low computation time. Therefore we run the procedure 100 times for different kinds of grids. We vary the parameters n_c and c_{max} and therefore get grids that differ in range and density of the observed values of c. Figure 4.6 shows the resulting p-values of all 100 simulation runs as boxplots. The used samples of size 2000 each in all 100 cases come from a mixture $\frac{1}{2}N(0,1) + \frac{1}{2}N(\mu_2, 1)$.

The results of this small simulation study show that wider grids improve the result under the null hypothesis (see Figure 4.6a). The value of c_{max} seems to have a bigger impact on the overall results than the number of observed c values n_c. We observe very good results for mixtures of densities with distance of three standard deviations (and above).

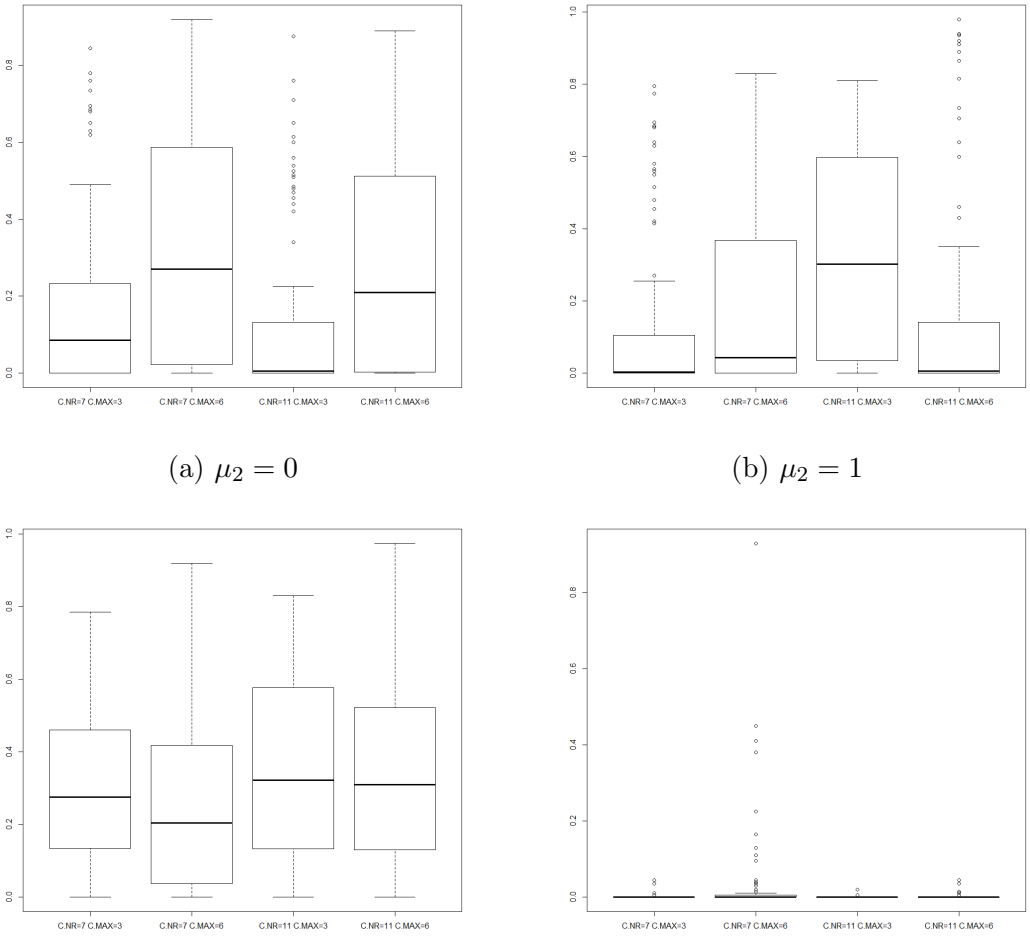
The unsatisfying result of the distribution of the p-values under the H_0 as described in the end of Section 4.3 and as seen in Figure 4.6a seems to be one of the biggest shortcomings of the algorithm. We will now try to improve the results and present a small extension of the procedure.

The output of the algorithm (p-value) is based on the formula

$$p = \left(\# \left\{ \max_{c \in \mathcal{C}} \left(\frac{T_n(c) - m(c)}{s(c)} \right) \leq \max_{c \in \mathcal{C}} \left(\frac{T_n^{*i}(c) - m(c)}{s(c)} \right), 1 \leq i \leq B \right\} + 1 \right) /(B+1) \tag{4.2}$$

where $T_n^{*i}(c)$ is the value of the target function for the i-th Monte Carlo sample and $m(c)$ and $s(c)$ the mean and standard deviation of all $T_n^{*i}(c)$ (see Section 3 in [43]). To simplify the calculation, one can easily skip the standardization and only calculate

$$p = \left(\# \left\{ \max_{c \in \mathcal{C}}(T_n(c)) \leq \max_{c \in \mathcal{C}}(T_n^{*i}(c)), 1 \leq i \leq B \right\} + 1 \right) /(B+1). \tag{4.3}$$



(a) $\mu_2 = 0$ (b) $\mu_2 = 1$

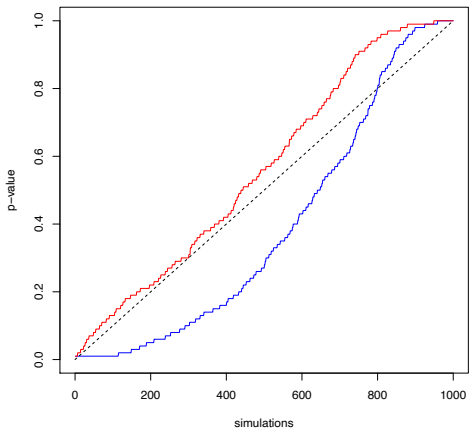
(c) $\mu_2 = 2$ (d) $\mu_2 = 3$

Figure 4.6.: Results of 100 simulation runs with $\frac{1}{2}N(0,1) + \frac{1}{2}N(\mu_2, 1)$ samples of size 2000 under varying c grids.

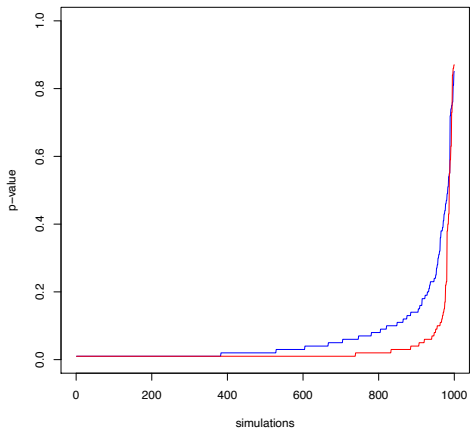
A big bias of the true p-value comes from the simplification of using only a grid of c values and not a theoretically optimal maximization over a steady function. The optimal solution would be to use a very fine and very wide grid (c_{max} and n_c very large). But another solution seems to work as well. The biggest shortcoming of a "bad" c-grid is the biased maximum of $T_n(c)$ (for the original data). Of course the other maximization(s) in (4.3) affect the result as well but as we compare a fixed value on the left hand side with B different values on the right hand side, a deviation on the left hand side from the true maximum has a much higher impact on the p-value.

Therefore we introduce a new variable c_{boost} which works as a multiplier for the number of observed c-values for the fitting based on real data. The set $\mathcal{C}$ for fitting densities based on real data is now created with $n_c \cdot c_{boost}$ equidistant points from 0 to c_{max}. The set $\mathcal{C}^*$ (for the Monte Carlo samples) is still created with n_c equidistant points from 0 to c_{max}. Thus we get a c_{boost} times finer grid for the left hand side in (4.3), which results in less deviation of the estimated maximum from the real maximum.

To show how well this small extension works, we perform some simulations and compare the results for the old version with ($c_{boost} = 10$) and without the extension (equivalent to $c_{boost} = 1$). 1000 unimodal densities ($N(0,1)$) and 1000 mixtures ($\frac{1}{2}N(0,1) + \frac{1}{2}N(3,1)$) are created and tested. The result is presented in Figure 4.7 and Table 4.2, which shows some percentiles of the results for the unimodal distributions analogous to Table 4.1 in Section 4.3 to check the distribution of the p-value under the null hypothesis.



(a) Results of 1000 simulated unimodal ($N(0,1)$) densities with $U(0,1)$ reference line.



(b) Results of 1000 simulated bimodal ($\frac{1}{2}N(0,1) + \frac{1}{2}N(3,1)$) densities.

Figure 4.7.: Comparison between results of the algorithm with (red) and without (blue) extension of a finer c-grid for real data. The boosting parameter $c_{boost} = 10$ is chosen.

From Figure 4.7 we can conclude, that the distribution under the H_0 looks much better now. Theoretically it should follow the dashed line but most important is the fact that we do not expect that much of false positive results that would have occurred without the extension since the number of very low p-values under the H_0 is much smaller now. Moreover we see an improvement of the results for samples

Percentiles								
1%	5%	10%	25%	50%	75%	90%	95%	99%
0.02	0.08	0.13	0.27	0.56	0.90	0.99	1.00	1.00

Table 4.2.: Quantitative overview of resulting p-values for 1000 simulation runs with a sample of size 2000 each from a standard normal distribution with the extension of a finer c-grid for real data fitting.

from mixtures as well, see Figure 4.7b. Thus we highly recommend the use of the parameter c_{boost} as it solves a problem but comes with no major disadvantage.

In the following computations we will use the parameter-set $c_{max} = 3$, $n_c = 4$ and $c_{boost} = 10$ as the standard parametrization to determine the grids $\mathcal{C}$ and $\mathcal{C}^*$.

4.5. Application to Artificial Data

We will now present the results of a small simulation study, that should give an insight in how well the procedure performs. Mainly we are interested in the performance on samples that come from mixtures with equal relative size (formally $p_1 = p_2 = \frac{1}{2}$ in Theorem 2.12) and different centers. Additionally we are also interested in the behavior of the procedure for samples from mixtures with differing relative sizes.

The following set of parameters is used:

- $BIN.DIST = 0.1$
- $B = 199$
- $c_{max} = 3$
- $n_c = 4$
- $c_{boost} = 10$

In Figure 4.8, we see the results for 1000 different $\left(\frac{1}{2}N(0,1) + \frac{1}{2}N(\mu_2, 1)\right)$-samples of size 2000 for several $\mu_2 \geq 0$. The points and the solid line describe the median of the resulting p-values and the two dashed lines the lower and upper quartile. Table 4.3 gives an overview of the relative number of samples that resulted in a p-value lower or equal the three mostly used levels of significance (α).

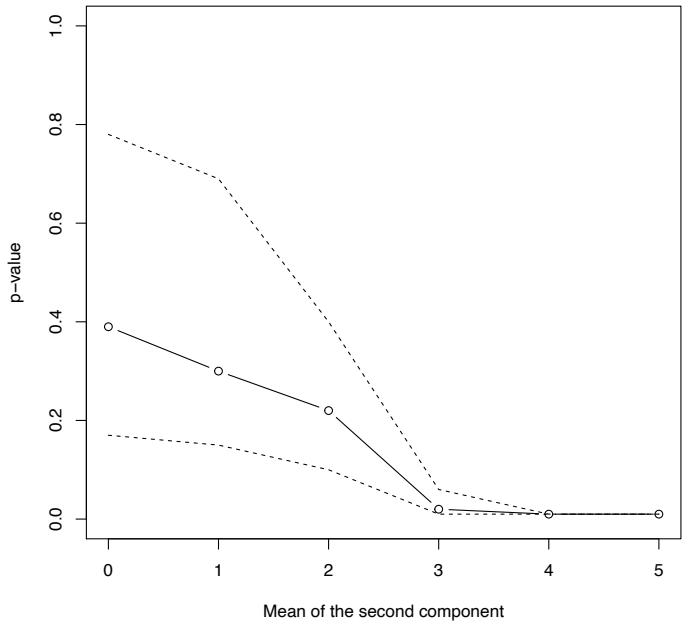


Figure 4.8.: Median and lower & upper quartile of 1000 $\frac{1}{2}N(0,1) + \frac{1}{2}N(\mu_2, 1)$-samples of size 2000.

	Relative number of cases $\leq \alpha$					
α	$\mu_2 = 0$	$\mu_2 = 1$	$\mu_2 = 2$	$\mu_2 = 3$	$\mu_2 = 4$	$\mu_2 = 5$
0.1	20%	22%	25%	91%	98%	100%
0.05	11%	15%	20%	82%	94%	96%
0.01	1%	3%	9%	54%	88%	94%

Table 4.3.: Relative number of samples that resulted in a p-value lower or equal than the given levels of significance on the left.

In Figure 4.9, we see the results for 1000 different $((1-p_2)\cdot N(0,1) + p_2 \cdot N(3,1))$-samples of size 2000 for several $p_2 \in (0;1)$. The points and the solid line describe the median of the resulting p-values and the two dashed lines the lower and upper quartile. Table 4.4 gives an overview of the relative number of samples that resulted in a p-value lower or equal the three mostly used levels of significance.

From the simulations described above we can conclude that for $p_1 = p_2$ a distance of ≈ 3 between the two mixture components is needed to detect the presence of mixing with the described test significantly sure enough. This fact is a good rule

of thumb for our use case when one thinks of the required distance between two equally sized components to detect mixing. Even if the components would not have equal size, the algorithm shows satisfying results as we can conclude from Figure 4.9 and Table 4.4. The presence of mixture seems detectable quite good even with proportions much more unequal than $p_1 = p_2$.

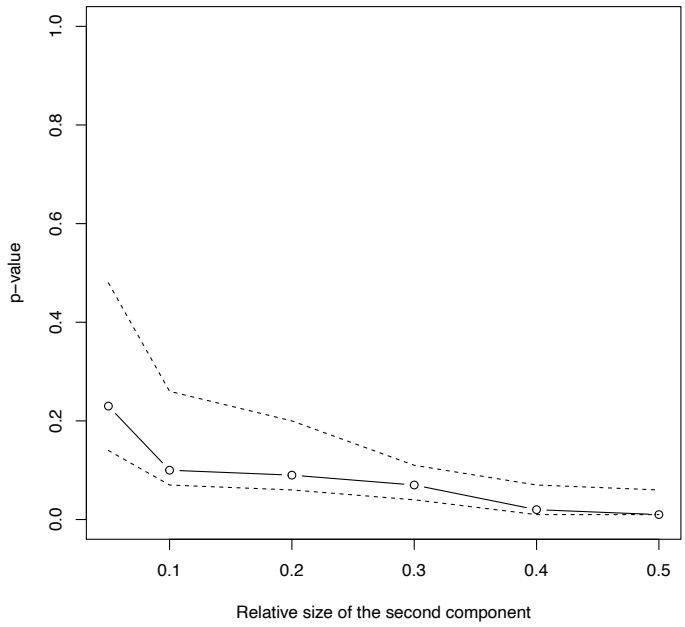


Figure 4.9.: Median and lower & upper quartile of 1000 $(1 - p_2) \cdot N(0, 1) + p_2 \cdot N(3, 1)$-samples of size 2000.

	Relative number of cases $\leq \alpha$					
α	$p_2 = 0.05$	$p_2 = 0.1$	$p_2 = 0.2$	$p_2 = 0.3$	$p_2 = 0.4$	$p_2 = 0.5$
0.1	22%	50%	73%	87%	90%	91%
0.05	16%	30%	41%	49%	64%	82%
0.01	7%	10%	18%	27%	46%	54%

Table 4.4.: Relative number of samples that resulted in a p-value lower or equal than the given levels of significance on the left.

5. Logarithmic Concave Densities in Extreme Value Theory

Extreme value theory generally consists of two approaches of catching the characteristics of the tails of a distribution. The first one is to investigate the properties of the maxima of several samples from a distribution. This approach and the connection to the class $\mathcal{F}_1$ is investigated in [24], from where we will now sketch the main idea. The fundamental result in this theory comes from [17] and [16]:

Theorem 5.1. *Let $X_1, \ldots, X_n$ be independent and identically distributed random variables with distribution function F and define $M_n := \max\{X_1, \ldots, X_n\}$. If sequences $(a_n)_{n=1}^{\infty} > 0$ and $(b_n)_{n=1}^{\infty} \in \mathbb{R}$ exist, such that*

$$\lim_{n\to\infty} P\left(\frac{M_n - b_n}{a_n} \leq x\right) = F(x)$$

then the distribution function F is in the Gumbel-, Weibull- or Frechet-family.

Proof. See [17] or [16]. ■

And we continue with the corresponding definition for which Theorem 5.1 is the building block.

Definition 5.2. *The probability family with distribution function*

$$F : D \to [0;1],\ x \mapsto \begin{cases} \exp\left(-\left(1+\gamma\frac{x-\mu}{\sigma}\right)^{-\frac{1}{\gamma}}\right) & \text{for } \gamma \neq 0 \\ \exp\left(-e^{-\frac{x-\mu}{\sigma}}\right) & \text{for } \gamma = 0 \end{cases}$$

*is called the **Generalized Extreme Value Distribution (GEVD)** with shape parameter γ and*

$$D = \begin{cases} \left[\mu - \frac{\sigma}{\gamma}; \infty\right) & \text{for } \gamma > 0 \\ \left]-\infty; \mu - \frac{\sigma}{\gamma}\right) & \text{for } \gamma < 0 \\ [0; \infty) & \text{for } \gamma = 0 \end{cases}.$$

Note, that the definition of the GEVD for $\gamma = 0$ is reasonable since

$$(1 + \gamma \cdot z)^{-\frac{1}{\gamma}} \to e^z \text{ as } \gamma \to 0.$$

We can observe, that Theorem 5.1 justifies Definition 5.2 in the following way:

- For $\gamma = 0$ and $y = \frac{x-\mu}{\sigma}$ we get
  the distribution function $F(y) = \exp(-e^{-y})$ and
  the density function $f(x) = \frac{1}{\sigma}\exp(-y - e^{-y})$,
  which belong to the **Gumbel Distribution**.
- For $\gamma > 0$, $y = \left(1 + \gamma\frac{x-\mu}{\sigma}\right)$ and $\gamma = \frac{1}{\alpha}$, we get
  the distribution function $F(y) = \exp(-y^{-\alpha})$ and
  the density function $f(y) = \alpha y^{-\alpha-1}\exp(-y^{-\alpha})$,
  which belong to the **Frechet Distribution** (with $\sigma = 1$ and $m = 0$).
- For $\gamma < 0$, $y = -\left(1 + \gamma\frac{x-\mu}{\sigma}\right)$ and $\gamma = -\frac{1}{\alpha}$, we get
  the distribution function $F(y) = \exp(-(-y)^{\alpha})$ and
  the density function $f(y) = \alpha(-y)^{\alpha-1}\exp(-(-y)^{\alpha})$, which belong to the **Reversed Weibull Distribution** (with $\sigma = 1$ and $m = 1$).

When we compare the parametrization of the GEVD (γ) with the three distributions above and the findings from Theorem 2.7, we can easily deduce:

Corollary 5.3. *The GEVD is log-concave for* $\gamma \in [-1; 0]$.

Proof. The GEVD is not log-concave for $\gamma > 0$ since the Frechet-distribution is not log-concave (see Theorem 2.7 (d)).

The GEVD is log-concave for $\gamma = 0$ since the Gumbel-distribution is log-concave (see Theorem 2.7 (c)).

The GEVD is log-concave for $\gamma \in [-1; 0)$ since the reversed-Weibull-distribution is log-concave for $\alpha \geq 1$ (recap $\gamma = -\frac{1}{\alpha}$) according to Theorem 2.7 (e). ■

The second approach to catch the characteristics of the tails of a distribution is to estimate the tail itself. This approach is widely known as the "Point Over Threshold" (POT) method. The idea is, that if the i.i.d. sample $x_1, \ldots, x_n$ of a random variable X is given, one can choose a relatively high threshold $u \in \left[x_{(1)}; x_{(n)}\right]$ and define the so called "exceedances" $x_{(n)} - u, x_{(n-1)} - u, \ldots, x_{(n-k_n+1)} - u$[1]. Note that the choice of the threshold u is by no means trivial and is a whole field of interest by itself, see for example [42]. Analogous to the fundamental theorem

[1] k_n describes the smallest natural number for which $x_{(n-k_n)} < u$.

in the first approach, described above in Theorem 5.1, the following Balkema-De Haan-Pickands theorem (see [4] and [26]) gives a statement on the asymptotic distribution of the exceedances.

Theorem 5.4. *Let X be a random variable with distribution function F. The distribution function of the exceedances is given by $F_u(y) := P(X - u \leq y | X > u)$. As $u \to \infty$, we get*

$$F_u(y) \to G_{\mu,\sigma,\gamma}(y)$$

where $G_{\mu,\sigma,\gamma}$ denotes the distribution function of the ***Generalized Pareto Distribution*** *(GPD)*

$$G_{\mu,\sigma,\gamma} : D \to [0;1]\,,\ y \mapsto \begin{cases} 1 - \left(1 + \gamma \frac{x-\mu}{\sigma}\right)^{-\frac{1}{\gamma}} & \text{for } \gamma \neq 0 \\ 1 - e^{-\frac{x-\mu}{\sigma}} & \text{for } \gamma = 0 \end{cases}$$

with shape parameter γ and

$$D = \begin{cases} [\mu; \infty) & \text{for } \gamma > 0 \\ \left[\mu; \mu - \frac{\sigma}{\gamma}\right) & \text{for } \gamma < 0 \\ [\mu; \infty) & \text{for } \gamma = 0 \end{cases}.$$

Proof. See [4] and [26]. ■

Note, that we already verified log-concavity for some parametrizations ($\gamma \in [-1; 0]$) of the GPD in Theorem 2.7. Thus, both distribution families occurring in extreme value theory - the GPD and the GEVD - are log-concave for the same values of their shape parameters.

5.1. Density Estimators in Tail Index Estimation

Our goal is now, to detect the shape of the tail of a density, given a sample $x_1, \ldots, x_n$ from this density function. To quantify the shape of the tail, we are interested in the shape parameter of the GPD γ, which we will from now on call "tail index". Therefore, we want to make use of the concept of tail index estimation, especially the approach by [26] from where we define the "Pickands estimator"[1]:

$$\hat{\gamma}^k_{Pick}(H) := \frac{1}{\log(2)} \log\left(\frac{H^{-1}\left(\frac{n-r_k(H)+1}{n}\right) - H^{-1}\left(\frac{n-2r_k(H)+1}{n}\right)}{H^{-1}\left(\frac{n-2r_k(H)+1}{n}\right) - H^{-1}\left(\frac{n-4r_k(H)+1}{n}\right)}\right) \tag{5.1}$$

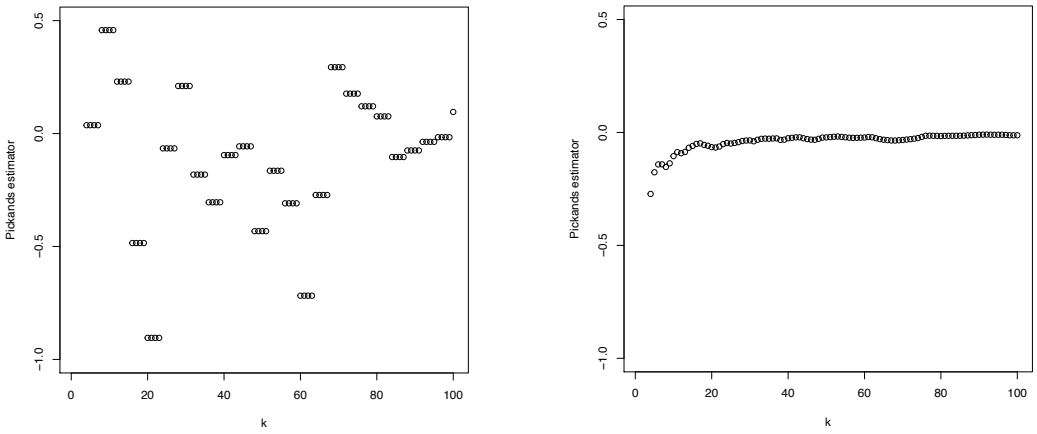
[1] The original form in our notation would be $\hat{\gamma}^k_{Pick}(\mathbb{F}_n)$

for $k = 4, \ldots, n$ and $H \in \left\{\mathbb{F}_n, \hat{F}_n\right\}$[1], where

$$r_k(H) := \begin{cases} \lfloor \frac{k}{4} \rfloor & \text{if } H = \mathbb{F}_n \\ \frac{k}{4} & \text{if } H = \hat{F}_n \end{cases} .$$

The actual notation we used in (5.1) comes from [25]. This paper describes the main idea of how to use the log-concave density estimator in the framework of tail index estimation. We simply replace the usually used empirical distribution function ($H = \mathbb{F}_n$) with the log-concave distribution estimator ($H = \hat{F}_n$). Therefore, we obviously get a much smoother estimation with $\hat{\gamma}^k_{Pick}(\hat{F}_n)$, than with $\hat{\gamma}^k_{Pick}(\mathbb{F}_n)$.

To illustrate the difference of these estimators we draw a sample of size 100 from a GPD with $\mu = 0$, $\sigma = 1$ and $\gamma = 0$. Figure 5.1 shows the so-called "Hill-plots"[2] of the different estimation approaches. One fact that we see, that is actually clear by construction, is that the original estimator using the empirical distribution function is equal for four consecutive values of k. Besides of this fact, we can easily see the big advantage of the extension of (5.1): the new estimator is much smoother than before.



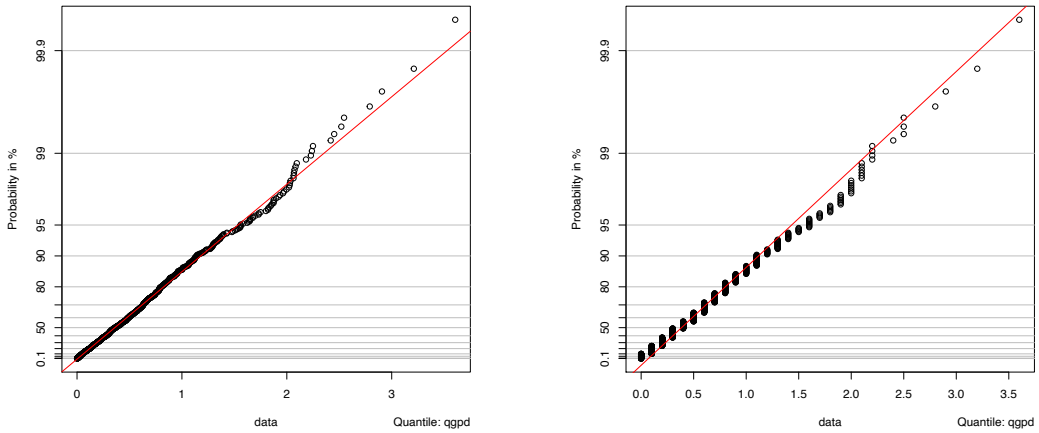
(a) $\hat{\gamma}^k_{Pick}(\mathbb{F}_n)$ (b) $\hat{\gamma}^k_{Pick}(\hat{F}_n)$

Figure 5.1.: Pickands estimator for a $GPD\,(\mu = 0, \sigma = 1, \gamma = 0)$-sample $(n = 100)$.

[1] $\mathbb{F}_n$ denotes the empirical distribution function based on the sample and $\hat{F}_n$ the log-concave distribution function coming from the log-concave density estimator.

[2] Hill plots show the result of $\hat{\gamma}^k_{Pick}(H)$ in dependence of k.

We want to point out another advantage of the smooth tail index estimator, because it plays a role in many samples in our practical application. It is possible and even very likely, that the precision of a machine that measures e.g. electrical values, is not good enough to catch the behavior of data that would actually be continuous but appears discrete due to rounding errors of the machine. The probability plots of an artificial example are given in Figure 5.2, but analogous data appears in real datasets as well.



(a) Data with precise values.

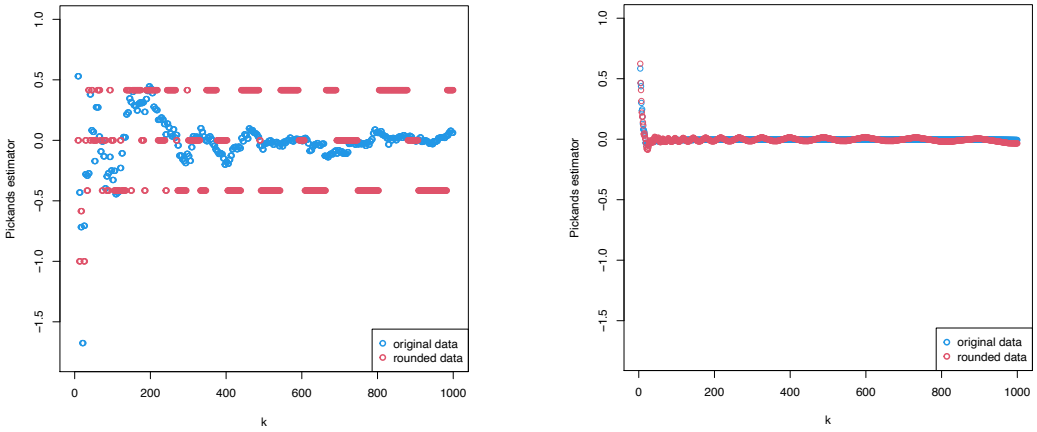
(b) Data with rounded values.

Figure 5.2.: Probability plots of a $GPD\left(\mu = 0, \sigma = \frac{1}{2}, \gamma = 0\right)$-sample of size 1000 for the original and rounded data.

One can guess that the discreteness of the rounded data can be problematic and in fact, we see that the estimator $\hat{\gamma}^k_{Pick}(\mathbb{F}_n)$ is not stable under such a rounding process. Figure 5.3 compares the Hill plots for both settings (actual and rounded data) from above.

The most obvious shortcoming already has been shown in Theorem 2.7. The GPD is only log-concave for $\gamma \in [-1; 0]$. Thus, the replacement of $\mathbb{F}_n$ by $\hat{F}_n$ is only possible if one can assume $\gamma \in [-1; 0]$. Nevertheless the results of $\hat{\gamma}^k_{Pick}(\hat{F}_n)$ can be outside of $[-1; 0]$. The recommended way of dealing with such estimates, if it is known that $\gamma \in [-1; 0]$, is "a truncation of the result to the closest boundary value", according to [25]. For our purpose, we will perform a quick test on log-concavity of the underlying data to make sure if the smoothed estimator $\hat{\gamma}^k_{Pick}(\hat{F}_n)$ can be used. The exact procedure is described in Chapter 6.

All previous and future computations are made with slightly adapted functions from the R-package *smoothtail*, see [34] and [13].



(a) $\hat{\gamma}^k_{Pick}(\mathbb{F}_n)$ (b) $\hat{\gamma}^k_{Pick}(\hat{F}_n)$

Figure 5.3.: Hill plots of a $GPD\left(\mu = 0, \sigma = \frac{1}{2}, \gamma = 0\right)$-sample of size 1000 for the original and rounded data.

5.2. Tail Index Estimation in an Automatic Procedure

The use of tail index estimators in an automatic manner is a bit problematic as they lead to a variety of possible values, more precisely one for every $k = 4, \ldots, n$. A common way to choose the "correct" k is to use Hill plots, see Figure 5.3. The common suggestion, is to use $\hat{\gamma}^k_{Pick}$ for a value of k for which the function $k \mapsto \hat{\gamma}^k_{Pick}(H)$ is as constant as possible. This approach is obviously hard to implement in an automatic procedure. In literature, several other approaches exist. For example in [8], theoretical and practical approaches are described and unfortunately they do not come to the same result in general. That is what motivates us to work out a solution that works especially for the type of problem we want to solve.

In Chapter 1 we sketched the goal of the algorithm, which is about the assignment to a class or a category that gives an idea of how the density looks like and what possible problems could occur in future production processes. Therefore we are not interested in the value of the tail index itself but we are mainly interested whether the tails of the density are heavier than the ones from a normal distribution or not. So we want to develop some rule that focuses on the separation between weaker and heavier tails.

We will simply use the mean of the computed estimations, precisely

$$\frac{1}{m-3} \cdot \sum_{k=4}^{m} \hat{\gamma}^k_{Pick}(\hat{F}_n), \tag{5.2}$$

where $m \approx \frac{5}{6} \cdot n$[1]. We will call (5.2) the "average tail index" even if it is not the mean tail index over every $k = 4, \ldots, n$.

We can then apply this technique to both sides of the sample to get separate tail indices for the lower and upper tail of the sample. The boundaries for the average tail index that classifies the tail of a sample to be heavy or weak are simulated with artificial data. The results of these simulations are shown in Figure 5.4.

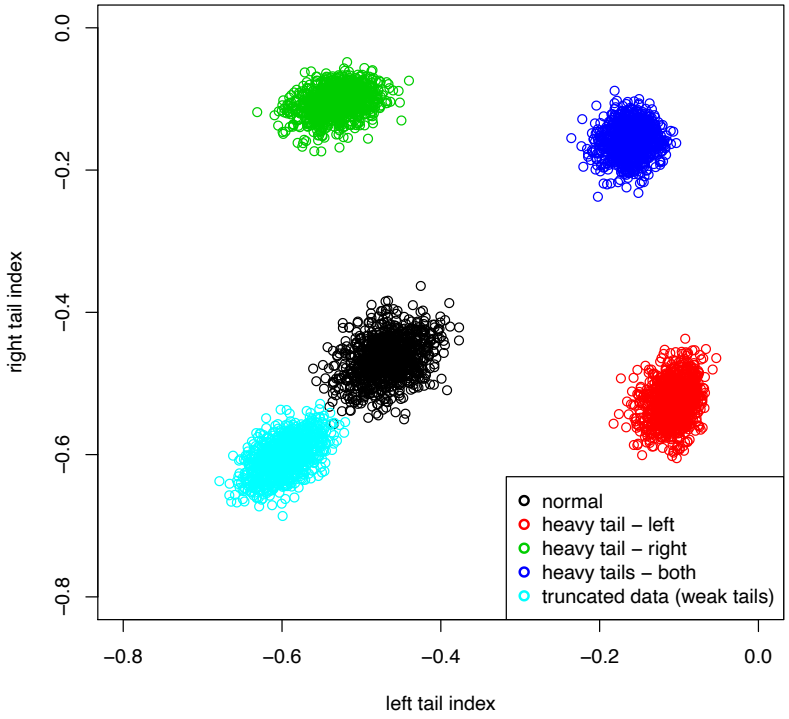


Figure 5.4.: Average tail index for simulated data in five different exemplary classes (1000 samples for every class with $n = 2000$ each) using $\hat{\gamma}^k_{Pick}(\hat{F}_n)$.

[1]The factor $\frac{5}{6}$ was found to give the best results out of several other factors. Nevertheless one could try to use other factors between $\frac{1}{2}$ and 1 to prevent an estimated tail index from one end of the sample's range to be too heavily violated from the other side's tail.

We can conclude that we get a good separation property of the average tail index at least for artificial data. Moreover we get an insight in the range of results of the average tail index for the main class, the normal distribution, from which we can deduce boundaries for this class. Table 5.1 gives an overview of these results and thus we decide to choose the boundaries of the class of normal distributions to be -0.55 and -0.35 with a smooth allocation to heavier or weaker classes beyond these boundaries. Even if tighter bounds would be possible with only little danger of misclassification, as we can see in Table 5.1, we choose those to be sure and keep Figure 5.4 in mind, where we can see that the most severe class to misclassify, the heavy tailed distributions, are still far enough away.

Percentiles								
min	1%	5%	25%	50%	75%	95%	99%	max
-0.541	-0.536	-0.516	-0.487	-0.467	-0.447	-0.419	-0.398	-0.363

Table 5.1.: Quantitative overview of the average tail index of 1000 $N(0,1)$-samples of size $n = 2000$ each.

To increase understandability of the calculated values, we add a transformed tail index to the result of the procedure. This transformation is a function $\mathbb{R} \to \{-3,-2,-1,0,1,2,3\}$, that assigns an easily understandable number to the data. A value of 0 is assigned for a normal distributed tail, positive numbers for heavier tails and negative numbers for weaker tails or data with an underlying density with finite endpoint. The transformation of a calculated average tail index γ works as follows:

$$\gamma \mapsto \begin{cases} 3 & \text{if } \gamma \in (-0.15; \infty) \\ 2 & \text{if } \gamma \in (-0.3; -0.15] \\ 1 & \text{if } \gamma \in (-0.35; -0.3] \\ 0 & \text{if } \gamma \in [-0.55; -0.35] \\ -1 & \text{if } \gamma \in [-0.65; -0.55) \\ -2 & \text{if } \gamma \in [-0.8; -0.65) \\ -3 & \text{if } \gamma \in (-\infty; -0.8) \end{cases} . \tag{5.3}$$

The transformed tail index is the basis of the assignment to the final classification. For values of 2 and 3, we declare the corresponding tail to be "heavy", for values of -2 and -3 to be "weak". An absolute value of 1 can be seen as semi-normal or slightly violated normal distribution and 0 is the desired case of a normal distribution. Examples of these cases can be seen in the next chapter.

To motivate the use of the smooth estimator $\hat{\gamma}^k_{Pick}(\hat{F}_n)$ over the primary version $\hat{\gamma}^k_{Pick}(\mathbb{F}_n)$ when it is possible, we will consider Figure 5.5, which shows artificial exemplary densities and their left and right tail index in analogy to Figure 5.4.

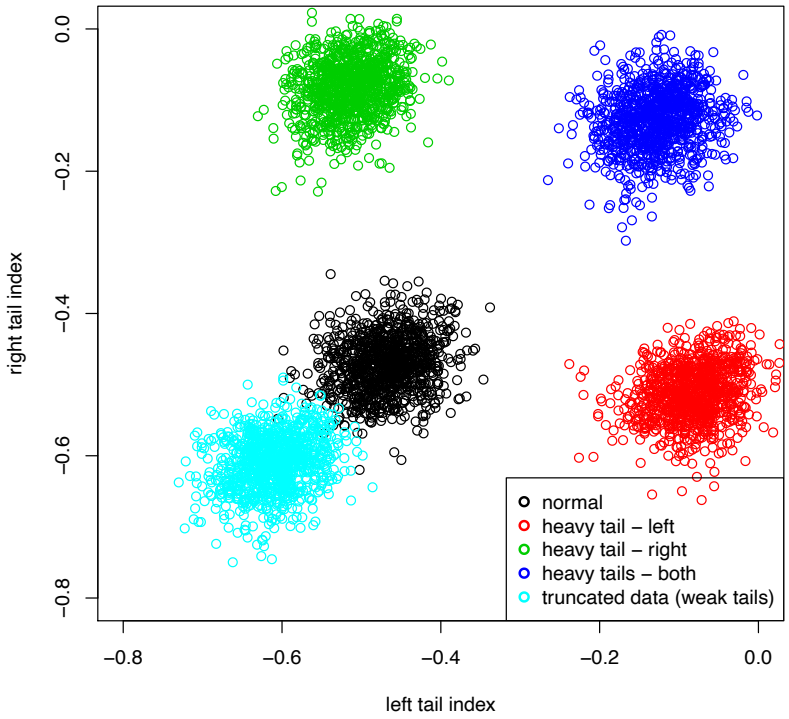


Figure 5.5.: Average tail index for simulated data in five different exemplary classes (1000 samples for every class with $n = 2000$ each) using $\hat{\gamma}^k_{Pick}(\mathbb{F}_n)$.

We can conclude that the tail indices in every of the observed exemplary cases have a higher variance than the smooth analogon. In fact, when we look at the empirical values in Table 5.2, the locations of each class are nearly the same while the standard deviation is about twice as large.

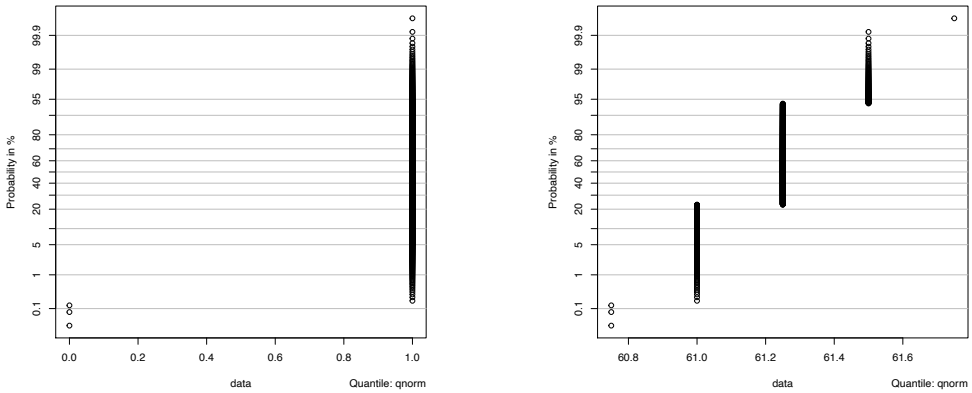
	$H = \hat{F}$		$H = \mathbb{F}_n$	
	left tail	right tail	left tail	right tail
"normal"	-0.47 (0.03)	-0.47 (0.029)	-0.47 (0.041)	-0.47 (0.04)
"heavy tail - left"	-0.11 (0.019)	-0.53 (0.028)	-0.08 (0.04)	-0.51 (0.038)
"heavy tail - right"	-0.53 (0.027)	-0.1 (0.019)	-0.51 (0.038)	-0.08 (0.04)
"heavy tails - both"	-0.16 (0.02)	-0.16 (0.022)	-0.13 (0.041)	-0.13 (0.042)
"truncated (weak) data"	-0.6 (0.026)	-0.6 (0.025)	-0.61 (0.04)	-0.61 (0.04)

Table 5.2.: Average tail index of simulated densities from Figure 5.4 and Figure 5.5 as means (and standard deviations) from all 1000 simulations.

6. NN 2.0 - An Algorithm for the Classification of Densities

We will now introduce the overall classification algorithm "NN 2.0", that is the main target of this thesis as described in Chapter 1. In the following enumeration we will describe the workflow of the procedure (sketched in Figure 6.4), that is implemented in R, see Section A.1 for the code. The input of the procedure is a univariate data-sample, which consists of some measured testing-parameter for all tested devices.

1. First of all, the suitability of the data is tested since many parameters in an overall dataset are of an informational type. Thus the user is asked to specify a minimal number of unique values that the sample should contain to avoid the (probably not even possible) classification of informational or categorical data. An example to this class could be a binary variable, see Figure 6.1a, or a variable that is obviously discrete, see Figure 6.1b.



(a) Probability plot of an exemplary binary variable from real data.

(b) Probability plot of an exemplary discrete variable from real data.

Figure 6.1.: Data of two exemplary variables from a real dataset that should be assigned to the class "categorical and informational data".

2. Another special case that appears quite often is shifted data that results in an extreme mixture or clustering (not in a strict mathematical sense). For example when a tested device causes a severe error of the testing program or of a machine, it is quite likely that all devices that are tested afterwards will result in completely different values and thus most likely a shift in the density, see Figure 6.2. We hope to detect these cases by searching for large enough "gaps" in terms of standard deviations. The difference between each point and its subsequent point is then compared with the empirical standard deviation of the sample multiplied by a user-specified number ($\approx$ 0.5 suggested). To prevent from the detection of single outliers in this step, a minimum number of observations is required on both sides of an occurring gap. The number of required observations on each side relative to the sample-size is once more specified by the user ($\approx$ 0.01 suggested). This procedure should only filter the data for such extreme examples and is not grounded on any mathematical theory. Nevertheless this can work as a fast way of finding extreme mixtures and such clustered data as presented in Figure 6.2, which would cause numerical problems in the next steps.

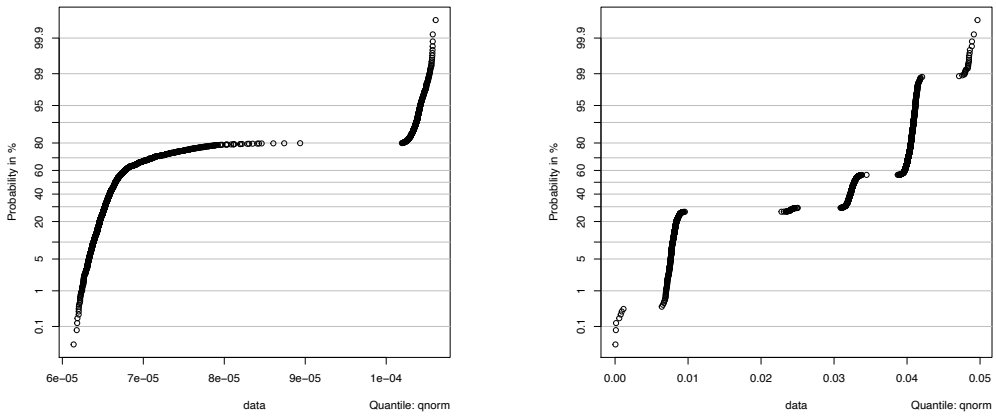


Figure 6.2.: Exemplary variables from a real dataset that should be assigned to the class "clustering".

3. To guarantee all following procedures to work numerically, we scan each sample for extreme outliers. This is simply done by detecting values that are far enough away from the mean value of the sample. The necessary distance is measured using the MAD (Mean Absolute Deviation) from the mean (not the median). Strangely the distance from the mean works out better here than the more frequently used distance from the median. A distance of

15 times the MAD is suggested but the choice is up to the user. We only want to detect extreme outliers in this step, for which an example is given in Figure 6.3, where the two points on the right are ≈ 25 times the MAD away from the mean value of the sample. The detected values are excluded in the following computations but the outlying devices are flagged and also exported to the user for informational purpose.

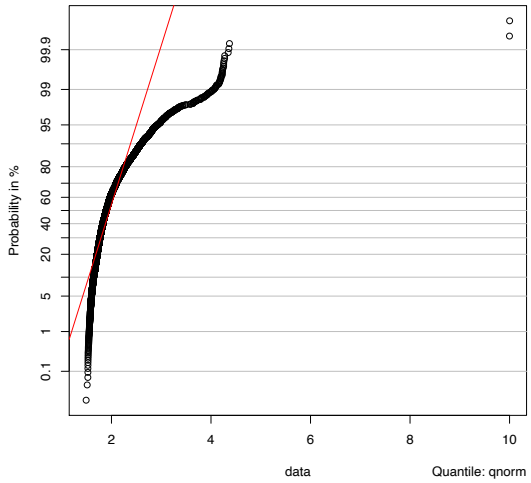


Figure 6.3.: Exemplary variable from a real dataset that should illustrate which type of extreme outliers should be detected.

4. Afterwards basically a test for log-concavity of the sample is performed, which was mentioned in the end of Chapter 2. For data that was declared not to be log-concave, the test on the presence of mixture, discussed in Chapter 4 can be performed. This decision is up to the user because the execution of this test causes a lot of computation time but it can help to detect the presence of mixtures. No matter if the test on the presence of mixtures is used or not, the result of the test on log-concavity will be used to determine whether to use $\mathbb{F}_n$ or $\hat{F}_n$ in the next step.

5. Finally the main part of the procedure, the estimation of the tail index of unimodal densities and in addition the shape of the tail, described in Chapter 5, is performed. Depending on the result of the previously performed test on log-concavity, we will either use the empirical distribution function $\mathbb{F}_n$ or the log-concave distribution estimator $\hat{F}_n$ to compute Pickands tail index estimator.

The workflow of the previous enumeration is sketched in Figure 6.4 to give a simple overview of the procedure.

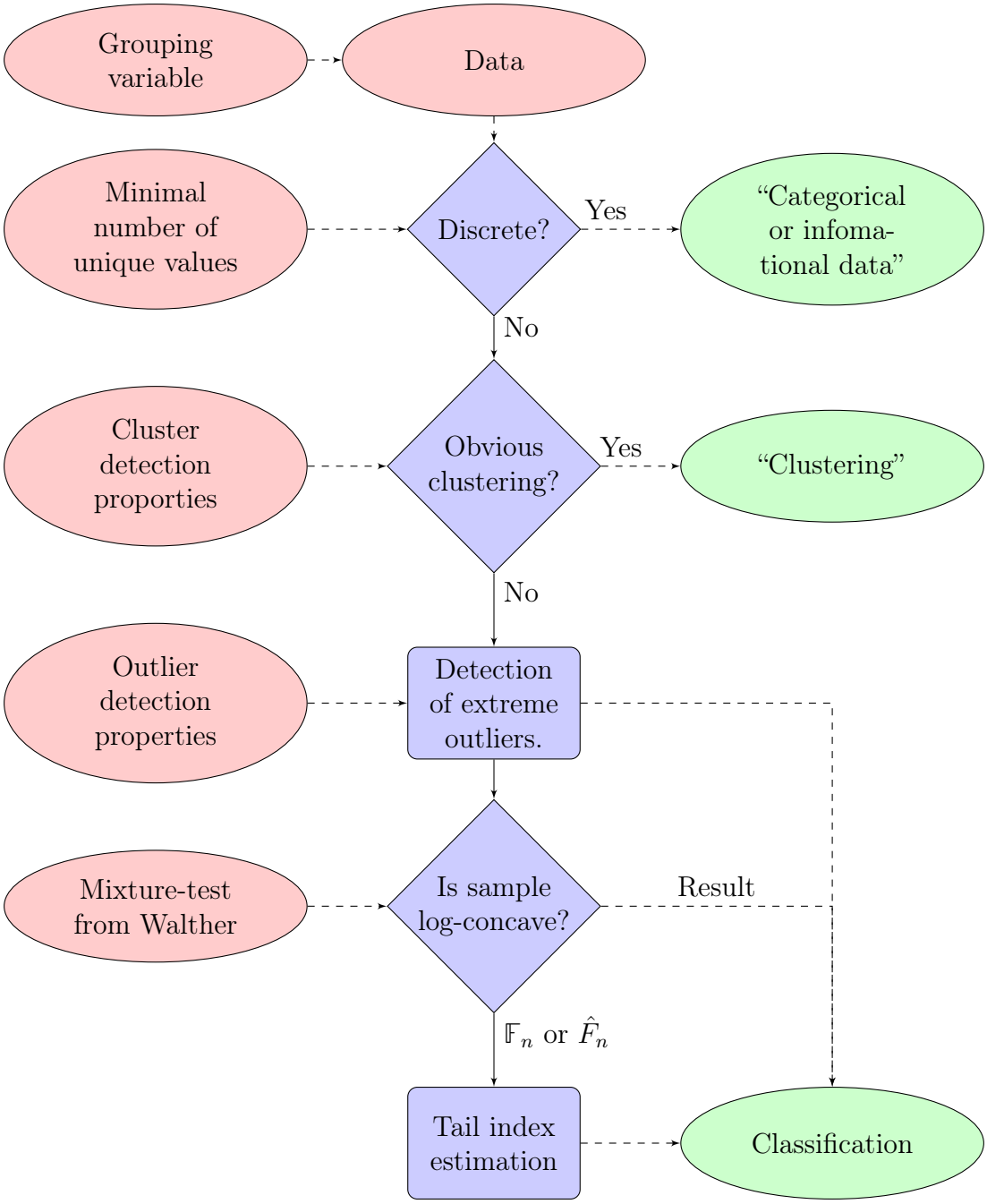
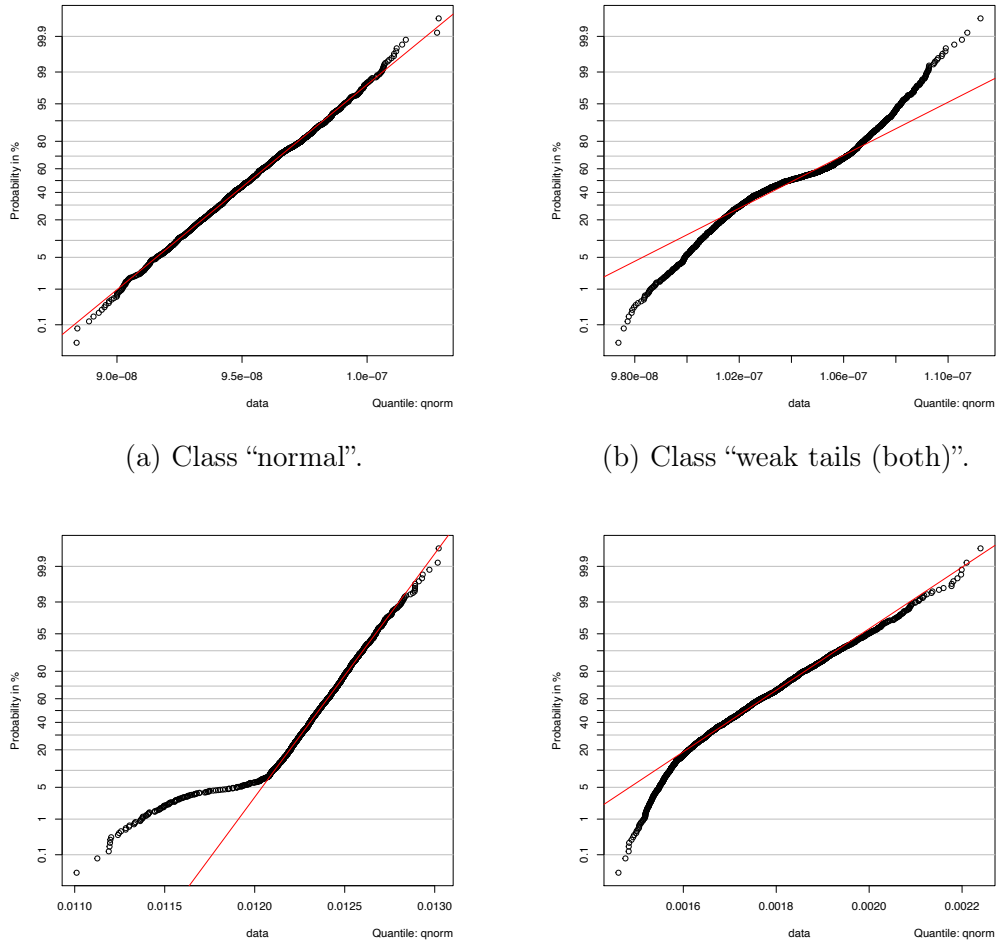


Figure 6.4.: Workflow of the classification algorithm "NN 2.0".

To give an insight in the range of different classes and to illustrate how representatives of some classes look like, exemplary densities and their desired assignment are presented in Figure 6.5. Note that not every density that is under investigation is as clearly assignable as these examples.



(a) Class "normal".

(b) Class "weak tails (both)".

(c) Class "heavy left tail".

(d) Class "weak left tail".

Figure 6.5.: Exemplary variables from real datasets to illustrate the assignment.

In addition, the algorithm provides the useful possibility to group data by a discrete variable in advance of the actual computation. Therefore a lot of problems that could occur can be solved easily. Consider the following example: when devices are tested on two or more machines simultaneously, it is very likely that results on these machines will be slightly different even if one would measure the same devices on both machines. Thus the overall dataset would consist of some shifted

densities that appear as mixtures for nearly every test as long as the data is not grouped by the proper variable. This case was also mentioned in the beginning of Chapter 4 to motivate the described test. When the user guesses the grouping variable correctly, a lot of computation time can be saved by skipping this test on the presence of mixtures and more importantly additional information can be gained by separate classification of the components of a mixture.

6.1. Application of NN 2.0 to Real Data

We will now apply the algorithm to four different datasets. The name of the product or the phase of development are not mentioned due to data privacy restrictions but the chosen datasets can be seen as representative and are all large enough to apply our algorithm.

1. The variables in the first dataset, "dataset A" are classified manually by the author in advance to check the result of the algorithm. The dataset consists of 1391 testing-parameters and 1956 tested devices. The accuracy of the result in comparison to the human classification can be seen in Table 6.1. For detailed results, see Table A.1.

Class	Correct	Not correct	Accuracy in %
Categorical or informational	261	0	100.00%
Clustering	47	30	61.04%
Mixture	162	182	47.09%
Normal	372	35	91.40%
Heavy left	9	23	28.13%
Heavy right	11	9	55.00%
Heavy tails (both)	2	23	8.00%
Weak left & heavy right	4	25	13.79%
Heavy left & weak right	9	32	21.95%
Weak left	15	21	41.67%
Weak right	9	13	40.91%
Weak tails (both)	18	79	18.56%
Overall Accuracy	919	472	**66.07%**

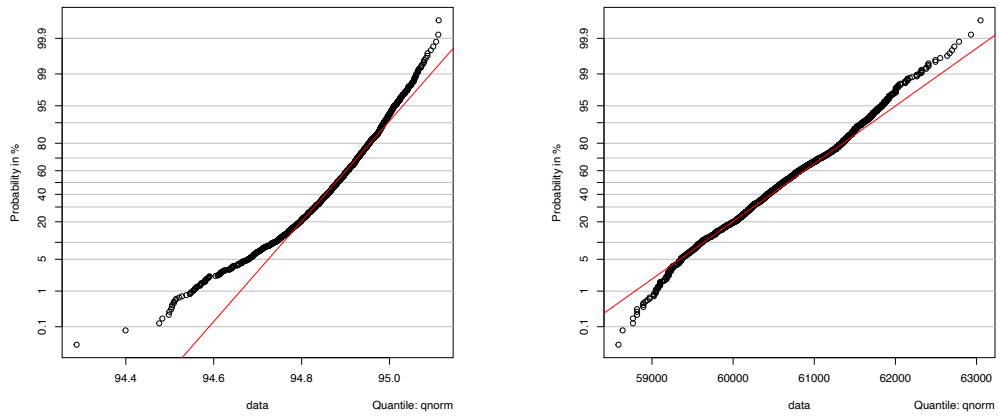
Table 6.1.: Accuracy of NN 2.0 relatively to manual classification for dataset A.

Especially for some (important) classes with heavy tails, the performance of the algorithm is not very satisfying when we look at the results. We will give

two examples for misclassified densities, that can be seen as representatives for many other misclassified densities in the overall dataset.

Figure 6.6a shows the probability plot of the first example. This density is classified as "heavy left & weak right" in the manually done classification. The algorithm rated the left tail as heavy and the right tail as normal but the computed average right tail index was -0.62, which is very close to the assignment as weak (< -0.65). Such cases occur quite often and are not a big problem since the assigned class can nevertheless be seen as a good rating even if it is no exact match with the previously assigned class.

The second example, presented in Figure 6.6b, gives a reason for the far too often assigned class "normal"[1]. The density was rated as "weak tails" in the human classification but were assigned to the class "normal". In fact one can argue, that the tails are weaker than the ones from a normal distribution but this difference is only very small and thus the classification is once more not as bad as it seems in Table 6.1.



(a) Density with "heavy left & weak right" in the human assignment and "heavy left" in the algorithm.

(b) Density with "weak tails" in the human assignment and "normal" in the algorithm.

Figure 6.6.: Exemplary variables from dataset A, that shows two misclassifications.

Moreover we want to mention that many misclassifications of mixture densities happened due to the fact that the test presented in Chapter 4 is not able to detect mixtures with only little differences of the components, as ob-

[1] About 49% of the wrong assignments were wrong assignments of the class "normal".

served in Section 4.8. Such densities are then most probably getting assigned to some class with weak tails (see Table A.1).

2. To avoid a subjective assignment in the manually done classification, the second exemplary dataset, "dataset B" is classified by a colleague in the product engineering team. The dataset was not rated with the same classes as in the first example. Therefore we compare the manually done assignment with the output of NN 2.0 mainly by the presence of heavy tails. Thus we only differentiate between the classes "categorical or informational data", "clustering", "normal or weak" and "heavy tail" (at least one side).

   In analogy to Table 6.1 we present the accuracy of this procedure in Table 6.2. The dataset consists of 2623 testing-parameters and 3682 devices. The detailed results can once more be seen in the appendix in Table A.3.

Class	Correct	Not correct	Accuracy
Categorical or informational	491	22	95.71%
Clustering	74	56	56.92%
Normal or weak	1519	195	88.62%
Heavy tail	152	114	57.14%
Overall accuracy	2236	387	**85.25%**

Table 6.2.: Accuracy of NN 2.0 relatively to manual classification for dataset B.

   The overall accuracy can be seen as satisfying since a lot of misclassifications are still densities with uncertain assignment. Consider the density, presented in Figure 6.7.

   The density is assigned to the class "normal", what can either be seen as correct or not depending on how strict the differentiation between the classes should be. To have the possibility to assign such densities neither to the class "normal" nor to some class with heavy or weak tails, we introduce a new class "semi-normal" for densities that are not perfectly normal but also not explicitly heavy or weak on one side. We will track such densities also with the transformation given in (5.3) and assign the class "semi-normal" when the absolute value of the transformed left and right tail index is 1.

3. In the previous examples, we saw that in some cases it is hard to assign a density to one single class. Thus we will present a third example where we apply the algorithm to "dataset C". Afterwards the assignment is controlled by a colleague of the team who defines whether the classification is "OK" or "not OK". The result is given in Table 6.3. The dataset consists of 2575 testing-parameters and 522 devices.

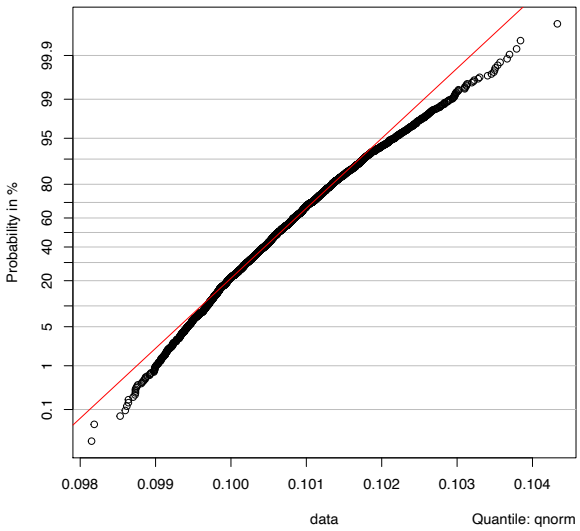


Figure 6.7.: Testing-parameter from dataset B with assignment "normal" and "semi-normal" in future computations.

Class	OK	not OK	OK in %
Categorical of informational	501	0	100.00%
Clustering or extreme mixture	248	163	60.34%
Mixture	57	10	85.07%
Normal	983	66	93.71%
Semi-normal	92	7	92.93%
Heavy left	19	32	37.25%
Heavy right	69	29	70.41%
Heavy tails (both)	9	2	81.82%
Weak left & heavy right	22	2	91.67%
Heavy left & weak right	0	0	-
Weak left	71	37	65.74%
Weak right	65	45	59.09%
Weak tails (both)	46	0	100.00%
$\sum$	2182	393	**84.74%**

Table 6.3.: Quality of NN 2.0 output.

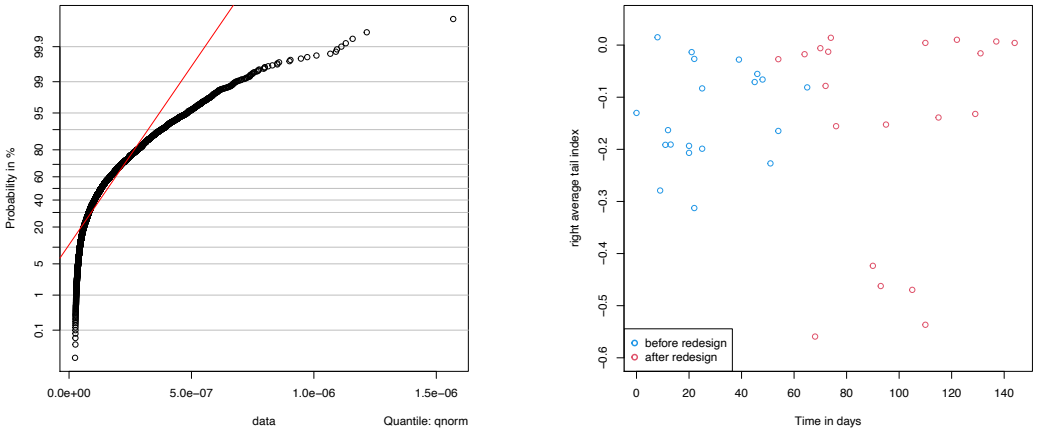
What is conspicuous about the result, is the bad performance for the class "clustering or extreme mixture". A possible explanation is the smaller size of each sample compared to previous examples because the gap-finding proce-

dure could then probably detect even a small amount of outliers as a cluster.

Another thing that comes in mind is the bad performance of classes with heavy tails, especially the classes “heavy left” and “heavy right”. But in fact when we think of possible applications, such misclassifications are not that fatal since testing-parameters get flagged, might get further investigation and will then nevertheless be treated properly.

4. The fourth example we will present, is an application that is a little different than the previous three. We will now analyze the outcome of the same test and the same product over time. The problem with the testing-parameter was the extreme tailing, that occurred in early samples as presented in an exemplary density in Figure 6.8a.

   We have data for this test for several months available. We will not focus on the assigned class but we will make use of the continuity of the presented average tail index. As we can see in Figure 6.8b, the occurrence of extreme tailing disappeared in later samples. In fact, the product was redesigned and after this redesign the problem at least seemed to be not that big as before and happened only under special conditions. This fact can be recognized using the average tail index and shows a possible application of the algorithm or at least of some parts of it.



(a) Testing-parameter with extreme heavy right tail at an early development phase.

(b) Evolution of the right tail index before and after a redesign that solved the heavy tail in some samples.

Figure 6.8.: A heavy tailed distribution and the evolution of the right average tail index of this testing-parameter over time.

7. Conclusion

In this thesis, we attempted to make use of log-concave density estimation in a real-life application and the development of a density classification tool. The main components of this tool were discussed in Chapter 4 and Chapter 5 and their qualitative possibilities were tested in Chapter 6. The application to real data showed satisfying results when we think of the advantageous objectiveness of the procedure. The accuracy is comparable with other techniques that were currently used by the team as described in Chapter 1. Nevertheless, there is room for improvement and further research we will also sketch below.

The algorithm to detect the presence of mixtures worked out quite well but comes with the big shortcoming of a high computation time. Some densities that consist of extremely different components (large difference in location) can be detected by searching for large "gaps" in the data as described in the second step of the overall algorithm. The way we did this detection of gaps was quite heuristic and could likely be improved by using other procedures and approaches. In addition to this, we implemented the possibility to group the data by a categorical variable. This can also be used to detect mixtures if the reason of the presence of mixtures is known or can be guessed. Note that calculations for the same data and different possible categorical variables, for which the data is grouped by in advance, are still faster than the application of the mixture-test on the whole dataset in most cases.

The second big part of the procedure can probably be summarized best with the term "applied tail index estimation". The used techniques worked out well to quantify the shape of each tail of a density. Note that we made use of only a small amount of the information that comes with each calculated value. The transformation of the tail index and the assignment to predefined classes, for which the creation of the boundaries happened heuristically, is something that matched with our predefined target but could be adapted in future work.

Overall we can conclude that the algorithm is by no means meant to replace the high-quality work of experienced engineers as the examples in Chapter 6 may suggest. The developed tool and possible modifications have a wide field of other

applications. We mainly mentioned the straight forward possibility to free engineers from a monotone and error-prone task. Another application was already sketched in the fourth example, presented in Section 6.1. The big advantage of the developed procedure over machine learning algorithms and manually done classification is the objectiveness and the flexibility of the procedure. Such an objective procedure, that rates densities the same way every time allows the tracking of changes of testing-parameters over time, as we sketched in the example. Another possible application is the development of a benchmark, that quantifies the quality of a product in terms of producibility, for which NN 2.0 could be the basis.

A few possible extensions or modifications of the algorithm can be seen as origin of further development:

- The assignment to classes happens through heuristically defined boundaries of the average tail indices on the left and right side separately. This could be adapted by rating the left and right tail index together. But the algorithm does not necessarily need to be limited by a procedure that results in discrete classification. One can probably find ways to make use of the continuity of the tail index and to present the results in an easy and still meaningful way.
- The extension of such an algorithm to multivariate data was not in the scope of this thesis but could be a promising approach to get a better insight in the data since many of the observed testing parameters can be assigned to test-groups. Testing-parameters from the same test group are likely to have dependent distributions and could thus be investigated together.
- Recap that the algorithm NN 2.0 obviously has a predecessor we mentioned in Chapter 1, which is based on a random forest algorithm. The features which were mainly used are quantiles and descriptive indices as skewness and kurtosis. The tail index could be a promising addition to the set of used features.

A. Appendix

A.1. R-Code of NN 2.0

In this section we present the R-code of the classification algorithm NN 2.0 as described in Chapter 6. The functions require the package *logcondens*, see [13] and make use of the function *mixing.test.walther()*, for which the R-code is presented in Section A.2.

Listing A.1: Main function *NN2.0()*.

```
require(logcondens)
# NN2.0() calculates raw results (unclassified and untransformed tail index)
# DATA... input as matrix: rows = tested devices
# cols = testing-parameters
# MIN.DISCRETE... minimum number of unique values for columns being non-discrete
# OUTLIER.REMOVAL... how many times the MAD does outliers have to be away
# CLUSTERING.DISTANCE.SD... "gap"-finding: necessary distance in terms of
# empirical standard deviations
# CLSUETRING.PROPORTION.POINTS.SD... "gap"-finding: necessary proportion of obs.
# on each side of a gap
# MIXTURE.TEST... perform mixture-test be performed for non-log-concave samples?
NN2.0 <- function(DATA, MIN.DISCRETE=10, OUTLIER.REMOVAL=15,
                  CLUSTERING.DISTANCE.SD=0.5,
                  CLUSTERING.PROPORTION.POINTS.SD=0.01,
                  MIXTURE.TEST=FALSE, GROUP.BY=NULL)
{
  nr.devices <- dim(DATA)[1]; nr.cols <- dim(DATA)[2];
  if(!is.null(GROUP.BY)){
    grouping.var.index <- which(colnames(DATA)==GROUP.BY)
    grouping.levels <- unique(DATA[,grouping.var.index])
    nr.levels <- length(grouping.levels)
    n <- nr.levels * nr.cols
  }
  else{
    nr.levels <- 1
    n <- nr.cols
  }
  result.logcon.test <- rep(NA,n)
  iui <- 1:nr.cols # indices under investigation
  data.list <- list(n) # value NA for categorical/informational data

  # step one: categorical or informational data
  ind <- iui
  for(k in ind){
```

```
  dat <- DATA[,k]
  if( (!is.numeric(dat)) | (length(unique(dat))<MIN.DISCRETE) ){
    iui <- iui[-which(iui==k)] # index not under investigation anymore
    for(j in 1:nr.levels){
      data.list[[(k-1)*nr.levels+j]] <- "categorical_or_informational_data"
    }
  }
}
print(paste(length(iui), "_densities_under_further_investigation_(",
            nr.cols-length(iui), "_columns_unconsidered")) # step one done

# step two: identify clustering and extreme mixtures
ind <- iui
for(k in ind){
  dat <- DATA[,k]
  na.indices <- which(is.na(dat)); dat <- na.omit(dat);
  # detect possible large enough gaps:
  gaps <- which(diff( sort((dat-mean(dat))/sd(dat)) )>=CLUSTERING.DISTANCE.SD)
  # chack if detected gaps have enough points on each side:
  if(length(gaps)>0){
    for(j in 1:length(gaps)){
      n.dat <- length(dat)
      clustering <- gaps[j] >= n.dat*CLUSTERING.PROPORTION.POINTS.SD &
                    gaps[j] <= n.dat*(1-CLUSTERING.PROPORTION.POINTS.SD)
      if(clustering){break}
    }
  }
  else{clustering <- FALSE}
  if(clustering){
    iui <- iui[-which(iui==k)] # index not under investigation anymore
    for(j in 1:nr.levels){
      # save result "clustering" to data.list
      data.list[[(k-1)*nr.levels+j]] <- list(data=na.omit(dat),
                                             excluded=na.indices,
                                             na=na.indices,
                                             outliers=numeric(),
                                             classification=
                                              "clustering_or_extreme_mixture")
    }
  }
}
print("Cluster_flagging_done!")


# step three: outliers and NAs:
ind <- iui
nr.sd <- OUTLIER.REMOVAL
for(k in ind){
  dat <- DATA[,k]
  na.indices <- which(is.na(DATA[,k]))
  if(is.null(GROUP.BY)){
    dist <- OUTLIER.REMOVAL*mean(abs(dat-mean(dat,na.rm=TRUE)),na.rm=TRUE)
    outlier.indices <- numeric()
```

```
    outlier.indices <- which(dat<(mean(dat,na.rm=TRUE)-dist) |
                               dat>(mean(dat,na.rm=TRUE)+dist))
    exclude <- c(na.indices, outlier.indices)
    if(length(exclude)==0){
      dat <- DATA[,k]
    }
    else{
      dat <- DATA[-exclude,k]
    }

    # na.indices and outlier.indices showing the index in the overall
    # dataset even if grouped by some discrete var
    data.list[[k]] <- list(data=dat,
                           excluded=c(na.indices, outlier.indices),
                           na=na.indices,
                           outliers=outlier.indices)
  }
  if(!is.null(GROUP.BY)){
    outlier.indices <- numeric()
    for(j in 1:nr.levels){
      dat <- DATA[which(DATA[,grouping.var.index]==grouping.levels[j]),k]
      na.indices.subgroup <- na.indices[which(na.indices %in%
                             which(DATA[,grouping.var.index]==grouping.levels[j])) ]
      dist <- OUTLIER.REMOVAL*mean(abs(dat-mean(dat,na.rm=TRUE)),na.rm=TRUE)
      outlier.indices.subgroup <- numeric()
      outlier.indices.subgroup <- which(dat<(mean(dat,na.rm=TRUE)-dist) |
                                          dat>(mean(dat,na.rm=TRUE)+dist))
      # adding subgroup outliers to outliers (with overall index)
      outlier.indices <- c(outlier.indices,
                           which(DATA[,grouping.var.index]==
                                   grouping.levels[j])
                           [outlier.indices.subgroup]
                          )
      if(length(outlier.indices.subgroup)==0){
        save <- list(data=na.omit(tmp),
                     excluded=c(na.indices.subgroup,which(
                       DATA[,grouping.var.index]==grouping.levels[j])
                       [outlier.indices.subgroup]),
                     na=na.indices.subgroup,
                     outliers=which(DATA[,grouping.var.index]
                                    ==
                                    grouping.levels[j])
                       [outlier.indices.subgroup])
        data.list[[(k-1)*nr.levels+j]] <- save
      }
      else{
        save <- list(data=na.omit(tmp[-outlier.indices.subgroup]),
                     excluded=c(na.indices.subgroup,which(
                       DATA[,grouping.var.index]==grouping.levels[j])
                       [outlier.indices.subgroup]),
                     na=na.indices.subgroup,
                     outliers=which(DATA[,grouping.var.index]
                                    ==
```

```
                                        grouping.levels[j])
                                [outlier.indices.subgroup])
            data.list[[(k-1)*nr.levels+j]] <- save
          }
        }
      }
    }
    print("Outlier flagging done!")

    print("Tail Index Estimation (TIE)")
    # step four: Tail Index Estimation
    ind <- iui
    perc <- 0 # show progress
    k.perc <- 5/6 * 100 # only use 5/6 of the tail index estimations
    for(k in ind){
      if(is.null(GROUP.BY)){
        if(which(ind==k) >= perc/100 * length(ind)){
          print(paste("TIE: approximately ", perc, " % done"))
          perc <- perc+1
        }
        dat <- data.list[[k]]$data
        # calculate LCDE:
        LCD.est <- try(logcondens::logConDens(dat, xgrid = NULL, smoothed=FALSE,
                                              print=FALSE, gam=NULL, xs=NULL),
                       silent=TRUE)
        if(is(LCD.est)=="try-error"){
          data.list[[(k-1)*nr.levels+j]]$TI <- c(NA,NA)
          next
        }
        end <- round(length(dat)*k.perc/100)
        ks.set <- round(seq(from=4, to=end, length.out=round(end-3)))
        samp <- rlogcon2(n=length(dat), LCD=LCD.est)
        p <- suppressWarnings(ks.test(dat,samp)$p.value)
        result.logcon.test[k] <- p
        if(p <= 0.05){
          data.list[[k]]$TI <- TI.calculation.both(LCD=LCD.est, ks=ks.set,
                                                   method="order")
        }
        else{
          data.list[[k]]$TI <- TI.calculation.both(LCD=LCD.est, ks=ks.set,
                                                   method="LCD")
        }
      }
      if(!is.null(GROUP.BY)){
        if(which(ind==k) >= perc/100 * length(ind)){
          print(paste("TIE: approximately ", perc, " % done"))
          perc <- perc+1
        }
        for(j in 1:nr.levels)
        {
          dat <- data.list[[(k-1)*nr.levels+j]]$data
          # calculate LCDE:
          LCD.est <- try(logcondens::logConDens(dat, xgrid=NULL, smoothed=FALSE,
```

```
                                                    print=FALSE, gam=NULL, xs=NULL),
                        silent=TRUE)
        if(is(LCD.est)=="try-error"){
          data.list[[(k-1)*nr.levels+j]]$TI <- c(NA,NA)
          next
        }
        end <- round(length(dat)*k.perc/100)
        ks.set <- round(seq(from=4, to=end, length.out=round(end-3)))
        samp <- rlogcon2(n=length(dat), LCD=LCD.est)
        p <- suppressWarnings(ks.test(dat,samp)$p.value)
        result.logcon.test[k] <- p
        if(p <= 0.05){
          data.list[[(k-1)*nr.levels+j]]$TI <- TI.calculation.both(LCD=LCD.est,
                                                ks=ks.set, method="order")
        }
        else{
          data.list[[(k-1)*nr.levels+j]]$TI <- TI.calculation.both(LCD=LCD.est,
                                                ks=ks.set, method="LCD")
        }
      }
    }
  }
  print("Tail Index Estimation done!")

  # Step five (optional) Mixture Detection
  perc <- 0
  if(MIXTURE.TEST){
    # only samples which were declared as non-log-concave are tested:
    ind <- which(result.logcon.test<=0.05)
    for(k in ind){
      if(which(ind==k) >= perc/100 * length(ind)){
        print(paste("Testing on the presence of mixtures: approximately ",
                    perc, " % done"))
        perc <- perc+1
      }
      dat <- data.list[[k]]$data
      data.list[[k]]$MIXTURE <- mixing.test.walther(DATA=dat, BIN.DIST=0.1,
                                                    C.MAX=3, C.NR=4, C.BOOST=10,
                                                    B=199, show.progress=FALSE)
    }
  }

  return(data.list)
}
```

Listing A.2: Function *TI.assessment()* (transformation of average tail indices and assignment to classes).

```
# TI.assessment() transformes the computed tail indices and classifies them
# DATA.LIST... output of NN2.0()
# weak.boundaries... boundaries for the transformed tail indices -1, -2 and -3
# heavy.boundaries... boundaries for the transformed tail indices 1, 2 and 3
TI.assessment <- function(DATA.LIST, weak.boundaries=c(-0.55,-0.65,-0.8),
```

```
                                        heavy.boundaries=c(−0.35,−0.3,−0.15))
{
  n <− length(DATA.LIST)
  assessment <− matrix(NA, nrow=3, ncol=n)
  rownames(assessment) <− c("class_name",
                            "left_TI_(transformed)", "right_TI_(transformed)")
  for(k in 1:n){
    if(!is.list(DATA.LIST[[k]]))
    {
      assessment[,k] <− c("no_TI_assessment", NA, NA) # class cat./inf. data
    }
    else
    {
      if(any(names(DATA.LIST[[k]])=="classification"))
      {
        if(DATA.LIST[[k]]$classification=="clustering_or_extreme_mixture")
        {
          assessment[,k] <− c("no_TI_assessment", NA, NA) # class clustering
        next
        }
      }
      if(any(is.nan(DATA.LIST[[k]]$TI),is.na(DATA.LIST[[k]]$TI)))
      {
        # Very rare case: when tail index was not computable
        assessment[,k] <− c("Special_Case", NA, NA)
        next
      }
      left <− DATA.LIST[[k]]$TI[1] # left tail index
      right <− DATA.LIST[[k]]$TI[2] # right tail index
      # transformed TI
      trafo.left <− 0
      trafo.right <− 0
      if(left < weak.boundaries[1]){
        trafo.left <− −1
        if(left < weak.boundaries[2]){
          trafo.left <− −2
          if(left < weak.boundaries[3]){
            trafo.left <− −3
          }
        }
      }
      if(left > heavy.boundaries[1]){
        trafo.left <− 1
        if(left > heavy.boundaries[2]){
          trafo.left <− 2
          if(left > heavy.boundaries[3]){
            trafo.left <− 3
          }
        }
      }
      if(right < weak.boundaries[1]){
        trafo.right <− −1
        if(right < weak.boundaries[2]){
```

```
        trafo.right <- -2
        if(right < weak.boundaries[3]){
          trafo.right <- -3
        }
      }
    }
    if(right > heavy.boundaries[1]){
      trafo.right <- 1
      if(right > heavy.boundaries[2]){
        trafo.right <- 2
        if(right > heavy.boundaries[3]){
          trafo.right <- 3
        }
      }
    }
    assessment[2,k] <- trafo.left
    assessment[3,k] <- trafo.right
    if(abs(trafo.left)+abs(trafo.right) <= 1){
      assessment[1,k] <- "normal"
    }
    if(abs(trafo.left)==1 & abs(trafo.right)==1){
      assessment[1,k] <- "semi-normal"
    }

    if(trafo.left <= -2 & trafo.right <= -2){
      assessment[1,k] <- "weak tails (both sides)"
    }
    if(trafo.left >= 2 & trafo.right >= 2){
      assessment[1,k] <- "heavy tails (both sides)"
    }
    if(trafo.left <= -2 & trafo.right >= 2){
      assessment[1,k] <- "weak left & heavy right"
    }
    if(trafo.left <= -2 & abs(trafo.right) <= 1){
      assessment[1,k] <- "weak left"
    }
    if(abs(trafo.left) <= 1 & trafo.right >= 2){
      assessment[1,k] <- "heavy right"
    }
    if(trafo.left >= 2 & trafo.right <= -2){
      assessment[1,k] <- "heavy left & weak right"
    }
    if(trafo.left >= 2 & abs(trafo.right) <= 1){
      assessment[1,k] <- "heavy left"
    }
    if(abs(trafo.left) <= 1 & trafo.right <= -2){
      assessment[1,k] <- "weak right"
    }
  }
 }
 return(assessment)
}
```

Listing A.3: Auxiliary functions for NN 2.0.

```
# Auxiliary functions and modifications

# rlogcon2()... draws a sample of size n from a given log-concave density (LCD)
# -> modification of logcondens::rlogcon() but with LCD as input instead of samp
# -> saves computation time if LCD is already computed
# n... sample size that should be drawn
# LCD... object of class 'dlc' (see logcondens-package)
rlogcon2 <- function(n, LCD)
{
  U <- runif(n)
  X <- logcondens::quantilesLogConDens(U, LCD)[, "quantile"]
  return(X)
}


# TI.calculation.both() calculates the average tail index of both sides
# with just one LCDE and pickands esstimator
# -> modification of smoothtail::pickands()
# LCD... object of class 'dlc' (see logcondens-package)
# ks... set of indices k for which the average tail index should be calculated
# method... either "order" or "LCD", depending on which distribution function
# estimator should be used
TI.calculation.both <- function(LCD, ks, method="order")
{
  n <- LCD$n
  x <- LCD$xn
  v1 <- 1:length(ks) * NA # v for left side
  v2 <- 1:length(ks) * NA # v for right side
  if(method=="order"){
    for (k in ks) {
      k2 <- floor(k/4)
      v1[k] <- (x[2 * k2] - x[k2])/(x[4 * k2] - x[2 * k2])
      v2[k] <- (x[n - k2 + 1] - x[n - 2 * k2 + 1])/
        (x[n - 2 * k2 + 1] - x[n - 4 * k2 + 1])
    }
    v1 <- suppressWarnings(log(v1)/log(2))
    v2 <- suppressWarnings(log(v2)/log(2))
  }

  if(method=="LCD"){
    for (k in ks) {
      q1 <- logcondens::quantilesLogConDens((k/4-1)/n, LCD)[, "quantile"]
      q2 <- logcondens::quantilesLogConDens((k/2-1)/n, LCD)[, "quantile"]
      q3 <- logcondens::quantilesLogConDens((k-1)/n, LCD)[, "quantile"]
      v1[k] <- (q2 - q1)/(q3 - q2)
      q1 <- logcondens::quantilesLogConDens((n - k/4 + 1)/n, LCD)[, "quantile"]
      q2 <- logcondens::quantilesLogConDens((n - k/2 + 1)/n, LCD)[, "quantile"]
      q3 <- logcondens::quantilesLogConDens((n - k + 1)/n, LCD)[, "quantile"]
      v2[k] <- (q1 - q2)/(q2 - q3)
    }
    v1 <- suppressWarnings(log(v1)/log(2))
    v2 <- suppressWarnings(log(v2)/log(2))
```

```
  }
  res <- c(mean(v1[v1!=Inf & v1!=-Inf], na.rm=TRUE),
           mean(v2[v2!=Inf & v2!=-Inf], na.rm=TRUE))
  return(res)
}
```

A.2. R-Code of the Test on the Presence of Mixtures

In this section we present the R-code of the test on the presence of mixtures, which we investigated in Chapter 4. The code was originally implemented in Matlab, as described in [43] and was transferred to R and only slightly modified.

Listing A.4: Main function *mixing.test.walther()*.

```
# mixing.test.walther() calculates p-value of walther-test
# DATA... input data sample
# BIN.DIST... number of standard deviations for the binning procedure
# C.MAX... maximal value of c-grid
# C.NR... number of equidistant vakues in c-grid
# C.BOOST... multiplicator of size of c-grid for null model
# B... monte carlo sample size
# L1... technical parameter (approximation for log(0)), suggeston: -10
# show.progress... boolean variable determining if progress should be shown
mixing.test.walther <- function(DATA, BIN.DIST=1/10, C.MAX, C.NR, C.BOOST, B,
                                L1=-10, show.progress=FALSE)
{
  nobs <- length(na.omit(DATA))
  new.dat <- data.prep(DATA=DATA, BIN.DIST=BIN.DIST, L1=L1)
  swork <- new.dat$SWORK; x <- new.dat$X; w <- new.dat$W; cw <- new.dat$CW;
  n <- length(x);

  swork <- ICMA(S=swork,X=x,C=0,CW=cw,L1=L1,show.progress=show.progress)
  snull <- swork[(n-1):1]; s <- snull; xorig<-x

  # test statistic(s) for original data
  c.increment.origdat <- C.MAX/(C.BOOST*C.NR-1)
  res.origdat <- numeric(C.BOOST*C.NR)
  for(k in 1:(C.NR*C.BOOST)){
    c <- (k-1)*c.increment.origdat
    swork <- ICMA(S=swork,X=x,C=c,CW=cw,L1=-10,show.progress=show.progress)
    s <- swork[(n-1):1]
    s <- s+c*(x[2:n]+x[1:(n-1)])
    res.origdat[k] <- STATISTIC(SLOPE=s,X=x,L1=L1)
  }

  # simulate test statistic(s) by sampling from nullmodel
  c.increment <- C.MAX/(C.NR-1)
  res.mat <- matrix(NA, nrow=B, ncol=C.NR)
  for(sim in 1:B){
```

```
    dat <- SAMPLE(X<-xorig,S=snull,M=nobs,L1=L1)
    new.dat <- data.prep(DATA=dat, BIN.DIST=BIN.DIST, L1=L1)
    swork <- new.dat$SWORK; x <- new.dat$X; w <- new.dat$W; cw <- new.dat$CW;
    n <- length(x);
    for(k in 1:C.NR){
      c <- (k-1)*c.increment
      swork <- ICMA(S=swork,X=x,C=c,CW=cw,L1=-10,show.progress=show.progress)
      s <- swork[(n-1):1]
      s <- s+c*(x[2:n]+x[1:(n-1)])
      res.mat[sim,k] <- STATISTIC(SLOPE=s,X=x,L1=L1)
    }
    if(show.progress){
      print(noquote(paste("simulation run ", sim)))
    }
  }

  # Extension to original code (no standardization)
  maximum.origdat <- max(res.origdat)
  nulldist <- 1:B
  for(sim in 1:B){nulldist[sim] <- max(res.mat[sim,2:C.NR])}
  p_value <- (sum(nulldist>maximum.origdat)+1)/(B+1)
  return(p_value)
}

# data.prep() is an auxiliary function and prepares data for mixing.test.walther
data.prep <- function(DATA, BIN.DIST, L1)
{
  dat <- na.omit(DATA)
  scaled.dat <- sort((dat-mean(dat))/sd(dat))

  # binning:
  x <- c(scaled.dat[1]-var(scaled.dat)/10, scaled.dat)
  n <- length(x) # n refers to the number of bins here
  n.obs <- n-1 # number of observations is one less than number of bins
  w <- rep(1, times=n) # default vector of weights
  i <- 2; x.max <- x[n]; # initalization of while loop (x is already sorted)
  while((x.max-x[i]) > BIN.DIST){
    next.ind <- which(x>(x[i]+BIN.DIST))[1]
    tmp.dist <- x[next.ind] - x[i]
    if((next.ind-i)>1){
      for(j in (i+1):(next.ind-1)){
        w[i] <- w[i] + (x[next.ind]-x[j])/tmp.dist
        w[next.ind] <- w[next.ind] + (x[j]-x[i])/tmp.dist
      }
      x <- x[-((i+1):(next.ind-1))]
      w <- w[-((i+1):(next.ind-1))]
    }
    i <- i+1
  }
  n <- length(x)
  x <- x[1:i];
  w[i] <- w[i]+n-i; w<-w[1:i];
  w <- w/n.obs
```

```
  n <- length(x)
  cw <- cumsum(w[n:2]); cw <- cw[(n-1):1]*diff(x)

  # initial s:
  # create swork, the default vector of slopes
  # => fitting a normal distribution over data as initial guess
  mu <- mean(x)
  v <- var(x)
  swork <- 0.5/v * c((-2*L1*v-v*log(2*pi*v)-(x[2]-mu)^2)/(x[2]-x[1]),
                     2*mu - x[3:n] - x[2:(n-1)])
  swork <- swork[(n-1):1]
  return(list(SWORK=swork, X=x, W=w, CW=cw))
}
```

Listing A.5: Iterative Convex Minorant Algorithm (ICMA).

```
# ICMA() performs the Iterative Convex Minorant Algorithm
# S,X,C,CW... set of properties of the sample as output of data.prep()
# ETA,EPS... technical parameters of ICMA
# W.LOW... lower bound for weights
ICMA <- function(S, X, C, CW, L1, ETA=0.00001, EPS=0.01, W.LOW=0.001,
                 show.progress)
{
  n <- length(X)
  xx <- S
  grad <- gradphi(S=S, X=X, C=C, CW=CW, L1=L1)
  w.ICMA <- grgrphi(S=S, X=X, C=C, L1=L1)
  w.ICMA[w.ICMA<W.LOW] <- W.LOW
  t1 <- abs(sum(xx*grad))
  t2 <- abs(sum(grad))
  t3 <- min(cumsum(grad[(n-1):1]))
  ctr <- 0; gain <- 1;
  while( (t1>ETA | t2>ETA | t3<(-ETA)) & ctr<100 & gain>ETA ){
    oldphi <- phi(S=xx, X=X, C=C, CW=CW, L1=L1)
    ytil <- MINLOWSET(G=(xx-grad/w.ICMA), W=w.ICMA)
    if(phi(S=ytil,X=X,C=C,CW=CW,L1=L1) < oldphi+EPS*sum(grad*(ytil-xx))){
      xx <- ytil
    }
    else{
      lam <- 1; ss <- 0.5; z <- ytil; ctr2 <- 0;
      t4 <- phi(S=z,X=X,C=C,CW=CW,L1=L1) < (phi(S=xx, X=X, C=C, CW=CW, L1=L1)+
                                            (1-EPS)*sum(grad*(z-xx)))
      t5 <- phi(S=z,X=X,C=C,CW=CW,L1=L1) > (phi(S=xx, X=X, C=C, CW=CW, L1=L1)+
                                            EPS*sum(grad*(z-xx)))
      while( (isTRUE(t4)|isTRUE(t5)) & ctr2<6){
        if(isTRUE(t4)){
          lam <- lam+ss
        }
        else{
          lam <- lam-ss
        }
        z <- xx+lam*(ytil-xx); ss <- ss/2;
        t4 <- phi(S=z,X=X,C=C,CW=CW,L1=L1) < (phi(S=xx, X=X, C=C, CW=CW, L1=L1)+
```

```
                                                (1−EPS)∗sum(grad∗(z−xx)))
        t5 <− phi(S=z,X=X,C=C,CW=CW,L1=L1) > (phi(S=xx, X=X, C=C, CW=CW, L1=L1)+
                                                EPS∗sum(grad∗(z−xx)))
        ctr2 <− ctr2+1
      }
      xx <− z
    }
    grad <− gradphi(S=xx,X=X,C=C,CW=CW,L1=L1)
    w.ICMA <− grgrphi(S=xx, X=X, C=C, L1=L1)
    w.ICMA[w.ICMA<W.LOW] <− W.LOW
    t1 <− abs(sum(grad∗xx))
    t2 <− abs(sum(grad))
    t3 <− min(cumsum(grad[(n−1):1]))
    ctr <− ctr+1; gain <− oldphi−phi(S=xx,X=X,C=C,CW=CW,L1=L1)
  }
  if(ctr>99 & show.progress){
    print(noquote("Number of Iterations in ICMA exceeded"))
  }
  return(xx)
}
```

Listing A.6: Auxiliary functions for *mixing.test.walther()* and *ICMA()*.

```
# Auxiliary functions for ICMA() and thus mixing.test.walther()

# phi() is target function of optimization in paper
phi <− function(S, X, C, CW, L1)
{
  n <− length(X)
  S <− S[(n−1):1]
  x1sq <− X[1]^2
  a <− L1 + cumsum(c(0, S∗(X[2:n]−X[1:(n−1)])))
  res <− (X[3]−X[2])∗exp(a[2]+C∗(X[2]^2−x1sq)) +
    (X[n]−X[n−1])∗exp(a[n]+C∗(X[n]^2−x1sq)) +
    sum((X[4:n]−X[2:(n−2)])∗exp(a[3:(n−1)]+C∗(X[3:(n−1)]^2−x1sq)));
  return(0.5∗res−sum(S∗CW))
}

# gradphi() is the gradient of target function in paper
gradphi <− function(S, X, C, CW, L1)
{
  n <− length(X)
  S <− S[(n−1):1]
  x1sq <− X[1]^2
  a <− L1 + cumsum(c(0, S∗(X[2:n]−X[1:(n−1)])))
  cumulative.res <− cumsum(c((X[n]−X[n−1])∗exp(a[n]+C∗(X[n]^2−x1sq)),
                             (X[n:4]−X[(n−2):2])∗exp(a[(n−1):3]+
                                                     C∗(X[(n−1):3]^2−x1sq)),
                             (X[3]−X[2])∗exp(a[2]+C∗(X[2]^2−x1sq))) )
  res <− 0.5∗cumulative.res[(n−1):1]∗(X[2:n]−X[1:(n−1)]) − CW
  return(res[(n−1):1])
}

# grgrphi() is diagonal of second derivative of target function in (5)
```

```
grgrphi <- function(S, X, C, L1)
{
  n <- length(X)
  S <- S[(n-1):1]
  x1sq <- X[1]^2
  a <- L1 + cumsum(c(0, S*(X[2:n]-X[1:(n-1)])))
  cumulative.res <- cumsum(c((X[n]-X[n-1])*exp(a[n]+C*(X[n]^2-x1sq)),
                             (X[n:4]-X[(n-2):2])*exp(a[(n-1):3]+
                                                     C*(X[(n-1):3]^2-x1sq)),
                             (X[3]-X[2])*exp(a[2]+C*(X[2]^2-x1sq))) )
  res <- 0.5*cumulative.res[(n-1):1]*(X[2:n]-X[1:(n-1)])^2
  return(res[(n-1):1])
}

# MINLOWSET(): minimum lower set algorithm
MINLOWSET <- function(G, W)
{
  n.tmp <- length(W)
  g <- G
  curr <- 1
  while(curr < (n.tmp+0.5)){
    h <- cumsum(g[curr:n.tmp]*W[curr:n.tmp])/cumsum(W[curr:n.tmp])
    ind <- which.min(h[(n.tmp-curr+1):1])
    val <- (h[(n.tmp-curr+1):1])[ind]
    ind <- n.tmp+1-ind
    g[curr:ind] <- base::rep(val, times=(ind-curr+1))
    curr <- ind +1
  }
  return(g)
}

# Test statistics
# SLOPES: slopes between points x_i and x_i+1 of log(f(x)) and NOT of the
# function phi(x) (notation "S" in previous functions!)
STATISTIC <- function(SLOPE, X, L1)
{
  n <- length(X)
  dpl <- numeric(n); dpl[1] <- L1;
  for(i in 2:n){
    dpl[i] <- dpl[i-1]+SLOPE[i-1]*(X[i]-X[i-1])
  }
  f <- exp(dpl)
  maxtoend <- 1:(n-1); ind <- 1:(n-1);
  maxtoend[n-1] <- SLOPE[n-1]; ind[n-1]<-n;
  for(i in (n-2):1){
    maxtoend[i] <- maxtoend[i+1]; ind[i] <- ind[i+1];
    if(SLOPE[i]>maxtoend[i+1]){maxtoend[i] <- SLOPE[i]; ind[i] <- i+1}
  }
  return(max((maxtoend-SLOPE)*f[ind]*f[1:(n-1)]/(f[ind]+f[1:(n-1)])))
}

# SAMPLE(): sampling from nullmodel
SAMPLE <- function(X, S, M, L1)
```

```
{
  n <- length(X)
  cdf <- numeric(n); lfold <- L1+S[1]*(X[2]-X[1]); sample <- 1:M;
  for(i in 3:n){
    lfcurr <- lfold+S[i-1]*(X[i]-X[i-1])
    cdf[i] <- cdf[i-1]+(X[i]-X[i-1])*(exp(lfcurr)+exp(lfold))/2
    lfold <- lfcurr
  }
  cdf <- cdf/(cdf[n])
  for(i in 1:M){
    tmp <- cdf-runif(1); tmp[tmp<0] <- 0;
    ind <- which.min(tmp[n:1]); tmp <- tmp[n:1][ind];
    ind <- n+1-ind
    if(abs(S[ind])<0.00001){
      tmp <- runif(1, X[ind], X[ind+1])
    }
    else{
      if(S[ind]<0){
        tmp <- rexp(1,1)/(-S[ind]*(X[ind+1]-X[ind]))
        tmp <- tmp-round(tmp-0.5)
        tmp <- X[ind]+(X[ind+1]-X[ind])*tmp
      }
      else{
        tmp <- rexp(1,1)/(S[ind]*(X[ind+1]-X[ind]))
        tmp <- tmp-round(tmp-0.5)
        tmp <- X[ind]+(X[ind+1]-X[ind])*(1-tmp)
      }
    }
    sample[i] <- tmp
  }
  return(sample)
}
```

A.3. Detailed Simulation Results

In the following tables we present more detailed simulation results of the examples in Section 6.1.

Class	0	C or M	M	N	HL	HR	HT	WH	HW	WL	WR	WT	$\sum$
0	261	0	0	0	0	0	0	0	0	0	0	0	261
C	3	47	11	7	1	0	0	1	0	2	4	1	77
M	0	14	148	70	1	2	2	1	0	23	32	51	344
N	0	10	11	372	3	1	0	0	0	8	2	0	407
HL	0	1	4	18	9	0	0	0	0	0	0	0	32
HR	0	1	0	8	0	11	0	0	0	0	0	0	20
HT	0	2	3	13	2	3	2	0	0	0	0	0	25
WH	0	0	3	12	0	6	0	4	0	4	0	0	29
HW	0	0	1	26	4	0	0	0	9	0	1	0	41
WL	0	0	1	20	0	0	0	0	0	15	0	0	36
WR	0	0	0	13	0	0	0	0	0	0	9	0	22
WT	0	2	10	44	0	0	0	0	0	4	19	18	97
$\sum$	264	77	192	603	20	23	4	6	9	56	67	70	**1391**

Table A.1.: Result of NN 2.0 for data A with the manual classification by row and the algorithm's classification by column (encoding in Table A.2).

Classification	Code
Categorical or informational	0
Clustering	C
Mixture	M
Normal	N
Heavy left	HL
Heavy right	HR
Heavy tails (both)	HT
Weak left & heavy right	WH
Heavy left & weak right	HW
Weak left	WL
Weak right	WR
Weak tails (both)	WT

Table A.2.: Encoding for different classes in DATA A.

Class	*Categorical or informational*	*Clustering*	*Normal or weak*	*Heavy component*	$\sum$
Categorical or informational	491	6	3	13	513
Clustering	0	74	44	12	130
Normal or weak	0	72	1519	123	1714
Heavy tail	0	8	106	152	266
$\sum$	491	160	1672	300	**2623**

Table A.3.: Result of NN 2.0 for data B with the manual classification by row and the algorithm's classification.

Bibliography

[1] An, M. Y. “Log-Concave Probability Distributions: Theory and Statistical Testing”. In: *Duke University Dept of Economics (Working Paper)* 95-03 (1997).

[2] Bagnoli, M. and Bergstrom, T. “Log-Concave Probability and its Applications”. In: *Economic Theory* 26.2 (2005), pp. 445–469.

[3] Balabdaoui, F., Rufibach, K., and Wellner, J. A. “Limit Distribution Theory for Maximum Likelihood Estimation of a Log-Concave Density”. In: *The Annals of Statistics* 37.3 (2009), pp. 1299–1331.

[4] Balkema, A. A. and de Haan, L. “Residual Life Time at Great Age”. In: *The Annals of Probability* 2.5 (1974), pp. 792–804.

[5] Cule, M., Gramacy, R., and Samworth, R. J. “LogConcDEAD: An R Package for Maximum Likelihood Estimation of a Multivariate Log-Concave Density”. In: *Journal of Statistical Software* 29.2 (2009), pp. 1–20.

[6] Cule, M. and Samworth, R. J. “Theoretical Properties of the Log-Concave Maximum Likelihood Estimator of a Multidimensional Density”. In: *Electronic Journal of Statistics* 4 (2010), pp. 254–270.

[7] Cule, M., Samworth, R. J., and Stewart, M. “Maximum Likelihood Estimation of a Multi-dimensional Log-Concave Density”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.5 (2010), pp. 545–607.

[8] Danielsson, J., Ergun, L. M., de Haan, L., and de Vries, C. G. *Tail Index Estimation: Quantile Driven Threshold Selection*. Available at SSRN: http://dx.doi.org/10.2139/ssrn.2717478. (2016).

[9] Dehnad, K. “Review of "Density Estimation for Statistics and Data Analysis"”. In: *Technometrics* 29.4 (1987), pp. 495–495.

[10] Doksum, K. A. and Bickel, P. J. “Test for Monotone Failure Rate Based on Normalized Spacings”. In: *The Annals of Mathematical Statistics* 40.4 (1969), pp. 1216–1235.

[11] Dümbgen, L: and Rufibach, K. "Maximum Likelihood Estimation of a Log-Concave Density and its Distribution Function: Basic Properties and Uniform Consistency". In: *Bernoulli* 15.1 (2009), pp. 40–68.

[12] Dümbgen, L., Hüsler, A., and Rufibach, K. "Active Set and EM Algorithms for Log-Concave Densities based on Complete and Censored Data". In: *Technical Report 61, University of Bern* (2007).

[13] Dümbgen, L. and Rufibach, K. "logcondens: Computations Related to Univariate Log-Concave Density Estimation". In: *Journal of Statistical Software* 39.6 (2011), pp. 1–28.

[14] Dümbgen, L., Samworth, R. J., and Schuhmacher, D. "Approximation by Log-Concave Distributions, with Applications to Regression". In: *The Annals of Statistics* 39.2 (2011), pp. 702–730.

[15] Epstein, B. "Tests for the Validity of the Assumption that the Underlying Distribution of Life is Exponential". In: *Technometrics* 2.1 (1960), pp. 83–101.

[16] Fisher, R. A. and Tippett, L. H. C. "Limiting Forms of the Frequency Distribution of the Largest or Smallest Member of a Sample". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 24. 1928, pp. 180–190.

[17] Gnedenko, B. "Sur La Distribution Limite Du Terme Maximum D'Une Serie Aleatoire". In: *Annals of Mathematics* 44.3 (1943), pp. 423–453.

[18] *Google Scholar*. URL: `https://scholar.google.at`.

[19] Grenander, U. "On the Theory of Mortality Measurement". In: *Scandinavian Actuarial Journal* 1956.2 (1956), pp. 125–153.

[20] Ibragimov, I. A. "On the Composition of Unimodal Distributions". In: *Theory of Probability & Its Applications* 1.2 (1956), pp. 255–260.

[21] *Infineon Technologies Austria AG*. URL: `www.infineon.com/cms/austria/`.

[22] Jongbloed, G. "The Iterative Convex Minorant Algorithm for Nonparametric Estimation". In: *Journal of Computational and Graphical Statistics* 7.3 (1998), pp. 310–321.

[23] Kim, Arlene K. H. and Samworth, Richard J. "Global Rates of Convergence in Log-Concave Density Estimation". In: *The Annals of Statistics* 44.6 (2016), pp. 2756–2779.

[24] Müller, S. and Rufibach, K. "On the Max-domain of Attraction of Distributions with Log-Concave Densities". In: *Statistics & Probability Letters* 78.12 (2008), pp. 1440–1444.

[25] Müller, S. and Rufibach, K. "Smooth Tail-index Estimation". In: *Journal of Statistical Computation and Simulation* 79.9 (2009), pp. 1155–1167.

[26] Pickands III, J. "Statistical Inference Using Extreme Order Statistics". In: *The Annals of Statistics* 3.1 (1975), pp. 119–131.

[27] Prekopa, A. "Contributions to the Theory of Stochastic Programming". In: *Mathematical Programming* 4.1 (1973), pp. 202–221.

[28] Proschan, F. and Pyke, R. "Tests for Monotone Failure Rate". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 3. 1967, pp. 293–313.

[29] *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: `www.R-project.org/`.

[30] Roberts, A. W. and Varberg, D. E. "Convex Functions". In: *Pure and Applied Mathematics*. 1973, pp. 88–120.

[31] Robeva, E., Sturmfels, B., and Uhler, C. "Geometry of Log-Concave Density Estimation". In: *Discrete & Computational Geometry* 61 (2019), pp. 136–160.

[32] Rufibach, K. "Computing Maximum Likelihood Estimators of a Log-Concave Density Function". In: *Journal of Statistical Computation and Simulation* 77.7 (2007), pp. 561–574.

[33] Rufibach, K. "Log-Concave Density Estimation and Bump Hunting for i.i.d. Observations". PhD thesis. Universität Bern & Georg-August Universität zu Göttingen, 2006.

[34] Rufibach, K. and Müller, S. *smoothtail: Smooth Estimation of GPD Shape Parameter*. R package version 2.0.5. 2006.

[35] Samworth, R. J. "Recent Progress in Log-Concave Density Estimation". In: *Statistical Science* 33.4 (2018), pp. 493–509.

[36] Saumard, A and Wellner, J. A. "Log-Concavity and Strong Log-Concavity: a Review". In: *Statistics Surveys* 8 (2014), pp. 45–114.

[37] Scott, D. W. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

[38] Scott, D. W. "On Optimal and Data-based Histograms". In: *Biometrika* 66.3 (1979), pp. 605–610.

[39] Silverman, B. W. *Density Estimation for Statistics and Data Analysis*. CRC Press, 1986.

[40] Silverman, B. W. "On the Estimation of a Probability Density Function by the Maximum Penalized Likelihood Method". In: *The Annals of Statistics* 10.3 (1982), pp. 795–810.

[41] Sturges, H. A. "The Choice of a Class Interval". In: *Journal of the American Statistical Association* 21.153 (1926), pp. 65–66.

[42] Süveges, M. and Davison, A. C. "Model Misspecification in Peaks Over Threshold Analysis". In: *Annals of Applied Statistics* 4.1 (2010), pp. 203–221.

[43] Walther, G. "Detecting the Presence of Mixing with Multiscale Maximum Likelihood". In: *Journal of the American Statistical Association* 97.458 (2002), pp. 508–513.

[44] Walther, G. "Inference and Modeling with Log-Concave Distributions". In: *Statistical Science* 24.3 (2009), pp. 319–327.

[45] Walther, G. "Multiscale Maximum Likelihood Analysis of a Semiparametric Model, with Applications". In: *The Annals of Statistics* 29.5 (2001), pp. 1297–1319.