technische universität
dortmund

fakultät
statistik

Dissertation

in partial fulfillment of the requirements for the degree of

Doktor der Naturwissenschaften

Sign Depth for Parameter Tests in Multiple Regression

by

M.Sc. Melanie Horn

Referees:	Prof. Dr. Christine Müller
	Prof. Dr. Roland Fried
Commission chairperson:	Prof. Dr. Jörg Rahnenführer
Assessor:	Dr. Uwe Ligges
Submitted:	March 23, 2021
Day of Oral Exam:	July 13, 2021

Department of Statistics
Statistics with Applications in the Field of Engineering Sciences
TU Dortmund University

Abstract

This thesis deals with the question how the sign depth test can be applied in the case of multiple regression. Because the result of this test depends on the ordering of the residuals and most times no inherent order is available for multidimensional values one has to think about suitable methods to order these values. In this thesis 13 different ordering methods are described, analyzed and compared with respect to characteristics, computational behavior and performance when using them in the context of sign depth tests. For the last one, several simulations of power functions for many different settings have been carried out. In the simulations different data situations as well as different multiple regression models and different parameters of the sign depth were examined. It is shown in this thesis that a group of so-called "distance based ordering methods" performs best and leads to satisfying results of the sign depth test. Also compared to other tests for regression parameters like the Wald test or the classical sign test the sign depth test performs satisfyingly and especially in the case of testing for model checks it performs clearly better. In addition, this thesis describes the contents and functionality of the `R`-package `GSignTest` which was written for this thesis and contains implementations of the sign depth, the sign depth test and the different ordering methods.

Table of Contents

Chapter 1

Motivation

Many questions and problems in science cannot be solved when it is looked for solutions only in the respective area of expertise. Often, thinking out of the box is necessary to find solutions in other areas of expertise or get inspired by problems in other fields. Some sub-areas of science are strongly connected to each other anyway: For example, the statistics could even not exist without mathematics and, in the last decades, also not without computer science. On the other hand, nearly all sub-areas of science need statistics. A good current example of this is the handling of the ongoing pandemic in politics and medicine. All actions are based on the analysis of the current infection numbers, hospitalization rates, numbers of deaths and vaccination rates. Also, for finding a vaccine statistics was/is very important for executing and analyzing the necessary clinical studies. But also in other fields of science statistics is very important: For example in the field of engineering sciences when analyzing the fatigue process of buildings and constructions. Or in economics when analyzing and predicting stock values. Further usage and applications of statistics can be found in nearly all fields of science: Biology, chemistry, physics, psychology, social sciences, sports (including e-sports) and many more. But also many methods in statistics and related sciences are inspired from different fields of science, not only for applications in the respective fields itself. Good examples of this are (optimization) algorithms in computer science which are inspired by processes from biology: Evolutionary algorithms, genetic algorithms, ant algorithms, neural networks and some more. Another example is the generation of random numbers. These methods are often inspired by physical processes like radioactive decay or thermal noise.

Also the statistical problem which led to this thesis could not be solved by staying completely in the field of statistics. The aim of this thesis sounds simple: Finding a way to apply the sign depth test to multiple regression. The sign depth test and the sign depth itself have their origins in the field of robust statistics, more precisely in the sub-area of data depths. Data depths were invented to have a more powerful generalization of the median, for example in the multivariate case. The first approach of this can be found in Tukey (1975) who proposed the so-called "halfspace depth". Over the years many extensions of the halfspace depth have been proposed and many further data depth concepts were developed. One of these concepts is the simplicial depth of Liu (1990), which was later applied to linear regression by Rousseeuw and Hubert (1999) and to generalized linear regression by Müller (2005). Rousseeuw and Hubert (1999) already noticed for the case of simple linear regression that the simplicial depth can also be obtained by counting the number of 3-tuples in the residuals which have alternating signs. This was generalized and proven for linear and non-linear regression and autoregression by Kustosz et al. (2016a). They have shown that for calculating the simplicial depth it is sufficient to count the number of alternating $(K+1)$-tuples in the residuals when having a K-dimensional parameter vector in the regression model. Leckey et al. (2020) defined this concept more generally as *sign depth* or *K-sign depth*. This K-sign depth can easily be used for testing on parameters in regression: If there are too few K-tuples with alternating signs in the residuals the parameter is not fitting the data. So far, this so-called *sign depth test* or *K-sign depth test* could only be applied in cases with an inherent order of the residuals, for example simple regression or time-series because the sign depth heavily depends on the order of the residuals. In situations without inherent order, for example multiple regression, the sign depth test was not applicable. This thesis will solve this problem. For this, suitable methods for ordering multidimensional data have to be found and it quickly became clear that the solution to this problem will not be found in the field of statistics, but in computer science. Ordering the data according to a shortest path through all data points leads to a good power of the sign depth test in all considered situations. Finding such a path is a transformation of the Traveling Salesman Problem which is well-known in computer science. As this thesis will show, this approach with origins in computer science has many advantages and is the most promising one compared with all other approaches which are presented in this thesis. So, this thesis will solve a problem of the sign depth test in many applications. This would not have been possible when not thinking out of the box and finding the solution for this statistical problem in the field of computer science.

1.1 Structure of the Thesis

This thesis is divided in six chapters. After this introductory chapter, in Chapter 2 robust estimators and tests for multiple linear regression are described. For this, at first an introduction of the notation in this thesis and for multiple linear regression itself is given. Afterwards, classical robust estimators and tests in the context of linear regression are described. This includes for example estimators and tests based on M- and MM-regression. In Section 2.4 the concept of using data depths for robust estimations and regressions is presented. This leads to the definition of the sign depth and the sign depth test in Section 2.5. All presented approaches for estimating and testing parameters of regression models are visualized and compared in the context of simple regression in Section 2.6. At the end of Chapter 2 the problem of the sign depth and the sign depth test in the context of multiple regression is described and the aim of this thesis is given.

In Chapter 3, 13 different ordering methods for multidimensional data are described and compared. These 13 methods can be assigned to four different ways of ordering: Naive ordering methods, scalarization based ordering methods, ordering methods based on partial sorting and ordering methods based on the pairwise distances of the regression vectors. All approaches have different advantages and disadvantages which will be shown in this chapter. Furthermore, it is looked at the theoretical time complexity and the empirical runtimes of these methods for ordering different numbers of multidimensional values in different numbers of dimensions.

Chapter 4 deals with the software which was developed as part of this thesis. For this software package the sign depth, the sign depth test and all ordering methods had to be implemented. As it can be read in Section 4.1, implementing the sign depth can be challenging. For this, it has to be paid attention to the computational runtimes of the implementations because calculating the sign depth has a large time complexity when using simple ways of implementation. Also, implementing the different ordering methods is challenging for some of them, see Section 4.3.

Afterwards, Chapter 5 shows, describes and analyzes the performance of the different ordering methods when applying them to the sign depth test in different situations. For this, power functions of the sign depth test are simulated. The simulations are described in Section 5.1. Afterwards, in Section 5.2 the simulated power functions are compared for all of the different ordering methods. For this, a rather simple

multiple regression model is chosen, for which simulated power functions are shown in several situations: The effects of the number of data points, of the underlying data set, of the error distribution in the model and of hyper-parameters of the sign depth test and the ordering methods are analyzed. This section will show that only the distance based ordering methods lead to completely satisfying power functions of the sign depth test. Because of this, in Section 5.3 only these methods are considered. In this section, the power of the sign depth test is analyzed for some more multiple regression models, for example models with interactions or with non-linear regressors. Afterwards, the power of the sign depth test is compared to three classical tests for parameters of multiple regression models and the performance of the sign depth test in the context of model checks is shown. At the end of this chapter, the sign depth test is applied on real data of a bridge monitoring.

All results obtained in Chapters 2 to 5 will be summarized in Chapter 6. Furthermore, an outlook to open questions and unsolved problems regarding the sign depth test is given.

1.2 Used Computational Tools

The simulations in this thesis were performed on the High Performance Cluster *LiDO3*[1] (Linux Cluster Dortmund, 3. Generation). LiDO3 consists of 366 nodes with overall 8 160 CPU cores which have a performance of up to 2.4 GHz and at least 64 GB RAM per node.

The simulations were executed in `R` (R Core Team, 2019) with the help of the package `batchtools` (Lang et al., 2017). Runtime measurements were performed with the package `microbenchmark` (Mersmann, 2019). Graphics were mostly made with `ggplot2` (Wickham, 2016) and `tikzDevice` (Sharpsteen and Bracken, 2019). Furthermore, the packages `BBmisc` (Bischl et al., 2017), `cowplot` (Wilke, 2019), `gridExtra` (Auguie, 2017), `rgl` (Adler and Murdoch, 2020), `scales` (Wickham and Seidel, 2019) and `xtable` (Dahl et al., 2019) were used.

[1] `https://www.lido.tu-dortmund.de/cms/en/home/index.html`

Chapter 2

Estimators and Tests for (Robust) Regression

This chapter describes tools for estimating and testing parameters in multiple linear regression models, especially in the context of robust regression, i.e. when having outliers in the design matrix and/or in the response vector of a linear model. After describing linear models in general, several approaches for robust linear models are presented. When a (robust) linear model is fitted to some data, usually the goodness of the fitted model is of interest. Therefore, classical tests on the parameter vector of the model like the F-test, the Wald test or the sign test can be used. In this thesis, more precisely in Section 2.5, the so-called *sign depth test* for multiple regression is presented. The sign depth test is a robust test on the parameter vector of a regression model which could be used so far only for simple regression models with only one regressor or models which have an inherent order in the data like time-series. In this thesis the sign depth test will be extended to multiple regression.

2.1 General Notations and Definitions

Before describing linear models in the next section, the general notation in this thesis is given. Here, the following notation is used: Normal written letters describe scalar values, where letters in lower case in most cases are indices or components of a vector. Lower case bold letters describe vectors and upper case bold letters describe matrices

Letter / Symbol	Meaning
$D \in \mathbb{N}$	Number of dimensions in a data set
$K \in \mathbb{N}$ with $K \geq 2$	Parameter of the sign depth
$N \in \mathbb{N}$	Number of data points in a data set
$\boldsymbol{e} = (e_1, \ldots, e_N)^\top \in \mathbb{R}^N$	Vector of errors in a regression model
$\boldsymbol{x}_n = (x_{n1}, \ldots, x_{nD})^\top \in \mathbb{R}^D$	A single regression vector
$\boldsymbol{x}_{\bullet d} = (x_{1d}, \ldots, x_{Nd})^\top \in \mathbb{R}^N$	A single regressor vector
$\boldsymbol{y} = (y_1, \ldots, y_N)^\top \in \mathbb{R}^N$	Response vector in a regression model
$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_D)^\top \in \mathbb{R}^D$	Parameter vector of a regression model
$\hat{\boldsymbol{\theta}} = (\hat{\theta}_1, \ldots, \hat{\theta}_D)^\top \in \mathbb{R}^D$	Estimated parameter vector
$\boldsymbol{X} = (\boldsymbol{x}_{\bullet 1}, \ldots, \boldsymbol{x}_{\bullet D}) \in \mathbb{R}^{N \times D}$	Design matrix in a regression model
$\boldsymbol{z}_n = (y_n, \boldsymbol{x}_n^\top)^\top \in \mathbb{R}^{D+1}$	Regression vector with its response
$\boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)^\top \in \mathbb{R}^{N \times (D+1)}$	Complete design matrix and responses
$\hat{\boldsymbol{e}}(\boldsymbol{\theta}) = (\hat{e}_1(\boldsymbol{\theta}), \ldots, \hat{e}_N(\boldsymbol{\theta}))^\top \in \mathbb{R}^N$	Vector of residuals in a regression model
$\mathbb{1} : \mathbb{R} \to \{0, 1\}$	Indicator function
$\varphi : \mathbb{R}^N \to \{0, 1\}$	A statistical test
$\gamma : \mathbb{R}^D \to [0, 1]$	The power function of a statistical test
$\alpha \in (0, 1)$	Level of a statistical test
$\boldsymbol{\Theta}_0 \subset \mathbb{R}^D$	Set of parameters in H_0
$\boldsymbol{\theta}_0 \in \mathbb{R}^D$	Single parameter vector in H_0
$\mathcal{O}(\cdot)$	Big-O-Notation
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution
$Cau(s, t)$	Cauchy distribution
$\mathcal{U}(a, b)$	Uniform distribution
$Bin(N, p)$	Binomial distribution
$F_{A,B}$	F-distribution
χ^2_A	χ^2-distribution

Table 2.1: Letters and symbols with a fixed meaning in this thesis.

and (data) sets. Some letters and symbols have a fixed meaning in the whole thesis. These letters and symbols are described in Table 2.1.

In the context of programming, software and its packages are written in teletype font. For indicating functions the function name is followed by round brackets. The

theoretical time complexity of algorithms is denoted with the *Big-O-Notation*, which is defined in Definition 2.1 below.

Definition 2.1. *Let f and g be real-valued functions and $x \in \mathbb{R}$. Then the following set can be defined (see Knuth (1976)):*

$$f \in \mathcal{O}(g) \Leftrightarrow \exists C > 0 \, \exists x_0 > 0 \, \forall x > x_0 : |f(x)| \leq C \cdot |g(x)|.$$

2.2 Regression Models

At first, a definition of a linear regression model is given:

Definition 2.2. *Let N, $D \in \mathbb{N}$ and*

- $\boldsymbol{x}_n = (x_{n1}, \ldots, x_{nD})^\top \in \mathbb{R}^D$ *a regression vector,*
- $\boldsymbol{y} = (y_1, \ldots, y_N)^\top \in \mathbb{R}^N$ *a response vector,*
- $\boldsymbol{e} = (e_1, \ldots, e_N)^\top \in \mathbb{R}^N$ *an error vector,*
- $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_D)^\top \in \mathbb{R}^D$ *a parameter vector,*
- $\boldsymbol{X} = (\boldsymbol{x}_{\bullet 1}, \ldots, \boldsymbol{x}_{\bullet D}) \in \mathbb{R}^{N \times D}$ *with* $\boldsymbol{x}_{\bullet d} = (x_{1d}, \ldots, x_{Nd})^\top \in \mathbb{R}^N$ *a design matrix.*

Then, a multiple linear regression model is given by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{e},$$

where the errors $e_1, \ldots, e_N$ are realizations of stochastically independent random variables $E_1, \ldots, E_N$.

When defining $\boldsymbol{z}_n := (y_n, \boldsymbol{x}_n^\top)^\top \in \mathbb{R}^{D+1}$ and $\boldsymbol{Z} := (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N)^\top$, the residual relating to $\boldsymbol{z}_n$ and a parameter vector $\boldsymbol{\theta}$ is given by $\hat{e}_n(\boldsymbol{\theta}) := y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta}$.

If the model includes an intercept θ_0, $\boldsymbol{\theta}$ is given by $(\theta_0, \theta_1, \ldots, \theta_{D-1})^\top$ and the design matrix has components $(\mathbf{1}, \boldsymbol{x}_{\bullet 1}, \ldots, \boldsymbol{x}_{\bullet D-1})$. Furthermore, components in the design matrix may also be transformations of other components, i.e. the design matrix would be $\boldsymbol{X} = (\boldsymbol{x}_{\bullet 1}, \ldots, \boldsymbol{x}_{\bullet R}, g_1(\boldsymbol{x}_{\bullet 1}, \ldots, \boldsymbol{x}_{\bullet R}), \ldots, g_{D-R}(\boldsymbol{x}_{\bullet 1}, \ldots, \boldsymbol{x}_{\bullet R}))$ with $R \in \mathbb{N}$, $R < D$ and functions $g_1, \ldots, g_{D-R}$.

Transformation functions can be for example functions to generate interactions or non-linear components, like quadratic regressors. Both will be appear in this thesis, especially in Subsections 5.3.2 and 5.3.3.

Ordinary *Least-Squares-Regression* assumes that the errors follow a normal distribution with expected value zero, i.e. $E_n \sim \mathcal{N}(0,\, \sigma^2)$ with $\sigma^2 > 0$. In this case, the least-squares estimator $\hat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$ is given by $\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$ if $rank(\boldsymbol{X}) = D$. For the classical scenario of least-squares-regression, well-known tests exist for testing the parameter vector of the regression model. When using the normal distribution of the errors, testing the hypothesis $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ vs. $H_1 : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0$ for a fixed vector $\boldsymbol{\theta}_0$ is done with the F-test, see Theorem A.1 on page 213. A generalization of the F-test is the Wald test, see Theorem 2.1 on page 14. In the context of non-parametric tests, the classical sign test can be used for this testing problem, see Theorem A.2 on page 213.

When having outliers in the data, the normal distribution assumption of the errors is violated. In the field of robust statistics, several approaches of modeling the errors exist. One idea is to use a more heavy-tailed distribution for modeling the errors, especially a t-distribution with only a few degrees of freedom seems to be a meaningful choice, see for example Lange et al. (1989) and Gelman et al. (2003). Another parametric approach is to use a "contaminated normal distribution", i.e. $E_n \sim (1-\epsilon)\mathcal{N}(0,\, \sigma^2) + \epsilon\mathcal{N}(0,\, c\sigma^2)$, where $c > 1$ and $\epsilon \sim Bin(1,\, p)$, normally with a rather small value for p. Also, non-parametric approaches for robust regression exist. Popular methods are regression on the basis of M-estimators, S-estimators, MM-estimators or LTS-estimators. Some of these approaches are described in more detail in the next section.

2.3 Classical Robust Estimators and Tests in the Context of Linear Regression

In ordinary least-squares-regression, the estimator $\hat{\boldsymbol{\theta}}$ is the solution of the optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^D}{\arg\min} \sum_{n=1}^{N} (y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 = \underset{\boldsymbol{\theta} \in \mathbb{R}^D}{\arg\min} \sum_{n=1}^{N} \hat{e}_n(\boldsymbol{\theta})^2. \tag{2.1}$$

The solution of this optimization problem 2.1 can be heavily affected by only a single outlier because the impact of a single large squared residual can be arbitrarily large. Because of this, many robust regression approaches use different underlying optimization problems for their models.

The oldest approach was already used in basic form by Galileo in the 17th century: the so-called L_1-estimation, see for example Dodge (2008). Instead of minimizing the sum of the squared residuals, this approach minimizes the sum of the absolute residuals

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^D}{\arg\min} \sum_{n=1}^{N} |y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta}| = \underset{\boldsymbol{\theta}\in\mathbb{R}^D}{\arg\min} \sum_{n=1}^{N} |\hat{e}_n(\boldsymbol{\theta})|. \tag{2.2}$$

Although this approach is intuitive and more robust than the ordinary least-squares method, very large outliers can still have a huge effect on the optimization problem. Also, the solution of the optimization problem 2.2 cannot be given as a explicit formula, but has to be derived iteratively with specialized algorithms.

Another approach is called *least-trimmed-squares-regression* (LTS-regression) and is described for example in Rousseeuw and Leroy (1987, p. 15). When denoting $\hat{e}^2_{(n)}(\boldsymbol{\theta})$ as the nth smallest squared residual, the optimization problem of the ordinary least-squares-regression can be written as $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^D} \sum_{n=1}^{N} \hat{e}^2_{(n)}(\boldsymbol{\theta})$. For LTS-regression, a hyper-parameter $L < N$ is needed which describes how many residuals should be taken into account for the optimization. In fact, only the L smallest residuals matter for the optimization, whereas the $N-L$ larger residuals are neglected. Consequently, the optimization problem of LTS-regression is

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^D}{\arg\min} \sum_{n=1}^{L} \hat{e}^2_{(n)}(\boldsymbol{\theta}). \tag{2.3}$$

Since it depends on the fitted model which residuals are the largest, also for this robust regression method no simple solution of the optimization problem 2.3 can be given, but the solution has to be computed iteratively with specialized algorithms, for example the *Fast LTS algorithm* proposed by Rousseeuw and Driessen (1999). Furthermore, the choice of L affects the solution of the optimization problem, which means that it should be clear how many outliers are in the data before applying this method to the data.

Furthermore, Huber (1964) proposed the so-called *M-estimators* as robust location estimators, which can be also used for robust regression. An M-estimator for regression models can be derived by the solution of the optimization problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^D}{\arg\min} \sum_{n=1}^{N} \rho(\hat{e}_n(\boldsymbol{\theta})), \tag{2.4}$$

where ρ is a symmetric function (i.e. $\rho(x) = \rho(-x)$ for all $x \in \mathbb{R}$) with an unique minimum at zero (Rousseeuw and Leroy, 1987, p. 12). If ρ is differentiable with

$\psi := \rho'$, the optimization problem 2.4 can be written as finding a vector $\boldsymbol{\theta}$ for which

$$\sum_{n=1}^{N} \psi(\hat{e}_n(\boldsymbol{\theta}))\boldsymbol{x}_n = \mathbf{0} \tag{2.5}$$

holds. In the literature, many different choices for ρ and ψ are proposed, especially the ordinary least-squares-regression and the L_1-regression are special cases of M-estimation. Often used ψ-functions for robust regression are for example the Huber function (Huber, 1964)

$$\psi(x) = \begin{cases} x, & |x| \leq k \\ k\,\mathrm{sign}(x), & |x| > k \end{cases}, \quad k > 0, \tag{2.6}$$

Tukey's bisquare function (Beaton and Tukey, 1974)

$$\psi(x) = \begin{cases} x\left(1 - \left(\frac{x}{k}\right)^2\right)^2, & |x| \leq k \\ 0, & |x| > k \end{cases}, \quad k > 0, \tag{2.7}$$

the Hampel function (Hampel et al., 1986, p. 150)

$$\psi(x) = \begin{cases} x, & |x| \leq a \\ a\,\mathrm{sign}(x), & a < |x| \leq b \\ a\,\mathrm{sign}(x)\frac{c-|x|}{c-b}, & b < |x| \leq c \\ 0, & c < |x| \end{cases}, \quad 0 < a < b < c, \tag{2.8}$$

an "optimal" function by Yohai and Zamar (1998)

$$\psi(x) = \begin{cases} x, & \left|\frac{x}{c}\right| \leq 2 \\ c\left(-1.944\frac{x}{c} + 1.728\left(\frac{x}{c}\right)^3\right. & 2 < \left|\frac{x}{c}\right| \leq 3 \\ \quad \left. - 0.312\left(\frac{x}{c}\right)^5 + 0.016\left(\frac{x}{c}\right)^7\right), & \\ 0, & \left|\frac{x}{c}\right| > 3 \end{cases}, \quad c > 0 \tag{2.9}$$

and an "optimal" function by Maronna et al. (2006, p. 145)

$$\psi(x) = \mathrm{sign}(x)\max\left\{0, -\frac{\phi'(|x|) + c}{\phi(|x|)}\right\}, \quad c \geq 0, \tag{2.10}$$

where ϕ describes the standard-normal density function. Since the above mentioned equation 2.5 is not easily solvable for most choices of ψ, specialized algorithms have to be used, for example Newton-Raphson algorithms or Iteratively Re-Weighted

Least Squares algorithms. The found solution of these algorithms can heavily depend on the starting point of the algorithm, so that a "good" initial estimator of $\boldsymbol{\theta}$ should be used as a starting point. Furthermore, the solution of M-estimators of course depends on the chosen ψ-function and its hyper-parameters, although for all above mentioned functions meaningful choices for the hyper-parameters exist in the respective literature. Normally, the hyper-parameters are chosen so that a specific efficiency and/or breakdown point is achieved.

While M-estimators reach good performance when having outliers in the response variable of a linear model, they perform poorly in case of leverage points (i.e. outliers in the regressors), see for example Rousseeuw and Leroy (1987, p. 13). Because of this, some other types of estimators, like *S-estimators*, were invented. S-estimators (Rousseeuw and Yohai, 1984) estimate the parameter vector $\boldsymbol{\theta}$ by minimizing the dispersion of the residuals. Formally written,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^D}{\arg\min}\, s(\hat{\boldsymbol{e}}(\boldsymbol{\theta})), \tag{2.11}$$

where $s(\hat{\boldsymbol{e}}(\boldsymbol{\theta}))$ is the solution of

$$\frac{1}{N} \sum_{n=1}^{N} \rho \left(\frac{\hat{e}_n(\boldsymbol{\theta})}{s(\hat{\boldsymbol{e}}(\boldsymbol{\theta}))} \right) = E_\Phi(\rho). \tag{2.12}$$

Here, $E_\Phi(\rho)$ describes the expected value of the function ρ in case of standard-normal inputs. For this, ρ must be symmetric and continuously differentiable with $\rho(0) = 0$ and there must be a $c > 0$ for which ρ is strictly increasing on $[0,\, c]$ and constant on $[c,\, \infty)$. Remembering that $\psi = \rho'$ holds, the requirements for ψ are that ψ is positive on $[0,\, c)$ and zero on $[c,\, \infty)$ with $\psi(-x) = -\psi(x)$ for all $x \in \mathbb{R}$ (so-called *redescending ψ-functions*). While above mentioned Tukey's bisquare function, the Hampel function and the optimal functions fulfill these requirements, for example the Huber function is not possible in this case.

Another extension of the M-estimators are the MM-estimators invented by Yohai (1987). MM-estimators are defined in three steps. At first, an initial estimator $\hat{\boldsymbol{\theta}}^*$ with high breakdown point (i.e. a high number of outliers is needed before the estimator gets arbitrarily affected) is computed. For this, often an LTS-estimator is used. In the second step, an S-estimator $s_{\hat{\boldsymbol{\theta}}^*}$ on the obtained residuals of the initial estimation is computed. Finally, in the third step, the MM-estimator is defined as

any solution of

$$\sum_{n=1}^{N} \psi\left(\frac{\hat{e}_n(\boldsymbol{\theta})}{s_{\hat{\boldsymbol{\theta}}^*}}\right) \boldsymbol{x}_n = \mathbf{0} \tag{2.13}$$

which fulfills $S(\boldsymbol{\theta}) \leq S(\hat{\boldsymbol{\theta}}^*)$ where $S(\boldsymbol{\theta})$ is defined via

$$S(\boldsymbol{\theta}) := \sum_{n=1}^{N} \rho\left(\frac{\hat{e}_n(\boldsymbol{\theta})}{s_{\hat{\boldsymbol{\theta}}^*}}\right). \tag{2.14}$$

For this, the function ρ and its derivative ψ must fulfill the same conditions as for S-estimations, which are mentioned above.

All above described methods are implemented in different `R`-packages for practical usage. The ordinary least-squares-regression can be found in the function `lm()` in the package `stats` which is part of `R` and automatically loaded when starting `R`. The robust methods can mostly be found in the package `robustbase` (Maechler et al., 2019a). In this package, the central function for robust regression is `lmrob()`. By default, it computes an MM-estimator with settings described in Koller and Stahel (2011). Further functions are `lmrob.lar()` for L_1-regression, `lmrob..M..fit()` for M-estimation, `lmrob.S()` for S-estimation and `ltsReg()` for LTS-regression. Another implementation of MM-estimation for robust regression can be found in the package `robust` (Wang et al., 2019). Its function `lmRob()` in default uses different settings than the function `lmrob()` in `robustbase`, especially the choice of the ψ-function is different: While `lmrob()` uses the so-called "linear-quadratic-quadratic" ψ-function described in Koller and Stahel (2011), `lmRob()` uses the above mentioned "optimal" ψ-function of Yohai and Zamar (1998). An example of the different regression methods is given in Section 2.6.

In the literature, there exist also some classical robust tests on parameters of linear regression. A good overview can be found in Chapter 7 of Hampel et al. (1986), where for example the so-called τ-test and R_n^2-test are described. These tests are mostly extensions of likelihood ratio tests which robustify classical tests. The classical Wald test can also be used directly for asymptotically normally distributed estimators, like M-estimators. For this, only a robust estimation of the covariance matrix of the estimated parameter vector is needed. In the literature, there exist several approaches for estimating this covariance matrix, see for example Rousseeuw and Leroy (1987, p. 130) or Maronna et al. (2006, p. 139). Since in Chapter 5 a Wald test based on the estimators provided by the `R`-function `lmRob()` of the package `robust`

is used as a comparison method, in the following its estimation of the covariance matrix is described in more detail. A complete description of the procedure used by `lmRob()` can be found in Yohai et al. (1991) and TIBCO Software Inc. (2010). As mentioned above, `lmRob()` estimates the parameter of a linear regression model by MM-estimation, for which the optimal ψ-function of Yohai and Zamar (1998) is used by default. In the following, let $\tilde{\boldsymbol{\theta}}$ be the estimator obtained in the second step of the MM-estimation process (i.e. the S-estimation step) and $\tilde{\sigma}$ its estimator of the dispersion, i.e.

$$\tilde{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^D}{\arg\min}\, s(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) \tag{2.15}$$

and

$$\tilde{\sigma} = \min_{\boldsymbol{\theta} \in \mathbb{R}^D} s(\hat{\boldsymbol{e}}(\boldsymbol{\theta})). \tag{2.16}$$

Let $\tilde{e}_n,\ n = 1, \ldots, N$ be the standardized residuals obtained in this step, i.e.

$$\tilde{e}_n = \frac{y_n - \boldsymbol{x}_n^\top \tilde{\theta}}{\tilde{\sigma}},\ n = 1, \ldots, N \tag{2.17}$$

and

$$w_n = \frac{\psi(\tilde{e}_n)}{\tilde{e}_n},\ n = 1, \ldots, N. \tag{2.18}$$

Then, the estimated covariance matrix of the final parameter vector $\hat{\boldsymbol{\theta}}$ is

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \hat{\boldsymbol{C}}^{-1} \tilde{\sigma}^2 \hat{\tau} \tag{2.19}$$

with

$$\hat{\boldsymbol{C}} = \left(\sum_{n=1}^{N} w_n \right)^{-1} \cdot \sum_{n=1}^{N} w_n \boldsymbol{x}_n \boldsymbol{x}_n^T \tag{2.20}$$

and

$$\hat{\tau} = \left(\frac{1}{N} \sum_{n=1}^{N} \psi'(\tilde{e}_n) \right)^{-2} \cdot \frac{1}{N} \sum_{n=1}^{N} \psi(\tilde{e}_n)^2. \tag{2.21}$$

The estimated parameter vector $\hat{\boldsymbol{\theta}}$ and the estimated covariance matrix $\hat{\boldsymbol{\Sigma}}$ obtained by `lmRob()` can then be used in a classical Wald test, which is described below in Theorem 2.1.

Theorem 2.1. ***The Wald test***
Let $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{e}$ *be a linear model with* $rank(\boldsymbol{X}) = D < N$, $\hat{\boldsymbol{\theta}}$ *an estimator for* $\boldsymbol{\theta}$ *and* $\hat{\boldsymbol{\Sigma}}$ *a consistent estimator for the covariance matrix of* $\hat{\boldsymbol{\theta}}$. *Assume that* $\sqrt{N}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)$ *follows asymptotically a normal distribution with expected value* $\mathbf{0}$ *and covariance matrix* $\hat{\boldsymbol{\Sigma}}$, *i.e.* $\sqrt{N}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0) \to \mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Sigma}})$. *The hypothesis* $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ *vs.* $H_1 : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0$ *can be rejected to the level* $\alpha \in (0, 1)$ *if*

$$(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)^\top (\hat{\boldsymbol{\Sigma}}/N)^{-1} (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0) > \chi^2_{D,1-\alpha},$$

where $\chi^2_{D,1-\alpha}$ *denotes the* $(1-\alpha)$*-quantile of the* χ^2*-distribution with* D *degrees of freedom.*

Remark 2.1.

(a) Since the χ^2*-distribution of the test statistic under* H_0 *only holds asymptotically, the Wald test may not maintain the level* α *for finite sample sizes.*

(b) The Wald test can be used as soon as the assumption $\sqrt{N}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0) \to \mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Sigma}})$ *holds. This is the case for many types of estimators, like for example maximum-likelihood estimators or M-estimators.*

2.4 Robust Estimators Based on Data Depth

Next to the above presented approaches for robust regression, there also exist further non-parametric and semi-parametric approaches. One branch in the field of robust estimation and testing deals with so-called *data depths*. Many different types of data depths exist, but they all have in common that they aim to find an estimator which is "deep" in the data.

The first approach in the context of data depths was presented by Tukey (1975) who proposed the so-called *halfspace depth* for location estimation. A description of the halfspace depth in a bit more detail can be found in Subsection 3.3.3. This halfspace depth has many extensions. Most of them concern multivariate data, see, for example simplicial depth of Liu (1990), zonoid depth treated in Mosler (2002), Mahalanobis depth as used in Hu et al. (2011) or functional data depths as in López-Pintado and Romo (2009), Claeskens et al. (2014) and Agostinelli (2018). Not all of them provide robust methods and most of them concern only estimation, but no testing. Also, there exist not many extensions of the halfspace depth in the context of robust

regression. One approach, called *regression depth*, was presented by Rousseeuw and Hubert (1999). They defined the regression depth as the minimal number of data points that have to be removed from the data set to make a fitted regression model a so-called non-fit. A more formal definition of the regression depth is given in Van Aelst et al. (2002). There, with the notation given in Table 2.1 on page 6, the regression depth is defined via

$$\begin{aligned} d_R(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) := \min_{\boldsymbol{u},v}(\#\{\hat{e}_n(\boldsymbol{\theta}) \geq 0 \text{ and } \boldsymbol{x}_n^\top \boldsymbol{u} < v\} + \\ \#\{\hat{e}_n(\boldsymbol{\theta}) \leq 0 \text{ and } \boldsymbol{x}_n^\top \boldsymbol{u} > v\}), \end{aligned} \tag{2.22}$$

where the minimum is over all unit vectors $\boldsymbol{u} \in \mathbb{R}^D$ and all $v \in \mathbb{R}$ with $\boldsymbol{x}_n^\top \boldsymbol{u} \neq v$. Estimating the parameter vector $\boldsymbol{\theta}$ in a regression model can be done by searching the parameter vector which has maximal regression depth, i.e.

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^D}{\arg\max}\, d_R(\hat{\boldsymbol{e}}(\boldsymbol{\theta})). \tag{2.23}$$

In general, this is a hard to solve optimization problem, for which a fast exact algorithm only exists for simple regression. For multiple regression, an approximate algorithm is described in Van Aelst et al. (2002), which is also implemented in the function `rdepthmedian()` in the `R`-package `mrfDepth` (Segaert et al., 2020).

The regression depth is an extension of Tukey's halfspace depth, but also Liu's simplicial depth can be extended to regression. A first approach of this was already given in Rousseeuw and Hubert (1999), but over the last years the research on this topic was pushed ahead, especially by Christine Müller and some of her co-workers, which has recently led to the definition of *sign depth*. Since this thesis is based on the sign depth, in the following this depth is described in detail.

2.5 Estimators and Tests Based on Sign Depth

As Rousseeuw and Hubert (1999) noticed for simple linear regression and Kustosz et al. (2016a) proved in the context of linear and non-linear regression and autoregression, the simplicial depth often reduces to what is called the K-sign depth. Hence, under certain conditions, the K-sign depth, with $K \geq 2$, can be seen as a generalization of the simplicial regression depth introduced by Müller (2005) for generalized linear models. The K-sign depth is defined as follows:

Definition 2.3. *Let $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$ be ordered according to some criteria and $K \in \mathbb{N}$ with $2 \leq K \leq N$. Then the K-*sign depth *d_S^K of a parameter $\boldsymbol{\theta}$ in the data set $\boldsymbol{Z}$ with residuals $\hat{\boldsymbol{e}}(\boldsymbol{\theta}) = (\hat{e}_1(\boldsymbol{\theta}), \ldots, \hat{e}_N(\boldsymbol{\theta}))^\top$ is defined via*

$$
\begin{aligned}
s_{n_k}^{(1)} &:= \mathbb{1}\{\hat{e}_{n_k}(\boldsymbol{\theta}) \cdot (-1)^k > 0\} \\
s_{n_k}^{(2)} &:= \mathbb{1}\{\hat{e}_{n_k}(\boldsymbol{\theta}) \cdot (-1)^k < 0\} \\
s_{n_k}^{(3)} &:= 1 - \mathbb{1}\{\hat{e}_{n_k}(\boldsymbol{\theta}) \neq 0\} \\
d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) &:= \binom{N}{K}^{-1} \sum_{1 \leq n_1 < n_2 < \ldots < n_K \leq N} \left(\prod_{k=1}^{K} s_{n_k}^{(1)} + \prod_{k=1}^{K} s_{n_k}^{(2)} + \prod_{k=1}^{K} s_{n_k}^{(3)} \right),
\end{aligned}
$$

where $\mathbb{1}\{\ldots\}$ denotes the indicator function, which is one if the condition in the curly braces holds and zero otherwise.

Remark 2.2. *In the case that $P(E_n = 0) = 0 \ \forall n = 1, \ldots, N$ holds, the K-sign depth is almost surely equal to*

$$
d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) = \binom{N}{K}^{-1} \sum_{1 \leq n_1 < n_2 < \cdots < n_K \leq N} \left(\prod_{k=1}^{K} s_{n_k}^{(1)} + \prod_{k=1}^{K} s_{n_k}^{(2)} \right),
$$

which corresponds to the definition given in Leckey et al. (2020).

In short, the K-sign depth is the relative number of ordered K-tuples with alternating signs of the residuals. It can be used as soon as a given ordering of the residuals is available. An example of calculating the 3-sign depth can be found in Figure 2.1.

In nearly all parts of this thesis, multiple linear regression is used with the condition $P(E_n > 0) = P(E_n < 0) = 0.5$, which means that no residuals with value zero should occur, but the median of the errors is zero. Especially, this condition means that no normal distribution assumption on the errors is needed when applying the sign depth since the sign depth only deals with the signs of the residuals but not with the values itself.

In simple regression, the data has an inherent order according to the values of the only regressor. So, the K-sign depth can easily be applied. Like the regression depth of Rousseeuw and Hubert, the sign depth can be used for estimating the parameter vector $\boldsymbol{\theta}$ of a regression model via searching the parameter vector which leads to the maximal depth, i.e. $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta} \in \mathbb{R}^D} d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta}))$. So far, no specialized algorithm for computing the parameter vector which leads to the maximal sign depth has been

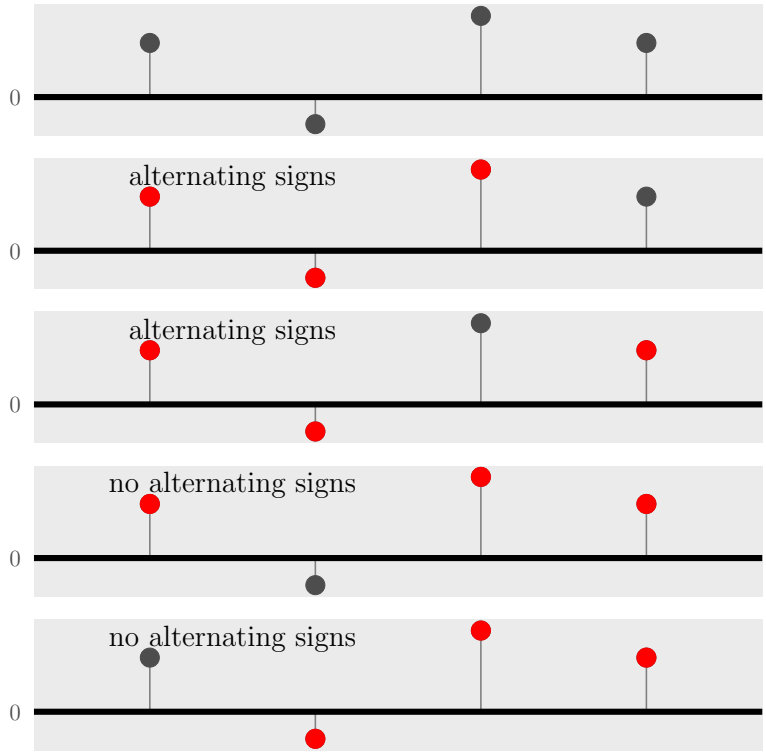


Figure 2.1: Example of calculating the K-sign depth with parameter $K = 3$. The topmost figure shows four residuals of an arbitrary model, the lower figures show the $\binom{4}{3} = 4$ combinations of 3-tuples of these residuals. Here, two out of four 3-tuples have alternating signs, so the 3-sign depth is 0.5.

developed. However, a naive approximate algorithm with large time complexity (i.e. $\mathcal{O}(\binom{N}{D})$) is available for linear models. This algorithm fits a hyperplane through all $\binom{N}{D}$ possibilities of choosing D regression vectors from all N regression vectors, calculates the sign depth of the residuals and takes the parameter vector which has led to the maximal sign depth. A formal description of this procedure for simple regression $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \boldsymbol{e}$ can be found in Algorithm A.1 on page 214.

In the last years, a lot of research has been done in the field of sign depths. The sign depth can be seen as a generalization of the simplicial regression depth for generalized linear models proposed already by Müller (2005). A few years ago, Kustosz et al. (2016b) showed that the values of the standardized version $N \cdot (d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) - (\frac{1}{2})^{K-1})$ of the distribution of the K-sign depth converges for $K = 3$ to an asymptotic distribution of which for example quantiles can be generated of. Kustosz et al. (2016b) also dealt with some simplified versions of the K-sign depth which are easier to compute and converge to the standard normal distribution.

Recently, Malcherczyk (2021+) found some efficient ways to compute the K-sign depth. Furthermore, a test on parameters of a regression model based on the sign depth has been developed, see below. But one problem has remained all the years: The sign depth and tests based on it could only be used when there is only one regressor in the regression model because the sign depth needs ordered data according to the values of the regressor. When there is more than one regressor, there is no inherent order in the data anymore and the values of the K-sign depth may change drastically when ordering the values according to different criteria. This problem will be discussed in more detail in Section 2.7.

By now, the K-sign depth has only been used for estimation of regression parameters. But, it can also easily be used for testing on the parameters. In the following, this will be explained in detail.

In the context of robust multiple regression, there exist already some tests on the parameter vector of a regression model. Widely used is the non-robust F-test (see Theorem A.1 on page 213), which depends on the assumption of normally distributed errors. Also for robust models which estimators are asymptotically normally distributed (like M-estimators) the F-test can be applied. As explained in the previous section, there exist also robust generalizations of the F-test which are based, for example, on M-estimators. Also the Wald test (see Theorem 2.1 on page 14) can be based on M-estimators which robustifies this test clearly. One approach for this was described in the previous section. In the field of non-parametric tests the sign test (see Theorem A.2 on page 213) is often used which is basically a binomial test on the number of positive (or negative) residuals to check whether they occur with probability 0.5 or not.

All three classical tests have some disadvantages: the F-test as well as the Wald test need the normal distribution, the robust Wald test is laborious to compute because of the robust estimation of $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\Sigma}}$ and the sign test has low power in many situations because it only looks at the number of positive residuals, but not on the order of the residuals. Because of this, an obviously non-fitting model could be regarded as a good model by the sign test if it has the same number of positive and negative residuals. The K-sign depth also only needs the signs of the residuals, but in contrast to the test statistic of the sign test also pays attention to the order of the signs. Because of this, the sign depth has great potential to be the basis of a powerful test. In addition, for the sign depth no assumption on the distribution of the errors is needed as long as the median of the errors is zero.

The basis of the K-sign depth test is the following: If the fitted parameter vector of a model is correct, the residuals scatter independently of each other around zero. In contrast, if the fitted parameter vector is not correct, the independence of the signs of the residuals is often violated and the residuals do not scatter independently around zero. This leads to fewer K-tuples with alternating signs and hence to a smaller K-sign depth. When the value of the K-sign depth is smaller than the α-quantile of the distribution of the corresponding K-sign depth, it is shown significantly to the level $\alpha \in (0, 1)$ that the considered model does not fit the data. The K-sign depth test can be written as follows:

Theorem 2.2. ***K-Sign Depth Test***
Let $\alpha \in (0, 1)$ and $K \in \mathbb{N}$ be fixed with $2 \leq K \leq N$, where N is the number of data points in the data set $\boldsymbol{Z}$ and $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$ denotes the vector of residuals when fitting a model to $\boldsymbol{Z}$ with parameter $\boldsymbol{\theta}$. Let $q_\alpha^{K,N}$ denote the α-quantile of the distribution of the K-sign depth for N data points. Then, the test

$$\varphi^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) = \mathbb{1}\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_0} d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) < q_\alpha^{K,N}\right\}$$

is a test to the level α for $H_0 : \boldsymbol{\theta} \in \boldsymbol{\Theta}_0$ vs. $H_1 : \boldsymbol{\theta} \notin \boldsymbol{\Theta}_0$.

In this thesis, only the case of a single valued $\boldsymbol{\Theta}_0$ is considered. In this case the K-sign depth test can be simplified as follows:

Corollary 2.1. ***K-Sign Depth Test for single valued $\boldsymbol{\Theta}_0$***
Let $\alpha \in (0, 1)$ and $K \in \mathbb{N}$ be fixed with $2 \leq K \leq N$, where N is the number of data points in the data set $\boldsymbol{Z}$ and $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$ denotes the vector of residuals when fitting a model to $\boldsymbol{Z}$ with parameter $\boldsymbol{\theta}$. Let $q_\alpha^{K,N}$ denote the α-quantile of the distribution of the K-sign depth for N data points. Then, the test

$$\varphi^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) = \mathbb{1}\left\{d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta}_0)) < q_\alpha^{K,N}\right\}$$

is a test to the level α for $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ vs. $H_1 : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0$.

Hence, the K-sign depth test rejects its null-hypothesis if the K-sign depth of all parameters $\boldsymbol{\theta}$ of the null-hypothesis is too small. For a proof of this simple test strategy, see Müller (2005). The K-sign depth test also is a generalization of the classical sign test since the 2-sign depth test is equivalent to the classical sign test, see Leckey et al. (2020). In addition, Kustosz et al. (2016b), Kustosz et al. (2016a) and Leckey et al. (2020) showed for several examples that the 3-sign depth test and

the 4-sign depth test are much more powerful than the classical sign test and reach the power of the classical non-robust tests based on least-squares of the residuals.

A small drawback of the sign depth test is the need of a quantile whose value depends on the number of data points as well as on the parameter K of the sign depth. Indeed, Kustosz et al. (2016b) showed that the standardized values of the 3-sign depth converge for large N to an asymptotic distribution of which quantiles can be generated of, but for smaller values of N or other values of K this distribution cannot be used. As described in Section 4.2, the quantiles of the distributions of the K-sign depth for smaller values of N have to be simulated.

2.6 Comparison of the Robust Estimators and Tests in the Context of Linear Regression

This section compares the different robust and non-robust estimators and tests on the basis of a given data set. For this, we will look at the Hertzsprung-Russell diagram data of star cluster CYG OB1 given in the data set `starsCYG` in the `R`-package `robustbase` and firstly described in Rousseeuw and Leroy (1987, p. 27). This data set contains the logarithm of the effective temperature at the surface of each star (Te) and the logarithm of its light intensity. Overall, values of 47 stars are in the data set, of which four are so-called "giants", which have a much smaller temperature and a slightly higher light intensity than the other stars and so can be declared as outliers.

At first, the classical (robust) estimators for linear regression described in Section 2.3 are compared. When looking at Figure 2.2, it can be seen that the ordinary least-squares model is heavily affected by the four outliers. Although there is a positive correlation between the temperature and the light intensity of the stars (when neglecting the giants), the slope of the regression line is negative. The same holds for the L_1-regression, since the outliers are quite large and so the L_1-regression is although heavily affected by them. As written in Section 2.3, the M-estimation leads to bad results in case of leverage points. Since the four giants have a much smaller temperature and only a slightly higher light intensity, they can be seen as leverage points and so it is not surprising that the M-estimator also is heavily affected by these points. All other robust estimators are not or not much affected by the outliers.

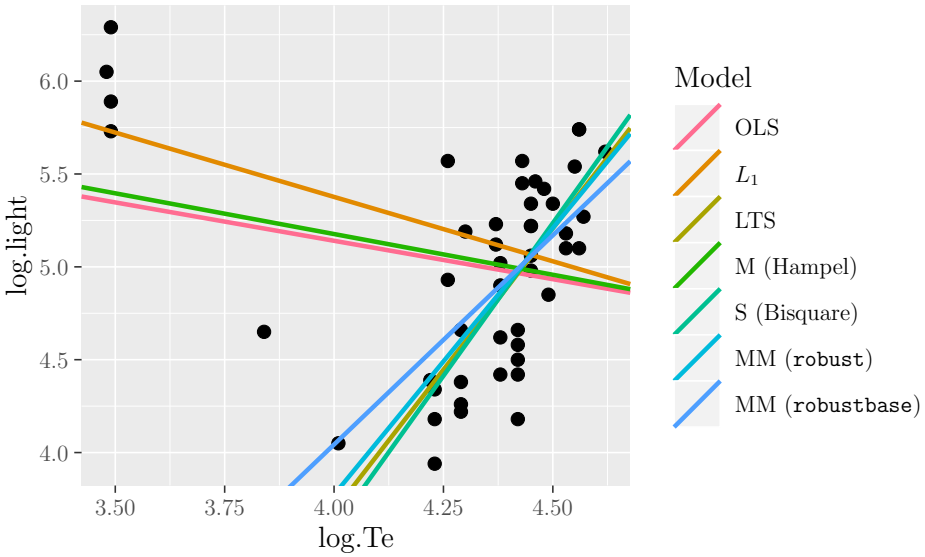


Figure 2.2: (Robust) linear regression models fitted to the Hertzsprung-Russell diagram data of star cluster CYG OB1. Here, OLS means the non-robust ordinary least-squares-regression and L_1 means regression based on least absolute deviations. The hyper-parameters of the other regression models are chosen as their default values given in their respective R-functions. In particular, for LTS $L = 0.5N$ is chosen, the Hampel M-estimator has parameters $a = 1.3521$, $b = 3.1549$, $c = 7.2112$ and the S-estimator is based on Tukey's bisquare function with $k = 4.69$. The MM-estimator in package `robust` is called via `lmRob()` with its default values ("optimal" ψ-function) and the MM-estimator in `robustbase` is called via `lmrob()` with its default values described in Koller and Stahel (2011).

Their slopes are positive and fit the data of the 43 "normal" stars quite well. Indeed, the MM-estimator implemented in the R-package `robustbase` fits a slightly different line than the three other methods, whose results are quite similar. Especially, it can be seen that the hyper-parameters of the regression methods can have a significant effect on the result, since the two MM-estimators come to quite different results.

Next, it is looked at the estimators based on data depth and sign depth described in Sections 2.4 and 2.5. The regression depth of Rousseeuw and Hubert can be easily fit to the Hertzsprung-Russel diagram data, whereas for the sign depth a value of K has to be chosen in advance. This parameter is often set to $K = 3$, especially for simple linear regression, where the 3-sign depth corresponds to Liu's simplicial depth, see Kustosz et al. (2016a). Because of this, in the following the sign depth is computed with parameter $K = 3$.

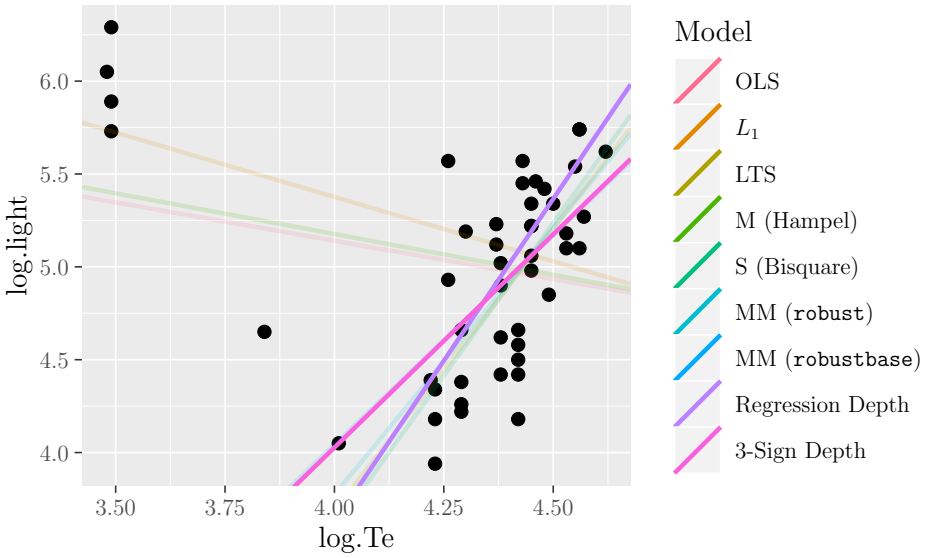


Figure 2.3: (Robust) linear regression models fitted to the Hertzsprung-Russell diagram data of star cluster CYG OB1. The regression depth is fitted with the help of the R-function `rdepthmedian()` from the package `mrfDepth`. The fit on the basis of the 3-sign depth is computed with self-written R-code based on Algorithm A.1 shown on page 214 and the function `calcDepth()` of the package `GSignTest` (Horn, 2021). For information on the other regression models and its hyper-parameters see Figure 2.2.

When using data depths for fitting a regression line to the Hertzsprung-Russel diagram data, it can be seen in Figure 2.3 that the regression depth of Rousseeuw and Hubert as well as the 3-sign depth are robust against the four outlying giants. But in detail, the fits of both models are different. Especially, the fit of the 3-sign depth has a smaller value of the slope than the line fitted by the regression depth and also than the lines of the LTS- and S-estimation and the MM-estimation given by the package `robust`. Indeed, the 3-sign depth fits nearly the same model as the MM-estimation by the package `robustbase`. Overall, both added regression models fit the data satisfyingly.

In the next step, the goodness of the fits of the different regression models can be analyzed. For this, the classical sign test and the sign depth test with parameter $K = 3$ will be used. All nine fitted regression models can be tested with both tests to analyze whether the respective models fit the data well. Table 2.2 shows the values of the regression parameters of the nine models. As already noted before, six slopes are positive, whereas the slopes of the ordinary least-squares regression, the

	Parameter		3-Sign Depth Test		Sign Test	
Model	Intercept	Slope	Statistic	p-value	Statistic	p-value
OLS	6.7935	-0.4133	0.1973	0.0067	27	0.3817
L_1	8.1492	-0.6932	0.2065	0.0138	24	1.0000
LTS	-8.5001	3.0462	0.2566	0.6062	28	0.2430
M (Hampel)	6.9307	-0.4385	0.1973	0.0067	27	0.3817
S (Bisquare)	-9.5708	3.2904	0.2514	0.4147	26	0.5601
MM (`robust`)	-7.7140	2.8717	0.2617	0.8640	26	0.5601
MM (`robustbase`)	-4.9694	2.2532	0.2609	0.8205	25	0.7709
Regression Depth	-10.3161	3.4848	0.2257	0.0613	22	0.7709
3-Sign Depth	-5.1548	2.2955	0.2615	0.8540	26	0.5601

Table 2.2: Table of regression parameters as well as values of the test statistic and p-values of the 3-sign depth test and the classical sign test for nine models which fit the Hertzsprung-Russel diagram data.

L_1-regression and the Hampel-M-estimation are negative because they are affected by the values of the giants and so, do not describe the remaining 43 data points well. Also, in Table 2.2 the test statistic of the classical sign test and the associated p-values are shown. It can be seen that the number of positive residuals (which is the test statistic of the sign test) of all models is about half of the number of observations (which is $N = 47$). All p-values are much larger than the usual level of $\alpha = 0.05$, which means that all models are regarded as well fitting models for the sign test. Furthermore, Table 2.2 shows the test statistics and the p-values of the 3-sign depth test for the nine models. Here, it can be seen that the 3-sign depths (i.e. the test statistic) of the OLS-, L_1- and M-regression are smaller than the depths of the other methods. The largest depth of course has been obtained by the 3-sign depth method, since this method is constructed by finding the parameter vector with the largest 3-sign depth. The p-values of the 3-sign depth tests show that the null-hypothesis H_0: "the fitted parameter vector describes the data well" can be rejected for $\alpha = 0.05$ for the OLS-, L_1- and M-regression. For the other six models, the null-hypothesis is not rejected which corresponds to the visual impression that all of these six models fit the data satisfyingly.

This section has shown that the sign depth and sign depth test work well in the case of simple linear regression. While even some robust regression methods were affected

by four outliers in the used data set, the regression line based on the maximal 3-sign depth has fitted the data quite well, like some of the other robust regression methods have also done. But in contrast to most of the other methods, the sign depth method needs less assumptions on the data and has less hyper-parameters. Furthermore, the sign depth test was able to detect the obviously non-fitting models whereas the classical sign test came to the conclusion that all fitted models describe the data well. But, as mentioned in the previous section, the sign depth and sign depth test have a problem in the case of multiple regression because this method needs ordered data and in multiple regression, often there is no inherent order of the data. This problem is described in detail in the next section.

2.7 Problem of the Sign Depth in the Context of Multiple Regression

In the context of multiple regression, there may occur a problem when applying the sign depth and sign depth test. As stated in Definition 2.3 on page 16, the sign depth needs ordered regression vectors or residuals respectively. In some applications there is an inherent order of the regression vectors although there may be more than one regressor, for example in the context of time-series or sometimes in the field of design of experiments, but often there is no inherent order in the data set. For these situations, a suitable and unique ordering has to be applied to the data because ordering the data on the basis of different ordering methods can have a crucial effect on the sign depth and the sign depth test. In the following, this is described in more detail based on a simple example.

Let $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{e}$ be a linear regression model with $\boldsymbol{x}_n = (1, x_{n1}, x_{n2})^\top$, where $x_{n1}, x_{n2} \in [-1, 1]$, $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2)^\top$ and $E_n \overset{i.i.d}{\sim} \mathcal{N}(0, 0.01)$. The true model has the parameter vector $\boldsymbol{\theta}_{true} = (0, 1, 1)^\top$. Here, $N = 16$ data points of this model are given. To these data points, a regression model with the parameter vector $\hat{\boldsymbol{\theta}} = (-0.05, -0.5, 1)^\top$ is fitted. A visualization of the data points and the fitted model can be found in Figure 2.4. It can be seen that the fitted model produces eight positive residuals and eight negative residuals. A possible method for ordering the regression vectors is to order the vectors according to the values of only one regressor and neglect the values of all others (see Subsection 3.2.3). In this example, this would lead to two possible orders which are shown in Figure 2.5.

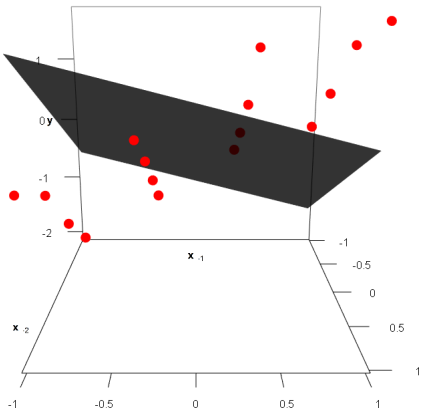
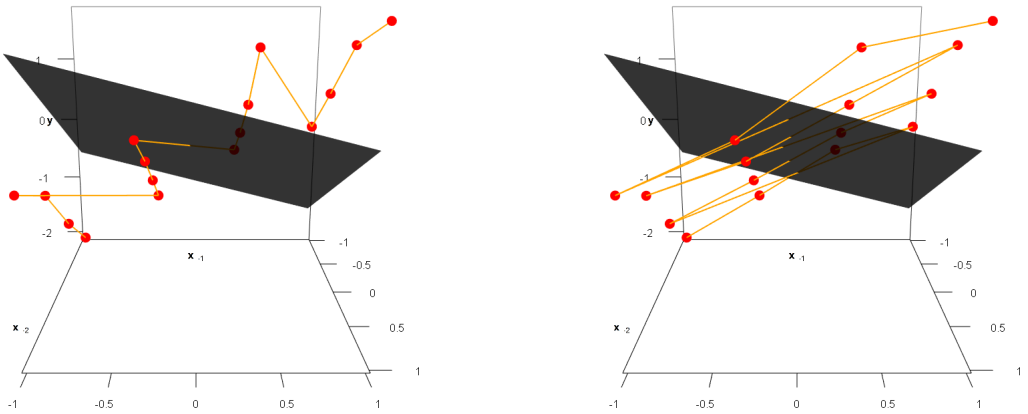


Figure 2.4: Visualization of the 16 data points (red) and the fitted model (dark grey).



(a) Ordering the regression vectors according to the values of $\boldsymbol{x}_{\bullet 1}$.

(b) Ordering the regression vectors according to the values of $\boldsymbol{x}_{\bullet 2}$.

Figure 2.5: Visualization of two possible orders for the 16 data points. The orders are marked as orange lines.

As discussed below, these two orders lead to very different values of the K-sign depth. When ordering according to the values of $\boldsymbol{x}_{\bullet 1}$, only one sign change in the residuals occurs. Ordering according to the values of $\boldsymbol{x}_{\bullet 2}$ leads to seven sign changes. This has crucial effect on all K-sign depths with $K \geq 3$, as it can be seen in Table 2.3. Like the K-sign depth also the K-sign depth test depends on the ordering of the regression vectors in the case of multiple regression. The 16 two-dimensional regression vectors

	$d_S^2(\hat{\boldsymbol{e}}(\hat{\boldsymbol{\theta}}))$	$d_S^3(\hat{\boldsymbol{e}}(\hat{\boldsymbol{\theta}}))$	$d_S^4(\hat{\boldsymbol{e}}(\hat{\boldsymbol{\theta}}))$	$d_S^5(\hat{\boldsymbol{e}}(\hat{\boldsymbol{\theta}}))$
(a)	0.533	0	0	0
(b)	0.533	0.286	0.176	0.088

Table 2.3: K-sign depth with $K \in \{2, 3, 4, 5\}$ for the orders (a) and (b) of Figure 2.5.

	$K = 2$	$K = 3$	$K = 4$	$K = 5$
(a)	1	0	0	0
(b)	1	0.8516	0.9903	0.8375

Table 2.4: p-values of K-sign depth tests with $K \in \{2, 3, 4, 5\}$ for the orders (a) and (b) of Figure 2.5. Values of the corresponding K-sign depths can be found in Table 2.3.

were ordered in two different ways, so that the fitted model with its eight positive and eight negative residuals led to one sign change in the residual vector in case (a) and seven sign changes in the residual vector in case (b). When testing the hypothesis whether the data is sufficiently fitted by the model, the p-values of different K-sign depth tests show the crucial dependance of the test on the ordering. Table 2.4 shows the p-values of the K-sign depth tests for both orderings and $K \in \{2, 3, 4, 5\}$. As Leckey et al. (2020) already noted, the 2-sign depth test is equivalent to the classical sign test which explains why both 2-sign depth tests have an p-value of one independently of the ordering. For $K \geq 3$, the sign depth and so the p-value of the sign depth test of ordering (a) is zero because there is only one sign change in the residual vector and for getting a positive value of the sign depth, at least $K - 1$ sign changes are necessary. Hence, the sign depth test comes independently of K to the conclusion, that the model does not fit the data well which corresponds to the visual impression of the model and the data. On the other hand, ordering (b) has led to seven sign changes within the 16 data points which implies a large value of the K-sign depth and so a large value of the p-value. On the basis of this ordering, the parameters of the obviously non-fitting model cannot be rejected to any appropriate level α.

This example shows how important the order of the multidimensional data is for getting meaningful results of the K-sign depth test. So far, this problem in the context of multiple regression is unsolved. This thesis aims to change this.

2.8 Aim of this Thesis

As in the previous sections described, a lot of research was made in the field of sign depths and sign depth tests in the past years. Many questions in this field are already answered: An asymptotic distribution of the 3-sign depth is known (Kustosz et al., 2016b), efficient algorithms for computing the (asymptotic) sign depth were developed by Dennis Malcherczyk and Kevin Leckey and it was shown that the sign depth test has similar power to classical tests like the F-test in many situations and outperforms the classical sign test nearly always (Leckey et al., 2020).

But, until now, one major drawback exists: The sign depth test can only be used when the regression vectors follow an inherent order, which is, for example, the case for time-series or simple regression, but most times not for multiple regression. Section 2.7 has shown that the order of the regression vectors and residuals has a crucial effect on the result of the sign depth and the sign depth test. This thesis will focus on this aspect of the sign depth. It has the aim to find a way of using the sign depth test for multiple regression, so that the test has the largest possible power.

Facing this goal, several research questions can be formulated which will be answered in this thesis:

1. Which possibilities exist for ordering multidimensional data? For each method, especially the following aspects are of interest:
   (a) Which types of data can be ordered? Only metric data or also ordinal and nominal values?
   (b) Is the result of the ordering affected by linear transformations of the data points?
   (c) Does the method obtain the inherent order of the data if there is one (for example, in the one-dimensional case)?
   (d) What is the time complexity to compute the order and how large are the empirical runtimes?
   (e) Which further advantages and disadvantages do the methods have?
2. How can the sign depth, the sign depth test and the ordering methods be efficiently implemented in `R`? How can these implementations be made usable for other people?

3. Which type(s) of ordering methods produce satisfying simulated power functions? Which ordering method is the best? Especially, the following aspects are of interest:

   (a) Different numbers of data points,

   (b) different values of the parameter K of the sign depth test,

   (c) different data sets,

   (d) different error distributions in the simulated data,

   (e) different hyper-parameters of the ordering methods (if there are some),

   (f) different linear models, especially with different numbers of dimensions.

4. How does the sign depth test perform in the different situations in contrast to the F-test, a robust Wald test and the classical sign test?

5. How does the sign depth test perform on real data?

Chapter 3

Methods for Ordering Multidimensional Data

As described in Section 2.7, the result of the calculation of the sign depth and sign depth test depends on the order of the given values. In the context of multiple regression, the value of the sign depth changes when the regression vectors (and so the residuals) are ordered differently. It can be thought of many possible ways to order the regression vectors of a statistical model. This chapter presents 13 methods of ordering multidimensional data which can be assigned to four different approaches for ordering. At first, two naive methods are described. Afterwards, some possibilities of scalarization of the multidimensional values are presented because the data can be easily ordered according to scalarized values. Another approach is to use partial sorting of the multidimensional data. The last presented approach is to use the pairwise distances of the values for ordering.

In the context of regression, it can be thought about two different ways of ordering: Either only the values of each design vector can be ordered or the complete regression vectors (i.e. including intercept and features which are derived from other features, for example quadratic terms or interactions) can be used for ordering. As an example, let the considered model be $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. In the first case, the values of the vectors $(x_{n1},\, x_{n2})^\top,\ n = 1, \ldots, N$ would be ordered. In the second case, $(1,\, x_{n1},\, x_{n2},\, x_{n1}x_{n2})^\top,\ n = 1, \ldots, N$ would be used for ordering. For this thesis, the second possibility was chosen. Reasons for this choice and a small simulation study on the effects of the different ways of ordering can be found in Section 5.4.

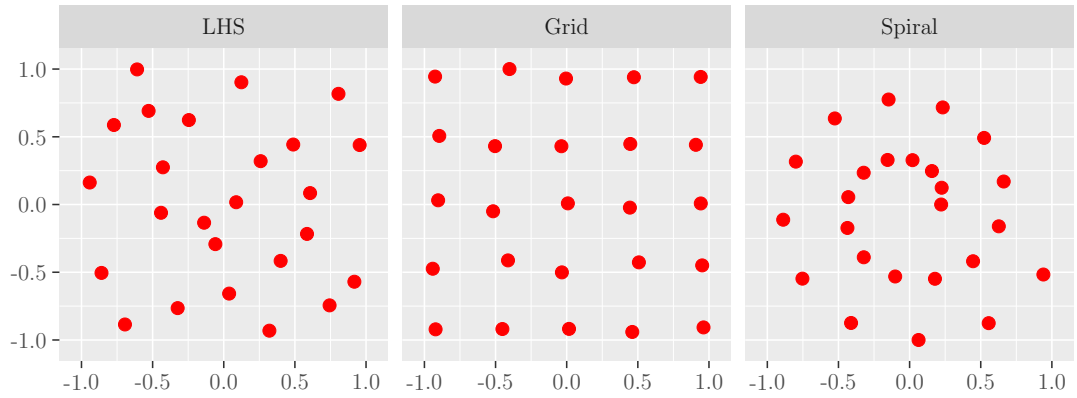


Figure 3.1: Visualization of the three data sets used in this chapter for comparing the presented ordering methods.

In this chapter the 13 ordering methods are described and visualized. For visualizations three different two-dimensional data sets with 25 data points each were chosen. The first one is created via *Latin Hypercube Sampling* (LHS). The so-called *maximin LHS* has the aim to maximize the minimal distance between its values (Stein, 1987). For creation of this data set the `R`-package `lhs` (Carnell, 2019) is used. The second data set consists of equidistant values to which a small noise is added, in this chapter called *Grid-data*. At last, a data set with values arranged as spiral is chosen. This data set is created with the help of the `R`-package `mlbench` (Leisch and Dimitriadou, 2010). While the first two data sets show designs which are often used in the field of *Design of Experiments* and therefore also in the context of (linear) regression, the third data set is chosen for visualization because it has an inherent order and so it can be analyzed which ordering methods preserve this order. Figure 3.1 shows the three described data sets.

Furthermore, in this chapter it is looked at the computational runtimes of the methods, both theoretically as well as empirically. For the theoretical time complexity, the *Big O-Notation* is used, whose definition can be found in Definition 2.1 on page 7. For the empirical runtimes, artificial data sets with different numbers of regression vectors N and dimensions D are used. Every data set contains $N \in \{100, 200, \ldots, 1\,000\}$ uniformly sampled values in $[0, 1]^D$ with $D \in \{1, 2, 3, 4, 10, 20, 50, 100\}$. For each combination of N and D 100 data sets are created and the respective times for ordering these data sets with each method is determined and visualized with the help of boxplots in every subsection.

Many of the approaches which will be presented in this chapter need some sorting or ordering. Hence, the algorithms and runtimes for sorting in `R` are mentioned here. As one can read on the help page of the function `sort()`, `R` uses different sorting algorithms for different situation. For less than $N = 200$ values an insertion sort (Knuth, 1998, pp. 80–83) is used, which has a time complexity of $\mathcal{O}(N^2)$. Otherwise, for numeric vectors of length $N < 2^{31}$ a radix sort (Knuth, 1998, pp. 168–177) is used, which has an asymptotic time complexity of $\mathcal{O}(N)$. If the vector for sorting is an integer of length $N < 100\,000$ (e.g. an index vector for ordering) a modification of the radix sort is used which is faster but also has complexity $\mathcal{O}(N)$. For all used sorting algorithms the sort is stable. This means, that if two or more values are equal, the order of the ties is preserved. As a conclusion, one can say that sorting or ordering values in `R` has an asymptotic time complexity of $\mathcal{O}(N)$.

The next sections and subsections describe 13 different ordering methods in detail. An overview and summary of the 13 ordering methods with its characteristics, advantages, disadvantages and worst case time complexity can be found in Section 3.5 and especially in Table 3.1 on page 82.

3.1 Naive Methods

When thinking about possible ordering methods for multidimensional data, at first it can be thought about simple approaches. If the simple approaches already perform well, one has not to think about more complicated methods. On the other hand, if the naive approaches perform poorly, these methods can be seen as baseline for other ordering methods.

The two presented methods, taking the order the regression vectors appear in the data set and using a random order, do not depend on the values of the regression vectors itself but only on its indices. This has the advantage that these methods can be used for any type of regressors, having only metric values is not necessary. Furthermore, the values in the data set can be transformed in an arbitrary way (additively and/or multiplicatively) without changing the order. But the dependance only on the indices has also the disadvantage that no information from the values are used for ordering and so, at least, these methods do not perform well in the special case of only one-dimensional data because the inherent order one-dimensional data has cannot be used.

3.1.1 Taking the Order of the Data Set

The most naive approach one can think of is using the order the regression vectors appear in the data set. If the regression vectors have an inherent order and are placed in the data set according to this order, this may be useful. In Figure 3.2, a visualization of this method is given. While this method does not provide a reasonable order on the LHS-data, the obtained orders of the two other data sets seem promising, especially the inherent order of the Spiral-data is preserved.

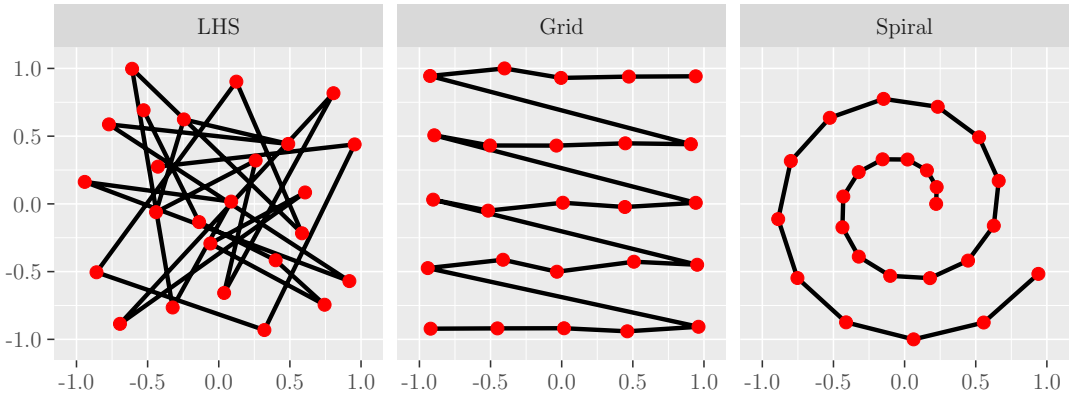


Figure 3.2: Visualization of the obtained orders when using the order the regression vectors appear in the data set.

Since nearly nothing has to be done for applying this method to data sets, the computational time complexity of this approach is obviously $\mathcal{O}(1)$, i.e. the runtime should be independent of the number of regression vectors and dimensions. When looking at Figure 3.3, it can be seen that although the runtimes are very small, there seems to be a small linear trend in the number of regression vectors N as well as in the number of dimensions D. This is no contradiction to the constant runtime in theory. It can be easily explained with the implementation of the ordering methods in `R`. The function `multiSorting()` of the self-written package `GSignTest`, which is used here, always returns the sorted data and the permutation vector of the indices (see Section 4.3) and creating the index vector `1:N` and returning it together with the data set has linear runtime in N and D. But despite of the overhead of the `multiSorting()`-function, it can be seen that the runtimes are very small (only small fractions of seconds). When using this ordering method there occur absolutely no problems because of large runtimes.

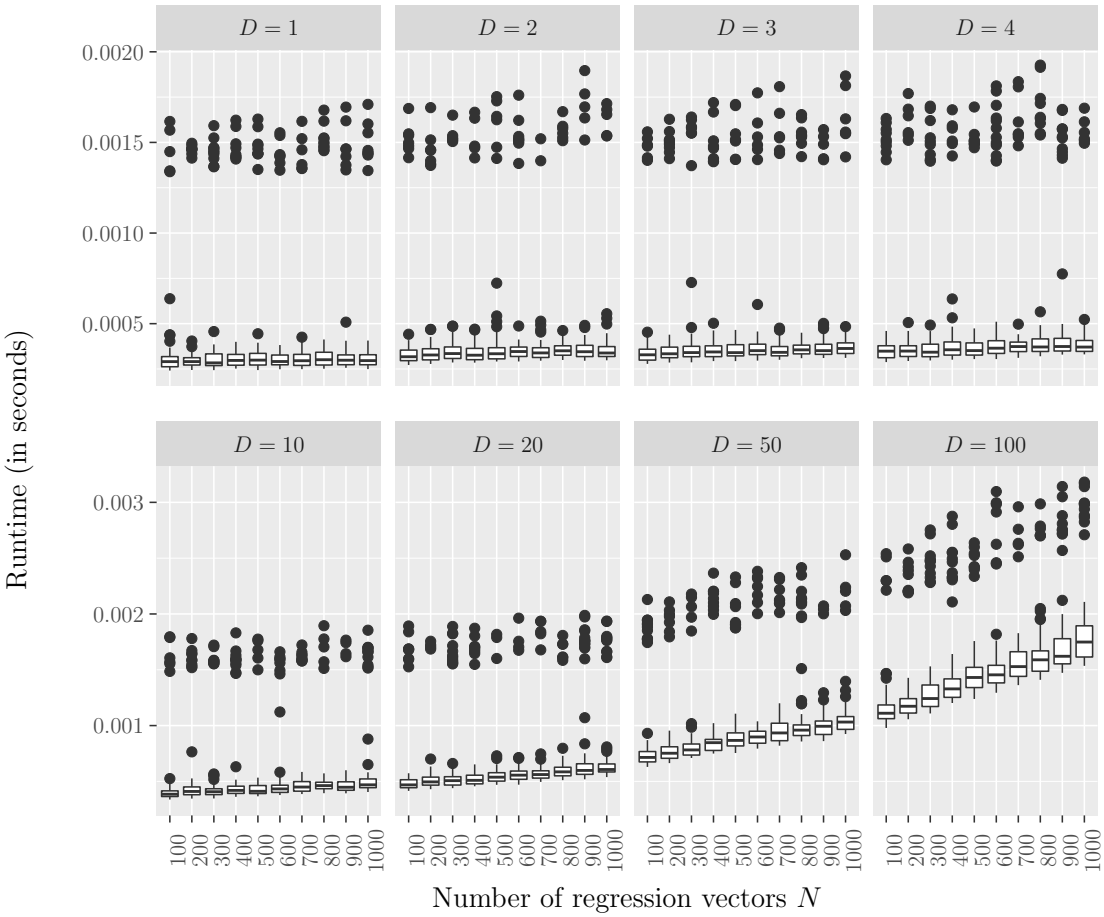


Figure 3.3: Empirical runtimes when ordering the data according to their appearance in the data set.

When looking again at Figure 3.3 it is noticeable that there are several outliers in the runtime data. This can be explained by the used HPC-Cluster LiDO3 for the runtime measurements. The nodes of this HPC-Cluster have different characteristics and different maximal speed. The simulations were distributed on the different nodes randomly, so that some simulation runs have taken slightly longer. This effect can only be seen for such small runtimes as in this subsection. For larger runtimes, this small overhead cannot be seen anymore.

3.1.2 Taking a Random Order

The second naive ordering method is using a random order. This method is very similar to using the order the regression vectors appear in the data set, especially when the regression vectors in the data set are already randomly ordered. An

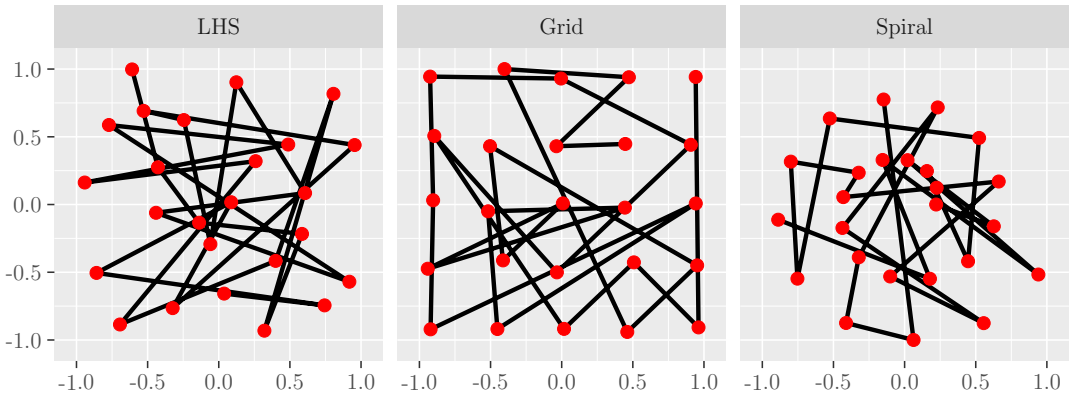


Figure 3.4: Visualization of the obtained orders when using a random order.

advantage of this method in contrast to the first naive method is that possibly unwanted orders in the data set are destroyed. For example, take a look at the Grid-data in Figure 3.4. One can easily think of this data being a design of a planned study. In this case, the order presented in the former subsection in Figure 3.2 may be unwanted because probably there is no reason why the regression vectors are ordered along the first axis and only afterwards along the second axis. But there are also two major disadvantages of the random order in contrast to the order the regression vectors appear in the data set: Firstly, the inherent order of the Spiral-data obviously gets destroyed, what is most probably unwanted. And secondly, generating a random order can lead to different orderings for the same data set when not using a seed in the procedure for generating the random order. Also, even if using a seed, the order may completely change when adding or removing only a single regression vector.

The time complexity for generating a random permutation of the index vector $(1, \ldots, N)^\top$ is $\mathcal{O}(N)$ when using the Fisher-Yates-algorithm (Durstenfeld, 1964), which is used in `R` by the function `sample()`. Since this is all what has to be done for generating a random order of the regression vectors, the overall time complexity of this ordering method is also $\mathcal{O}(N)$. As it can be seen in Figure 3.5, the empirical runtimes of this ordering method are almost as small as the runtimes when using the order the regression vectors appear in the data set (for a more accurate comparison take a look at Table C.1 on page 221 which shows the median empirical runtimes of all presented ordering methods). Furthermore, the outliers and the increase of the runtimes in the number of dimensions D can be explained the same way as in the previous subsection.

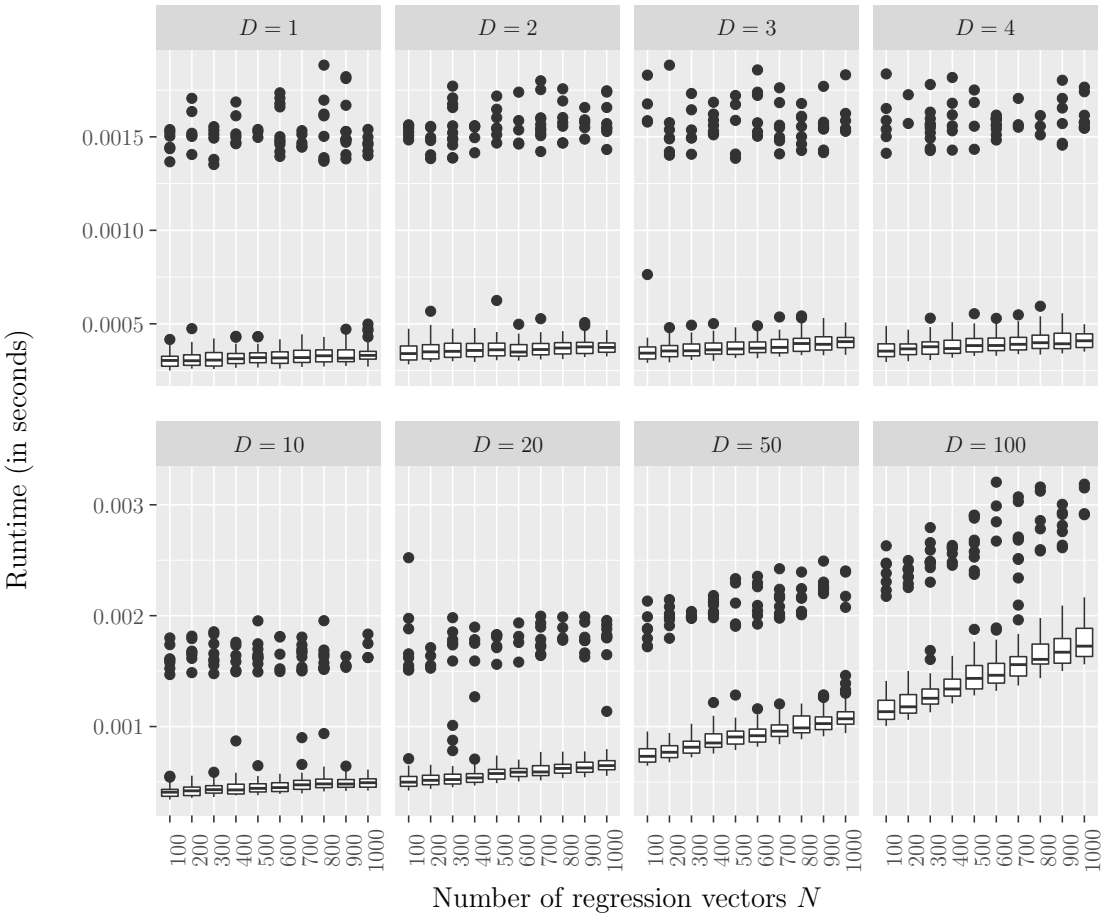


Figure 3.5: Empirical runtimes for generating a random order.

3.2 Scalarization Based Methods

As it can be seen later in Chapter 5, the naive methods do not perform well in most cases. Because of this, it has to be thought of further ordering methods. One general approach is to calculate a single value out of each multidimensional regression vector and order the vectors according to this value. In the following subsections, five scalarization based ordering methods are described.

In general, an advantage of these methods is their fast runtime and easy understanding of the ordering approaches. A disadvantage most of the five presented methods have in common is that they can only be used for purely metric data. In regression, also categorical features are often part of a regression vector. In this case, these methods cannot be applied.

3.2.1 Taking a Norm of Each Regression Vector

A purely metric vector containing the values of a single regression vector can be scalarized by calculating an arbitrary vector norm. The vector norm of a vector $\boldsymbol{x}_n$ is defined via

$$||\boldsymbol{x}_n||_p := \begin{cases} \left(\sum_{d=1}^{D} |x_{nd}|^p \right)^{\frac{1}{p}}, & 1 \leq p < \infty \\ \max_{d=1,\ldots,D} |x_{nd}|, & p = \infty \end{cases} . \tag{3.1}$$

Vectors which are near to the coordinate system origin get small values, whereas vectors which are far away from the coordinate system origin get large values. Thereby, since vector norms are not robust, only a single large value of a component of the regression vector can lead to a large vector norm. Furthermore, ordering the regression vectors according to their vector norm, might lead to an order where the distance between two consecutive regression vectors is large. This can be seen and explained in Figure 3.6, where the obtained orders for $p \in \{1, 2, \infty\}$ are shown. When looking at the Grid-data and $p = 1$ and $p = 2$, it is obvious that the regression vectors in the corners of the grid have the largest values of the norm, so that they are placed one after the other in the ordering, although they are as far away from each other as it can be. In Figure 3.6 it can also be seen that the inherent order of the Spiral-data is preserved only for $p = 2$, whereas in the other parts of Figure 3.6 no real structure of the ordering can be seen.

A big disadvantage of this method for ordering multidimensional data is that the obtained order completely changes if the data is additively transformed. While multiplicative transformation of the data changes the norm of every regression vector by the same factor so that the order is preserved (except it is multiplied by zero), additive transformation affects the regression vectors differently. This can be seen in Figure 3.7. In this figure, the obtained orders of the three data sets are shown when ordering according to the values of the euclidean norm ($p = 2$) and adding the value one to every component of every regression vector in the data sets. When comparing Figure 3.7 with the middle row of Figure 3.6, it can be seen that the orders are completely different. Even the inherent order of the Spiral-data is not there anymore. Although this problem could be solved by always transforming the regression vectors to a given interval (i.e. $[0, 1]^D$ or $[-1, 1]^D$), this might destroy an inherent order which is in the non-transformed data. For example, when saying all

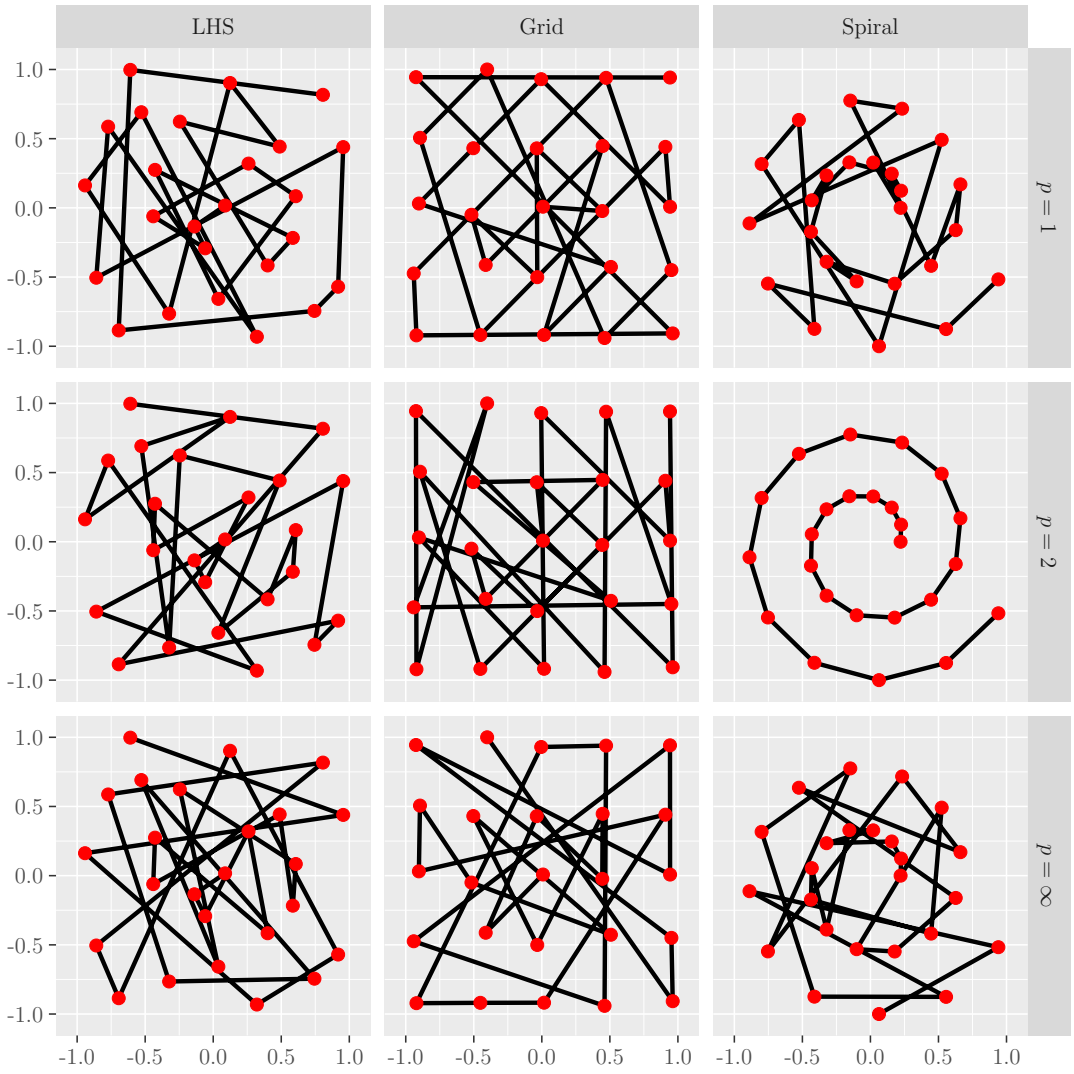


Figure 3.6: Visualization of the obtained orders when using different vector norms for scalarization. The upper row shows orderings according to the manhattan norm ($p = 1$), the middle row shows ordering according to the euclidean norm ($p = 2$) and the lower row shows orderings according to the maximum norm ($p = \infty$).

data is transformed to $[0, 1]^D$ the inherent order of the two-dimensional Spiral-data, which is given on $[-1, 1]^2$, would be destroyed when calculating the euclidean norm.

Another disadvantage of this ordering method is that the inherent order of the values in the one-dimensional case is not obtained when both positive and negative values exist.

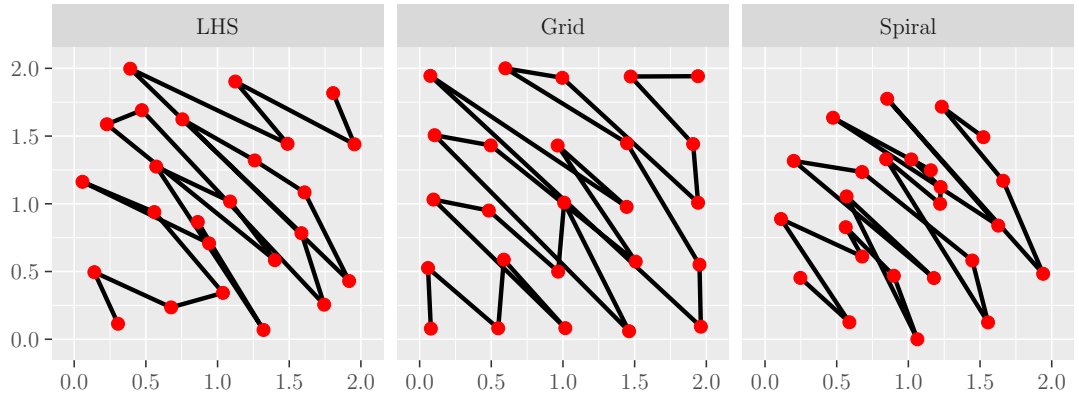


Figure 3.7: Visualization of the obtained orders when using the euclidean norm ($p = 2$) for scalarization and transforming the data by adding the value one to every component of every regression vector.

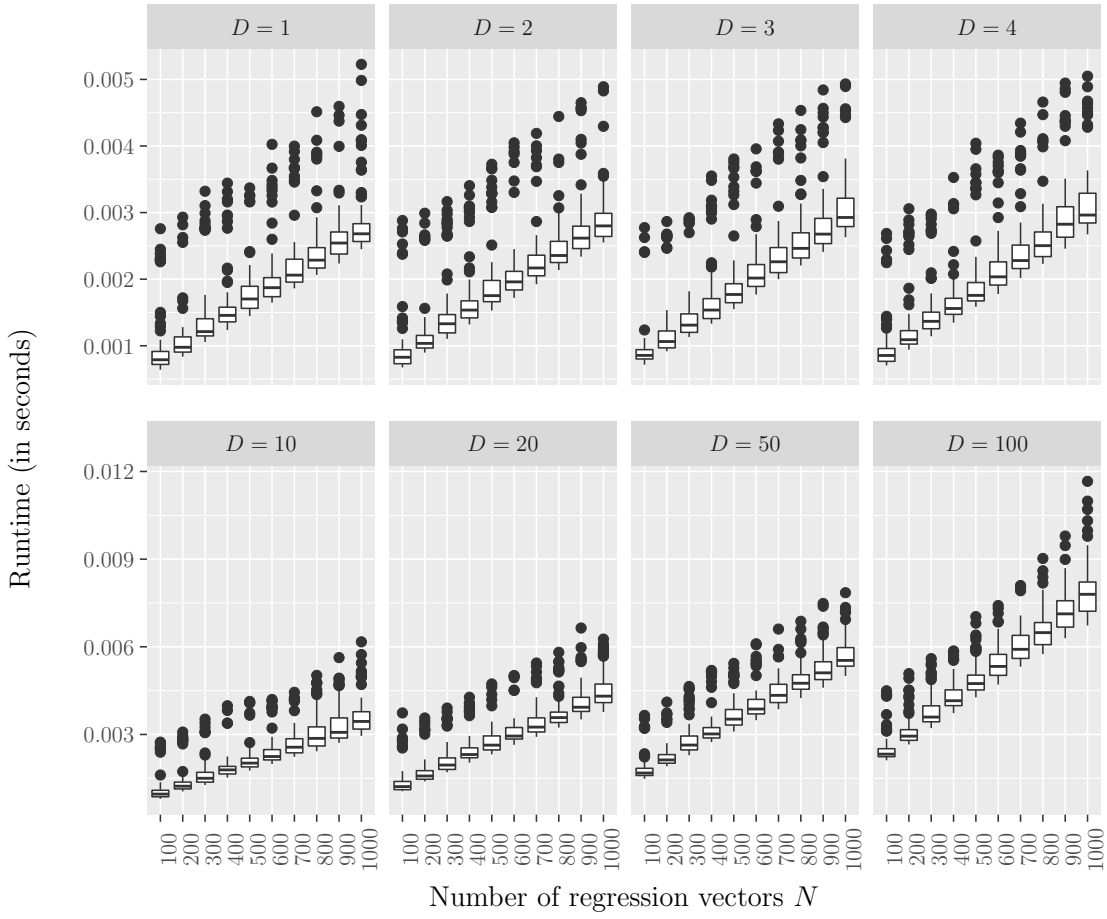


Figure 3.8: Empirical runtimes when using an euclidean norm for ordering. Since computing any vector norm needs linear runtime in D, the empirical runtimes would not change much when using a different vector norm.

An advantage of this ordering method is its rather small time complexity. Calculating an arbitrary norm of a single D-dimensional regression vector needs linear runtime in D because every component of the vector is used once for calculation. When doing this N times, the time complexity for calculating all needed norms is $\mathcal{O}(ND)$. As mentioned above, sorting these N values has approximately linear time complexity, i.e. $\mathcal{O}(N)$, so that the overall time complexity of this sorting method is $\mathcal{O}(ND + N) \in \mathcal{O}(ND)$.

Empirical runtimes of this ordering method can be found in Figure 3.8. The linear runtime in the number of regression vectors N can be clearly seen, whereas the rise in D is very small. Overall, the runtimes are very small (clearly smaller than a second), but they are approximately ten times larger than the runtimes of the naive ordering methods (see also Table C.1 on page 221).

3.2.2 Taking the Median of Each Regression Vector

Since ordering according to the values of a vector norm of the regression vectors is not robust against outliers (even if they occur only in one component of the regression vector), it was thought about a more robust ordering method. The result is an ordering where the median of every regression vector is calculated and the ordering process is done according to these medians. Here, the median of a D-dimensional vector $\boldsymbol{x}_n$ is defined as follows:

$$\tilde{x}_n := \begin{cases} x_{n(k)}, & k = \frac{D+1}{2} \text{ and } D \text{ is odd,} \\ \frac{1}{2}(x_{n(k)} + x_{n(k+1)}), & k = \frac{D}{2} \quad \text{and } D \text{ is even,} \end{cases} \tag{3.2}$$

where $x_{n(1)}, \ldots, x_{n(D)}$ describe the ordered components of the vector $\boldsymbol{x}_n$. In theory, large outliers do not affect the ordering method. In practice, this is only the case if all components of the regression vectors have the same scale. For example, if one regressor of a linear model is the intercept (which is always one), a second regressor has values in the range [10, 20] and a third regressor has values in the range [100, 200], for ordering only the values from the second regressor would be used. And if there is an outlier in the second regressor with, for example, value 50, this outlier affects the ordering. Furthermore, this ordering method does only make sense for three or more dimensions, since the median in the two-dimensional case is the mean of both components which is not robust against outliers. Last but not least, it has to

be ensured that the median of not all regression vectors is the intercept component since then all median values would be the same.

Besides all the disadvantages, this ordering method has the advantage that the inherent order in the one-dimensional case is preserved. Also, in the two-dimensional case when one component is an intercept, the inherent order is preserved. Furthermore, the data can be transformed additively as well as multiplicatively without changing its order because such transformations affect all regression vectors equally and so the medians are also affected the same way.

Figure 3.9 shows the obtained orders for the three presented data sets. The upper row of Figure 3.9 shows the orders in the two-dimensional case, where the median of every regression vector is the mean of the two components. In the lower row, every regression vector has an additional third component: A constant of value one, which is not shown in the plots and corresponds to an intercept in a regression model. Since the first two components have values in the range $[-1, 1]$, the median of the three

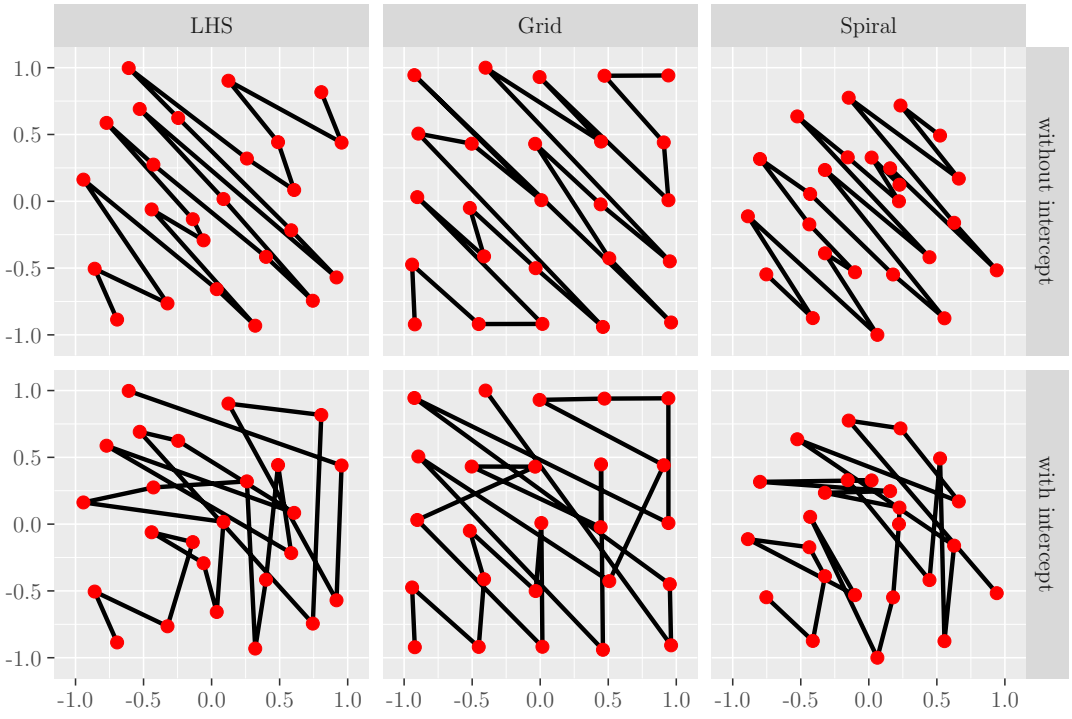


Figure 3.9: Visualization of the obtained orders when using the median as ordering method. The upper row shows the ordering for the presented two-dimensional cases and in the lower row every regression vector has an additional (not shown) intercept of value one.

values is the larger value of the first two components and the smaller one does not affect the ordering in this case.

Calculating a median in `R` is done via partial sorting.[1] This partial sorting has linear time complexity (Blum et al., 1973), i.e. the time complexity for calculating a median of a D-dimensional regression vector is $\mathcal{O}(D)$. Consequently, computing all needed medians has time complexity $\mathcal{O}(ND)$ and ordering these values has time complexity $\mathcal{O}(N)$, so that the overall time complexity of this ordering method is $\mathcal{O}(N + ND) \in \mathcal{O}(ND)$.

[1]Please note, that here "partial sorting" does not mean the same as in Section 3.3. While in Section 3.3 this terms means sorting multidimensional values as far as possible, here "partial sorting" means sorting several one-dimensional values only as far as necessary, i.e. only as long as the median value has been found, the rest of the data may be remained unsorted.

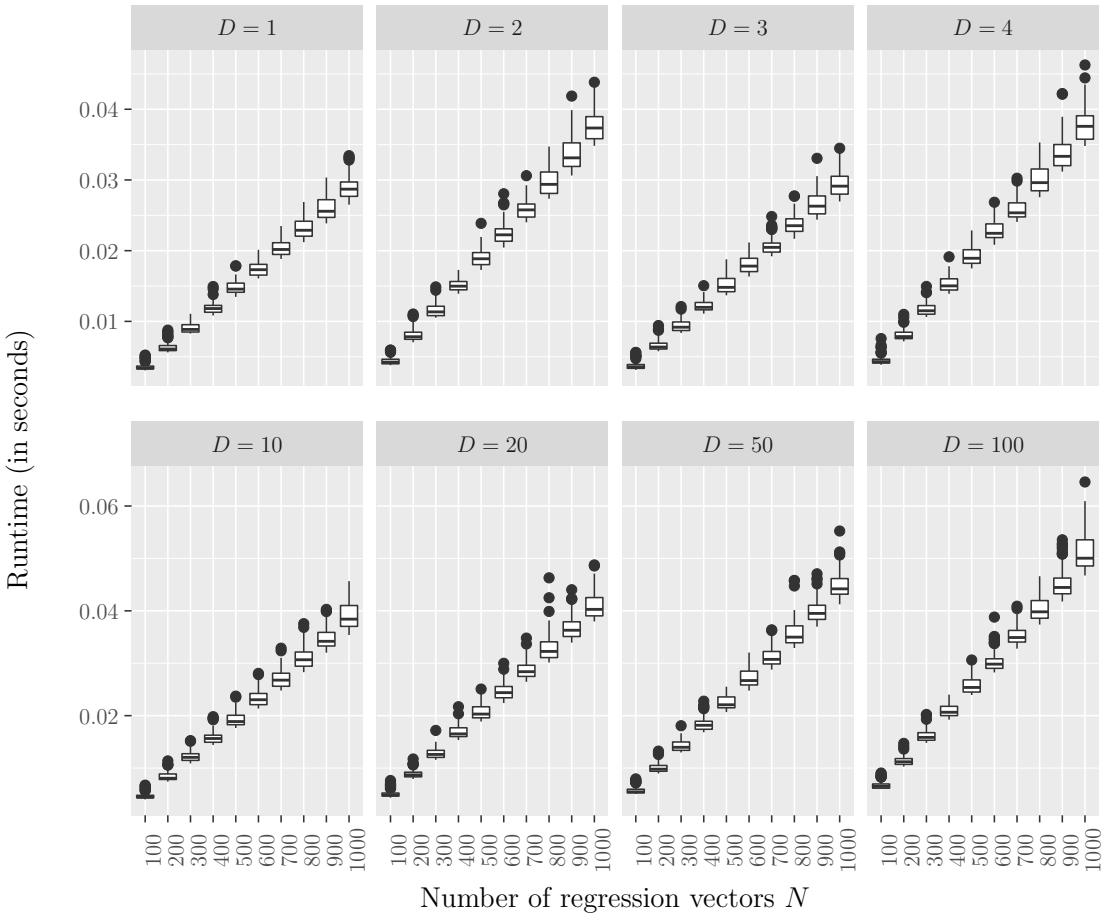


Figure 3.10: Empirical runtimes when using medians for ordering.

The empirical runtimes are shown in Figure 3.10. Overall, the runtimes are quite small, clearly smaller than a second, but they are larger than the runtimes of the former presented methods. The linear trend in the number of regression vectors N can be clearly seen, whereas the linear trend in D cannot be seen very well, especially because the runtimes for $D = 2$ and $D = 4$ are larger than the runtimes for $D = 1$ and $D = 3$. This phenomenon can be explained by the fact that for even numbers of dimensions the mean of two values has to be calculated. While calculating the mean of two values is done in a very small constant runtime, finding these two values instead of finding only one value when the number of dimensions is odd takes some more time. Here, it is remarkable that the runtimes of the two-dimensional case are comparatively large, because it could be thought that the R-function `median()` would be quicker in this case, since it has not to search for the two values of which the mean has to calculated, but apparently this is not the case.

3.2.3 Taking Only One Component of Each Regression Vector

One of the most intuitive ordering methods may be an ordering according to only one component of each regression vector by neglecting all other components. The advantages of this ordering method are clear: it is easy to understand, easy to compute and the inherent order when having only one dimension will be preserved. Furthermore, nominal and ordinal components in the regression vectors are allowed, if they are not chosen for the ordering and additive and multiplicative transformations of the data do not affect the ordering. Also, the disadvantages are obvious: the information of $D - 1$ components of the regression vectors gets lost and the ordering heavily depends on the selection of the component for ordering. This may have crucial effect on the sign depth and the sign depth test, see Section 2.7. Figure 3.11 shows the obtained orders on the three given data sets when choosing the first component of each regression vector for ordering.

The time complexity of this ordering method is easy to determine. Since sorting N values has an approximate linear time complexity (see beginning of this chapter for explanation), the time complexity of this method is $\mathcal{O}(N)$. In practice, Figure 3.12 shows that the runtimes are very small, indeed they are not much larger than the runtimes of the naive ordering methods. The linear trend in the number of regression vectors N can be seen well, but as in Subsections 3.1.1 and 3.1.2 there seems to be also a linear trend in the number of dimensions D which should not be there in

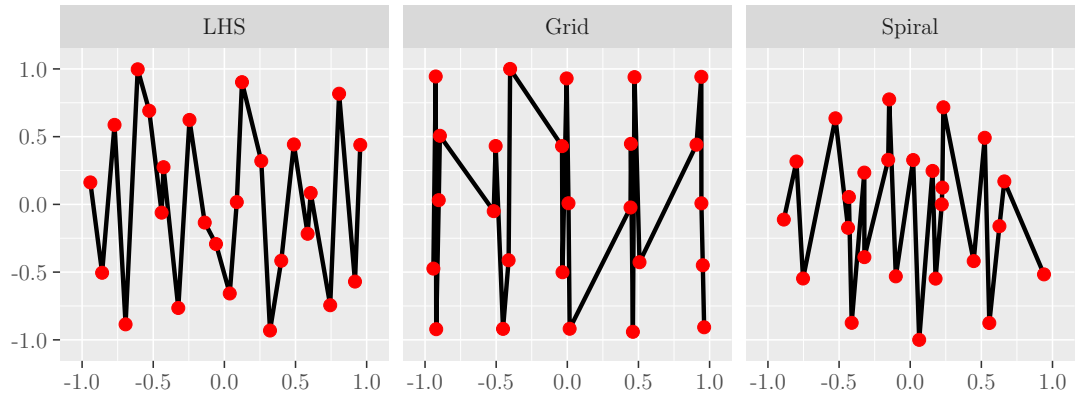


Figure 3.11: Visualization of the obtained orders when using only the first component of each regression vector for ordering.

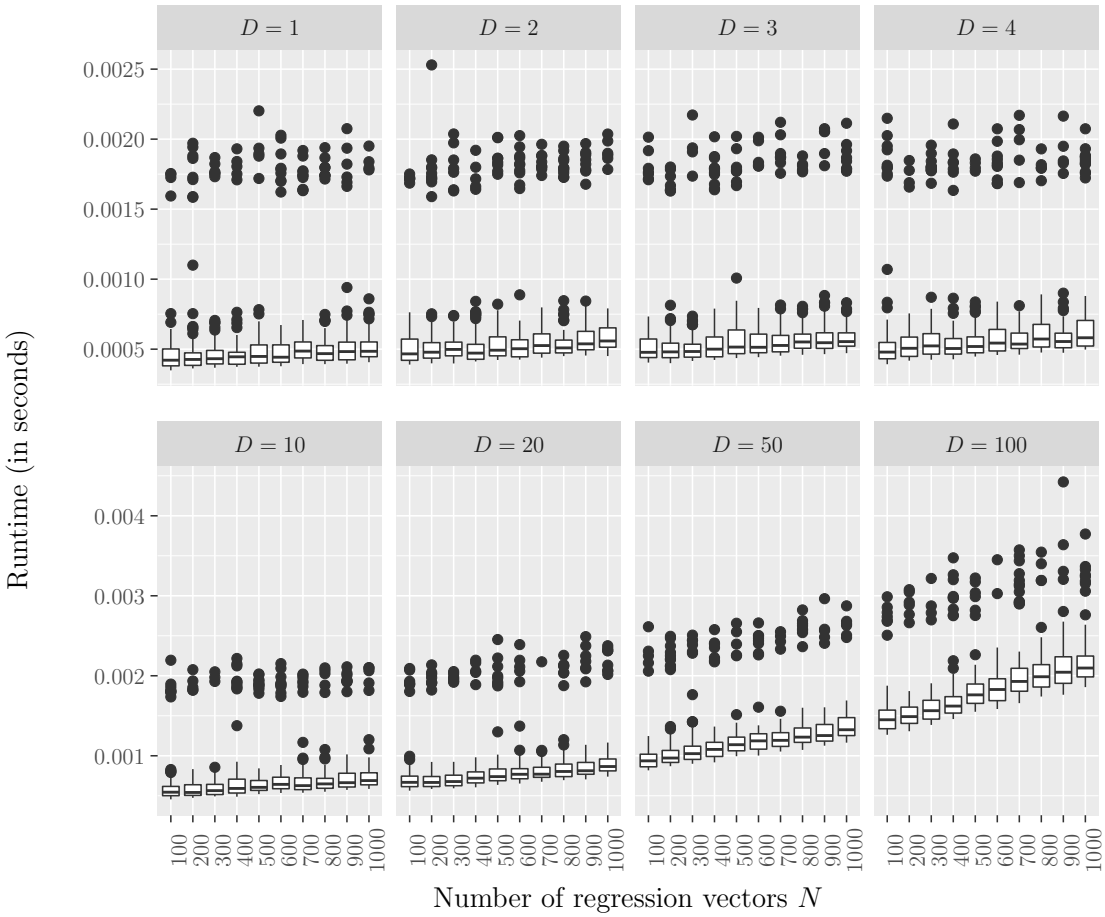


Figure 3.12: Empirical runtimes when using the first component of each regression vector for ordering.

theory. As in Subsection 3.1.1 explained, the increasing runtimes in D are overhead from the implementation of the sorting function in `R`.

3.2.4 Taking a Weighted Sum of Each Regression Vector

A generalization of taking only one component of each regression vector for ordering is to take several components, possibly with different weightings. This leads to an ordering method where the order is determined by the values of a weighted sum of each regression vector.

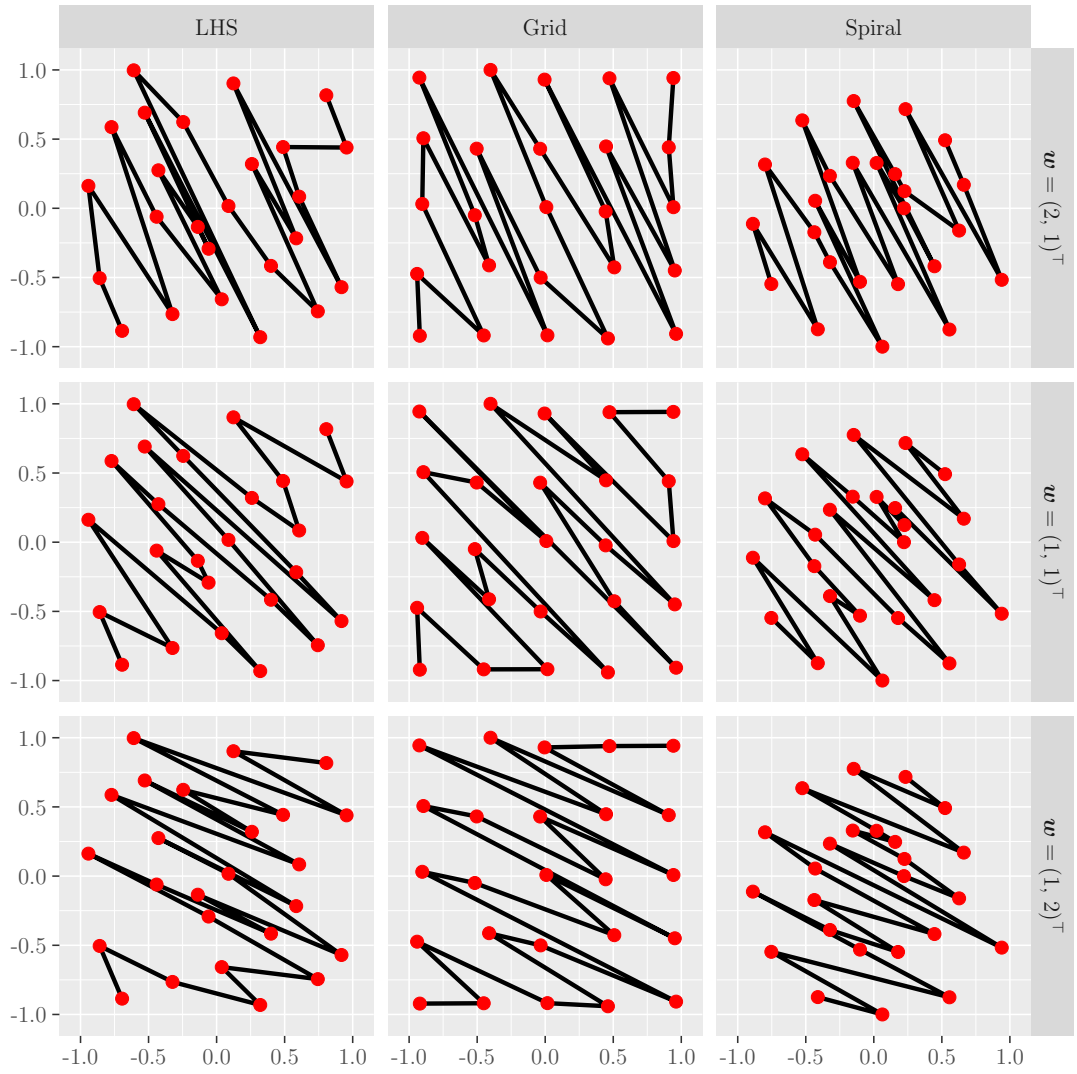


Figure 3.13: Visualization of the obtained orders when using different weighted sums for ordering.

For this, let $\boldsymbol{w} \in \mathbb{R}^D$ with $\forall d \in \{1, \ldots, D\} : w_d \geq 0$ and $\exists d \in \{1, \ldots, D\} : w_d > 0$. Then, the ordering is done on the basis of the values of $\boldsymbol{w}^\top \boldsymbol{x}_n$, calculated for each $n = 1, \ldots, N$. If only one component of $\boldsymbol{w}$ is positive, the obtained order is the same as in the previous subsection. If all components of $\boldsymbol{w}$ have the same value, in the two-dimensional case the ordering is the same as in subsection 3.2.2 (the middle row of Figure 3.13 is the same as the upper row of Figure 3.9). Advantages of this ordering method are the easy and understandable calculation and the possibility to rescale all components of the regression vectors to the same level via the magnitude of the components of the weight vector. Furthermore, additive and multiplicative transformations do not affect the ordering and the inherent order for $D = 1$ is preserved. A big disadvantage is that the ordering heavily depends on the choice of the weight vector $\boldsymbol{w}$. This can also be seen in Figure 3.13, where orderings with three different weight vectors are shown. In addition, only metric values in the regression

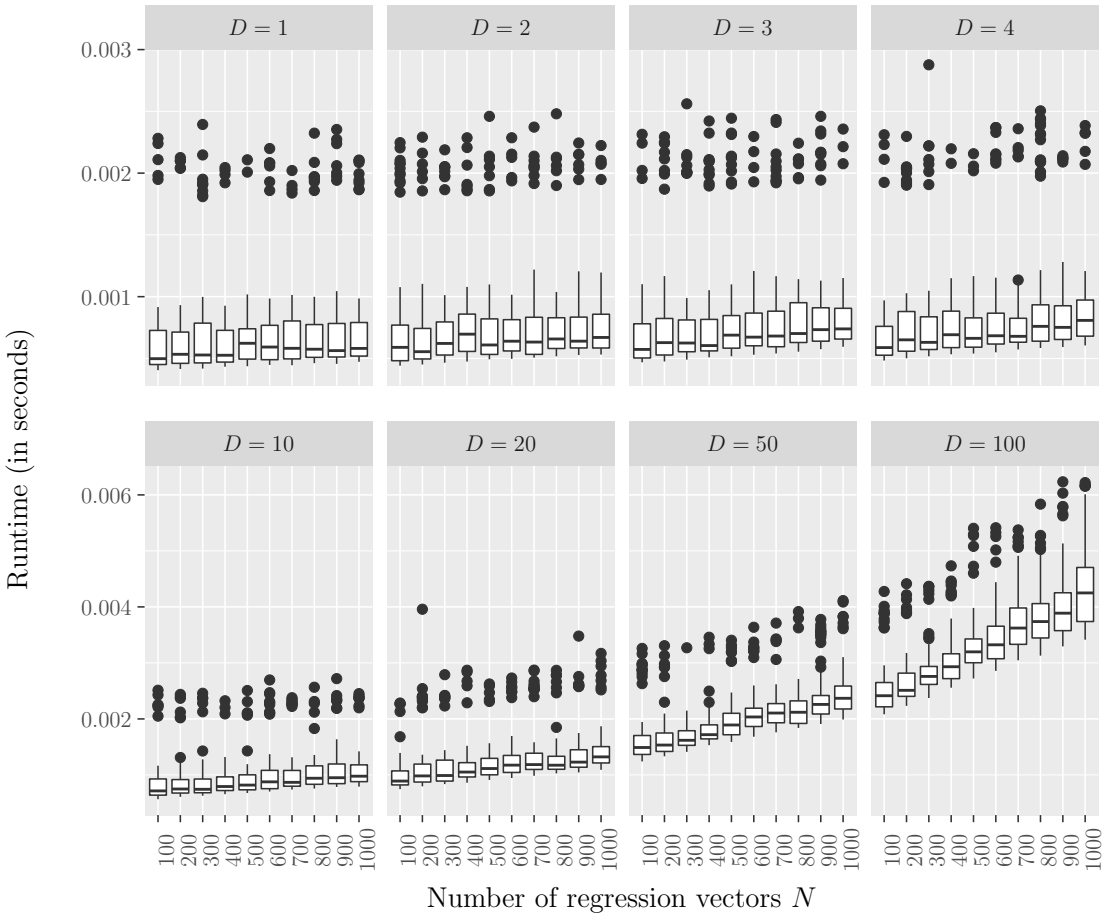


Figure 3.14: Empirical runtimes when using a weighted sum (with weight vector $\boldsymbol{w} = (1, \ldots, 1)^\top \in \mathbb{R}^D$) for ordering.

vectors are possible. Even if the corresponding weight is zero the component of the regression vector is not allowed to be ordinal or nominal because then $\boldsymbol{w}^\top \boldsymbol{x}_n$ cannot be calculated anymore.

Calculating $\boldsymbol{w}^\top \boldsymbol{x}_n$ has time complexity $\mathcal{O}(D)$ for a single regression vector $\boldsymbol{x}_n$ since $\boldsymbol{w}$ and $\boldsymbol{x}_n$ are both D-dimensional. Consequently, calculating all weighted sums has time complexity $\mathcal{O}(ND)$, so that the overall time complexity including the sorting of the weighted sums is $\mathcal{O}(ND + N) \in \mathcal{O}(ND)$.

Figure 3.14 shows that the empirical runtimes are as small as the runtimes of the naive methods. Also, a small linear increase in N and D can be seen in Figure 3.14, which is independent of the choice of the weight vector $\boldsymbol{w}$.

3.2.5 Taking an Orthogonal Projection of Each Regression Vector

A further generalization of the described methods in Subsections 3.2.3 and 3.2.4 is to project all regression vectors orthogonal on a line and order the regression vectors according to these values. Given a location vector $\boldsymbol{u} \in \mathbb{R}^D$ and a direction vector $\boldsymbol{v} \in \mathbb{R}^D$ the orthogonal projection on the line $\boldsymbol{u} + \lambda \boldsymbol{v},\ \lambda \in \mathbb{R}$ of each regression vector $\boldsymbol{x}_n$ can be determined via

$$\lambda_n = \frac{(\boldsymbol{x}_n - \boldsymbol{u})^\top \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}} = \frac{\boldsymbol{x}_n^\top \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}} - \frac{\boldsymbol{u}^\top \boldsymbol{v}}{\boldsymbol{v}^\top \boldsymbol{v}},\ n = 1, \ldots, N. \tag{3.3}$$

Obviously, calculating the value of λ_n for every $\boldsymbol{x}_n$ and ordering the regression vectors according to their values of λ_n is sufficient for this ordering method. Based on the formula for calculating λ_n, it can be seen that the location vector $\boldsymbol{u}$ is completely irrelevant for the obtained orders, since $\boldsymbol{u}$ affects only the second summand which is independent from $\boldsymbol{x}_n$ and so, the second summand is the same constant value for every regression vector $\boldsymbol{x}_n$. In fact, this ordering method is equivalent to an ordering according to a weighted sum, see Subsection 3.2.4, if the direction vector consists of non-negative values. In contrast to the weights of an ordering according to the values of a weighted sum, here negative values in the direction vector are allowed. The only reason for the requirement of non-negative weights in the previous subsection is the general understanding that weights are non-negative. If one would allow negative weights in the weighted sums, the projection method would have no additional benefit compared with the weighted sum method. On the contrary, the projection method is more difficult to understand. Apart from that, this ordering

method has the same advantages and disadvantages as the weighted sum method, since it is mostly the same ordering method.

Figure 3.15 shows the ordering on the three presented data sets with two different location and direction vectors. The lower row of Figure 3.15 shows the ordering according to the orthogonal projection on the bisecting line, which leads to the same ordering as an ordering according to a weighted sum with equal weights for every component of the regression vectors (see middle row of Figure 3.13). It can be seen that the choice of the direction vector $\boldsymbol{v}$ has a crucial effect on the obtained order.

Computing the value of a single λ_n has time complexity $\mathcal{O}(D)$ because it consists of addition and multiplication of D-dimensional vectors. So, as for the weighted sum method, the overall time complexity for computing all λ_n and sorting them is $\mathcal{O}(ND + N) \in \mathcal{O}(ND)$ because linear time complexity in N and D is needed for computing all values of λ_n and additionally it takes linear time complexity in N for sorting these values.

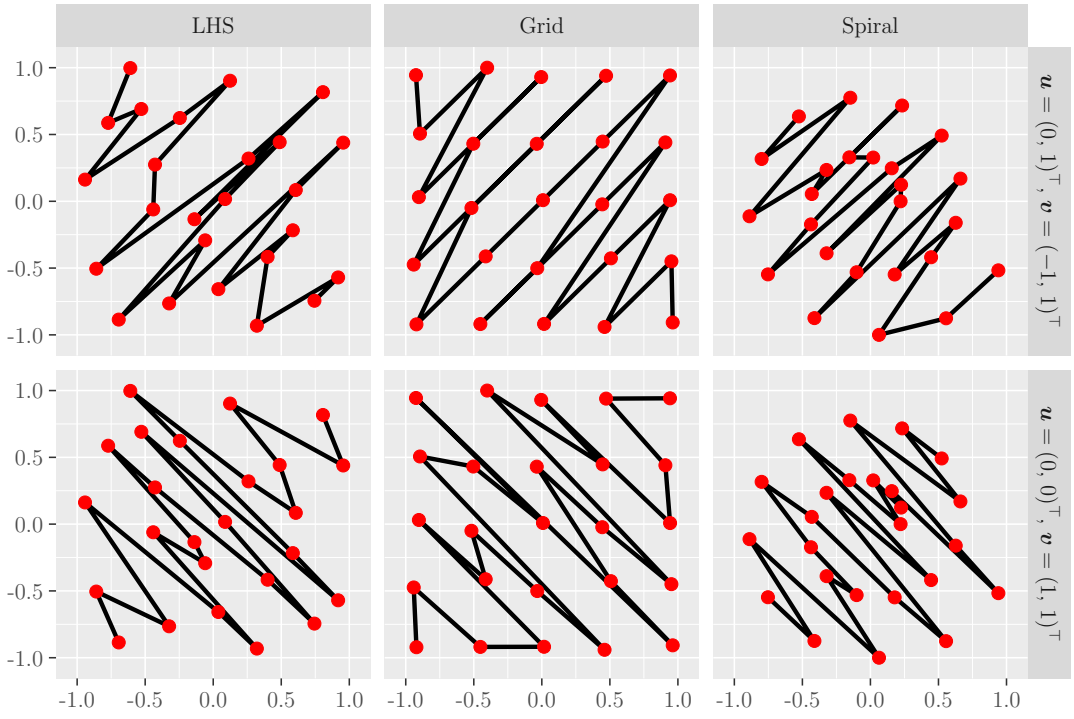


Figure 3.15: Visualization of the obtained orders when using different orthogonal projections for ordering. The upper row shows orderings based on orthogonal projections on the line $\binom{0}{1} + \lambda \cdot \binom{-1}{1}$ and the lower row shows orderings based on orthogonal projections on the line $\binom{0}{0} + \lambda \cdot \binom{1}{1}$.

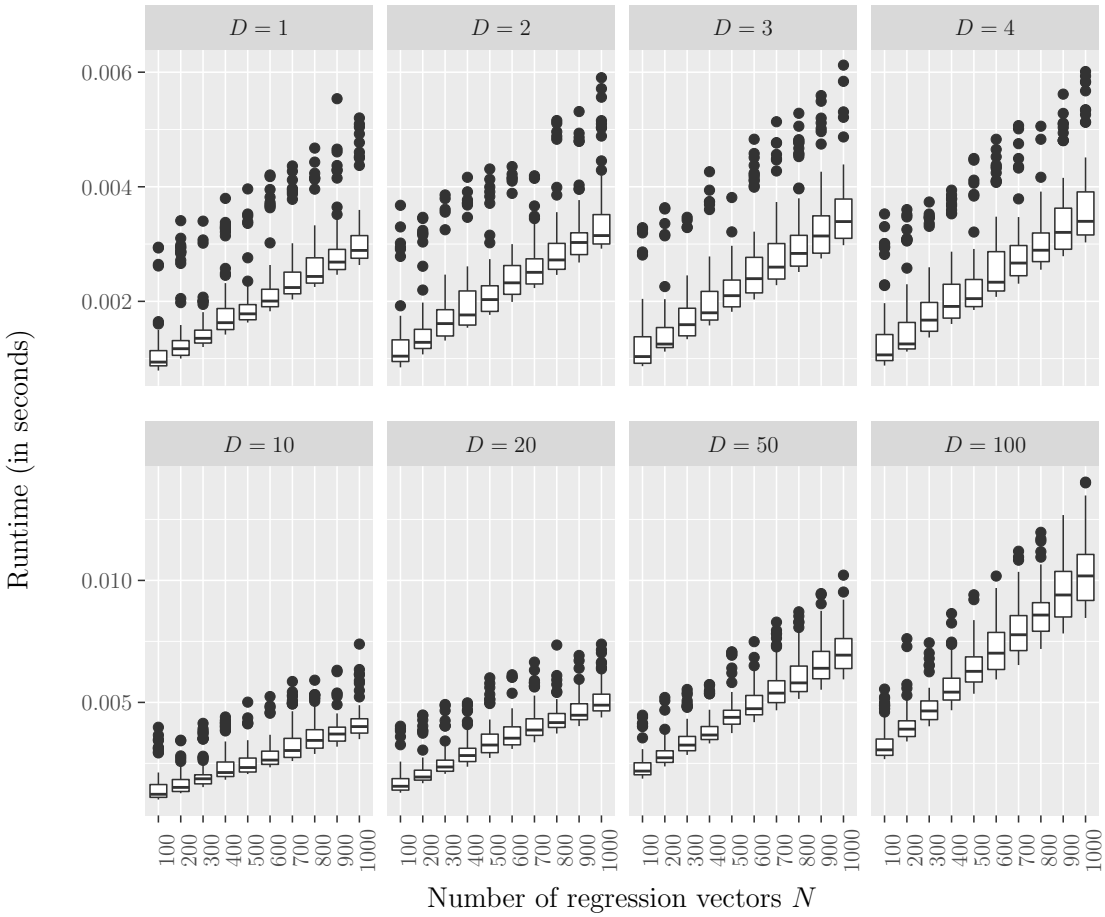


Figure 3.16: Empirical runtimes when using a projection on the bisecting line for ordering ($\boldsymbol{u} = (0, \ldots, 0)^\top \in \mathbb{R}^D$, $\boldsymbol{v} = (1, \ldots, 1)^\top \in \mathbb{R}^D$).

But, as it can be seen in Figure 3.16, the empirical runtimes are slightly larger than the runtimes in the previous subsection. So, as a conclusion, this ordering method has not much additional benefit compared with the weighted sum method, but it is less understandable and slower to compute.

3.3 Orders Based on Partial Sorting

Multidimensional values cannot only be sorted with the help of scalarizations, but it is also possible to order these values directly. However, in the multidimensional case often it is not possible to determine if a specific vector is smaller or greater than another, these vectors are then called *incomparable*. For example, it might be easy to say that the vector $(1, 2)^\top$ is smaller than the vector $(2, 3)^\top$, but it is not possible to

compare the vectors $(1, 2)^\top$ and $(2, 1)^\top$. So, it is possible to rank multidimensional values, but the ranks are not unique. All incomparable values get the same rank. This procedure is called *partial sorting*.

For our purpose, we need a clear sorting, so the incomparable values have also be ordered somehow. Three different methods for partial sorting and tie-breaking are described in the following subsections. All three methods have in common that they need metric data, ordinal or nominal values are not allowed.

A detailed description of the three ordering methods can be found in the following subsections, but because a big disadvantage of these methods applies to all of the three methods, it is described below and because of this a short overview of the methods is given at this point. The first presented method is an ordering on the basis of a nondominated sorting. The nondominated sorting has its origin in the context of multicriteria optimization. It defines a multidimensional value to be smaller than another multidimensional value if it is smaller or equal in all components of the regression vector and smaller in at least one component of the regression vector. The second presented method is an ordering according to convex hulls. For this method, repetitively convex hulls of the multidimensional values are computed as long as every value belongs to exactly one hull. The third method is an ordering according to the values of Tukey's halfspace depth. The halfspace depth aims to give every multidimensional value a one-dimensional value describing how "deep" the value is in the data set, so that values "on the edge" of the data set get small values and values "in the center" of the data set get the largest values.

The above mentioned big disadvantage all three methods have in common, is the following: When the number of dimensions D is growing, more and more multidimensional values get incomparable. This leads to the fact that many (or nearly all) values get the same rank (usually the first rank) and the partial sorting has no benefit, since a tie-breaking with nearly all values has to be applied. Figure 3.17 shows this behavior. For this figure, $N = 100$ random D-dimensional points were created and all three ordering methods were applied. This was made 100 times independently. The left plot shows the number of different ranks in the data. The lines are the means of the 100 independent simulation runs and the ribbons describe minimums and maximums. It can be seen that whereas for two dimensions the 100 multidimensional values get split in mean in more than 15 groups for the nondominated sorting method, in more than 10 groups for the convex hull method and in more than 30 groups for the halfspace depth method, in higher dimensions the number of groups (i.e. the

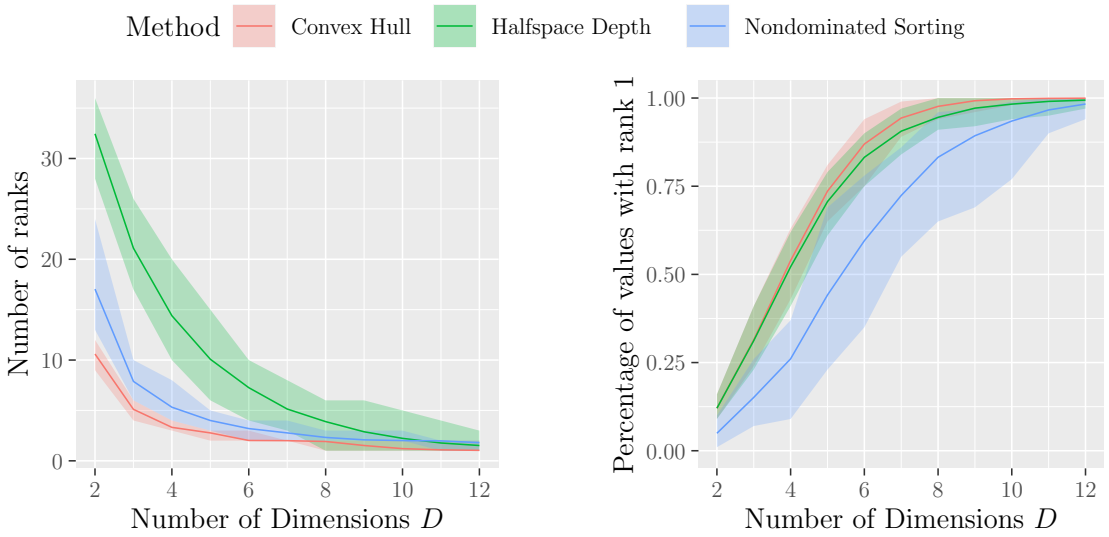


Figure 3.17: Visualization of the characteristics in high dimensions of the three presented partial sorting methods. The left plot shows how many different ranks were assigned and the right plot shows the percentage of values which have gotten the first rank. Here, $N = 100$ multidimensional values are used and the values in the plots are the means, minimums and maximums of 100 independent simulation runs.

number of different ranks) decreases very fast. In five dimensions, the nondominated sorting method and the convex hull method split the data in less than five groups in mean and in twelve dimensions, there is not a single simulation run of the both methods which has split the data in more than two groups. Although the halfspace depth method performs slightly better (i.e. it splits the data nearly always in more groups than the other methods), it shows the same behavior. Moreover, there were simulation runs of all three methods where all values got the same rank, so that the partial sorting had absolutely no benefit. The right plot of Figure 3.17 shows the percentage of values, which got rank 1. It can be seen that this percentage increases very fast to a value near one. This means, that not only the number of groups decreases, but also nearly all values are assigned to the first group and the remaining groups contain nearly no values. Overall, all three methods are in practice not useful for more than four or five dimensions.

3.3.1 Partial Sorting via Nondominated Sorting

Nondominated sorting is a technique which has its origin in the context of multicriteria optimization. It bases on the concept of *pareto dominance*. A regression vector $\boldsymbol{x}_n$ is said to dominate another regression vector $\boldsymbol{x}_m$ if all components of $\boldsymbol{x}_n$ are at most as large as the respective components of $\boldsymbol{x}_m$ and at least one component of $\boldsymbol{x}_n$ is smaller than the respective one of $\boldsymbol{x}_m$. Formally written, pareto dominance can be defined via (see for example Emmerich and Deutz (2018)):

$$\boldsymbol{x}_n \prec \boldsymbol{x}_m :\Leftrightarrow \forall d \in \{1, \ldots, D\} : x_{nd} \leq x_{md} \ \wedge \ \exists d \in \{1, \ldots, D\} : x_{nd} < x_{md}. \quad (3.4)$$

If $\boldsymbol{x}_n \not\prec \boldsymbol{x}_m$ and $\boldsymbol{x}_m \not\prec \boldsymbol{x}_n$, $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ are *incomparable* or *nondominated*. A partial order can now obtained when finding groups where all regression vectors in the respective group are incomparable, but all regression vectors in the first group dominate all regression vectors in the second group and all regression vectors in the second group dominate all regression vectors in the third group and so on. This procedure is called *nondominated sorting*.

Algorithm 3.1 Nondominated sorting algorithm as described in Deb et al. (2000).

Input: (Data) Set of multidimensional values $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
Output: Sets of nondominated values $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, ...

$\boldsymbol{F}_1 = \emptyset$
for all $\boldsymbol{x}_n \in \boldsymbol{X}$ **do**
  $m_{\boldsymbol{x}_n} = 0$; $\boldsymbol{S}_{\boldsymbol{x}_n} = \emptyset$
  for all $\boldsymbol{x}_m \in \boldsymbol{X}$ with $\boldsymbol{x}_n \neq \boldsymbol{x}_m$ **do**
    if $\boldsymbol{x}_n \prec \boldsymbol{x}_m$ **then**
      $\boldsymbol{S}_{\boldsymbol{x}_n} = \boldsymbol{S}_{\boldsymbol{x}_n} \cup \{\boldsymbol{x}_m\}$
    else if $\boldsymbol{x}_m \prec \boldsymbol{x}_n$ **then**
      $m_{\boldsymbol{x}_n} = m_{\boldsymbol{x}_n} + 1$
    end if
  end for
  if $m_{\boldsymbol{x}_n} = 0$ **then**
    $\boldsymbol{F}_1 = \boldsymbol{F}_1 \cup \{\boldsymbol{x}_n\}$
  end if
end for
$i = 1$
while $\boldsymbol{F}_i \neq \emptyset$ **do**
  $\boldsymbol{H} = \emptyset$
  for all $\boldsymbol{x}_n \in \boldsymbol{F}_i$ **do**
    for all $\boldsymbol{x}_m \in \boldsymbol{S}_{\boldsymbol{x}_n}$ **do**
      $m_{\boldsymbol{x}_m} = m_{\boldsymbol{x}_m} - 1$
      if $m_{\boldsymbol{x}_m} = 0$ **then**
        $\boldsymbol{H} = \boldsymbol{H} \cup \{\boldsymbol{x}_m\}$
      end if
    end for
  end for
  $i = i + 1$
  $\boldsymbol{F}_i = \boldsymbol{H}$
end while
return $\boldsymbol{F}_1$, $\boldsymbol{F}_2$, ...

The state-of-the-art algorithm for nondominated sorting was invented two decades ago by Deb et al. (2000). A pseudocode of this algorithm is given in Algorithm 3.1. Its idea is to first compute for every regression vector $\boldsymbol{x}_n$ a set $\boldsymbol{S}_{\boldsymbol{x}_n}$ of values which are dominated by $\boldsymbol{x}_n$ and a number $m_{\boldsymbol{x}_n}$ of values which dominate $\boldsymbol{x}_n$ (see lines 2 to 10 of Algorithm 3.1). All regression vectors $\boldsymbol{x}_n$ which are not dominated by any other regression vector (i.e. $m_{\boldsymbol{x}_n} = 0$) build the first group $\boldsymbol{F}_1$ (see line 11 to 13). The while-loop from line 16 to 28 builds iteratively further sets $\boldsymbol{F}_i$. Basically, all regression vectors $\boldsymbol{x}_n$ with the smallest positive value $m_{\boldsymbol{x}_n}$ build the second group $\boldsymbol{F}_2$, all regression vectors $\boldsymbol{x}_n$ with the next larger value of $m_{\boldsymbol{x}_n}$ build the third group $\boldsymbol{F}_3$ and so on.

A visualization of the values of $m_{\boldsymbol{x}_n}$ for the three presented data sets (LHS, Grid and Spiral) can be found in Figure 3.18. Since $m_{\boldsymbol{x}_n}$ describes the number of regression vectors which dominate a specific regression vector $\boldsymbol{x}_n$, in every case there must be at least one regression vector with $m_{\boldsymbol{x}_n} = 0$. These regression vectors (marked

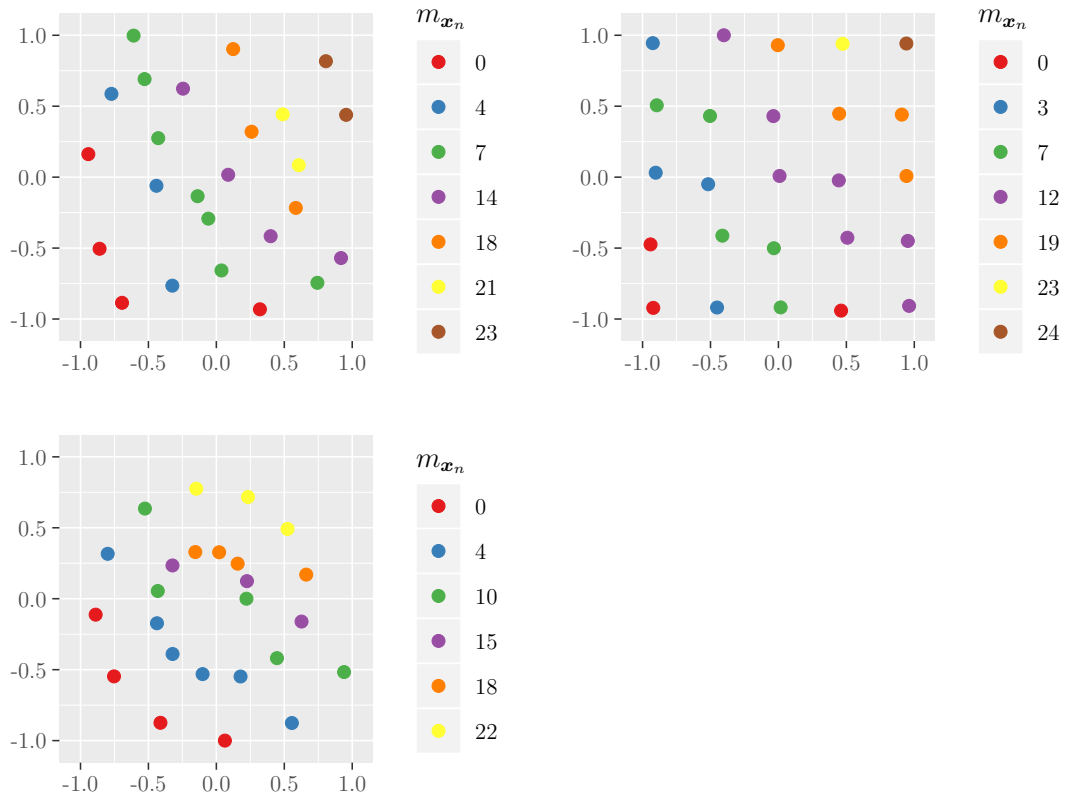


Figure 3.18: Visualization of the values of $m_{\boldsymbol{x}_n}$ for the three presented data sets - LHS, Grid and Spiral. The value $m_{\boldsymbol{x}_n}$ specifies the number of regression vectors which dominate the respective regression vector $\boldsymbol{x}_n$.

with red in the plots) build the respective group $\boldsymbol{F}_1$. The respective groups $\boldsymbol{F}_2$ are marked with blue in every plot, even though they have slightly different values of $m_{\boldsymbol{x}_n}$. While every blue marked regression vector in the Grid data is dominated by three red marked regression vectors, in the other two data sets every blue marked regression vector is dominated by four red marked regression vectors. Overall, the LHS-data and the Grid-data are split into seven groups and the Spiral-data is split into six groups.

When having obtained the partial order, one has to decide how to order the incomparable values. Basically, for this, one can use every other ordering method presented in this chapter. Figure 3.19 shows the obtained orders on the three presented data sets with two different approaches for tie-breaking. The plots in the upper row show the orders when ordering the incomparable regression vectors according to their

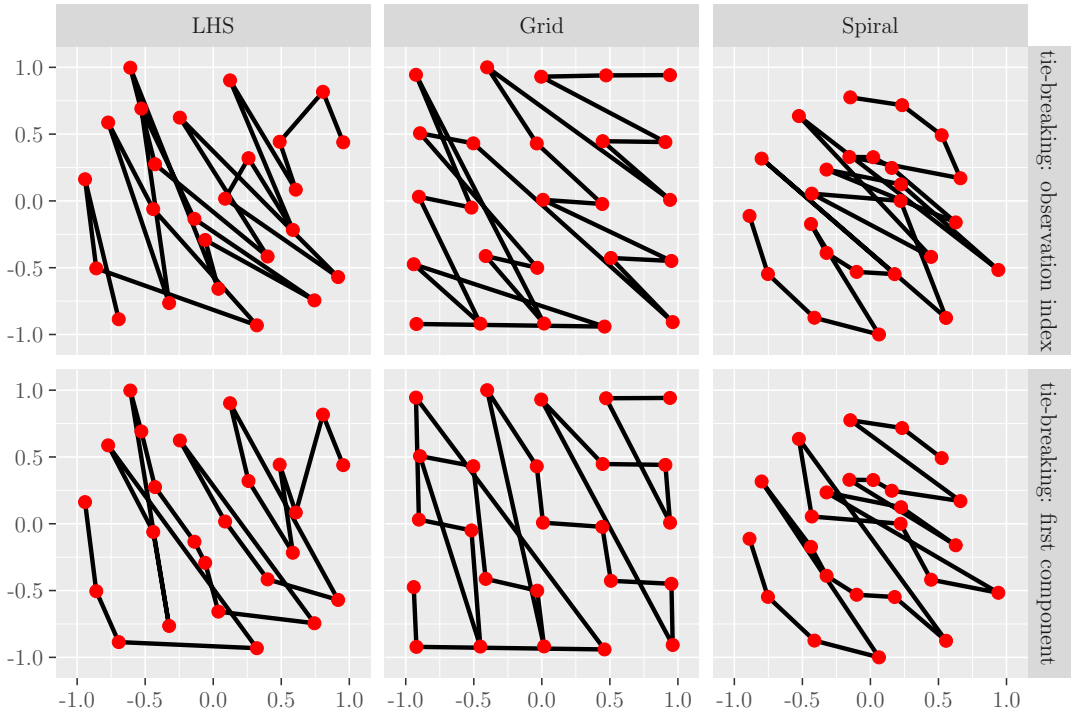


Figure 3.19: Visualization of the obtained orders when using nondominated sorting for partial sorting. The upper row shows the obtained orders when the incomparable regression vectors get ordered according to their appearance in the data set (see Subsection 3.1.1) and the lower row shows the obtained orders when ordering the incomparable regression vectors according to the value of their first component (see Subsection 3.2.3).

appearance in the data set (see Subsection 3.1.1) and the plots in the lower row show the orders when ordering the incomparable regression vectors according to the value of their first component (see Subsection 3.2.3). It can be seen that the tie-breaking method has definitely an effect on the orders. Even though the approximate order is given in both rows through the same partial ordering, in detail the obtained orders differ a lot.

Next to the before mentioned problem of this ordering method in high dimensions, also the great dependance on the tie-breaking method is a big disadvantage. Furthermore, the obtained orders may change when transforming the data. While additive transformation does not change the obtained partial order (but possibly the order of the incomparable regression vectors depending on the chosen tie-breaking method), multiplying the data with negative values will change the obtained partial order. But, for one-dimensional data the inherent order of the data is preserved when using nondominated sorting, which is an advantage of this ordering method.

The time complexity for the nondominated sorting is quite large. In Deb et al. (2000) the time complexity of the presented algorithm for nondominated sorting is given by $\mathcal{O}(DN^2)$. This time complexity can easily be comprehended when looking at Algorithm 3.1. The two loops in lines 2 to 14 compare all $N \cdot (N-1)$ pairs of regression vectors. For comparing them, every time (up to all) D components of the respective two regression vectors have to be compared. This results in a time complexity of $\mathcal{O}(DN(N-1)) \in \mathcal{O}(DN^2)$ for the first 14 lines of the algorithm. The worst case time complexity for the remaining lines of Algorithm 3.1 will be reached when all N regression vectors are comparable and so N sets $\boldsymbol{F}_1, \ldots, \boldsymbol{F}_N$ are created. In this case, the while-loop makes N iterations, the for-loop beginning at line 18 makes one iteration and the for-loop beginning at line 19 makes $N-i$ iterations for set $\boldsymbol{F}_i$. This results in $\mathcal{O}(N^2)$ iterations where the individual iterations have constant runtime. So overall, the time complexity of this algorithm is $\mathcal{O}(DN^2 + N^2) \in \mathcal{O}(DN^2)$.

The quadratic increase in N can be seen pretty well in the empirical runtimes shown in Figure 3.20. Also, an increase of the runtimes for larger values of D can be seen. Furthermore, it can be seen that the runtimes are much larger than the runtimes in Sections 3.1 and 3.2. While in previous sections the runtimes always were fractions of seconds, the runtimes of the nondominated sorting are more than two minutes when having several hundred regression vectors and a small number of dimensions. For the case $D = 100$, the runtimes are larger than half an hour for several hundreds of regression vectors. Remembering that for such large numbers of dimensions the

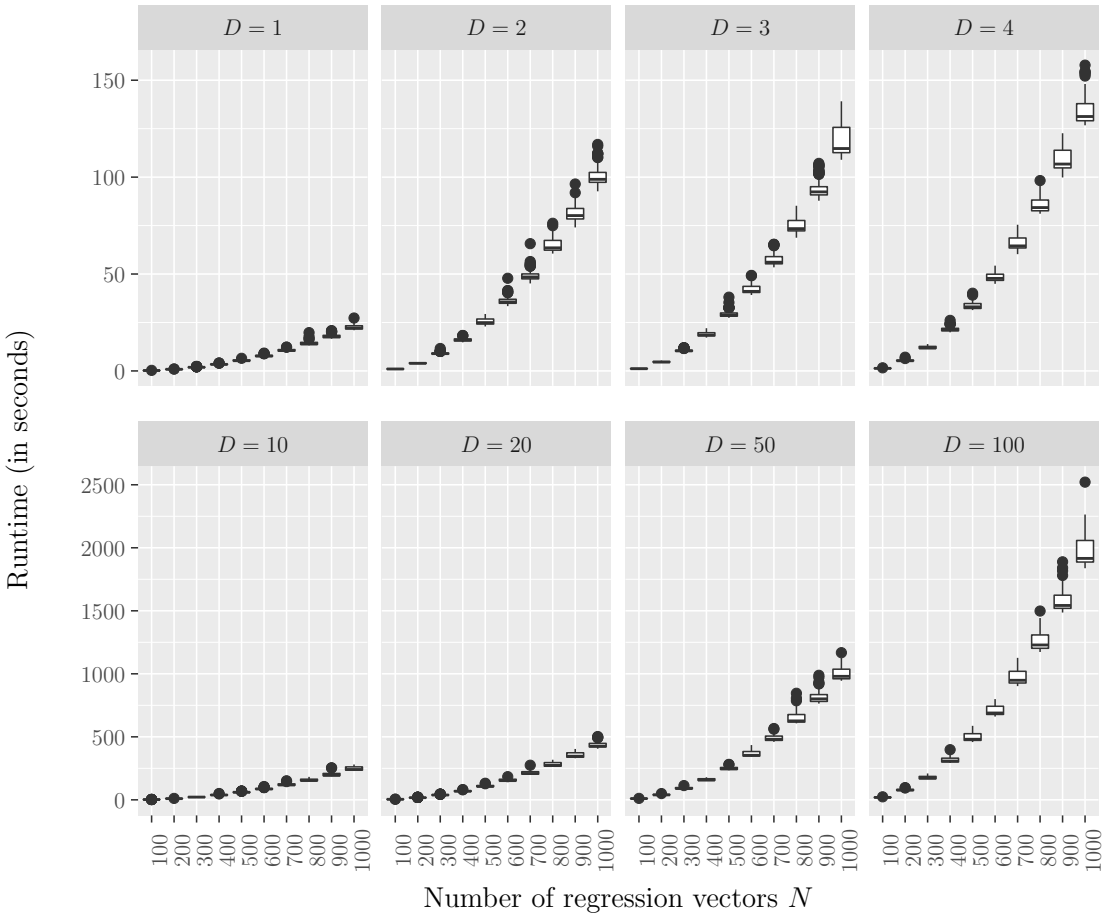


Figure 3.20: Empirical runtimes when using a nondominated sorting for ordering. Here, tie-breaking is made according to the observation index (see Subsection 3.1.1) which has time complexity $\mathcal{O}(1)$, so that the shown runtimes are obtained solely by the nondominated sorting algorithm.

nondominated sorting will have no real benefit because (nearly) all multidimensional values will have the same rank these large runtimes are not acceptable.

Overall, this ordering method has large runtimes and many further disadvantages, but not many advantages. So, the usability of this ordering method can be regarded as low in contrast to other ordering methods.

3.3.2 Partial Sorting via Convex Hulls

Another approach for partial sorting multidimensional values is to use convex hulls. The convex hull of a (data) set $\boldsymbol{X}$ which consists of N D-dimensional regression

vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is defined via (see for example Rockafellar (1970))

$$conv\ \boldsymbol{X} := \left\{ \sum_{n=1}^{N} \lambda_n \cdot \boldsymbol{x}_n \middle| \boldsymbol{x}_n \in \boldsymbol{X},\ \sum_{n=1}^{N} \lambda_n = 1,\ \lambda_n \geq 0 \right\}. \tag{3.5}$$

In Büning (1991), convex hulls were used for detecting outliers in multidimensional data sets. For this, the convex hull of the data set is computed and all values which are on the edge of the convex hull are marked as outliers. This strategy will now be adapted for our purpose: We compute the convex hull of our data set and assign all regression vectors on the edge of the hull to the first group of vectors. Afterwards, we eliminate these regression vectors from the data set and repeat the procedure. This is done until all regression vectors are assigned to a specific group.

For calculating convex hulls in two dimensions, there exist many different algorithms (for an overview see for example Preparata and Shamos (1990)). In higher dimensions, the problem is more complex. One of the few convex hull algorithms that work also for more than two dimensions is the *Quickhull Algorithm* which was invented by Barber et al. (1996). Below, a pseudocode and visualization of the Quickhull algorithm in two dimensions is given, the generalization to more dimensions can be looked up in above mentioned article of Barber et al.

Algorithms 3.2 and 3.3 show the recursive procedure of the Quickhull algorithm in two dimensions. For better understanding the procedure of this algorithm is visualized in Figure 3.21 based on the LHS data. At first, the regression vectors with minimal and maximal values of their first component are detected. These values are added to the span of the convex hull. The line which goes through these values

Algorithm 3.2 Quickhull algorithm for computing convex hulls in two dimensions. The used procedure *FindHull* can be found in Algorithm 3.3.

Input: A (data) set $\boldsymbol{X}$ with N two-dimensional regression vectors

Output: The values which span the convex hull of $\boldsymbol{X}$

1: In $\boldsymbol{X}$, find values $\tilde{\boldsymbol{x}}_1$ with minimal value of the first component and $\tilde{\boldsymbol{x}}_2$ with maximal value of the first component. Add them to the hull.
2: Divide $\boldsymbol{X}$ in two sets $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, where $\boldsymbol{X}_1$ contains all values above the line from $\tilde{\boldsymbol{x}}_1$ to $\tilde{\boldsymbol{x}}_2$ and $\boldsymbol{X}_2$ contains all values below the line from $\tilde{\boldsymbol{x}}_1$ to $\tilde{\boldsymbol{x}}_2$.
3: FindHull($\boldsymbol{X}_1$, $\tilde{\boldsymbol{x}}_1$, $\tilde{\boldsymbol{x}}_2$)
4: FindHull($\boldsymbol{X}_2$, $\tilde{\boldsymbol{x}}_1$, $\tilde{\boldsymbol{x}}_2$)
5: **return** All values which span the convex hull of $\boldsymbol{X}$.

Algorithm 3.3 Procedure *FindHull* which is necessary for the Quickhull algorithm presented in Algorithm 3.2.

Input: A (data) set $\boldsymbol{X}$; Values $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$

Output: Values which belong to the convex hull of $\boldsymbol{X}$

1: **if** $\boldsymbol{X} = \emptyset$ **then**
2: **return** $\emptyset$
3: **end if**
4: Find the value with the maximal distance to the line from $\tilde{\boldsymbol{x}}_1$ to $\tilde{\boldsymbol{x}}_2$ and name it $\tilde{\boldsymbol{x}}_3$. Add it to the hull.
5: Delete all values inside the triangle $\tilde{\boldsymbol{x}}_1$, $\tilde{\boldsymbol{x}}_2$, $\tilde{\boldsymbol{x}}_3$ from $\boldsymbol{X}$.
6: Divide the remaining values in $\boldsymbol{X}$ in two groups $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, where $\boldsymbol{X}_1$ contains all values which are "outside" the line from $\tilde{\boldsymbol{x}}_1$ to $\tilde{\boldsymbol{x}}_3$ and $\boldsymbol{X}_2$ contains all values which are "outside" the line from $\tilde{\boldsymbol{x}}_2$ to $\tilde{\boldsymbol{x}}_3$.
7: FindHull($\boldsymbol{X}_1$, $\tilde{\boldsymbol{x}}_1$, $\tilde{\boldsymbol{x}}_3$)
8: FindHull($\boldsymbol{X}_2$, $\tilde{\boldsymbol{x}}_2$, $\tilde{\boldsymbol{x}}_3$)

divides the data set in two groups: One group with regression vectors above the line and one group with regression vectors below the line. Plot 1 of Figure 3.21 and lines 1 and 2 of Algorithm 3.2 show this. Afterwards, the procedure *FindHull* described in Algorithm 3.3 is recursively executed on both groups. For this, at first the regression vector with the farthest distance to the above mentioned line has to be found. This regression vector belongs to the span of the convex hull and builds a triangle together with the two values which were before added to the span of the convex hull. This triangle can be seen in Plot 2 of Figure 3.21 and is described in line 4 of Algorithm 3.3. All values inside the triangle obviously do not belong to the span of the convex hull and can be deleted from the data set. The regression vectors outside the triangle are split in two groups again: One group of regression vectors outside the one edge of the triangle and one group with regression vectors outside the other edge of the triangle (all design outside the third edge of the triangle are irrelevant in this step of the recursion). If no regression vectors are left outside an edge of the triangle, the recursion breaks up. In Figure 3.21 this can be seen in Plots 2 and 3. In Plot 2, there are no regression vectors left on the one edge of the triangle, so that *FindHull* has to be called recursively only on the other edge. This can be seen in Plot 3. Plots 4 to 8 of Figure 3.21 show the recursive procedure on the values below the first line. At the end the convex hull has been found.

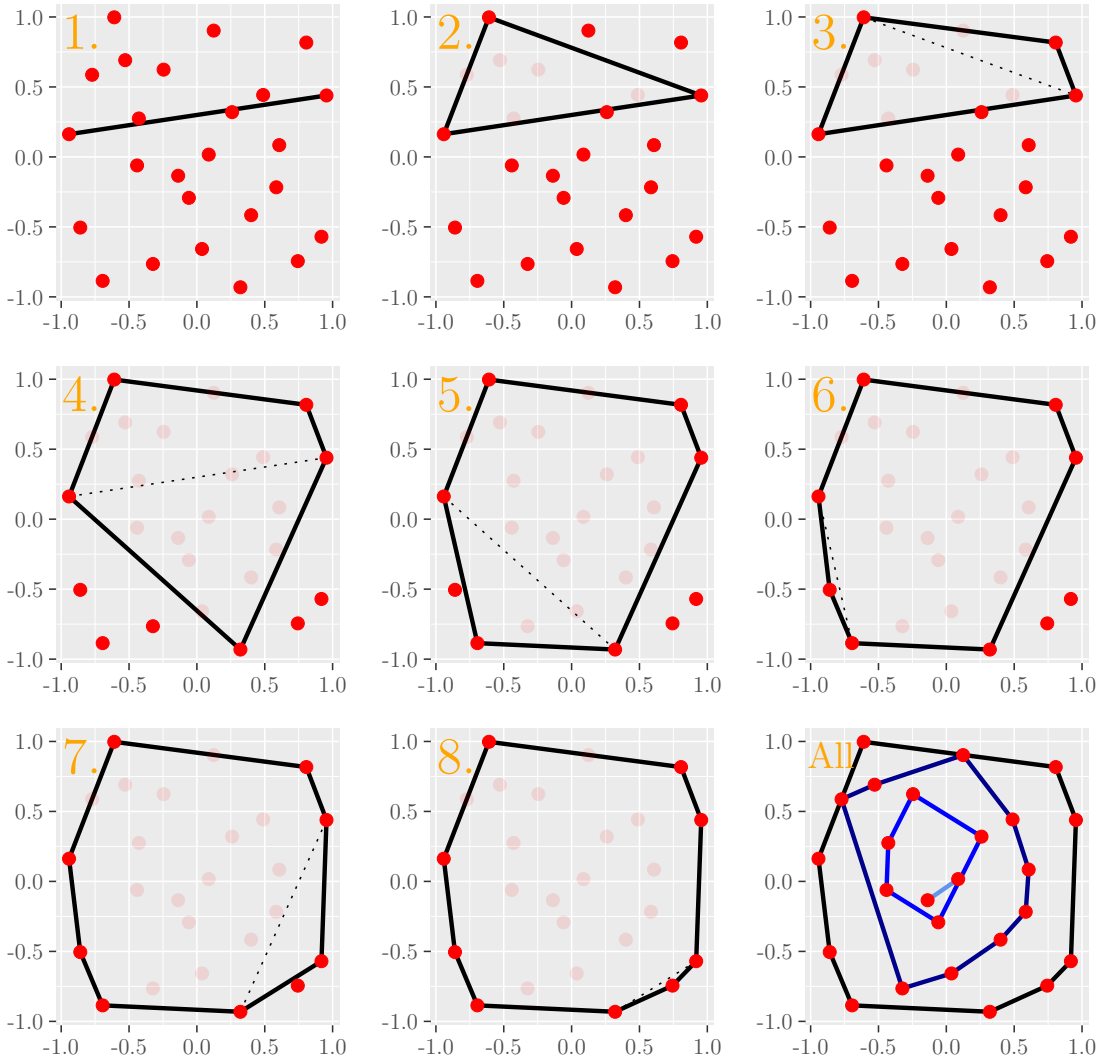


Figure 3.21: Visualization of the Quickhull algorithm based on the LHS data. Plots 1 to 8 show the procedure of the algorithm, where the black line shows the current status of the convex hull, the red points show values which are (possibly) part of the span of the convex hull and light red points show values which are inside the convex hull. The plot in the right bottom corner shows all convex hulls which can be found in the data set.

The plot in the bottom right corner of Figure 3.21 shows all convex hulls that can be found in the LHS data. These hulls define the groups on which the partial sorting is based on. In contrast to the partial sorting via nondominated sorting presented in Subsection 3.3.1, the regression vectors within every group have an inherent order along the edges of the convex hull. Unfortunately, Quickhull returns the regression vectors on the edges of the convex hull not in this inherent order. Finding the inherent

order is a Traveling Salesman Problem (TSP), which cannot be solved exactly in polynomial runtime so far (see Subsection 3.4.1), but methodologically the problem is simple. Another problem might be methodological more complex: Where do the respective groups start and end? Since a circle has no inherent start and end, one has to think about good cutpoints of the convex hulls for good transitions from one group to the next in the ordering process. In the best case, the transition from one group to another should be done where the distance between two points of the respective groups is the smallest. But this is really a hard problem since the transition from the first group (the most outer convex hull) to the second group conditions all further transitions because when the starting point in a specific hull is fixed, also the end point in this hull is fixed. Also, the question comes up whether the regression vectors in every hull should be visited left around or right around? And what does "left around" and "right around" mean for more than two dimensions? Since this problem is methodological rather complex and a small investigation of the performance of this ordering method has shown that it does not perform very well in the context of sign depth tests, no great effort was made on the solution of this problem and the most easy solution has been chosen: The regression vectors in every group are ordered according to the order the TSP-solver returns the values. This solution does not care about good transitions between the groups and it is random whether the regression vectors in every hull are visited left around or right around.

Figure 3.22 shows the obtained orders on the three data sets. It can be seen that the cutpoints between the partial ordered groups are random as well as the direction the regression vectors in every group are visited.

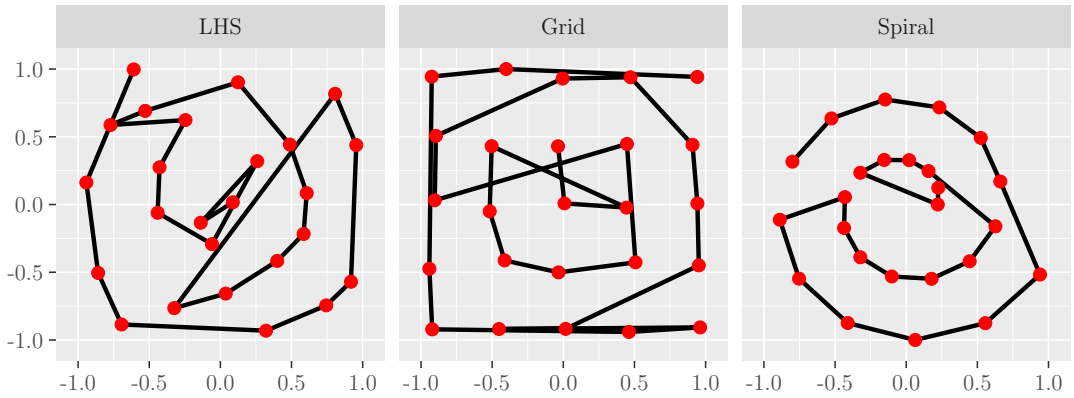


Figure 3.22: Visualization of the obtained orders when using convex hulls for partial sorting.

In contrast to the nondominated sorting, this ordering method does not need a criterion how to order the values within a partial ordered group because applying a TSP-solver to every group leads to a good order of every group. But, since the cutpoints and the direction of a tour of a TSP are arbitrary and might be even random, there may be different orders when applying this ordering method twice to the same data. Although the obtained order does not change when transforming the data additively or multiplicatively, the fact that the order might change without external influence is a big disadvantage. Furthermore, it cannot be guaranteed that the inherent order in the one-dimensional case will be obtained. Although in the one-dimensional case all values are assigned to a single hull and the TSP-solver orders the regression vectors according to their values, the cutpoint of the TSP-tour is not necessarily between the smallest and the largest value. Remembering the before mentioned disadvantage that in higher dimensions (nearly) all regression vectors are part of a single hull, this ordering method has much more disadvantages than advantages.

The time complexity of this ordering method composes of the time complexity for detecting all convex hulls in the data and finding the inherent order of the regression vectors in each hull. Barber et al. (1996) calculate the time complexity of finding a convex hull with their Quickhull algorithm when in every step of the algorithm the remaining values are split in two equal sized groups. This average time complexity is given as $\mathcal{O}(N \log(N))$ for two and three dimensions and $\mathcal{O}(N^{\lfloor D/2 \rfloor} / \lfloor D/2 \rfloor !)$ for higher dimensions. A proof of the average time complexity of the presented two dimensional case can be found in Proof C.1 starting on page 224. Also in Proof C.1 it is shown that in the two-dimensional case the time complexity degenerates to $\mathcal{O}(N^2)$ in the worst case which is given when in every step all regression vectors are assigned to one group and the other group is empty. It is hard to say how many convex hulls have to be built for ordering all regression vectors. This depends strongly on the number of dimensions and also on the number of regression vectors and the structure of the given data set. But, as it can be seen in Figure 3.17 on page 50, already for a single-digit number of dimensions (nearly) all regression vectors are assigned to a single hull, so that the number of hulls in the data is not as important as the time complexity for finding the order in the most outer hull. As it can be read in Subsection 3.4.1, the worst case time complexity for exactly solving a Traveling Salesman Problem is $\mathcal{O}(DN^2 + N^2 2^N)$, which is much larger than the time complexity for finding several convex hulls. As a conclusion it can be said that for only a little

number of dimensions the time complexity seems to be polynomial whereas the worst case time complexity in the higher dimensional case is exponential.

Figure 3.23 shows the empirical runtimes of this ordering method. It can be seen that the runtimes are much larger than the runtimes of the naive and scalarization based methods, but smaller than the empirical runtimes of the nondominated sorting (except for $D = 10$). Also, it can be seen in the upper row of Figure 3.23 that the runtimes and also the variance of the runtimes increase for larger D and that the increase in N is more than linear and maybe more like N^2 than $N \log(N)$. The lower row of 3.23 shows some strange behavior: The runtimes of the 10-dimensional case are very large, whereas the runtimes for $D = 20$ and $D = 50$ are much smaller. Moreover, the runtimes for $D = 100$ are missing. The missing runtimes can be easily explained: The Quickhull algorithm needs much memory for its calculations. For a small number of dimensions, a few gigabyte of RAM are sufficient, but for larger dimensions, calculations are not possible with home computers anymore. Some of the

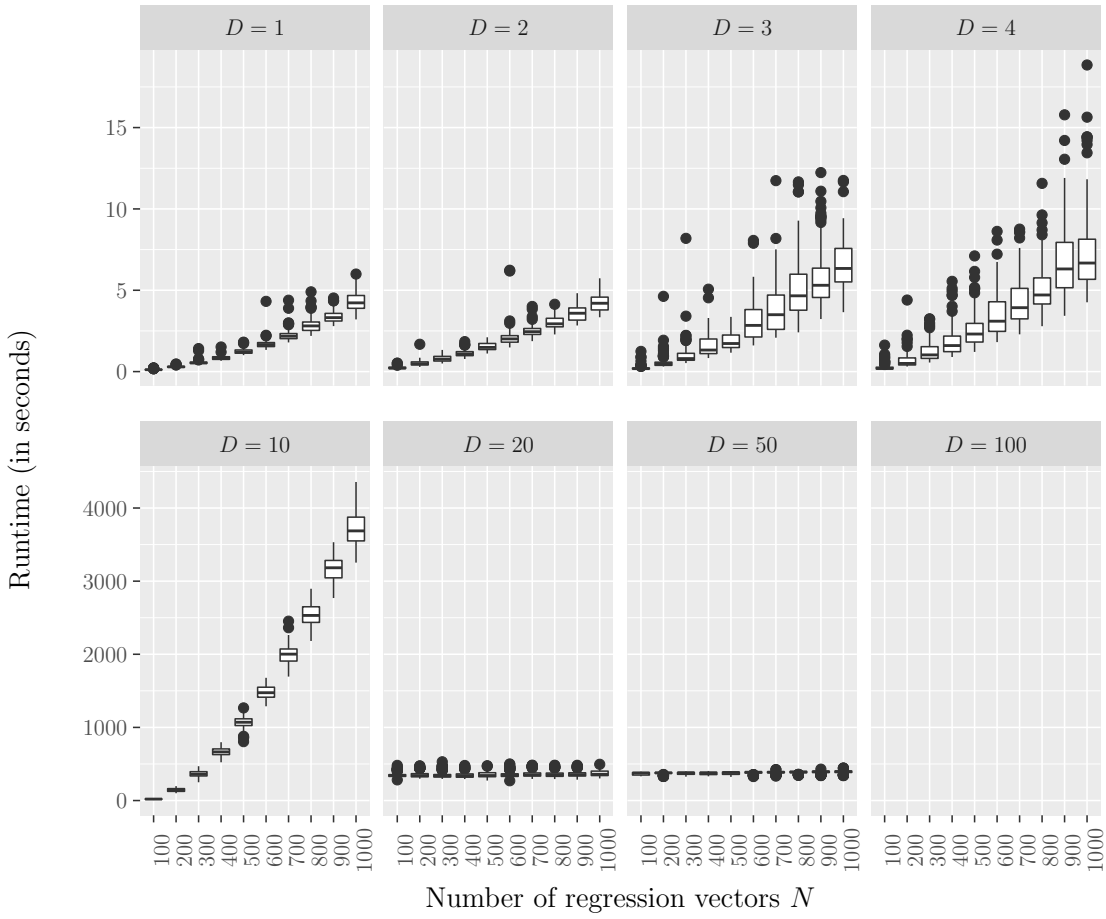


Figure 3.23: Empirical runtimes when using convex hulls for ordering.

simulation runs for $D = 10$, $D = 20$ and $D = 50$, have needed up to 64 GB RAM and for $D = 100$ not a single simulation run was possible even with 200 GB RAM. This shows that this ordering method can only be used for small numbers of dimensions (less than ten) where the size of the RAM of a normal home computer is sufficient. This is also said by the developers of the Quickhull algorithm. The documentation of their implementation *Qhull*, which is also used in this thesis, states: "For convex hulls (...), Qhull may be used for 2-D up to 8-D. (...) In higher dimensions, the size of the output grows rapidly and Qhull does not work well with virtual memory."[2] Although no final explanation for the decrease of the runtimes for $D = 20$ and $D = 50$ in contrast to $D = 10$ could be found, the fact that the Quickhull algorithm is optimized only up to eight dimensions, may be responsible for the strange behavior of the runtimes. It is hard to say whether the obtained results in the high dimensional cases are sensible and correct, but since Figure 3.17 on page 50 shows that in high dimensions (nearly) always all regression vectors are assigned to a single convex hull, it may be correct, that the runtimes decrease because the Quickhull algorithm has to be applied only once to the data to find all convex hulls.

Overall, this ordering method has many disadvantages and large runtimes. In contrast to the nondominated sorting method, it has no further hyper-parameters which have to be chosen and it has smaller empirical runtimes in most cases, but the partial sorting leads more quickly to only one group of incomparable values, see Figure 3.17. Furthermore, in high dimensions, this ordering method needs a lot of memory for its computation. This makes this ordering method in many cases unusable.

3.3.3 Partial Sorting via Tukey's Halfspace Depth

Another approach for partial sorting multidimensional data is to use data depths. Its idea is to determine how "deep" a specific value is in the data set, so that regression vectors "in the center" of the data set get the largest values whereas regression vectors "on the edge" of the data set get small values, see also Section 2.4 for this. Tukey (1975) invented the so-called *halfspace depth* or *location depth* for getting a robust location estimator by finding the value(s) in the data set with the maximal depth value. The halfspace depth of a specific regression vector $\boldsymbol{x}_n$ describes the minimal number of values which are in any closed halfspace with boundary hyperplane through $\boldsymbol{x}_n$. Optionally, this number is divided by N to obtain the minimal fraction

[2] `http://www.qhull.org/html/index.htm#when`, accessed on July 22, 2020.

of values in any halfspace. Formally written, the halfspace depth value of a regression vector $\boldsymbol{x}_n$ relative to a data set $\boldsymbol{X}$ can be calculated via (see for example Struyf and Rousseeuw (1999))

$$hd(\boldsymbol{x}_n, \boldsymbol{X}) = \frac{1}{N} \min_{||\boldsymbol{u}||=1} \#\{i : \boldsymbol{u}^\top \boldsymbol{x}_i \leq \boldsymbol{u}^\top \boldsymbol{x}_n\}, \tag{3.6}$$

where $\boldsymbol{u} \in \mathbb{R}^D$. The halfspace depth has some relations to convex hulls. One can easily prove that all regression vectors on the span of the convex hull of the data set have a halfspace depth value of $\frac{1}{N}$, since there always exist halfspaces where only the respective value itself is in.

For ordering multidimensional data the halfspace depth can be used by using the depth values for a partial sorting. All regression vectors with the same value of the halfspace depth have to be ordered afterwards. Since this ordering method has many similarities to an ordering according to convex hulls (see Subsection 3.3.2), it makes sense to use the same method for ordering regression vectors with the same values of the halfspace depth as for convex hulls. Because of this, a TSP-solver is applied to every group of values. As in the previous subsection, it is not cared about good transitions from one group to the next because this would be a hard optimization problem which it is not worth to be solved because the performance of this ordering method in the context of sign depth tests is much worse than the performance of some other ordering methods.

Figure 3.24 shows the obtained orders of this ordering method for the three data sets. It can be seen that the regression vectors on the convex hull of the data sets are first

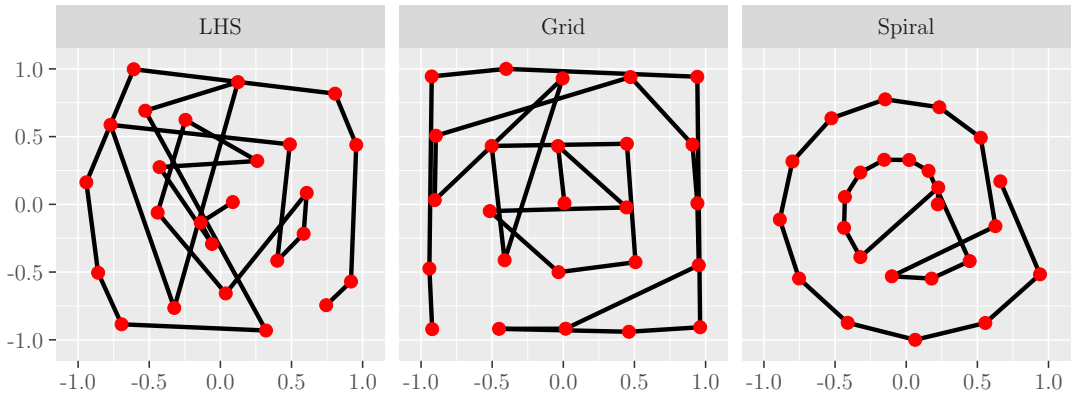


Figure 3.24: Visualization of the obtained orders when using Tukey's halfspace depth for partial sorting.

in the ordering, but afterwards the ordering is much more messy than the ordering of the convex hull method. This is because one can easily prove that the regression vectors on all following convex hulls do not necessarily have the same values of the halfspace depth.

Since this ordering method is very similar to an ordering according to convex hulls, it has similar advantages and disadvantages. The data can be additively and multiplicatively transformed without changing the obtained order, but the inherent order in the one-dimensional case is not preserved, since the regression vectors with minimal value of the halfspace depth are the regression vectors with minimal and maximal value. Furthermore, the before mentioned disadvantage of partial sorting in high dimensions applies also to this ordering method, see Figure 3.17 on page 50.

The time complexity of this ordering method depends on some details. For a given vector $\boldsymbol{u} \in \mathbb{R}^D$ and a fixed regression vector $\boldsymbol{x}_n$, determining how many regression vectors are in the respective halfspace has time complexity $\mathcal{O}(DN)$ because calculations with N D-dimensional vectors have to be carried out. Doing this for all N regression vectors leads to a time complexity of $\mathcal{O}(DN^2)$. For an exact calculation of the halfspace depth $\binom{N}{D}$ different vectors $\boldsymbol{u}$ have to be looked at. This would lead to an overall time complexity of determining all halfspace depths of $\mathcal{O}(\binom{N}{D}DN^2)$. Since $\binom{N}{D}$ gets too large very quickly for a computation in acceptable runtime, in this thesis an approximate version of the halfspace depth invented by Cuesta-Albertos and Nieto-Reyes (2008) is computed. The so-called *Random Tukey Depth* determines the halfspace depth as the minimum univariate halfspace depth of the data, which is projected on lines in a fixed number of directions. Here, 100 000 directions were chosen, so that the factor $\binom{N}{D}$ of the time complexity decreases to a (large) constant, what leads to an overall time complexity of computing all approximate halfspace depths of $\mathcal{O}(DN^2)$. In addition to the time complexity of computing all halfspace depths, as in the previous subsection, the time complexity of computing the solution of the Traveling Salesman Problem is needed, which is $\mathcal{O}(DN^2 + N^2 2^N)$ in the worst case. So, the overall time complexity of this ordering method is $\mathcal{O}(DN^2 + DN^2 + N^2 2^N) \in \mathcal{O}(DN^2 + N^2 2^N)$.

Figure 3.25 shows the empirical runtimes of this ordering method. It can be seen that the runtimes increase quadratically in the number of regression vectors N. Also a small increase in the number of dimensions D is visible. The exponential runtime of computing the TSP seems to be empirical negligible. In Figure 3.25, the runtimes for $N = D = 100$ are missing because the used implementation for computing the

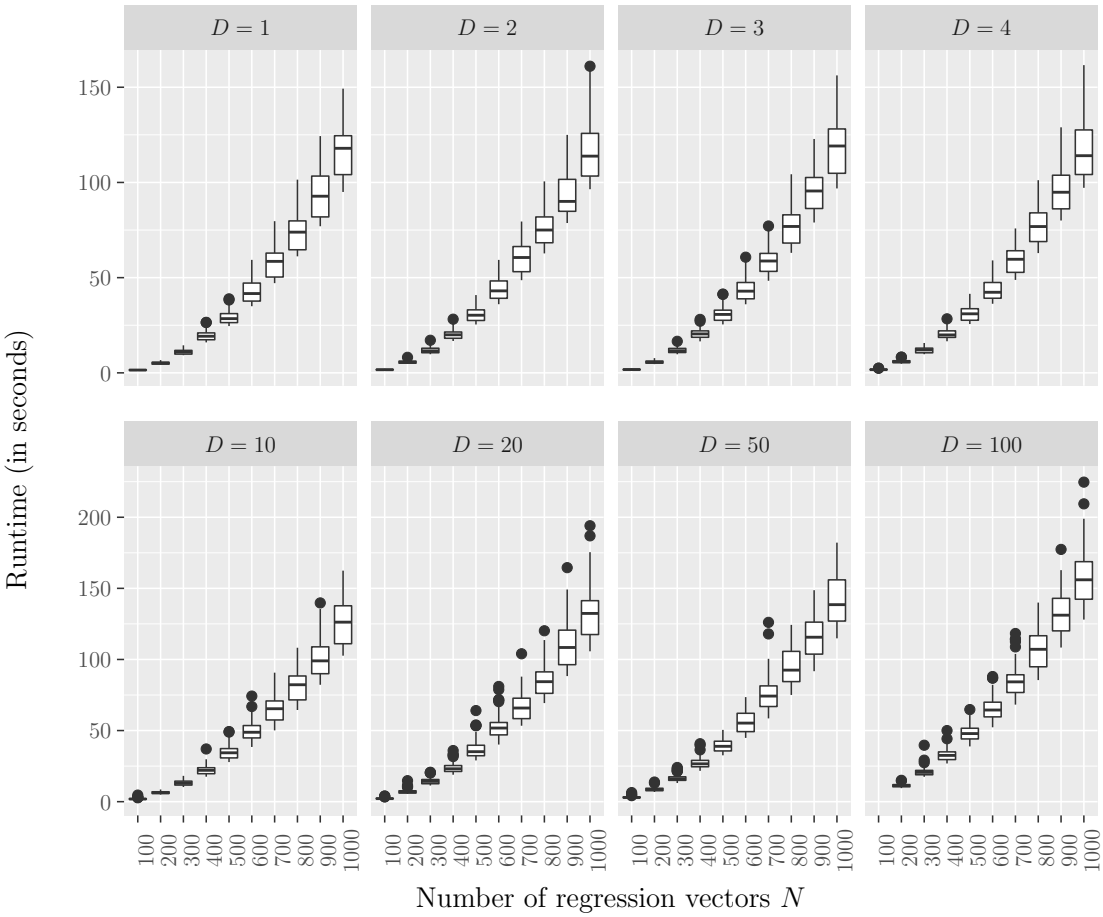


Figure 3.25: Empirical runtimes when using Tukey's halfspace depth for ordering.

halfspace depth requires that $N > D$. In comparison to the similar ordering method via convex hulls, this ordering method is much slower for $D \leq 4$ and much quicker for $D \geq 10$ (see also Table C.1 in the appendix for comparison of the runtimes).

Overall, this ordering method has large runtimes and several further disadvantages. Its inherent order in the one-dimensional case is not preserved and when the number of dimensions gets large, (nearly) all regression vectors have the same value of the halfspace depth. Although its runtimes are faster for $D \geq 10$ than the runtimes of the two other partial sorting methods, in comparison to the other ordering methods, this method has no real benefit.

3.4 Distance Based Methods

An intuitive idea when ordering multidimensional data may be that values which are similar shall be near to each other in the ordering. Here, similar means that the distance between the values is small. Consequently, values which have a large distance to each other shall not get ordered one after the other.

This section describes three different distance based ordering methods even though one of the three presented methods is only an approximation of another presented ordering method. Since all three methods base on the pairwise distances of the regression vectors, the actual values of the regression vectors do not matter for the ordering anymore. This leads to the fact that not only pure metric regression vectors can be ordered, but also regression vectors which have ordinal or even nominal components, since distance measures also exist for these values and also for mixtures of metric, ordinal and nominal regression vectors. For example, for this the so-called *Gower's distance* can be used which bases on Gower's coefficient (Gower, 1971). Furthermore, all three presented methods allow to transform the regression vectors additively and multiplicatively without changing the order, since additive transformations do not change the pairwise distances at all and multiplicative transformations preserve the proportion of the pairwise distances.

All three presented methods need to calculate a distance matrix of pairwise distances. Because of this, the time complexity for computing this distance matrix is given at this point. Although we are working nearly always with the euclidean distance, also nearly all other distance measures need to look at every component of both regression vectors once for calculating the distance between two regression vectors. This leads to a time complexity of $\mathcal{O}(D)$ for computing the distance between two given regression vectors of dimension D. Overall, $\frac{N(N-1)}{2}$ distances have to be computed because the needed distance matrix has N^2 entries of which N entries are zero (the N distances of a regression vector to itself) and all other entries occur twice because distance measures are symmetric. This leads to an overall time complexity of $\mathcal{O}(DN^2)$ for calculating the needed distance matrix.

3.4.1 Ordering on the Basis of the Exact Solution of the Shortest Path Problem

The idea that regression vectors which have a small distance to each other shall be also near to each other in the ordering leads to the problem of finding a path through all regression vectors with minimal path length. If starting point and end point would be the same, the problem is well known in computer science as *Traveling Salesman Problem* (TSP). Here, starting point and end point should not be the same. This problem is known as *Shortest Hamiltonian Path Problem* (SHP) and is a quite simple variation of the TSP.

For the TSP, several definitions and formulations of the problem exist. One popular possible formulation was invented by Dantzig et al. (1954). They represented the TSP as the solution of a linear programming problem:

Definition 3.1. *Let* $\boldsymbol{V} \in \{0, 1\}^{N \times N}$ *with entries*

$$v_{mn} = \begin{cases} 1, & \textit{the path goes from regression vector } \boldsymbol{x}_m \textit{ to regression vector } \boldsymbol{x}_n \\ 0, & \textit{otherwise} \end{cases}.$$

Let $c_{mn} \in \mathbb{R}_0^+$ *be the fixed distance between regression vector* $\boldsymbol{x}_m$ *and* $\boldsymbol{x}_n$*. Then the solution of the TSP is*

$$\min_{\boldsymbol{V}} \sum_{m=1}^{N} \sum_{n=1, m \neq n}^{N} v_{mn} c_{mn}$$

under the conditions

1. $\sum_{m=1, m \neq n}^{N} v_{mn} = 1$ *(every regression vector has exactly one predecessor),*
2. $\sum_{n=1, m \neq n}^{N} v_{mn} = 1$ *(every regression vector has exactly one successor),*
3. $\sum_{m \in Q} \sum_{n \in Q, m \neq n} v_{mn} \leq |Q| - 1 \; \forall Q \subsetneq \{1, \ldots N\}, \; |Q| \geq 2$ *(ensures that there exist no subtours, but only a single tour with all regression vectors is computed).*

The TSP can now be transformed to a SHP when adding another regression vector to the data which has distance zero to all other regression vectors, i.e. $c_{mn} = 0$ if $m = N + 1$ or $n = N + 1$, see for example Applegate et al. (2006). This "dummy value" allows the solver to go once from an arbitrary regression vector to another with zero distance when visiting the dummy value in-between. When cutting the obtained tour at the dummy value, one has obtained the Shortest Hamiltonian Path.

Many different algorithms for solving the TSP (and SHP) exist, for example dynamic programming approaches, branch-and-bound approaches and branch-and-cut approaches. A good overview of many different approaches can be found in Applegate et al. (2006). The state-of-the-art solver for the TSP is called *Concorde* (Applegate et al., 2004), a branch-and-cut algorithm, which is also described in above mentioned book.

For our purpose, taking the solution of the SHP as ordering criterion has many advantages. Not only the before mentioned fact that ordinal and nominal components in the regression vectors are allowed, but also the fact that in the one-dimensional case the inherent order is preserved. Furthermore, the idea of this ordering method is easy to understand and the ordering method does not depend on further hyperparameters which affect the solution of the ordering process (except of the choice of the distance measure, but in this thesis this is said to be always the euclidean distance). In addition, additive and multiplicative transformations of metric data can be done without changing the result of the ordering process.

Figure 3.26 shows the obtained orders on the three data sets. It can be seen that the inherent order of the Spiral-data is preserved and in general, that there is no ”disorder” in the data.

A big disadvantage of this ordering method is its time complexity. Imagine the most simple solving algorithm for a TSP which is trying all possible tours through the data and selecting the shortest one. This combinatorial problem has $N!$ possible solutions (basically it is the enumeration of all possible permutations of the numbers

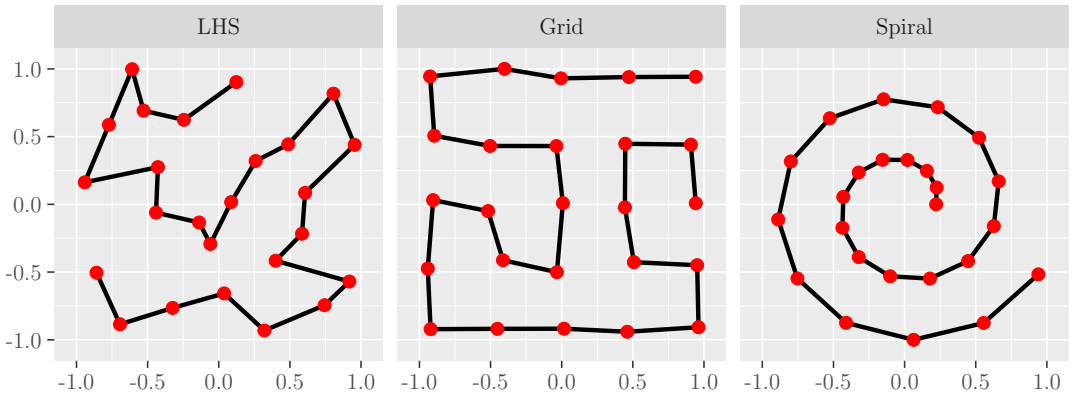


Figure 3.26: Visualization of the obtained orders when using a Shortest Hamiltonian Path for ordering.

1 to N). A time complexity of $\mathcal{O}(N!)$ is not practical, since it gets very large already at very small N. For example, for trying all possible combinations of one of our data sets with $N = 25$ data points more than $1.5 \cdot 10^{25}$ iterations would be necessary. In general, the TSP belongs to the *NP-hard problems*, which means that it is no algorithm known which solves the problem exactly in polynomial time complexity. The smallest known worst case time complexity for solving the TSP is $\mathcal{O}(N^2 2^N)$ which was reached for a dynamic programming approach presented by Held and Karp (1962). Although the time complexity of the Concorde-solver apparently has never been analyzed, here it is assumed that its time complexity is not larger than $\mathcal{O}(N^2 2^N)$ because it is the only solver so far which could solve a TSP-problem with 85 900 values[3] (Applegate et al., 2006). Combining the time complexity for creating the necessary distance matrix and the time complexity of solving a TSP with $N + 1$

[3] `http://www.math.uwaterloo.ca/tsp/pla85900/index.html`

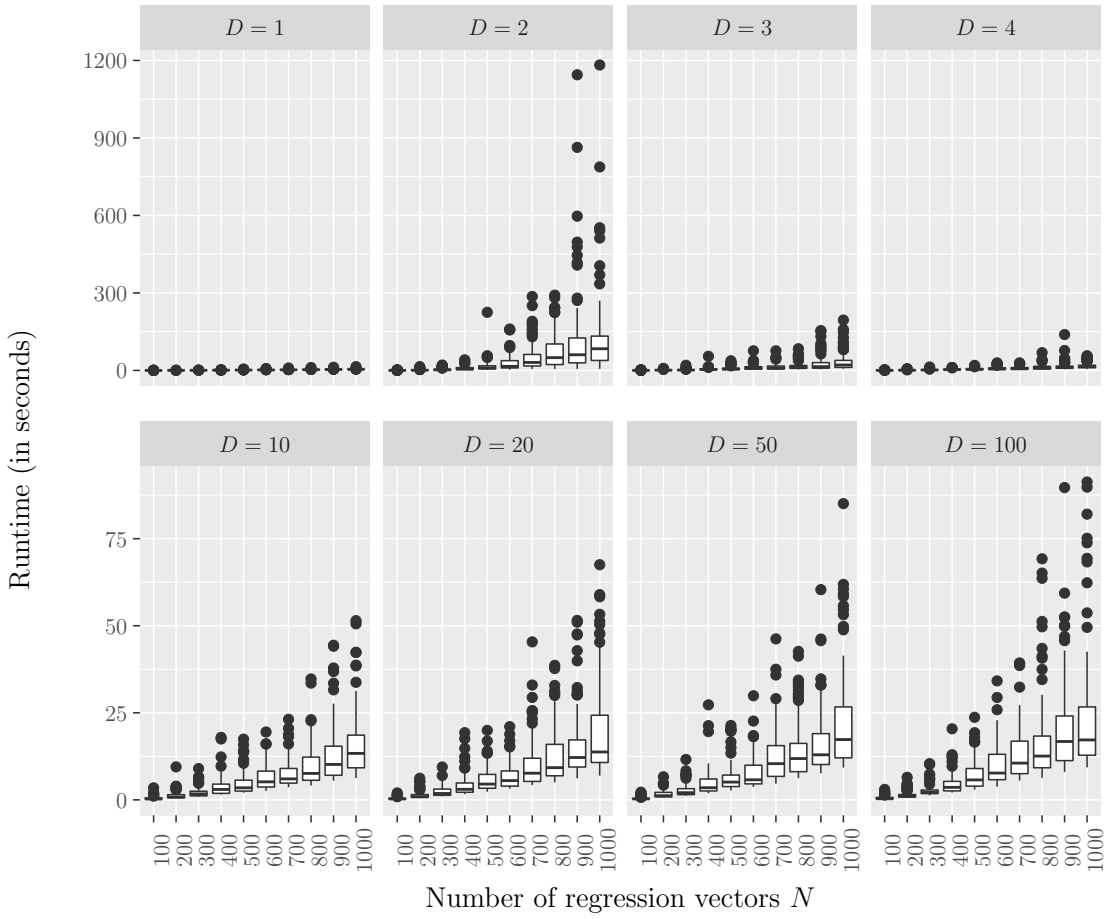


Figure 3.27: Empirical runtimes when using the Shortest Hamiltonian Path for ordering.

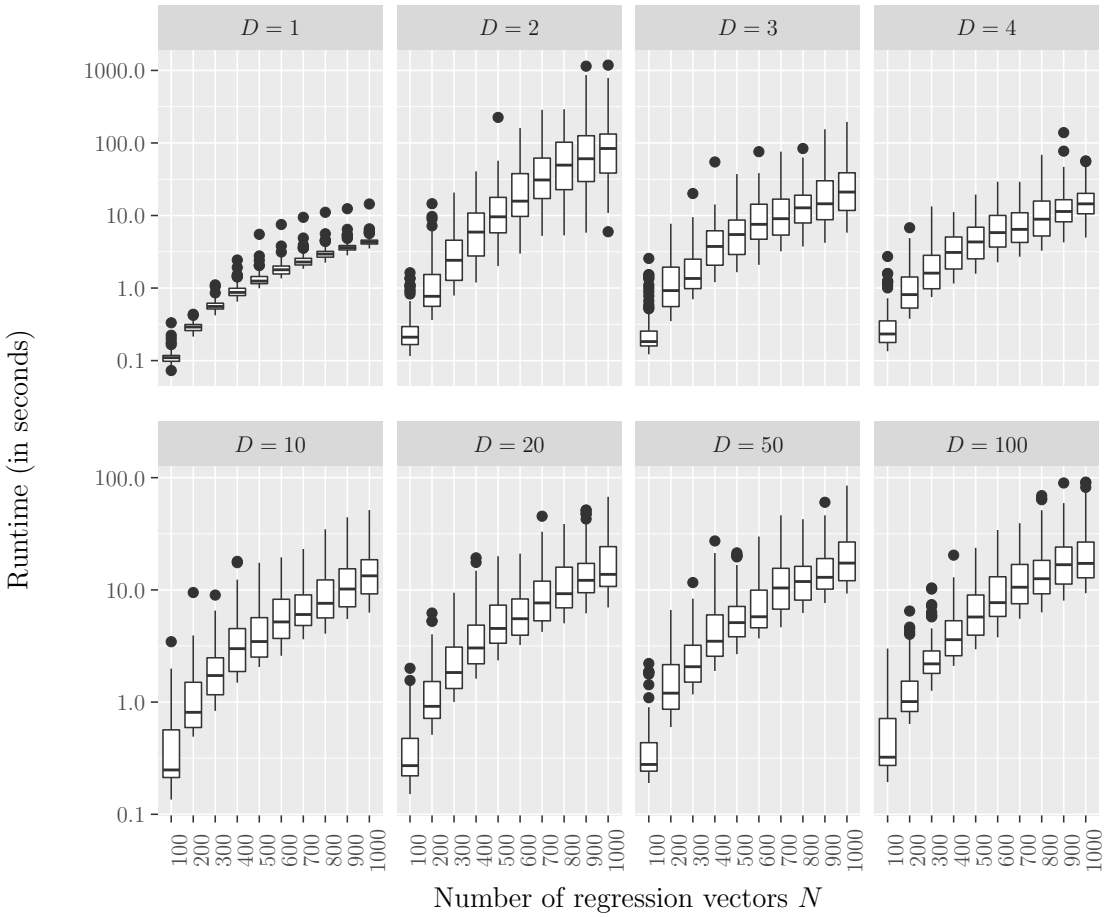


Figure 3.28: Empirical runtimes when using the Shortest Hamiltonian Path for ordering. Here, the values are plotted on a logarithmic scale.

regression vectors (N real regression vectors and one dummy value), computing the SHP has a time complexity of $\mathcal{O}(DN^2 + (N+1)^2 2^{N+1}) \in \mathcal{O}(N^2(D + 2^N))$.

Figure 3.27 shows the empirical runtimes of this ordering method. Because the runtimes have different magnitudes for different values of D, the same runtimes are also plotted on a logarithmic scale for better visibility in Figure 3.28. Looking at these plots, two things are remarkable: Firstly, the runtimes are much larger than the runtimes of most other so far presented ordering methods (except the runtimes of the ordering according to a nondominated sorting and according to convex hulls). Secondly, strangely the runtimes in the two-dimensional case are much larger than the other runtimes and also the runtimes of the four-dimensional case are smaller than the runtimes of the three-dimensional case. So far, no real explanation for this behavior has been found. Since the Concorde-solver is called with the before calculated distance matrix and not with the regression vectors itself, the structure

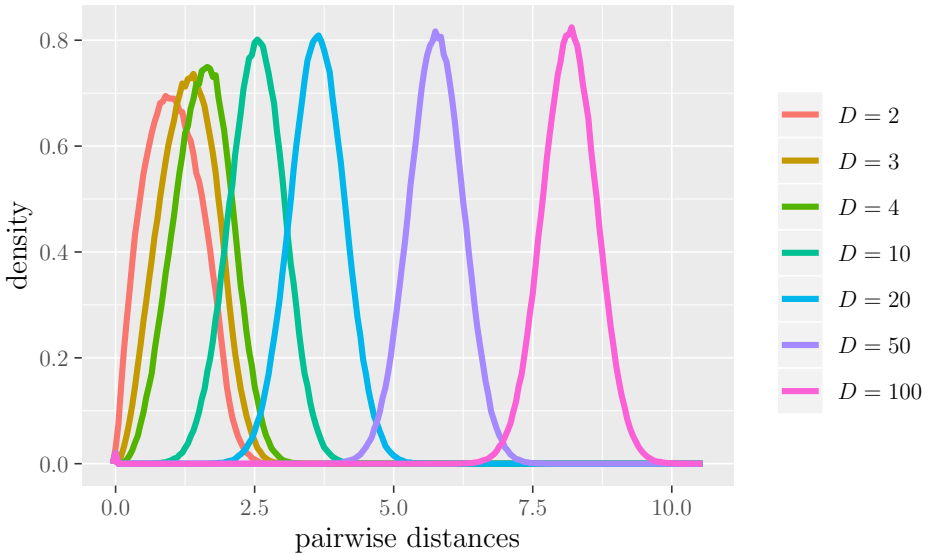


Figure 3.29: Estimated densities of the values in the distance matrices when using the euclidean distance for 1 000 uniformly sampled values in $[-1, 1]^D$.

and characteristics of the values in the distance matrix have to be responsible for this behavior. When looking at the characteristics of the pairwise distances in Figure 3.29, it can be seen that the values of the euclidean distance get larger in higher dimensions, which is not surprising since the euclidean distance of two regression vectors $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ is calculated via $d(\boldsymbol{x}_m, \boldsymbol{x}_n) = \sqrt{\sum_{d=1}^{D}(x_{md} - x_{nd})^2}$ and so the maximal distance of values in the range $[-1, 1]^D$ is $2 \cdot \sqrt{D}$ which is increasing in D. Since the minimal distance is zero independent from the number of dimensions, the range of the distance values gets larger when D is increasing. Because of this, the presumption is that the branch-and-cut algorithm implemented in Concorde has to look at less possible orderings because it can cut many branches with nonsensical solutions very quickly when the pairwise distances have a larger range.

However, the runtimes of this ordering method are quite large, especially in the two-dimensional case. But this seems to be the only disadvantage of this ordering method. Besides the runtime, ordering the regression vectors according to a Shortest Hamiltonian Path has many advantages like the easy understanding of this method or the possibility to have ordinal and nominal components in the regression vectors.

3.4.2 Ordering on the Basis of an Approximate Solution of the Shortest Path Problem

Since the only obvious disadvantage of an ordering according to the solution of a Shortest Hamiltonian Path is its runtime, one can try to decrease the runtime by using approximate solutions of the Shortest Hamiltonian Path Problem. Many different approximations of the TSP (and SHP) exist, see for example Laporte (1992). Here, we focus on the *nearest neighbor approach* because it is one of the most intuitive heuristics and many theoretical aspects of this algorithm are known like the maximal deviation from the optimal solution.

Although the nearest neighbor heuristic for computing a SHP can be represented as the nearest neighbor heuristic for computing a TSP with an additional dummy value, Algorithm 3.4 shows a more intuitive version of the nearest neighbor algorithm for computing a SHP. The characteristics and time complexity of the presented version and the version with dummy value are the same. The idea behind this heuristic is quite easy: it is started at an arbitrary regression vector and from this regression vector on, a tour through the data set is computed by always going to the nearest (i.e. smallest distance) not so far visited regression vector until all regression vectors are visited once. This procedure is done N times with each regression vector as starting vector once and the tour with shortest tour length is chosen at the end. This method is described in Algorithm 3.4.

Figure 3.30 shows the obtained orders when using the nearest neighbor approximation for ordering. It can be seen that the data seems to be "ordered" although there are some crossings in the orders which is easy to prove cannot occur when computing the exact solution of the SHP. But the approximation also preserves the inherent order of the Spiral data. Overall, it can be proven that the tour length of the nearest neighbor approximation is at most longer than the optimal tour by a factor of $\frac{1}{2}\lceil \log_2(N) \rceil + \frac{1}{2}$ (Rosenkrantz et al., 1977).

The time complexity of the nearest neighbor heuristic is quite easy to determine when looking at Algorithm 3.4. As mentioned before, calculating the distance matrix in line 1 has time complexity $\mathcal{O}(DN^2)$. Finding the successor of a specific regression vector in line 8 needs to look at a specific row (or column) of the $N \times N$ distance matrix, which has consequently time complexity $\mathcal{O}(N)$. All other lines within the while-loop have constant time complexity. The while-loop itself makes

Algorithm 3.4 Nearest neighbor heuristic for computing a Shortest Hamiltonian Path.

Input: A data set $\boldsymbol{X}$, which consists of N regression vectors

Output: Ordered data set on the basis of the nearest neighbor heuristic

1: Compute the distance matrix with the pairwise distances of all N regression vectors.
2: $opt\_tour\_length := \infty$
3: **for all** $\boldsymbol{x}_n \in \boldsymbol{X}$ **do**
4: $\quad \boldsymbol{S} = \boldsymbol{X} \setminus \{\boldsymbol{x}_n\}$
5: $\quad \tilde{\boldsymbol{x}} = \boldsymbol{x}_n$
6: $\quad tour\_length = 0$
7: $\quad$ **while** $\boldsymbol{S} \neq \emptyset$ **do**
8: $\quad\quad$ Find regression vector $\tilde{\boldsymbol{x}}_{tmp}$ in $\boldsymbol{S}$ which has minimal distance to $\tilde{\boldsymbol{x}}$.
9: $\quad\quad tour\_length \; = tour\_length +$ distance from $\tilde{\boldsymbol{x}}$ to $\tilde{\boldsymbol{x}}_{tmp}$
10: $\quad\quad \boldsymbol{S} = \boldsymbol{S} \setminus \{\tilde{\boldsymbol{x}}\}$
11: $\quad\quad \tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}_{tmp}$
12: $\quad$ **end while**
13: $\quad$ **if** $tour\_length \; < opt\_tour\_length$ **then**
14: $\quad\quad opt\_tour\_length = tour\_length$
15: $\quad\quad$ Cache the tour with the current optimal tour length.
16: $\quad$ **end if**
17: **end for**
18: **return** Tour (i.e. ordered data) with minimal tour length

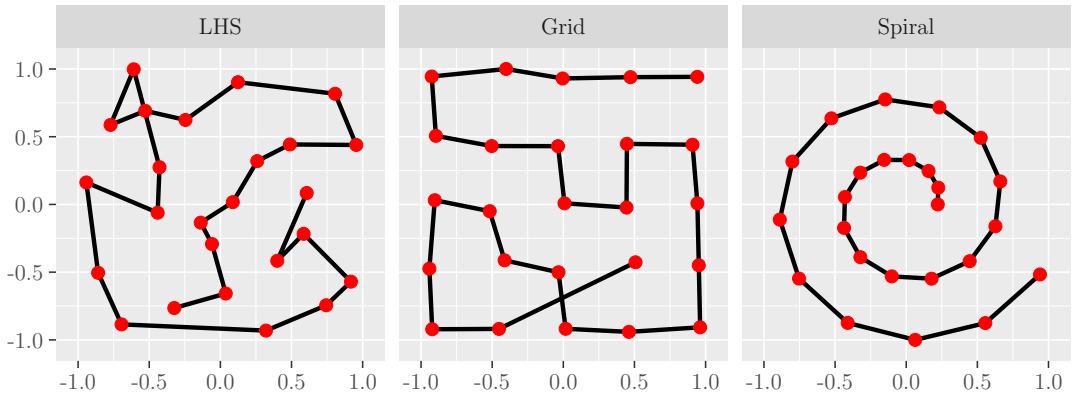


Figure 3.30: Visualization of the obtained orders when using a nearest neighbor approximation of the SHP for ordering.

$N-1$ iterations within every iteration of the outer for-loop because $\boldsymbol{S}$ has $N-1$ entries at the beginning and is reduced by one element in every iteration. Since the outer for-loop obviously has to made N iterations and except the while-loop all lines within in the for-loop have constant time complexity, the time complexity of lines 3 to 17 of Algorithm 3.4 is $\mathcal{O}(N \cdot (N-1) \cdot N) \in \mathcal{O}(N^3)$. Overall, the time complexity of the nearest neighbor approximation is $\mathcal{O}(DN^2 + N^3)$.

Since the empirical runtime of calculating a distance matrix is rather small, in Figure 3.31 almost exclusively the cubic increase of the loop can be seen. But this cubic increase can be seen very well and is independent from the number of dimensions D. Overall, the runtimes are quite large. Computing the nearest neighbor approximation for 1 000 regression vectors can take up to approximately two minutes. Although the variability of the runtimes is not as large as the variability of the runtimes of the exact solution of the SHP, it can be seen that the runtimes have the same magnitude. Rather, the median runtimes of the approximation are larger

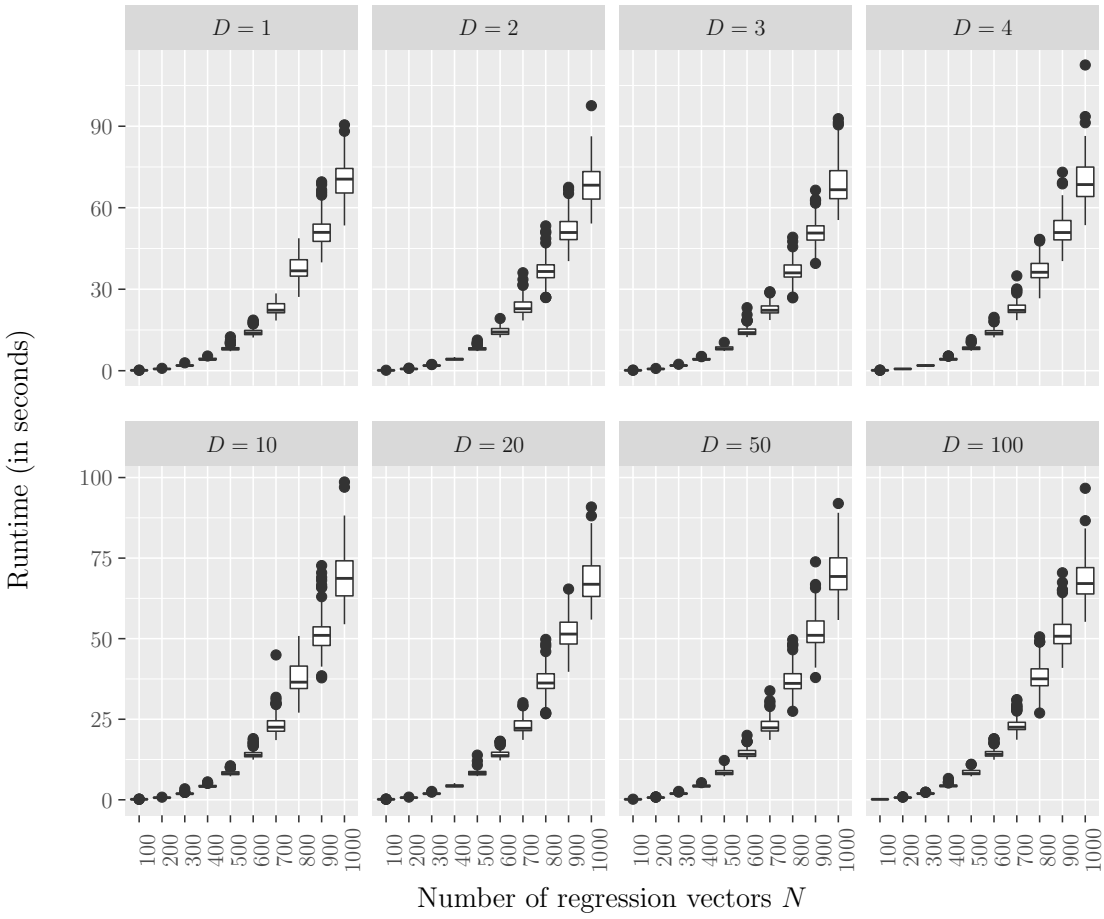


Figure 3.31: Empirical runtimes when using the nearest neighbor heuristic for ordering.

than the median runtimes of the exact solution for three and more dimensions, see Table C.1 on page 221 and following. This is a clear disadvantage compared with computing the exact solution. For sure, the runtime of the exact solver will increase more quickly than the runtime of the approximate solver and so the exact solver will need longer runtimes when N gets very large, but as it can be seen, for several hundreds of regression vectors, there is no need to take the approximate solver.

Overall, this ordering method has the same advantages as the exact solver described in the previous subsection: it is easy to understand, it does not depend on hyper-parameters (when choosing always the euclidean distance as distance measure), it is able to deal with ordinal and nominal components in the regression vectors, it preserves the inherent order in the one-dimensional case and the regression vectors can be transformed additively and multiplicatively without changing the result of the ordering process. Furthermore, this method can be easily self-implemented, whereas computing the exact solution in acceptable runtime needs specialized solver which are sometimes not easy to connect to the used software. But if the exact Concorde-solver is useable, it should be used when having only several hundreds of regression vectors, since its empirical median runtime is smaller than the empirical median runtime of the approximate solver.

3.4.3 Ordering on the Basis of a Hierarchical Clustering

The last presented ordering method is an ordering on the basis of a hierarchical clustering. Actually, hierarchical clustering is a method for unsupervised machine learning, but we can adapt this method for our purpose of ordering multidimensional data. A common way of visualizing the result of a hierarchical clustering of a given data set is to plot a dendrogram, which shows the single steps of the clustering process as a tree. The leafs of the tree are the single regression vectors (i.e. clusters of size one) and the order the dendrogram uses for arranging the regression vectors can be used for our purpose.

At first, the hierarchical clustering itself is described, as it is described for example in Hastie et al. (2009). The clustering process depends on two things: The chosen distance measure for calculating the pairwise distances and a dissimilarity measure for calculating the dissimilarity of two clusters. Since in this thesis always the euclidean distance is used, we will not focus on the distance measure any longer. For computing

the dissimilarity of two clusters, several measures exist. Here, three of them (the *complete linkage* (CL), the *average linkage* (AL) and the *single linkage* (SL)) are described. Let $d(\boldsymbol{x}_m, \boldsymbol{x}_n)$ be the pairwise euclidean distance of regression vector $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$, $m, n \in \{1, \ldots, N\}$ and $\boldsymbol{G}$ and $\boldsymbol{H}$ two clusters of regression vectors. Then the dissimilarity of the clusters $\boldsymbol{G}$ and $\boldsymbol{H}$ is given via

$$\begin{aligned} u_{CL}(\boldsymbol{G}, \boldsymbol{H}) &:= \max_{x_m \in \boldsymbol{G}, x_n \in \boldsymbol{H}} d(\boldsymbol{x}_m, \boldsymbol{x}_n), \\ u_{AL}(\boldsymbol{G}, \boldsymbol{H}) &:= \frac{1}{|\boldsymbol{G}| \cdot |\boldsymbol{H}|} \sum_{x_m \in \boldsymbol{G}} \sum_{x_n \in \boldsymbol{H}} d(\boldsymbol{x}_m, \boldsymbol{x}_n), \\ u_{SL}(\boldsymbol{G}, \boldsymbol{H}) &:= \min_{x_m \in \boldsymbol{G}, x_n \in \boldsymbol{H}} d(\boldsymbol{x}_m, \boldsymbol{x}_n). \end{aligned} \tag{3.7}$$

One approach for hierarchical clustering, called *agglomerative clustering*, is a bottom-up procedure, which means that it starts with every regression vector being an own cluster and in every step the two clusters with the smallest dissimilarity get merged until only one cluster of all regression vectors is left. This approach is formalized in Algorithm 3.5.

Algorithm 3.5 Agglomerative hierarchical clustering.

Input: Data set $\boldsymbol{X}$ with N regression vectors; dissimilarity measure u

Output: Result of a hierarchical clustering

1: Compute the distance matrix with entries $d(\boldsymbol{x}_m, \boldsymbol{x}_n)$, $m, n \in \{1, \ldots, N\}$.
2: Initialize N clusters with one regression vector each.
3: **while** # clusters > 1 **do**
4: Find the two clusters with smallest dissimilarity u and merge them to a single cluster.
5: **end while**
6: **return** All cluster partitions from all N iterations.

A visualization of this clustering procedure can be found in Figure 3.32. There, the clustering is shown on the LHS data set with complete linkage. The regression vectors are displayed as numbers which represent in which iteration the respective regression vector has been used for clustering the last time. It can be seen that until iteration 7 the clustering exclusively merges two regression vectors in every step. In iteration 8, the first time a cluster with three regression vectors occur. This cluster consists of the two regression vectors which were merged in iteration 2 already and a third regression vector. This process goes on and the single clusters get larger and larger until in iteration 25 the two remaining clusters get merged to a single one.

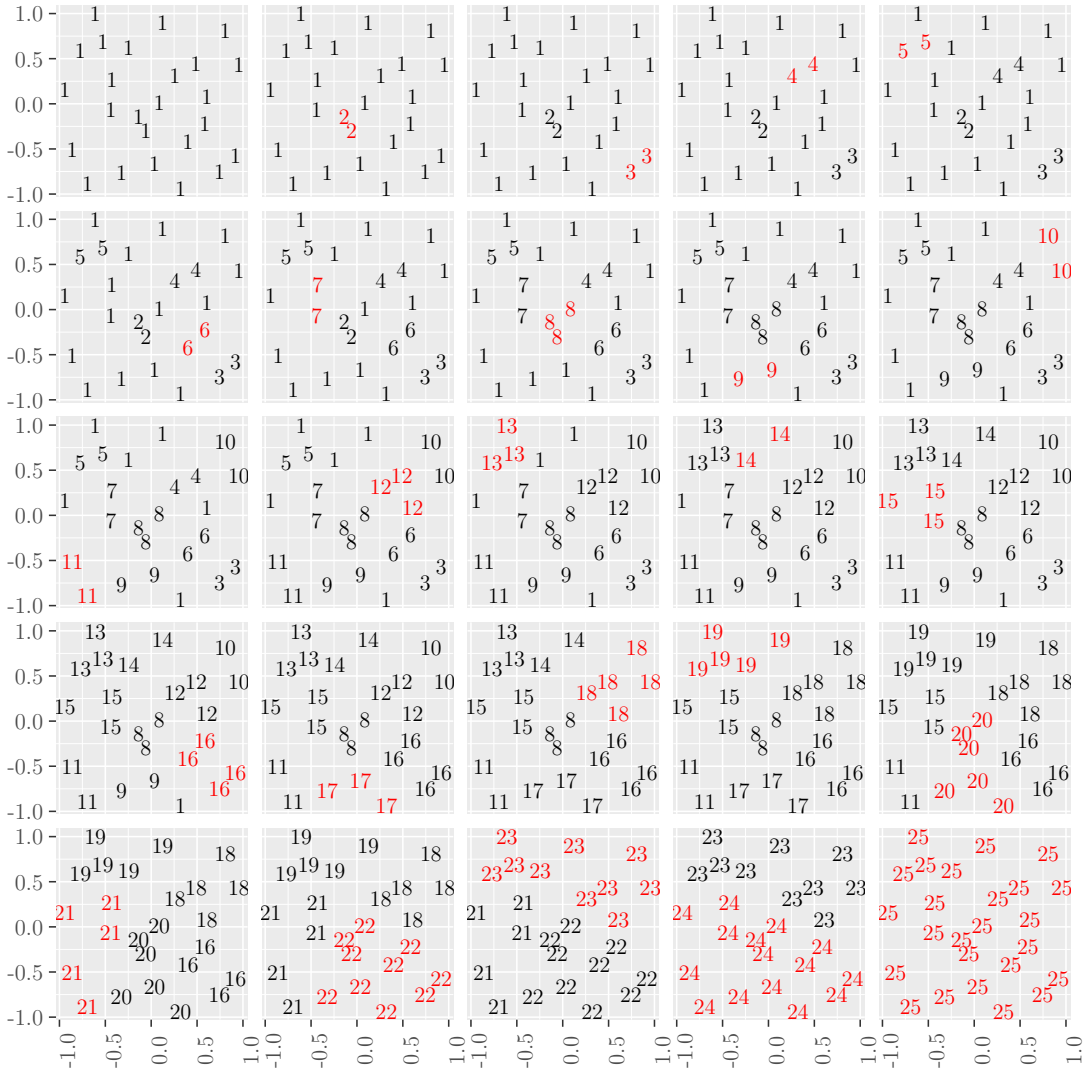


Figure 3.32: Visualization of the agglomerative hierarchical clustering on the LHS data when using complete linkage. The regression vectors are displayed as numbers which describe in which iteration the respective regression vector has been used the last time. Regression vectors marked in red are the values which are merged to a single cluster in the respective iteration.

When having done the clustering, the dendrogram of the clustering process can be created. For this, it is started at the latest partition of the data set. All regression vectors belonging to the first cluster go to the left branch of the dendrogram and all regression vectors belonging to the second cluster go to the right branch. This procedure is continued with the previous partitions the clustering process has been done until every regression vector belongs to its own cluster. The only question in each partition is: Which cluster goes to the left branch of the dendrogram and which

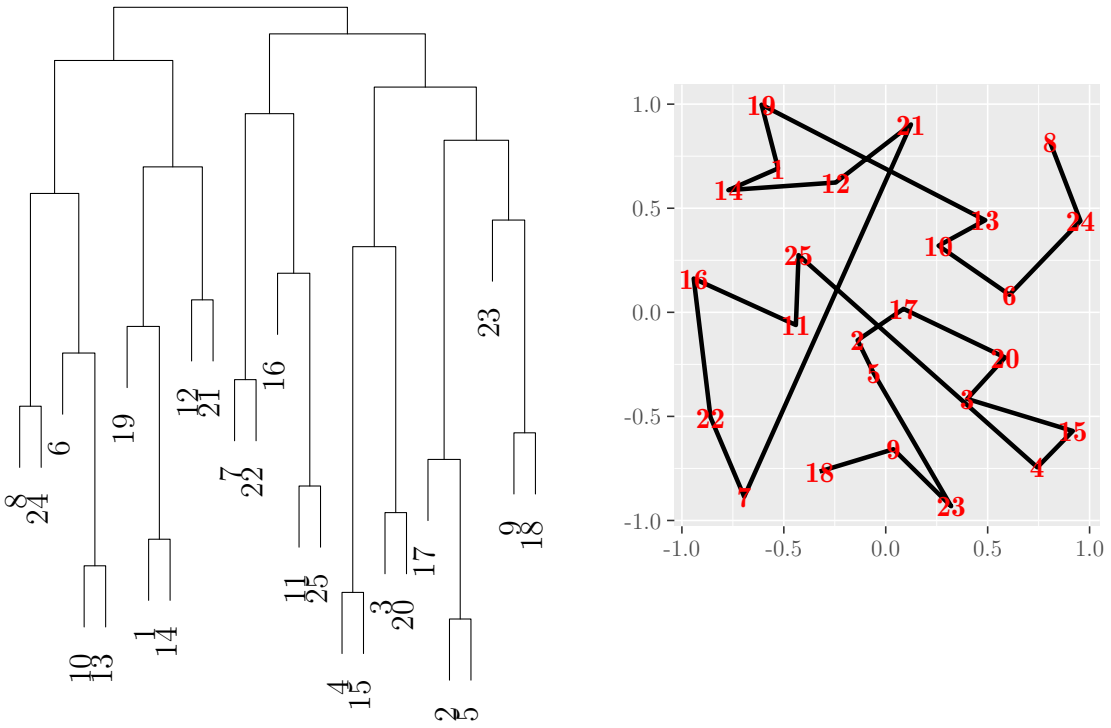


Figure 3.33: Dendrogram of the hierarchical clustering with complete linkage of the LHS data on the left and the obtained order according to the order in the dendrogram on the right. The regression vectors of the data set are represented as unique numbers.

cluster to the right? To this question, several meaningful solutions exist. Here, we will use the solution `R` uses for its hierarchical clustering function `hclust()`: "The algorithm used in `hclust()` is to order the subtree so that the tighter cluster is on the left (the last, i.e., most recent, merge of the left subtree is at a lower value than the last merge of the right subtree). Single observations are the tightest clusters possible, and merges involving two observations place them in order by their observation sequence number." (cited from the `R` help page of `hclust()`.) Figure 3.33 shows the obtained dendrogram of the hierarchical clustering with complete linkage on the LHS data set. Looking at the dendrogram bottom-up, the clustering process visualized in Figure 3.32 is visible: For example, the first seven merges are merges of two regression vectors respectively, before the first time a larger cluster occurs. On the left hand side of Figure 3.33, a scatterplot of the data set is shown with the obtained order. It can be seen that the regression vectors are ordered according to the order of the leafs of the dendrogram.

Figure 3.34 shows the obtained orders not only for the LHS data with complete linkage, but also for the two other data sets and also with average linkage and single linkage. It can be seen that the obtained orders highly depend on the choice of the dissimilarity measure. Furthermore, it can be seen that nearly all orders are much more "in a mess" than the obtained orders in the two previous subsections, apart from the Spiral data with single linkage where the inherent order of the data is preserved.

The dependance of the dissimilarity function is a disadvantage of this ordering method. But, besides of this disadvantage, there are many advantages: Like the other distance based ordering methods, this ordering method can be applied to ordinal

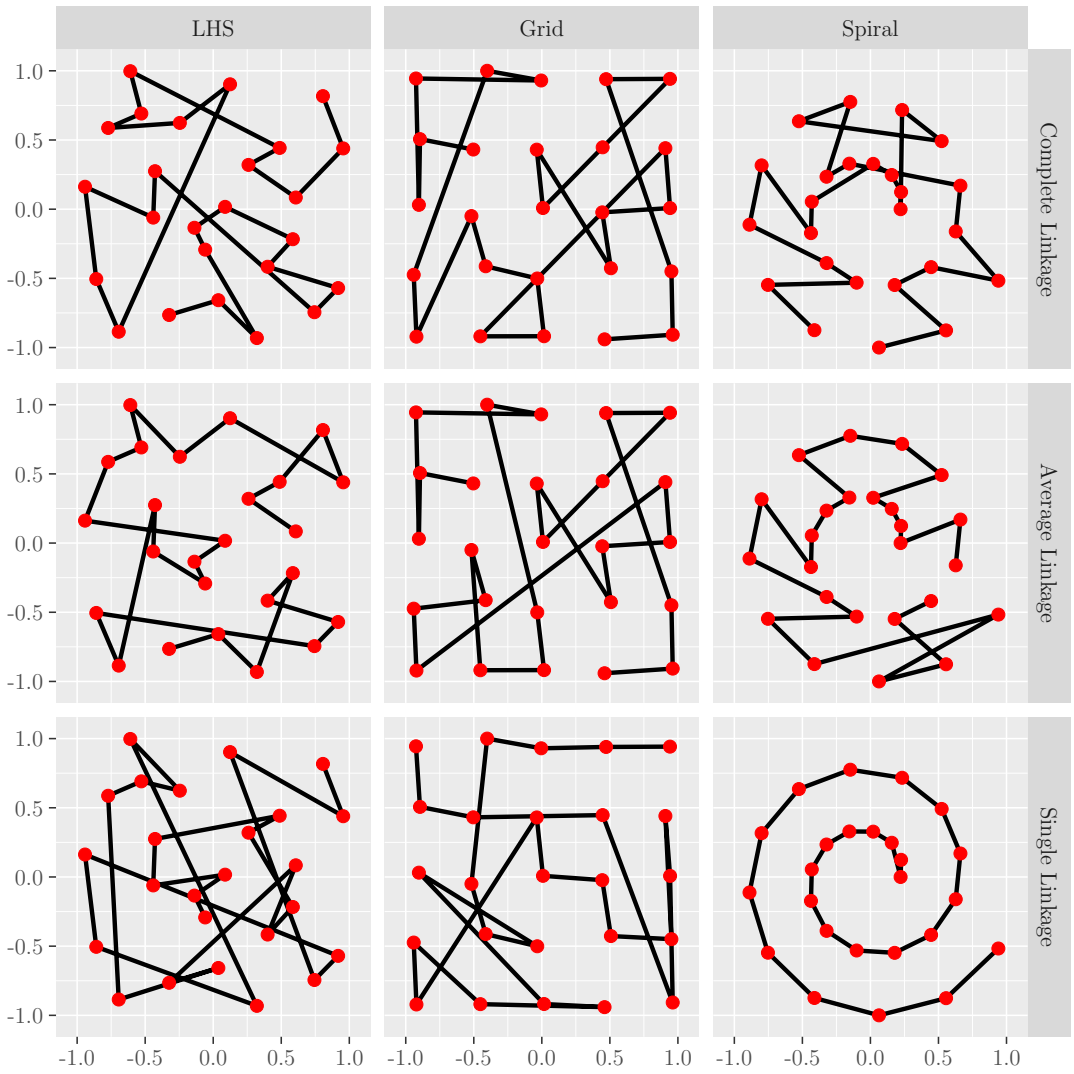


Figure 3.34: Visualization of the obtained orders when using hierarchical clustering with different dissimilarity measures for ordering.

and nominal data, metric data can be transformed additively and multiplicatively without changing the order and in the one-dimensional case the inherent order of the data is preserved.

The time complexity of the hierarchical clustering approach presented in Algorithm 3.5 is $\mathcal{O}(DN^3)$. This can be quite easy seen when realizing that line 4 of the algorithm (finding the two clusters with smallest dissimilarity) has runtime $\mathcal{O}(DN^2)$ since a matrix with pairwise dissimilarities of D-dimensional regression vectors has to be computed and that this matrix can have size up to $N \times N$. Since this has to be done $N - 1$ times, the time complexity of this "naive" algorithm is $\mathcal{O}(DN^3)$. As shown for example in Manning et al. (2008), the time complexity of the agglomerative hierarchical clustering can be reduced to $\mathcal{O}(N^2 \log(N))$ when using updating strategies for the dissimilarity matrices. Together with the time complexity of calculating a

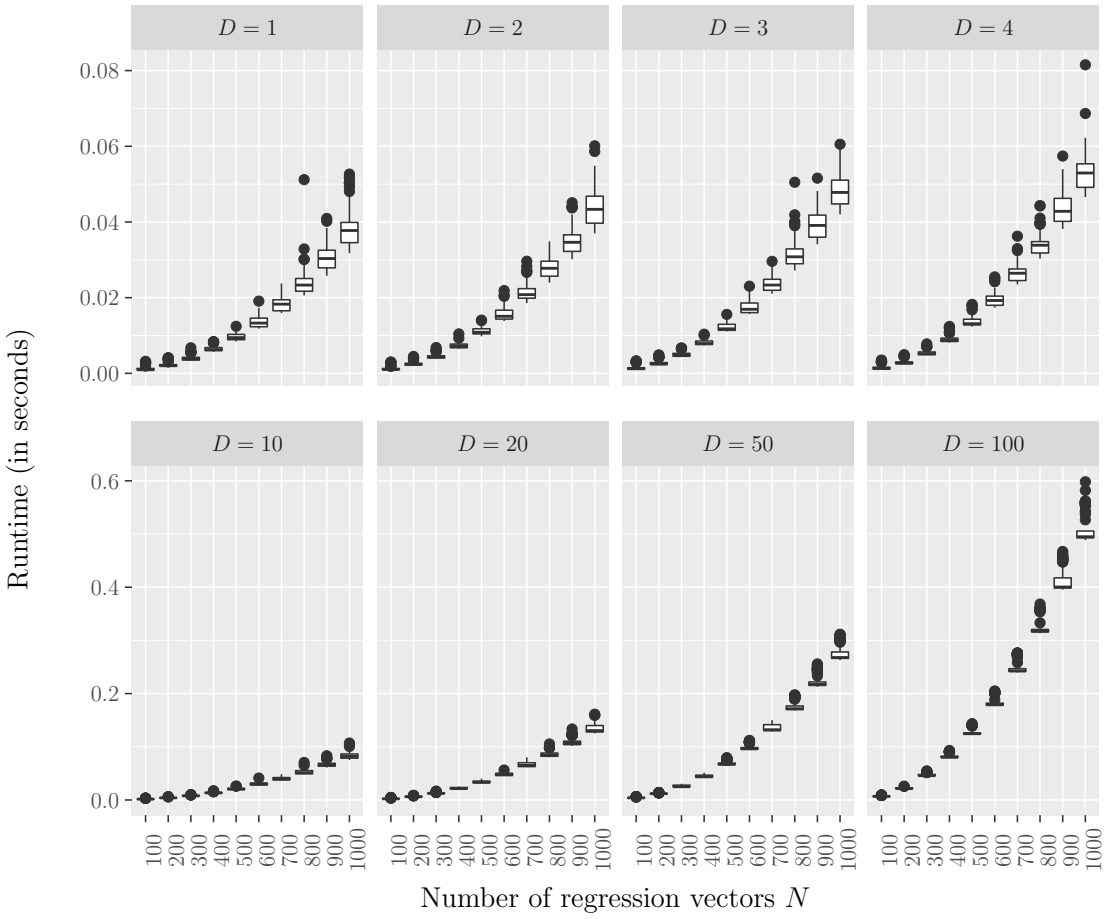


Figure 3.35: Empirical runtimes when using a hierarchical clustering with complete linkage for ordering.

pairwise distance matrix at the beginning, the time complexity of this ordering method can be given as $\mathcal{O}(N^2 \cdot (D + \log(N))$.

Figure 3.35 shows the empirical runtimes of this ordering method when using a complete linkage. The quadratic increase of the runtimes can be seen very well. Remarkably, the empirical runtimes are very small, only fractions of a second. This is a big difference to the other distance based ordering methods and a big advantage of this method. Although the theoretical time complexity is rather large, the hierarchical clustering can be done so efficiently that the empirical runtimes are very small.

3.5 Summary and Comparison of the Described Ordering Methods

In the previous sections 13 different ordering methods were described. Their characteristics, advantages and disadvantages were explained and their theoretical and empirical computational runtimes were analyzed. In this section, the obtained knowledge gets summarized and compared. For this, especially Table 3.1 is useful.

This table shows that additive and multiplicative transformations do not change the obtained orders of most methods. Only, ordering the regression vectors according to a vector norm and the nondominated sorting method get affected by data transformations. Furthermore, most of the scalarization based ordering methods (except the ordering according to a vector norm) are affected by multiplicative transformations in the way that multiplying all regression vectors with the same negative value will lead to a reversed order.

Most ordering methods need purely metric data to be able to compute an ordering. Nominal and ordinal data can only be ordered by the naive methods and the distance based methods which do not need the values of the regression vectors for ordering but only the indices of the regression vectors in the data set or the pairwise distances, respectively. Also, some nominal and ordinal components are allowed when using only one component of the regression vector for ordering, as long as a metric component of the regression vector is chosen for the ordering process.

Another aspect considered in this chapter is whether the described ordering methods obtain the inherent order of the data in the one-dimensional case. This is the case for

Method	Transformation		Data			Inherent Order for $D = 1$	Runtime	
	additive	multiplicative	nominal	ordinal	metric		Theoretical (Worst Case)	Empirical (Magnitude)
Order of the Data Set	✓	✓	✓	✓	✓	✗	$\mathcal{O}(1)$	10^{-4} sec
Random Order	✓	✓	✓	✓	✓	✗	$\mathcal{O}(N)$	10^{-4} sec
Norm	✗	✓	✗	✗	✓	✗	$\mathcal{O}(ND)$	10^{-3} sec
Median	✓	(✓)	✗	✗	✓	✓	$\mathcal{O}(ND)$	10^{-2} sec
Taking only one Regressor	✓	(✓)	(✓)	(✓)	✓	✓	$\mathcal{O}(N)$	10^{-4} sec
Weighted Sum	✓	(✓)	✗	✗	✓	✓	$\mathcal{O}(ND)$	10^{-3} sec
Orthogonal Projection	✓	(✓)	✗	✗	✓	✓	$\mathcal{O}(ND)$	10^{-3} sec
Nondominated Sorting	✓	✗	✗	✗	✓	✓	$\mathcal{O}(N^2D)$	10^{3} sec
Convex Hull	✓	✓	✗	✗	✓	✗	$\mathcal{O}(N^2 \cdot (D + 2^N))$	10^{2} sec
Halfspace Depth	✓	✓	✗	✗	✓	✗	$\mathcal{O}(N^2 \cdot (D + 2^N))$	10^{2} sec
Shortest Hamiltonian Path	✓	✓	✓	✓	✓	✓	$\mathcal{O}(N^2 \cdot (D + 2^N))$	10^{2} sec
Nearest Neighbor Heuristic	✓	✓	✓	✓	✓	✓	$\mathcal{O}(DN^2 + N^3)$	10^{2} sec
Hierarchical Clustering	✓	✓	✓	✓	✓	✓	$\mathcal{O}(N^2 \cdot (D + \log(N)))$	10^{-2} sec

Table 3.1: Summary of the characteristics and runtimes of the 13 ordering methods. The "Transformation" columns describe whether the order is preserved when transforming the data, where (✓) means that the order is inverse when multiplying with negative values (but this does not affect the value of an arbitrary sign depth). The "Data" columns describe which levels of measurement the data may have for the respective method, where (✓) means that nominal and ordinal components of regression vectors are allowed if they are not chosen for ordering. "Inherent Order for $D = 1$" means that the respective methods order the regression vectors according to their values in the one-dimensional case and the "Runtime" columns describe the theoretical worst case time complexity and the magnitude of the empirical runtimes.

only eight of the 13 ordering methods. Both naive ordering methods do not obtain this inherent order since the ordering is achieved by looking at the order in the data set or by randomly permuting the index vector. Also, when the ordering is based on the values of a vector norm, the inherent order is not obtained if both positive and negative values are in the data set because all vector norms in the one-dimensional case reduce to taking the absolute value of the respective value. Furthermore, two of the three presented ordering methods based on partial sorting do not obtain the inherent order in the one-dimensional case. Although the ordering method based on convex hulls obtains the correct sequence of regression vectors, its obtained order

does not necessarily start with the smallest value. Additionally, the partial sorting based on the values of Tukey's halfspace depth orders the one-dimensional values from the extremes of the data to the "center" of the data.

The computational runtimes are quite different among the 13 ordering methods. The smallest theoretical worst-case time complexity has the ordering according to the indices of the data set, which has a constant runtime. The largest time complexity has been calculated for all methods which need an exact solution of the Traveling Salesman Problem, i.e. the convex hull method, the halfspace depth method and the SHP method which have an exponential runtime increase in the number of regression vectors. Most methods also have a linear runtime increase in the number of dimensions, but this is empirically in most cases negligible. In general, the empirical runtimes showed in this chapter do not necessarily match the theoretical time complexity. The expected increases were visible very well in most cases, but this does not necessarily mean anything for the magnitude of the empirical runtimes. Especially, the hierarchical clustering ordering method has very small empirical runtimes whereas the time complexity is more than quadratic in the number of regression vectors. The smallest empirical runtimes are obtained by the naive ordering methods. Also, the scalarization based methods and the before mentioned hierarchical clustering method have very small empirical runtimes. In contrast, the partial sorting methods and the two others distance based ordering methods have quite large runtimes.

Overall, it can be said that all ordering methods have advantages and disadvantages. The performance of the ordering methods in the context of the sign depth test will be discussed in Chapter 5. There, it will be seen which characteristics of the ordering methods are important for a good performance and which can be neglected.

Chapter 4

Developed Software

As part of this thesis, all ordering methods as well as the sign depth and the sign depth test were implemented in `R` (R Core Team, 2019). The following sections describe these implementations and the self-written `R`-package `GSignTest` (Horn, 2021). Particular attention is paid to the time complexities and empirical runtimes of the implementations because especially implementing the sign depth is challenging.

4.1 Implementation of the Sign Depth

As it can be seen in Remark 2.2 on page 16, the sign depth in the case $P(E_n = 0) = 0$ is defined via a quite easy formula:

$$\begin{aligned} s_{n_k}^{(1)} &:= \mathbb{1}\{\hat{e}_{n_k}(\boldsymbol{\theta}) \cdot (-1)^k > 0\} \\ s_{n_k}^{(2)} &:= \mathbb{1}\{\hat{e}_{n_k}(\boldsymbol{\theta}) \cdot (-1)^k < 0\} \\ d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) &:= \binom{N}{K}^{-1} \sum_{1 \leq n_1 < n_2 < \cdots < n_K \leq N} \left(\prod_{k=1}^{K} s_{n_k}^{(1)} + \prod_{k=1}^{K} s_{n_k}^{(2)} \right) \end{aligned} \tag{4.1}$$

The case $P(E_n = 0) > 0$ will not be considered in this chapter because this thesis deals with multiple regression with metric values where the probability of having a residual with value zero is zero. Looking at this formula, it quickly becomes clear that there is a very simple way of implementing the sign depth via nested loops. This naive way of implementing the sign depth is formalized in Algorithm 4.1. As it can be seen, this algorithm has two drawbacks: Firstly, the algorithm has to be implemented for every value of K separately because the number of loops depends on the value

Algorithm 4.1 Naive way of computing the K-sign depth.

Input: Vector of residuals $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$ of length N; parameter K of the sign depth

Output: The value of the K-sign depth

1: Determine the vector $\boldsymbol{r}$ of the signs of $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$.
2: $result = 0$
3: **for all** $n_1 = 1$ **to** $n_1 = N - K + 1$ **do**
4: **for all** $n_2 = n_1 + 1$ **to** $n_2 = N - K + 2$ **do**
5: $\vdots$ # Altogether K loops
6: **for all** $n_K = n_{K-1} + 1$ **to** $n_K = N$ **do**
7: **if** $r_{n_1}, \ldots, r_{n_K}$ are alternating **then**
8: $result = result + 1$
9: **end if**
10: **end for**
11: $\vdots$
12: **end for**
13: **end for**
14: $result = result \,/ \binom{N}{K}$
15: **return** $result$

of K. And secondly, since it has to be checked $\binom{N}{K}$ times whether K residuals are alternating, the time complexity of this algorithm is $\mathcal{O}\left(K \cdot \binom{N}{K}\right) \in \mathcal{O}\left(K \cdot N^K\right)$. The first problem can be avoided by calculating all combinations of residuals before checking whether they are alternating. In `R`, the function `combn()` can be used for creating a matrix with all needed combinations of indices. With the help of this function, a general implementation of the K-sign depth for all values of K is possible in `R`. Such an implementation is given in Code 4.1. Besides the drawback of the large time complexity, the approach of calculating all needed combinations of indices beforehand has another disadvantage: Storing the complete combination matrix needs a lot of memory space. In `R`, the maximal length of a vector (and so the maximal number of entries in each dimension of a matrix) is $2^{31} - 1$.[1] Since `combn()` in Code 4.1 builds a matrix with K rows and $\binom{N}{K}$ columns, especially for larger values of K the number of residuals N is very limited. Furthermore, in practice it is hardly possible to create such large matrices because the random access memory (RAM) of private used computers is too small for such matrices. On 32bit-Windows machines, the maximal size an `R`-object is allowed to have is 2 GB. Although the

[1] see help page of `Memory-limits` in `R`

```
signDepth <- function(residuals, K) {
  N <- length(residuals)
  r <- sign(residuals)
  comb <- combn(N, K)
  result <- apply(comb, 2, function(x) {
    all(r[x] == rep(c(1, -1), length.out = K) |
          all(r[x] == rep(c(-1, 1), length.out = K)))
  })
  return(mean(result))
}
```

Code 4.1: First implementation of the K-sign depth in `R`.

limit for 64bit-Windows machines and Unix-like computers is larger, it is not realistic to work with objects which are larger than a few gigabyte. Since `combn()` creates an integer matrix and every integer in `R` needs 4 bytes memory space, in every gigabyte approximately 2^{28} integers can be stored.[2] Here, it has to be counted in that every column of the matrix consists of K rows. Table 4.1 shows the maximal number of residuals depending on the value of K that can be used in Code 4.1. It can be seen that for practical purpose this implementation reaches its limit very fast, especially for larger values of K. Since it seems not realistically possible to compute the sign depth of only a few hundreds of residuals, this implementation is unusable in practice.

[2] 1 gigabyte $\hat{=}$ 1 024 megabyte $\hat{=}1\,024^2$ kilobyte $\hat{=}1\,024^3$ byte. Since one gigabyte are $1\,024^3 = 2^{30}$ bytes and a single integer needs 4 bytes disc space, in one gigabyte disc space $2^{30}/4 = 2^{28}$ integers can be stored (Overhead, for example from creating a vector in `R`, is not counted in).

		Size of the matrix		
K	theoretically	1 GB	2 GB	4 GB
3	2 345	813	1 025	1 291
4	477	201	239	284
5	193	93	107	122

Table 4.1: Maximal number of residuals whose combination matrix can be stored in `R`. The theoretical values can be obtained by solving $\max_{N\in\mathbb{N}} \binom{N}{K} < 2^{31} - 1$ for fixed values of K and the values for the three right columns are determined via $\max_{N\in\mathbb{N}} K \cdot \binom{N}{K} < \gamma \cdot 2^{28}$ for fixed values of K and $\gamma \in \{1, 2, 4\}$.

Moreover, not only the limited memory space is problematic, but also the time to create such large matrices which often amounts to several minutes.

Because of these drawbacks it is necessary to calculate the combinations one after the other, so that storing the complete combination matrix is not necessary. For this, it has to be thought about an algorithm to compute all combinations without being forced to remember and store all previous ones. Algorithm 4.2 shows a pseudocode for computing a new combination vector on the basis of an old combination vector. The idea behind this algorithm is to increase the value in the last position of the given vector by one. If this is not possible because this value is already the maximal allowed value N, the value in the previous position is increased. Here, the maximal allowed value is $N-1$. If this is also not possible, it is again taken the value in the previous position, and so on. All values before this position stay the same. The value at the determined position is increased by one and all following values are larger by one than the previous. Starting with the vector $(1,\ldots,K)^\top$ calling this

Algorithm 4.2 Pseudocode for computing a new combination vector $\boldsymbol{n}$ on the basis of an old one $\boldsymbol{o}$. Iteratively calling this function starting with the vector $(1,\ldots,K)^\top$ will give all $\binom{N}{K}$ vectors of possible combinations. Name this function `nextComb()` for usage in Algorithm 4.4.

Input: Vector of combinations $\boldsymbol{o}$ of length K; Maximal possible value in the combination vector N

Output: Vector of combinations $\boldsymbol{n}$

1: Initialize vector $\boldsymbol{n}$ of length K.
2: $j = K$
3: **while** $o_j == N - K + j$ **do**
4: $\quad j = j - 1$
5: **end while**
6: **for all** $i = 1$ **to** $j - 1$ **do**
7: $\quad n_i = o_i$
8: **end for**
9: $n_j = o_j + 1$
10: **for all** $i = j + 1$ **to** K **do**
11: $\quad n_i = n_{i-1} + 1$
12: **end for**
13: **return** $\boldsymbol{n}$

function iteratively will give all possible combinations in the same order `combn()` is computing the combinations.

Another disadvantage of the first implementation of the sign depth in Code 4.1 is that always all K signs are checked, even if for example the first two residuals in the given K-tuple are positive, nevertheless it is looked at all following residuals in the tuple. By going iteratively through the vector of residuals one can stop checking the respective vector when two residuals with the same sign one after the other have been found. A pseudocode of this rather simple procedure is shown in Algorithm 4.3. If all residuals have the same sign, in each iteration only one comparison between the first two elements of each K-tuple is needed for detecting that the respective tuple has no alternating signs. And also if the probability of each residual is 0.5 to be positive, it follows from the geometric distribution that in the expected value only two comparisons are needed to detect a non-alternating tuple. This can be explained by understanding that the probability of a sign change is also 0.5 when every residual is positive with probability 0.5. So, the probability of having no sign change within two successive residuals is 0.5, too. Since the geometric distribution describes the probability of getting a specific event the first time when the event occurs independently with probability p in every step and the expected value of the geometric distribution is $\frac{1}{p}$, here the expected first occurrence of two successive residuals with equal signs happens after $\frac{1}{0.5} = 2$ comparisons.

Algorithm 4.3 Efficiently checking for alternating signs in a vector $\boldsymbol{v}$. Name this function `signSwitch()` for usage in Algorithm 4.4.

Input: Vector $\boldsymbol{v} \in \mathbb{R}^K$

Output: 1, if $\boldsymbol{v}$ has alternating signs, 0 otherwise

1: Compute the vector $\boldsymbol{r}$ of signs of $\boldsymbol{v}$.
2: $result = 1$
3: **for all** $i = 2$ **to** $i = K$ **do**
4: **if** $r_i == r_{i-1}$ **then**
5: $result = 0$
6: **break**
7: **end if**
8: **end for**
9: **return** $result$

Algorithm 4.4 Pseudocode for computing the K-sign depth when creating iteratively index vectors and efficiently checking for alternating signs in the vector of residuals.

Input: Vector of residuals $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$; Parameter K of the sign depth

Output: The value of the K-sign depth

1: N = length of vector $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$
2: $\boldsymbol{o} = (1, \ldots, K)^T$
3: Check whether $\boldsymbol{o}$ has alternating signs with the help of Algorithm 4.3: $c =$`switchSign`$(\boldsymbol{o})$
4: **for all** $i = 1$ **to** $i = \binom{N}{K}$ **do**
5: $\quad \boldsymbol{n} =$`newComb`$(\boldsymbol{o}, N)$
6: $\quad c = c +$`switchSign`$(\boldsymbol{n})$
7: $\quad \boldsymbol{o} = \boldsymbol{n}$
8: **end for**
9: **return** $c/\binom{N}{K}$

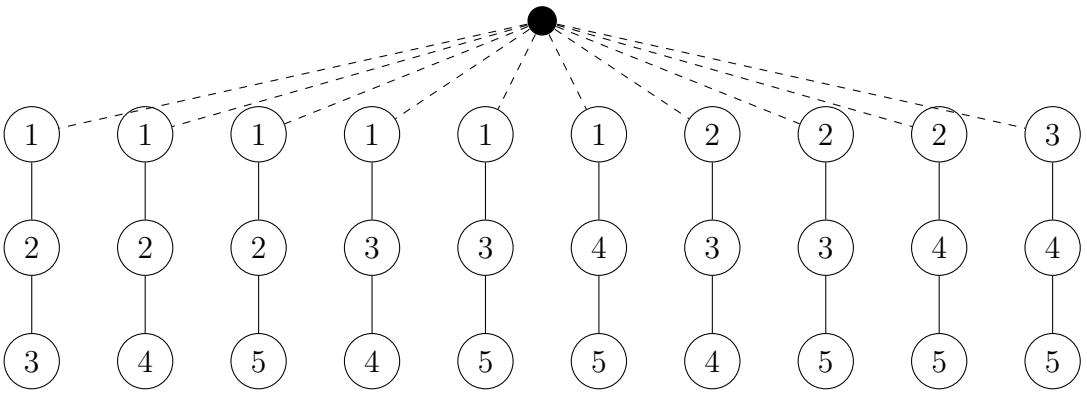


Figure 4.1: Visualization of needed comparisons when calculating iteratively the 3-sign depth with five residuals.

A pseudocode for computing the K-sign depth in this way is given in Algorithm 4.4. In addition, an example of calculating the 3-sign depth with five residuals is visualized in Figure 4.1. For this, the iterative computation of the index vectors from Algorithm 4.2 is used and additionally, the second and third residual of each 3-tuple is only checked for alternating signs if the first and second residual were alternating. The drawn tree shows the residual indices of the five residuals and every branch of the tree corresponds to one step of the algorithm. By looking at the tree from the left to the right the procedure of the algorithms gets clear: At first the index vector $(1, 2, 3)^\top$ is created and checked for alternating signs. Therefore, the residuals with indices

1 and 2 are compared and if they are alternating, residuals 2 and 3 are compared. In the next step, the index vector $(1, 2, 4)^\top$ is examined, and so on until the index vector $(3, 4, 5)^\top$ is reached.

Of course, this theoretical approach has been implemented in `R`. Indeed, both the implementation of the iterative way to create combination vectors and the sign depth itself were written in `C++` and connected to `R` with the help of the package `Rcpp` (Eddelbuettel and Balamuta, 2017). The code of the implementation of Algorithm 4.2 can be found in Code B.3 in the appendix on page 217. The `C++`-code for checking iteratively the signs of the residuals is given in Code B.2. Also in Appendix B the `C++`-implementation of the sign depth with the help of above mentioned implementations can be found. It is given in Code B.4. The programming language `C++` instead of `R` has been chosen for these implementations because of the above mentioned problem with the large time complexities of the sign depth calculation. In most cases, implementations in `C++` are faster than the same implementations in `R`.

Although this implementation of the sign depth reduces the time complexity from $\mathcal{O}(K \cdot \binom{N}{K})$ to $\mathcal{O}(2 \cdot \binom{N}{K})$ in the average, there are still some non-necessary comparisons. Imagine there are five residuals for which the 3-sign depth should be computed and the first two residuals are positive. Then the comparison of the first two residuals is done three times because the index-tuples $(1, 2, 3)^\top$, $(1, 2, 4)^\top$ and $(1, 2, 5)^\top$ are checked independently of each other. To avoid these non-necessary comparisons one can compute the sign depth recursively by looking only at the jth element of a tuple if all $j - 1$ elements before have alternating signs. If the residual at position j has the same sign as the residual at position $j - 1$, the recursion breaks up for all tuples which contain the same values until the jth position. This strategy is formalized in Algorithms 4.5 and 4.6 and implemented in Code B.5 and B.6 in Appendix B. As before, the implementations were written in `C++` and afterwards connected to `R` to speed up the calculations. The loop in line 4 to 6 of Algorithm 4.6 computes the number of K-tuples with alternating signs starting at each index separately. This is done because in the first line of `recFun()` (Algorithm 4.5) it is checked whether the sign of the current residual is the same as the sign of the previous residual and since there is no previous residual at the beginning of a tuple the previous sign is set to zero (see the last parameter in the call of `recFun()` in line 5 of Algorithm 4.6). If the current and the previous sign are the same, `recFun()` returns zero which means that the recursion breaks up because all tuples with the respective previous and current sign cannot be alternating. If the previous and the current sign are not the

Algorithm 4.5 Pseudocode for function `recFun()` used in Algorithm 4.6.

Input: Vector of signs $\boldsymbol{r}$; length of considered tuples K; index j; sign of previous residual r_{j-1}

Output: Number of alternating K-tuples starting at index j

```
if r_j == r_{j-1} then
    return 0
end if
if K == 1 then
    return 1
end if
N = length of vector r
result = 0
for all i = j + 1 to i = N − K + 2 do
    result = result + recFun(r, K − 1, i, r_j)
end for
return result
```

Algorithm 4.6 Algorithm for computing the sign depth which reduces the number of K-tuples to be checked for alternating signs.

Input: Vector of residuals $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$; Parameter K of the sign depth

Output: The value of the K-sign depth

```
N = length of vector ê(θ)
Determine the vector r of the signs of the residuals ê(θ).
c = 0
for all i = 1 to i = N − K + 1 do
    c = c + recFun(r, K, i, 0)
end for
return c/(N choose K)
```

same, `recFun()` is called recursively on the next indices with the former current sign now as previous sign. Since the remaining tuple is smaller by one `recFun()` is called with argument $K - 1$. If the recursion has not broken up when reaching $K = 1$, an alternating tuple has been found for which one is returned (see line 4 to 6 in Algorithm 4.5).

During this thesis the three above mentioned implementations of the sign depth were developed: The primitive `R`-implementation, the iterative `C++`-implementation and the recursive `C++`-implementation. While the primitive `R`-implementation always needs $K \cdot \binom{N}{K}$ comparisons independent of the data situation, the number of needed comparisons gets smaller with the `C++`-implementations. In the algorithm's best case, when all residuals have the same sign, the iterative algorithm needs only one comparison for every tuple. This leads to a best case runtime of $\mathcal{O}(1 \cdot \binom{N}{K})$. As mentioned above, the average runtime of the iterative algorithm is $\mathcal{O}(2 \cdot \binom{N}{K})$, if every residual is positive with probability 0.5. An upper bound for the runtime can be given by $\mathcal{O}((K-1) \cdot \binom{N}{K})$. This number of comparisons would be reached if every tuple has strictly alternating signs, which is impossible.

In this scenario, the recursive algorithm would make $\sum_{k=0}^{K-2} \binom{N-k}{K-k} \in \mathcal{O}(\binom{N}{K})$ comparisons, for understandig see Figure 4.2. This figure shows the recursion tree when computing the 3-sign depth with five residuals. It can be seen that the first entry of a triple is residual number 1, 2 or 3 and that they are compared with residual numbers 2, 3 and 4 in the first step. This leads to $6 = \binom{4}{2} = \binom{5-1}{3-1}$ comparisons. In

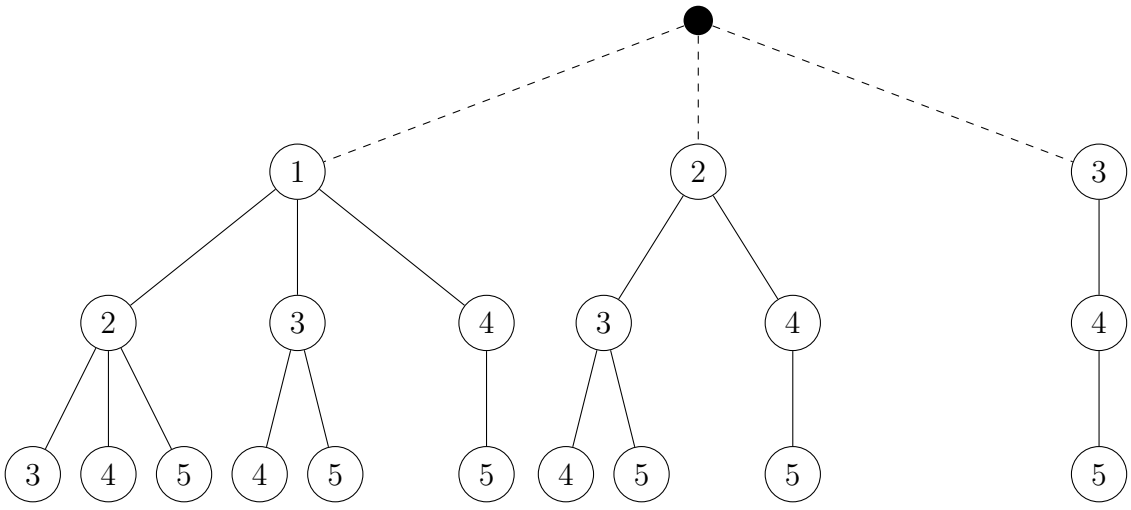


Figure 4.2: Visualization of needed comparisons when calculating recursively the 3-sign depth with five residuals.

the second step, the respective second entry of these six tuples gets compared with residual numbers 3, 4 and 5, which leads to $10 = \binom{5}{3}$ comparisons. So overall, in this example, an upper bound for the number of comparisons is $\binom{5}{3} + \binom{4}{2} = 16$.

Assumed that every entry has probability 0.5 for being positive, the number of comparisons reduces in the average to $\sum_{k=0}^{K-2} (\frac{1}{2})^{K-2-k} \cdot \binom{N-k}{K-k} \in \mathcal{O}((\frac{1}{2})^{K-2} \cdot \binom{N}{K})$ because the probability of going a layer deeper in the recursion tree is everytime 0.5. Specifically, that means that the comparisons between the first two entries of every tuple are made for sure, i.e. with probability 1. The comparison of the second and third entry of the tuple is only made if the first two entries had alternating signs which occured with probability 0.5. For $K > 3$, the comparison of the third and fourth entry is only made if both previous comparisons showed alternating signs, i.e. this event has probability $0.5 \cdot 0.5 = 0.25$, and so on. In the example tree in Figure 4.2 the number of comparisons reduces in the average to $1 \cdot \binom{4}{2} + \frac{1}{2} \cdot \binom{5}{3} = 11$ comparisons.

The smallest time complexity of the recursive algorithm will be reached when all residuals have the same sign. In this case, only the comparisons between the residuals in the first and second layer of the recursion tree have to be done, which are $\binom{N-(K-2)}{K-(K-2)} = \binom{N-K+2}{2} = \frac{1}{2} \cdot (N-K+1) \cdot (N-K+2) \in \mathcal{O}((N-K)^2)$ comparisons.

	Best Case	Average Case	Worst Case
Primitive `R`	$\mathcal{O}\left(K \cdot \binom{N}{K}\right)$	$\mathcal{O}\left(K \cdot \binom{N}{K}\right)$	$\mathcal{O}\left(K \cdot \binom{N}{K}\right)$
Iterative `C++`	$\mathcal{O}\left(1 \cdot \binom{N}{K}\right)$	$\mathcal{O}\left(2 \cdot \binom{N}{K}\right)$	$\mathcal{O}\left((K-1) \cdot \binom{N}{K}\right)$
Recursive `C++`	$\mathcal{O}\left((N-K)^2\right)$	$\mathcal{O}\left(\left(\frac{1}{2}\right)^{K-2} \cdot \binom{N}{K}\right)$	$\mathcal{O}\left(1 \cdot \binom{N}{K}\right)$

Table 4.2: Overview of the time complexities of the three presented algorithms for computing the sign depth. "Best Case" means that all residuals have the same sign, "Average Case" means that every residuals has probability 0.5 to be positive and "Worst Case" means that all $\binom{N}{K}$ tuples are have strictly alternating signs. Since the worst case cannot occur, the values for the worst case have to be interpreted as upper bound for the time complexities of the algorithms.

An overview of the theoretical time complexities of the three different algorithms can be found in Table 4.2.

Empirical runtimes of the algorithms can be found in Figure 4.3 and Table C.2. For these results 100 times $N \in \{20, 40, 60, 80, 100\}$ residuals were drawn independently from a $\mathcal{U}(-1, 1)$ distribution. This simulates the above mentioned "Average Case". Figure 4.3 clearly shows the differences in the runtimes of the three algorithms (please note the logarithmic y-axis). While the primitive `R`-implementation needs about seven minutes in median for computing the 5-sign depth with 100 residuals (and several Gigabytes of RAM), the recursive `C++`-implementation needs only about a third of a second for the same setting. In general, the runtimes increase with larger K and the iterative `C++`-implementation always is much faster than the primitive `R`-implementation and the recursive `C++`-algorithm is much faster than the iterative one.

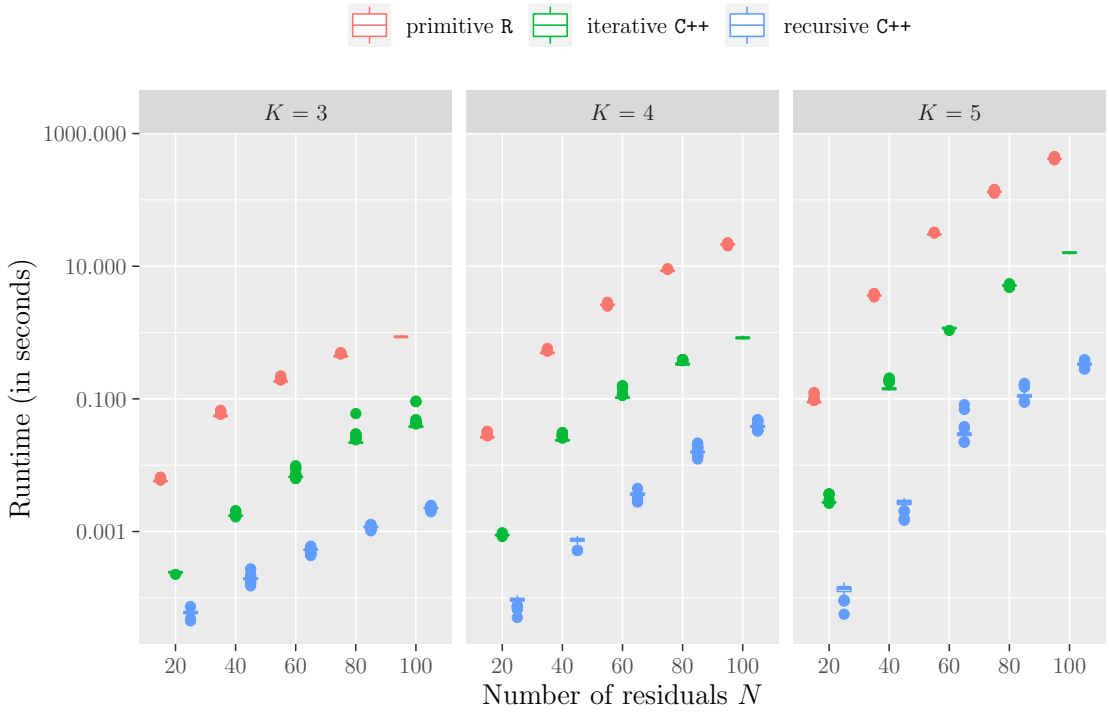


Figure 4.3: Empirical Runtimes of the three algorithms for computing the sign depth with $K \in \{3, 4, 5\}$. For these results 100 times N residuals were drawn independently from a $\mathcal{U}(-1, 1)$ distribution, so that every residual has probability 0.5 to be positive which refers to the above mentioned "Average Case". Please note the logarithmic scale on the y-axis. A table of the median of these runtimes can be found in Table C.2 on page 227.

Although the recursive `C++`-implementation seems to be quite fast, a time complexity which has a factor of $\binom{N}{K}$ in it becomes large when N or K gets large. For example, calculating the sign depth for 1 000 residuals needs 3.9 seconds for the 3-sign depth, 7 minutes for the 4-sign depth and 11 hours for the 5-sign depth in the median. Computing the sign depth for $K \in \{10, 50\}$ needs 1 and 38 hours respectively for $N = 100$ residuals. Because of this, Kevin Leckey, Dennis Malcherczyk and Christine Müller worked on implementations of the sign depth which are linear in N. So far, two linear implementations were developed, one is an asymptotically equivalent implementation of the sign depth. For this implementation also exact solutions are available for $K \in \{3, 4, 5\}$. This asymptotically equivalent form of the K-sign depth bases on the fact that the sign depth can be written as a sum of products of sign functions with some additional terms for $K > 3$, which converge almost surely to zero for large values of N. For $K \in \{4, 5\}$, Dennis Malcherczyk found a way of calculating these terms also in linear time complexity, so that an exact implementation of the sign depth for these values of K is available. The additional terms for larger values of K cannot be calculated in linear time complexity so far, but for practical purpose the general asymptotically equivalent implementation of the K-sign depth in linear time complexity is sufficient and will be used in Subsection 5.3.4, whereas the exact linear implementation will be used in all other parts of Chapter 5. The second developed implementation in linear time complexity computes the sign depth exactly for all values of K. This so-called "block-implementation" is explained in the dissertation of Dennis Malcherczyk (Malcherczyk, 2021+) and has even smaller empirical runtimes than the above mentioned implementation and is numerically more stable. Unfortunately, when performing the simulations of Chapter 5 and writing this thesis, this implementation was not yet available and so, in this thesis only the above mentioned implementation in linear time complexity could be used. Because of this, in the following and especially in Chapter 5 also all interpretations of the results are written as if the block-implementation would not exist.

4.2 Implementation of the Sign Depth Test

With the help of the sign depth, in linear models hypotheses of the form

$$H_0 : \boldsymbol{\theta} \in \boldsymbol{\Theta}_0 \quad \text{vs.} \quad H_1 : \boldsymbol{\theta} \notin \boldsymbol{\Theta}_0$$

can be tested when a linear models is defined via $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{e}$. Since in this thesis it is focused on hypotheses of the form

$$H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0 \quad \text{vs.} \quad H_1 : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0,$$

the implementation of this case is described further on. When looking at the definition of the sign depth test in Section 2.5, it can be seen that the general case can be obtained by calculating the sign depth for every $\boldsymbol{\theta} \in \boldsymbol{\Theta}_0$ and taking the supremum of the obtained sign depths, so that the described implementation for a null-hypothesis consisting of only one element can also be used for the general case if the largest depth in H_0 was calculated beforehand.

When looking at the definition of the sign depth test in Theorem 2.2 in Section 2.5, it can be seen that the test statistic of the sign depth test is compared with $q_\alpha^{K,N}$, the α-quantile of the distribution of the K-sign depth with N residuals. For implementing the sign depth test, these quantiles have to be computed beforehand. Since for large N the distribution of $N \cdot (d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) - (\frac{1}{2})^{K-1})$ converges to an asymptotic distribution, especially the quantiles and distributions for small N are of interest. For this purpose, the exact distribution of the K-sign depth for $K \in \{2, 3, 4, 5\}$ and $N \in \{K, \ldots, 25\}$ has been computed. This is done by calculating the K-sign depth of all 2^N possible sign combinations, which are up to $2^{25} = 33\,554\,432$. For larger values of N the distribution of the K-sign depth is approximately computed by calculating the K-sign depth of one million random sign combinations of length N.

When having the distribution of the K-sign depth, calculating the K-sign depth test with given residuals is quite easy. A pseudocode of this procedure can be found in Algorithm 4.7. Here, the K-sign depth of the residuals is computed and afterwards the p-value of the test is calculated via the probability function of the distribution of the K-sign depth. If no residuals but data and a model are given, the procedure is the same with the difference that the residuals have to be computed beforehand. For

Algorithm 4.7 Pseudocode of the K-sign depth test when input is a vector of residuals.

Input: Vector of residuals $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$; Parameter K of the sign depth

Output: p-value of the K-sign depth test

1: Calculate the test statistic d_S^K, which is the K-sign depth of $\hat{\boldsymbol{e}}(\boldsymbol{\theta})$.
2: Calculate the p-value of the test via the formula $p = F^{K,N}(d_S^K)$, where $F^{N,K}$ denotes the probability function of the K-sign depth for N residuals.
3: **return** p

Algorithm 4.8 Pseudocode of the K-sign depth test when input is a data set and a model.

Input: Formula `f` of the model to be fitted; Data Set $\boldsymbol{Z}$; Vector $\boldsymbol{\theta}_0$ to be tested on; An ordering method; Parameter K of the sign depth

Output: p-value of the K-sign depth test

1: Extract the design matrix $\boldsymbol{X}$ and the response vector $\boldsymbol{y}$ of $\boldsymbol{Z}$ with the help of `f`.
2: Order the values in $\boldsymbol{X}$ and $\boldsymbol{y}$ via the given ordering method.
3: Compute the residuals under H_0: $\hat{\boldsymbol{e}}(\boldsymbol{\theta}_0) = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}_0$.
4: Calculate the test statistic d_S^K, which is the K-sign depth of $\hat{\boldsymbol{e}}(\boldsymbol{\theta}_0)$.
5: Calculate the p-value of the test via the formula $p = F^{K,N}(d_S^K)$, where $F^{N,K}$ denotes the probability function of the K-sign depth for N residuals.
6: **return** p

this, the values in the design matrix $\boldsymbol{X}$ and the response vector $\boldsymbol{y}$ have to be ordered according to a given ordering method (see Chapter 3 and Section 4.3) before the residuals under H_0 can be calculated. This procedure is described in Algorithm 4.8.

4.3 Implementation of the Ordering Methods

All ordering methods described in Chapter 3 are implemented in `R`, partly with the help of different `R`-packages. The self-written `R`-function `multiSorting()` (available in the package `GSignTest`) gets a data set (as `data.frame`), a `character`-string describing the ordering method and possibly further control arguments of the ordering methods as input and returns the ordered data set and an index vector which describes the permutation of the original data set.

The naive ordering methods are easy to implement: For taking the order of the data set only the index vector `1:N` has to be created. For a random ordering this index vector has to be shuffled. This is done with the help of the function `sample()`.

The scalarization based methods are also quite easy to implement. The scalarized values are computed via standard `R`-functions and the index vector of the ordered data set is obtained with the help of the function `order()`. Since these ordering methods depend on different hyper-parameters, these parameters can be set via the `control`-argument of the function `multiSorting()`. This applies for the parameter p of the vector norm, for the choice of the component to be ordered on when ordering

only according one component is conducted, for the weights of a weighted sum of the regression vectors and for the position vector and the direction vector of the line on which the regression vectors are orthogonal projected on. When not setting any specific parameter, the defaults are used, which are set to $p = 2$ (i.e. euclidean norm) for an ordering according to a vector norm. For an ordering according to only one component of the regression vectors, the first component is used as default with a possible ordering according to the second component for regression vectors which have the same value in the first component and an ordering according to the third component if ties also occur in the second component and so on. If two regression vectors are completely equal, here (and for all other ordering methods) the order the regression vectors appear in the data set is used. The default value for the weights of an ordering according to a weighted sum of the regression vectors is a vector where all values are one, which means that all components of the regression vectors get equal weights. And for the projecting method the default is an orthogonal projection on the bisecting line, i.e. the location vector consists of entries with all values zero and the direction vector is a vector where all entries have value one.

The ordering methods based on partial sorting use some specialized `R`-packages. When using nondominated sorting as method for partial sorting the `R`-function `fastNonDominatedSorting()` from the `R`-package `nsga2R` (Tsou, 2013) is used. For ordering the regression vectors which got the same rank in the nondominated sorting process any other ordering method can be used. This can be set via the `control`-argument of the `multiSorting()` function, the default uses the order the regression vectors appear in the data set. For using convex hulls for partial sorting the function `convhulln()` from the package `geometry` (Habel et al., 2019) is repeatedly used. When assigned the regression vectors to the different convex hulls, the order of the regression vectors in every hull has to be determined. Because `convhulln()` does not return the regression vectors of a hull in an inherent order (i.e. an order which runs along the edges of the hull) the regression vectors in every hull have to be ordered manually. For example, the usage of the solution of a Traveling Salesman Problem leads to the wanted order, see below. The same holds for the ordering according to Tukey's halfspace depth. The regression vectors with the same value of the halfspace depth can be ordered with a Traveling Salesman Problem solver. For determining the values of the halfspace depth the function `depth.halfspace()` of the package `ddalpha` (Pokotylo et al., 2019) is used. Here, an approximate solution of the halfspace depth with 100 000 uniformly distributed directions is used because

computing the exact solution with $\binom{N}{D}$ directions would be computationally too expensive in many cases.

Making the distance based ordering methods usable is partially rather difficult. Computing the Shortest Hamiltonian Path and the nearest neighbor approximation is done with the `R`-package `TSP` (Hahsler and Hornik, 2019) which can be used for solving a Traveling Salesman Problem. This package also provides a function for including a dummy value with distance zero to every other regression vector, so that the Traveling Salesman Problem can be transformed to a Shortest Hamiltonian Path problem. While the nearest neighbor approximation can be computed solely with the `TSP`-package, the exact solution of the Shortest Hamiltonian Path needs the connection to an external solver written in `C`. This solver is not available or connected to `R` directly, but everybody has to do this on his/her own when wanting to use this solver. The *Concorde TSP Solver* (Applegate et al., 2004) uses a cutting-plane method which bases on linear optimization for solving the Traveling Salesman Problem, see Applegate et al. (2006) and Applegate et al. (2001). For running the linear optimization within the solving process it is recommended to use the *QSopt Linear Programming Solver* (Applegate et al., 2003). Compiling this rather old `C`-code can be challenging, especially when using Windows computers because the code is optimized for Windows 98/ME/NT/2000/XP, but it is possible even for new Windows 10 computers. The Concorde-solver is called by the `R`-function `solve_TSP()` of the `TSP`-package when setting the option `method = "concorde"`. When setting the option `method = "repetitive_nn"` the nearest neighbor approximation is used. Since the solution of the Traveling Salesman Problem and the Shortest Hamiltonian Path problem is based on the pairwise distances of the regression vectors a distance matrix has to be computed beforehand. This is done via the function `dist()`. All hyper-parameters of the `dist()`-function (like the distance measure) can be set via the `control`-argument of `multiSorting()`. When using the default the euclidean distance is used. Also via the `control`-argument the path of the executables of the Concorde-solver and the precision parameter (i.e. the number of decimal places used for the internal representation of distances in Concorde) can be set. On the other hand, the implementation of the hierarchical clustering for ordering is quite easy. For this, the function `hclust()` is applied on the distance matrix of the pairwise distances and the indices of the permutation are given in the list entry `order` of the returned object. The hyper-parameter for specifying the dissimilarity function can be given via the `control`-argument of `multiSorting()`. The default value for this is the complete linkage function.

4.4 The R-package GSignTest

For easy usage of the sign depth and the sign depth test including the ordering methods the R-package GSignTest (Horn, 2021) has been written as part of this thesis. Writing this package was made with the help of several R-packages: devtools (Wickham et al., 2019b) for technical infrastructure, testthat (Wickham, 2011) for testing the written functions on correctness, roxygen2 (Wickham et al., 2019a) for documentation and checkmate (Lang, 2017) for argument checks in the written functions. Next to the sign depth, the sign depth test and the ordering methods, GSignTest provides also functions for computing the F-test and the classical sign test as well as the density function, the probability function and the quantiles of the distribution of the sign depth with $K \in \{2, 3, 4, 5\}$ and a function for visualization of computed multidimensional orders.

The sign depth is implemented three times in the package GSignTest. The first and oldest implementation can be found in the function calcDepth_old(), which contains the recursive implementation presented in Algorithms 4.5 and 4.6 in Section 4.1. The function calcDepth_asymp() uses the before mentioned asymptotic implementation in linear time complexity in N based on the work of Dennis Malcherczyk, Kevin Leckey and Christine Müller. In default, the argument exact of this function is set to TRUE, which means that the exact solution of the sign depth based on the asymptotic version is computed for $K \in \{3, 4, 5\}$. If $K \notin \{3, 4, 5\}$, this function computes the asymptotic sign depth. The third function for computing the sign depth in GSignTest is named calcDepth() and contains the recently developed block-implementation.

As mentioned in Section 4.2 the sign depth test is implemented for two different situations: Either input of a vector of residuals or input of a data set and a model description. The function depth.test() is written as S3-method, so that both variants can be addressed via the same interface. The output of depth.test() is an object of class htest like most in R implemented tests. This was chosen on the one hand because of consistency to other tests implemented in R and on the other hand because for class htest there exists already some useful methods (for example a print()-method) which can be used for depth.test().

The package contains a table of values of the K-sign depth for each $K \in \{2, 3, 4, 5\}$ respectively. These four tables cannot be addressed directly, but are internally used

for the functions `qdepth()`, `ddepth()` and `pdepth()` which return quantiles, values of the density function and values of the probability function of the distribution of the K-sign depth. As mentioned in Section 4.2, up to $N = 25$ the sign depth of all possible combinations of signs of residuals were computed and for $25 < N \leq 100$ one million random sign combinations were used for approximating the values of the distribution of the sign depth. Since `R`-packages should not be too large, it was decided to include for every N and K only 10 001 values, so that all quantiles from zero to one with four decimal places are available. The function `qdepth()` directly accesses the respective table in which in each column the sorted values (i.e. quantiles) for one specific number of residuals are available. The values of the probability function are calculated with the help of the function `ecdf()` and the values of the density function are approximated via `density()`. All three functions have the possibility to return the transformed values $N \cdot (d_S^K(\hat{\boldsymbol{e}}(\boldsymbol{\theta})) - (\frac{1}{2})^{K-1})$.

As mentioned in Section 4.3 the function `multiSorting()` provides all ordering methods described in Chapter 3. For visualization of computed orders the function `plotMultiSorting()` can be applied to an object returned by `multiSorting()`. This function draws the regression vectors via `ggplot2` in a two-dimensional scatterplot and connects the points with lines which describe the ordering. When the ordered regression vectors have more than two dimensions the argument `dims` in `plotMultiSorting()` can be used to determine which two dimensions shall be shown in the plot.

The F-test and the sign test can be found in the self-written functions `f.test()` and `sign.test()`. Like the sign depth test, the sign test is implemented as `S3`-method for input of residuals as well as input of model description and data. Since the F-test in the context of regression is not defined for residuals only, but bases on an estimation of the parameter vector $\boldsymbol{\theta}$, the input of the function `f.test()` has always be a model description and a data set. Both, `f.test()` and `sign.test()`, return an object of class `htest` like `depth.test()` also does.

The whole package `GSignTest` can be found on the development platform Github via the address `https://github.com/melaniehorn/GSignTest`. The implementation of all mentioned functions can be found there. `GSignTest` can be freely downloaded, installed and used. The easiest way to install the package in `R` is to use the command `devtools::install_github("melaniehorn/GSignTest")`.

Chapter 5

Results and Analysis of the Power of the Sign Depth Test

This chapter shows, describes and analyzes the power of the sign depth test in many different situations of multiple linear regression and with different ordering methods. Most of the results are based on simulations which are described in the first section of this chapter. The simulated power functions and further comparisons of the different ordering methods are presented in the following sections. In addition, the sign depth test will be compared to classical (robust) tests in the context of parameter testing as well as for model checks. At the end of the chapter the sign depth test will be applied to real data from a bridge monitoring.

5.1 Description of the Simulations

For examining the behavior of the sign depth test when applying different ordering methods and in comparison to other tests, many simulations have been carried out. These simulations have examined many different aspects of the test, the data and the linear regression model. The aim of the simulations is to get empirical power functions of the sign depth test for the different situations and compare them. While the theoretical power function of a test is defined as the expected value of the test in dependance of the unknown parameter $\boldsymbol{\theta}$, i.e. $\gamma(\varphi) = E_{\boldsymbol{\theta}}(\varphi) = P_{\boldsymbol{\theta}}(\varphi = 1)$, one can obtain empirical power functions by replacing the expected value of the test by the relative number of rejections of the null-hypothesis among several simulation runs.

Overall, ten different regression models will be examined. In Section 5.2, the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ will be used and the results will be evaluated in detail. This model is the only possible multiple linear regression model with only two parameters. Because of this, it is possible to look at the simulated three-dimensional power functions directly without being forced to summarize the results any further. In Subsection 5.3.1 to 5.3.3 the summarized results of five further models with higher dimensional parameter vectors $\boldsymbol{\theta}$ will be shown. These models will belong to three groups: Simple linear models with intercept ($\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ and $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$), models with interactions without and with intercept ($\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ and $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$) and a quadratic regression model ($\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$). In Subsection 5.3.4, four high-dimensional models will be analyzed. These models have the form $\boldsymbol{y} = \sum_{d=1}^{D} \theta_d \boldsymbol{x}_{\bullet d} + \boldsymbol{e}$ with $D \in \{10, 20, 40, 80\}$.

The entries of the error vector $\boldsymbol{e}$ are chosen to be independent copies from a normal distribution with parameters $\mu = 0$ and $\sigma = 0.2$ in all simulations. Note that the choice $\mu = 0$ is necessary to have a median of zero in the errors whereas the standard deviation is chosen arbitrarily and could be replaced by any other positive value. In addition, the model in Section 5.2 is also analyzed for errors with a Cauchy distribution and a uniform distribution in order to examine the effect of the error distribution on the sign depth test. Both distributions were set up to have median zero and the same interquartile range as the normal distribution to make the results more comparable. Hence, if $u_{0.75}$ denotes the 75% quantile of the standard normal distribution then the parameters of the Cauchy distribution were chosen as $s = 0$ and $t = 0.2 \cdot u_{0.75} \approx 0.135$ while the ones of the uniform distribution were set to be $a = -0.2 \cdot 2 \cdot u_{0.75} \approx -0.27$ and $b = 0.2 \cdot 2 \cdot u_{0.75} \approx 0.27$.

Regarding the underlying data set $\boldsymbol{X}$, overall three different types of data sets have been used. Basically, in Section 5.2 the same data sets as in Chapter 3 were used: A data set where the design vectors are arranged as a spiral, a data set where the design vectors are arranged as a grid and a data set with random design vectors. For comparison see Figure 3.1 on page 30. All three data sets have values in the interval $[-1, 1]^2$. In Section 5.3 only the data set with random design vectors has been used because this data set can easily be extended to more than two dimensions if there are more than two regressors in the regression model. The simulations were carried out with two different numbers of data points: Rather small data sets with $N = 25$ and medium data sets with $N = 100$ data points.

The sign depth test itself depends on the ordering method and the parameter K. In Section 5.2 all ordering methods presented in Chapter 3 will be shown and compared whereas in Section 5.3 only the best performing ordering methods will be used. The simulation study is mostly focused on the parameter choices $K \in \{3, 4, 5\}$ since for these values the sign depth can be computed exactly in linear time complexity. In Subsection 5.3.4, also $K \in \{11, 21\}$ is considered because of the high dimensions of the models. For this, only an asymptotic implementation of the sign depth in linear time complexity can be used.

For the simulations the statistical hypothesis $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ is used. In Section 5.2, the power functions of the tests are simulated on the interval $[-1, 1]$ for each of the two components of $\boldsymbol{\theta}$. For this, the interval is discretized to the values $\boldsymbol{\theta} \in \{-1, -0.95, -0.9, \ldots, 0.9, 0.95, 1\}^2$. In Subsections 5.3.1 to 5.3.3 the power functions are only simulated on the interval $\{-0.5, -0.45, -0.4, \ldots, 0.4, 0.45, 0.5\}^D$ because it turned out in the previous section that this is sufficient to see the general behavior of the test and thus the runtime can be reduced significantly by considering less points, especially because the dimension of $\boldsymbol{\theta}$ can be up to four in this section. In contrast, in Subsection 5.3.4, the power functions are simulated in the range of -1 to 1 with a step width of 0.02 because in this subsection only two one-dimensional aspects of the power functions are of interest and not the whole power functions and so it is computationally possible to simulate the power functions on a larger interval with a smaller step width. Computing the whole power functions in this section is not possible because of the high dimensions. This would computationally too expensive.

All power functions are simulated by conducting the tests 1 000 times on independently generated data sets and calculating the relative number of simulation runs which have rejected the null-hypothesis. In this thesis, always $\alpha = 0.05$ for the level of the tests is used.

5.2 Results of Model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$

We will start with the most simple multiple regression model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. When having only a two-dimensional parameter vector $\boldsymbol{\theta}$, it is possible to look at the simulated three-dimensional power functions in detail. In the following subsections, every aspect which could have an effect on the power of the sign depth test will get

analyzed: The number of data points, the underlying data set, the distribution of the errors and the parameter K of the sign depth test. Afterwards, some further analysis of the results and a summary of them is given.

5.2.1 Effect of the Number of Data Points on the Results

At first, the effect of the number of data points on the simulated power functions gets analyzed. It should be not surprising that, usually, the power increases in the area of the alternative hypothesis when the number of data points (and so the number of regression vectors) gets larger. So, the number of data points will have a large effect on the power and the power functions of the sign depth test.

In the following, it is looked at simulated power functions with $N = 25$ and $N = 100$ regression vectors in the underlying data set. Because it would be too much to present really all results in detail, only the power functions on the data set with random values in the regression vectors $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$, with normally distributed errors $\boldsymbol{e}$ and with parameter $K = 3$ of the sign depth test are shown. As it will be shown in the next subsections, the data set and the error distribution do not have a large effect on the power of the test anyway and the parameter $K = 3$ for the test is chosen because for the two-dimensional parameter vector $\boldsymbol{\theta}$ the 3-sign depth is equivalent to the simplicial depth which was one of the origins for the research about sign depths.

Figure 5.1 shows the simulated power functions for both naive ordering methods presented in Section 3.1. Since the power functions are three-dimensional, the power is shown as color in the two-dimensional plots from black (low power) to white (high power), a so-called "heatmap". In addition, points which have a simulated power less than or equal to the level $\alpha = 0.05$ are marked in red. A "good" power function would have a power less than or equal to α at H_0 and the power would increase in every direction and converge to one in the area of the alternative hypothesis. As it can be seen in Figure 5.1, these characteristics do not apply to the simulated power functions of the naive ordering methods. Here, the power is always about 0.05, independent of the values of θ_1 and θ_2 and the number of data points. It is not surprising that both methods behave the same here since the underlying data set consists of random values and so, shuffling these values again for getting a "Random Order" has no effect on the power function. All in all, it can be seen that the naive ordering methods perform very poorly and should not be used when applying the

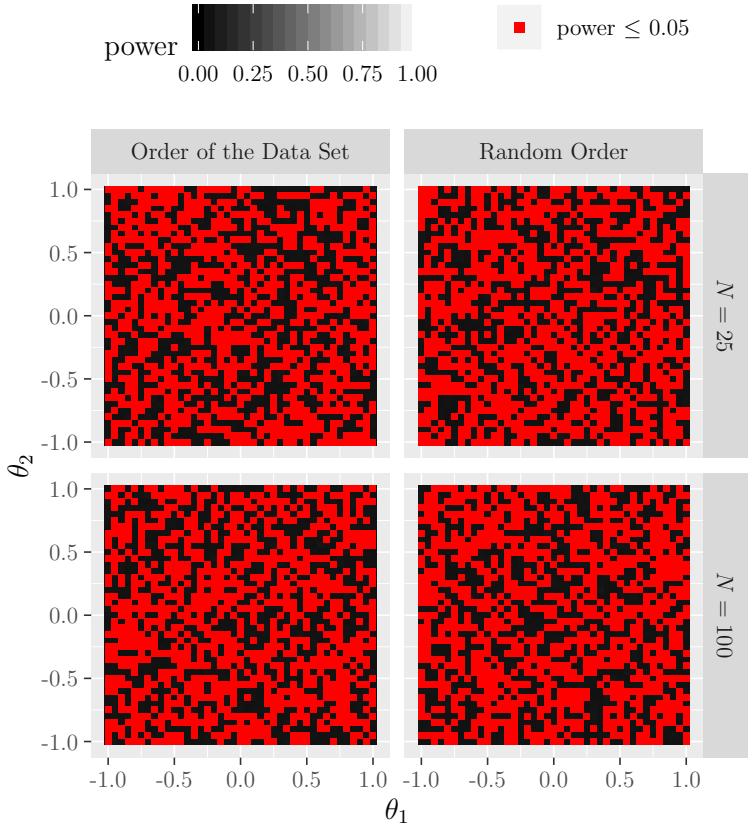


Figure 5.1: Simulated power functions of the naive ordering methods presented in Section 3.1 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ and the error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

sign depth test to some data. This is an important result because it shows that one has to think about the ordering of the regression vectors when using the sign depth test. Additionally, it shows that this thesis is important for the possibility of applying the sign depth test in the context of multiple regression. It clearly shows that thinking about the ordering method is necessary and a simple random order is not sufficient for getting a good test. Also, taking the order the regression vectors appear in the data set is bad, at least when the regression vectors do not follow an inherent order.

When looking at Figure 5.2, which shows the simulated power functions for the scalarization based methods described in Section 3.2, it can be seen that ordering

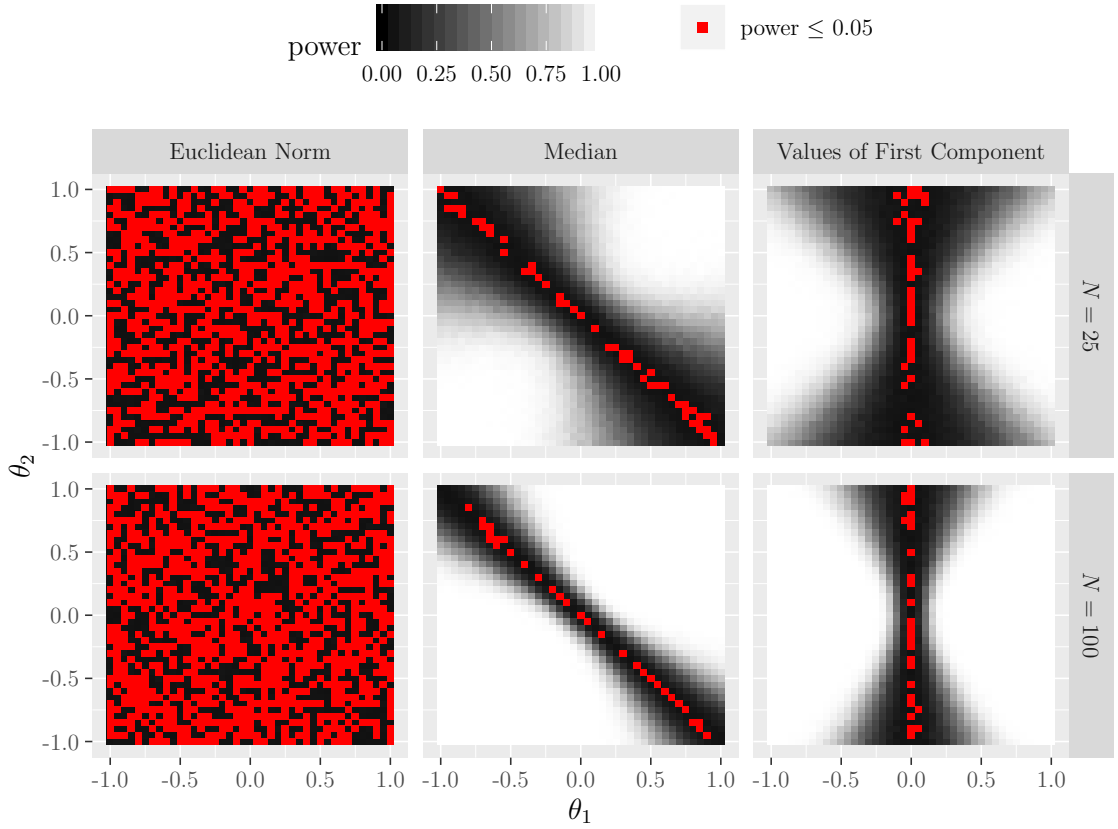


Figure 5.2: Simulated power functions of the scalarization based ordering methods presented in Section 3.2 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ and the error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The ordering methods based on calculating a weighted sum and projecting all regression vectors orthogonal on a line are not plotted here because the results are identical to calculating the median when setting all weights to an equal value or all entries in the direction vector have equal values, respectively.

the regression vectors according to the values of a vector norm leads to the same bad results as the naive ordering methods. Here, and in all following subsections, the vector norms are calculated with parameter $p = 2$ which leads to the euclidean norm. Choosing different values for p would have led to similar results, see Figure 5.18 on page 134. The small power of the euclidean norm in Figure 5.2 can be explained by the fact that the underlying data set has values in the range $[-1, 1]^2$ and so the

smallest values of the norm are in the center of the data set and the largest values are on the edge of the data set. Because of this, the distance between the regression vectors which are ordered next to each other gets larger and larger and the ordering looks more like a random order without any sorting in it, for this see also the left plot in the middle row of Figure 3.6 on page 37. If the same data would be shifted to the range $[0, 2]^2$ (see left plot of Figure 3.7 on page 38), the power function of the sign depth test would look different, see middle plot of Figure C.1 on page 228. This shows that it is important to think about the ordering method one wants to use because some ordering methods are very sensitive to the underlying data set. More on this topic can be read in Subsection 5.2.2.

In the middle column of Figure 5.2, the simulated power functions for ordering the regression vectors according to their median value can be seen. One can easily realize that in the case where $\boldsymbol{\theta}$ is only two-dimensional, an ordering according to the median values leads to the same ordering like an ordering according to a weighted sum when all weights are equal and like an ordering based on an orthogonal projection of the regression vectors when the direction vector consists of equal values. Because of this, the results for an ordering according to a weighted sum and according to an orthogonal projection are not shown in Figure 5.2. It can be seen that the power functions do look better than the power functions of the naive methods and the euclidean norm. But the power is not only at $\boldsymbol{\theta} = \mathbf{0}$ less than or equal to 0.05, but also on a line which is orthogonal to the "direction of ordering". If we look at the upper row of Figure 3.9 on page 40 for example, it can be seen that the ordering starts at the the lower left corner of the plot and ends at the upper right corner of the plot (or vise versa). Such a "direction of ordering" leads to the fact that the test has low power in the orthogonal direction. When looking at Figure 2.5 on page 25 this behavior is getting clear: The model (marked as dark gray plane in this figure) can be rotated around a specific line without changing the signs of the residuals (or nearly all signs are changed simultaneously). So, there is always a line with power values about $\alpha = 0.05$ in all power functions which have this "direction of ordering" which is a big disadvantage of these ordering methods. Since ordering according to the values of the median has no further hyper-parameters, this line is always located where it can be seen in Figure 5.2, whereas the direction of this line can be affected by the hyper-parameters of an ordering according to a weighted sum or an orthogonal projection, see Figure 5.20 on page 136. When changing the weights in a weighted sum or the direction of the projection, the line with low power runs differently. This can be also seen when looking at the right column of Figure 5.2. Here, an ordering

according to only the first component of every regression vector is made and the values of the second component are neglected. Also, this ordering method can be regarded as an ordering according to a weighted sum (all but one of the weights are zero) or an ordering according to an orthogonal projection (projecting all regression vectors on a line parallel to the first axis) and it can be seen that the line with low power runs in a different direction than it does when using the median.

In both columns of Figure 5.2, the middle column and the right column, it can be seen that outside above mentioned line the power functions have greater power values when the number of data points is greater. Also, the line with low power values gets tighter the more data points are available. So, in contrast to all former described results, here the number of data points has a crucial effect on the power functions.

In Figure 5.3 the power functions of the methods based on partial sorting are shown. The left column shows the simulated power functions when using a nondominated sorting as ordering method. Like the scalarization based methods (except the ordering according to the euclidean norm) the nondominated sorting has a "direction of ordering" (see Figure 3.19 on page 53). Because of this, also these power functions have a direction with low power values. While for $N = 25$, in general the power values seem to be smaller than the respective power values of both scalarization based methods, for $N = 100$ the power function looks better than the ones of the scalarization based methods. Especially the number of points with power less than or equal to 0.05 is smaller. This is caused by the fact that there is a little bit more "mess" in the ordering than it is for the scalarization based methods. Because of this, the model cannot be rotated so much without affecting the signs of single residuals.

The power functions of the convex hull method and the ordering according to the values of Tukey's halfspace depth look very different than the power functions of the nondominated sorting method. The power is everywhere really low and often less than 0.05. This is a phenomenon quite similar to it was for the euclidean norm method: The ordering starts at the edges of the data set and ends in the center (or vise versa). Because of this, there is no real structure in the ordering and the power gets low. In contrast to the ordering on the basis of the euclidean norm, here the data cannot be shifted to a different interval of values for increasing the power of the sign depth test because the ordering does not depend on the location of the values directly. Interestingly, the power values get smaller when the number of data points increases. This is a behavior which is totally undesirable, but it can be explained logically: For a rather small number of data points (e.g. $N = 25$), there are less

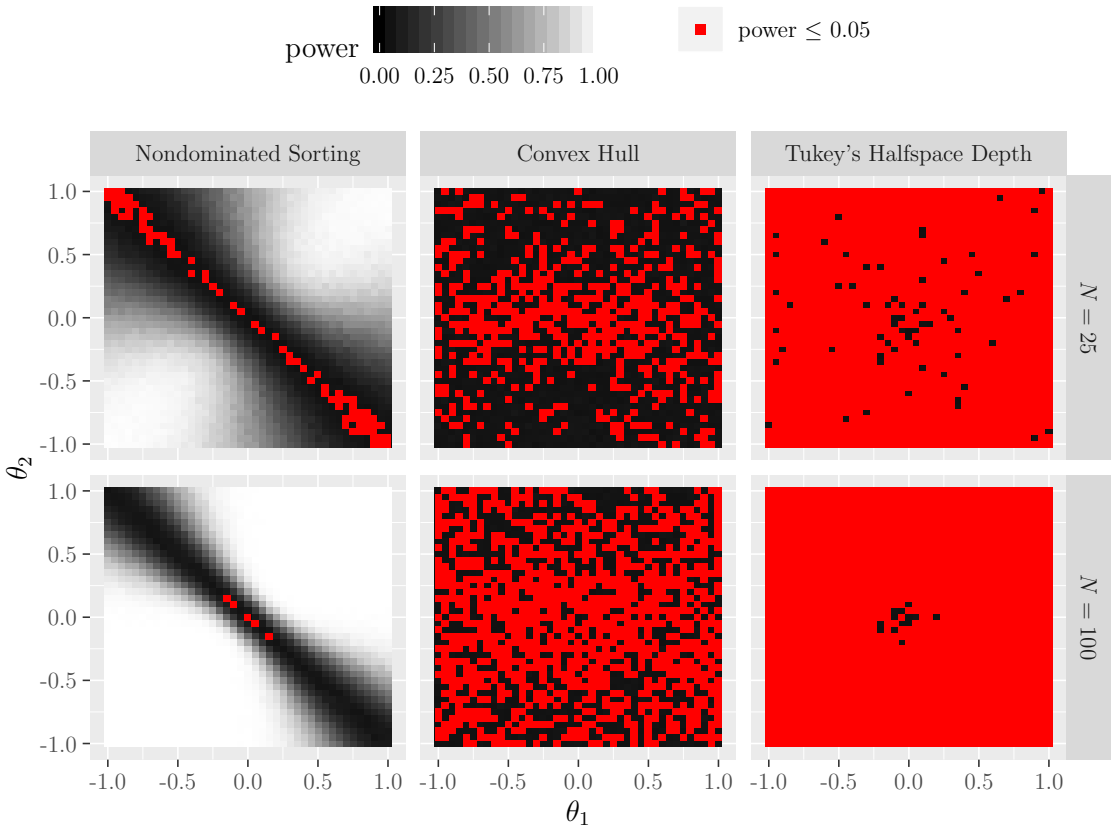


Figure 5.3: Simulated power functions of the ordering methods based on partial sorting presented in Section 3.3 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ and the error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The regression vectors with the same rank of the nondominated sorting method are ordered according to their appearance in the data set, which is a random order here.

different ranks (i.e. less convex hulls or regression vectors with the same value of Tukey's halfspace depth) than for a larger number of data points (e.g. $N = 100$). Because of this, for fewer data points the "mess" in the order is not as large as it is for larger numbers of data points because not so many transitions between the regression vectors with different ranks are needed. But because this behavior is totally undesirable, these both ordering methods should not be used when applying the sign depth test to data.

In Figure 5.4 the power functions of the distance based ordering methods are shown. As it can be seen, all three methods produce satisfying results. The power is about $\alpha = 0.05$ at H_0 for all power functions and the power increases the larger the absolute values of θ_1 and θ_2 are. Obviously, the power in the area of H_1 is greater for $N = 100$ than for $N = 25$, which is the desired behavior. The best of the three methods seems to be the Shortest Hamiltonian Path method whose power values are (slightly) greater than the values of the hierarchical clustering and the nearest neighbor approximation.

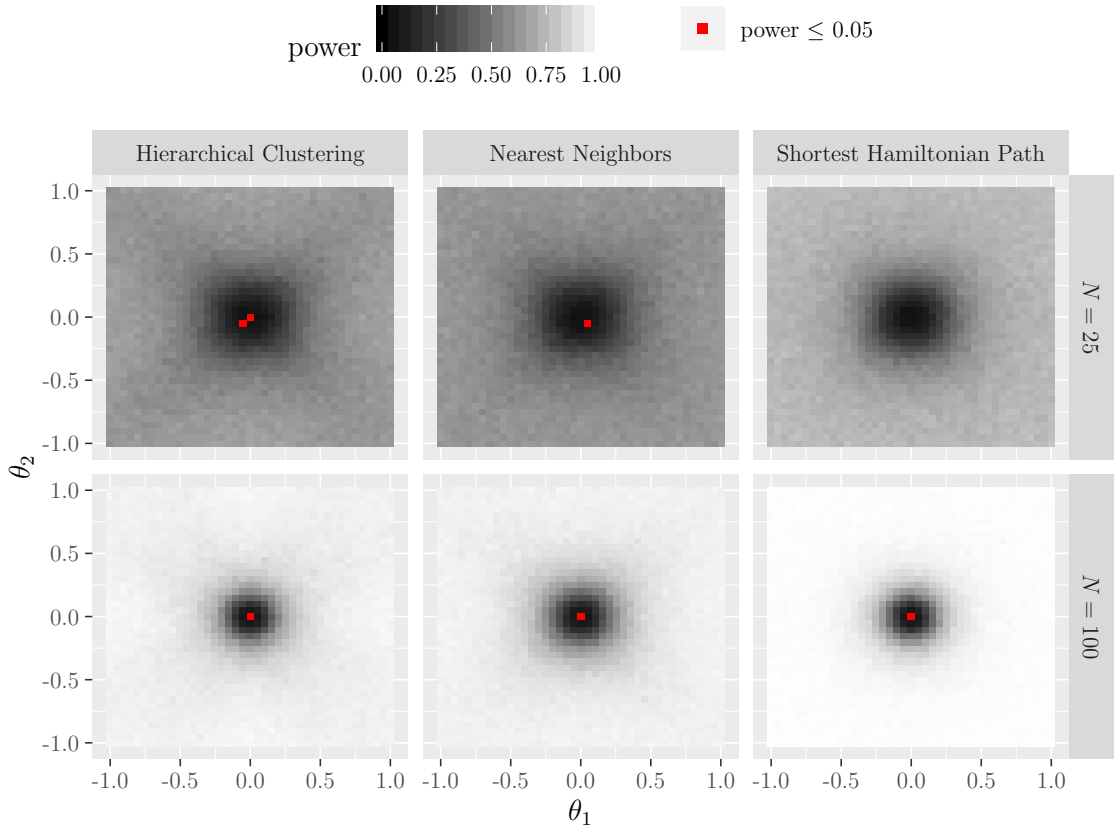


Figure 5.4: Simulated power functions of the distance based ordering methods presented in Section 3.4 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ and the error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). For the hierarchical clustering method a complete linkage was chosen to order the regression vectors.

But overall, all three power functions are satisfying which was not the case for any other ordering method.

As a conclusion, it can be said that the number of data points has a crucial effect on most of the ordering methods. The effect on the convex hull method and the ordering according to the values of Tukey's halfspace depth is the opposite of the desired effect that the power should increase when the number of data points increases. So, these ordering methods perform really badly in this situation. The effect of the number of data points on the naive ordering methods and on the ordering according to the values of the euclidean norm is rather small or even non-existent. The other scalarization based methods and the nondominated sorting perform better, but have a direction in which the power is low, independent of the number of data points. Only the distance based methods produce really good results. The power functions of these methods fulfill all criteria to be satisfying power functions of a statistical test. Here, the method based on the exact solution of the Shortest Hamiltonian Path problem performs best.

5.2.2 Effect of the Data Set on the Results

In this subsection the effect of the underlying data set on the sign depth test gets analyzed. A good statistical test should perform well independently from the data set, but as it could be guessed from the results of the previous subsection the results of at least some ordering methods will get affected by the fact whether there is an inherent order in the data set or not. In this subsection all data sets consist of $N = 100$ data points, the model has normally distributed errors and the parameter of the sign depth test is set to $K = 3$. The three used data sets have different characteristics each: The "Random" data set consists of purely random regression vectors, the "Grid" data set arranges its regression vectors on a two-dimensional grid and the "Spiral" data set arranges all regression vectors on a two-dimensional spiral. While there is no inherent order in the "Random" data set, the "Spiral" data set has an inherent two-dimensional order. Although the regression vectors in the "Grid" data set also have a structure, there are only one-dimensional inherent orders in both dimensions, but the data set has no inherent two-dimensional order.

Figure 5.5 shows the simulated power functions of the naive ordering methods for these three different types of data sets. Taking the order the regression vectors appear

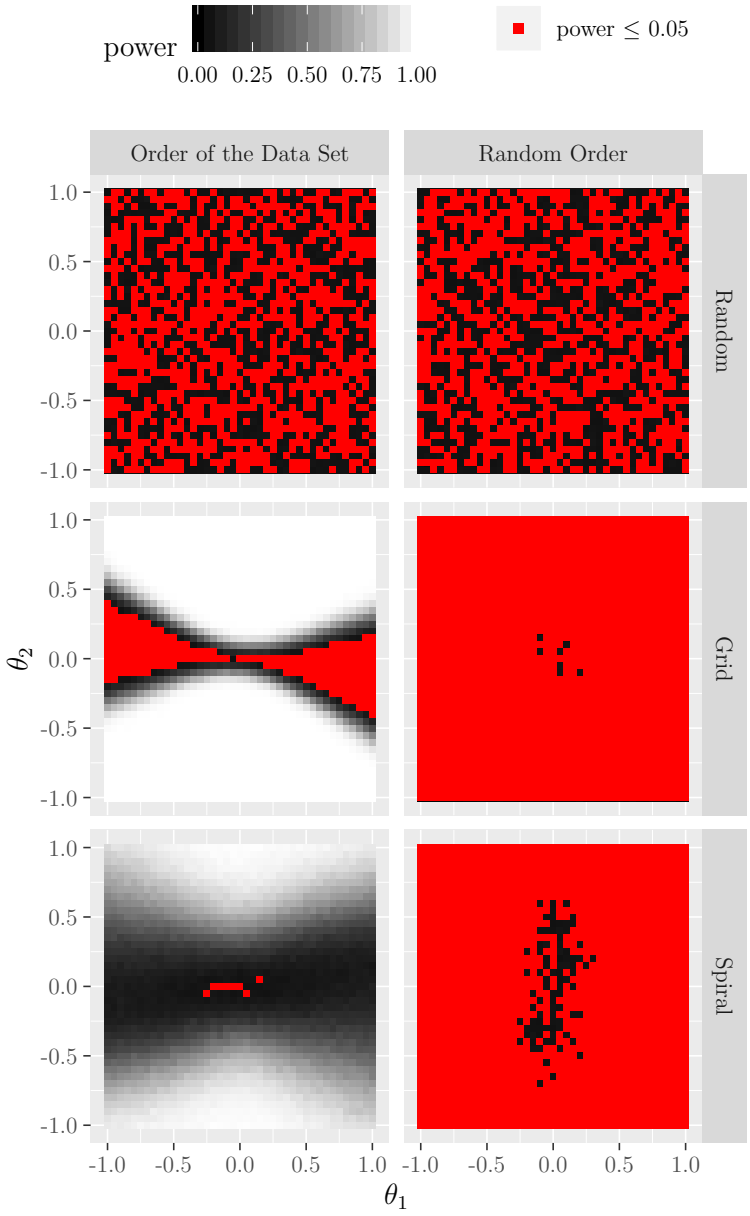


Figure 5.5: Simulated power functions of the naive ordering methods presented in Section 3.1 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. All data sets consists of $N = 100$ data points. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

in the data set leads to bad results when having random regression vectors, this was also seen in the previous subsection. How the regression vectors of the "Grid" data set are arranged can be seen in Figure 3.2 on page 32. As described in the previous subsection, there is a "direction of ordering" in this arrangement which is the reason for the low power in the orthogonal direction of this direction of ordering. But in all other directions the power is quite large. The regression vectors of the "Spiral" data set are arranged according to their inherent order in the data set. So, all ordering methods which find this inherent order in this data set will have the same power function. Unfortunately, this power function has also large areas with low power. But this phenomenon can be easily explained when looking at Figure 5.6. Here, the data situation for the sign depth test is visualized for a parameter vector from the alternative hypothesis ($\theta_1 = -1$ and $\theta_2 = 0$). For a better visualization the error vector $\boldsymbol{e}$ is neglected in this figure. The null-hypothesis ($\theta_1 = \theta_2 = 0$) is displayed as gray plane. It can be seen that there are only four sign changes in the residuals when ordering the regression vectors according to their inherent order. Because of this, there are many 3-tuples of residuals with alternating signs and so the 3-sign depth

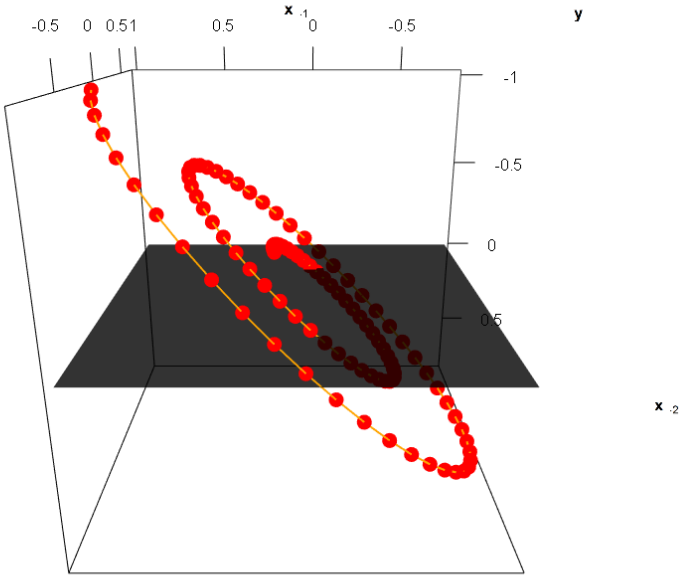


Figure 5.6: Visualization of the situation for the sign depth test in case the regression vectors are arranged according to a spiral and $\boldsymbol{y} = -1 \cdot \boldsymbol{x}_{\bullet 1} + 0 \cdot \boldsymbol{x}_{\bullet 2}$ holds. The null-hypothesis $H_0 : \boldsymbol{\theta} = \mathbf{0}$ is visualized as gray plane.

has a quite large value. To avoid a large value of the sign depth and so a low power of the sign depth test, there are two possibilities: The number of data points can be increased or the parameter K of the sign depth can be increased. When having more data points the relative number of alternating tuples decreases and so the value of the sign depth decreases. And when increasing the parameter K, more alternating residuals are needed for increasing the value of the sign depth which also leads to a lower value of the sign depth in this situation. The effect of both, the number of data points N and the parameter K, is visualized in Figure C.2 on page 229. There, it can be seen that the power functions get better when increasing N and/or K and that $N = 100$ data points and a value of $K = 4$ is sufficient to obtain a good power function.

So, although the power of the 3-sign depth test for the "Spiral" data set is not the best in Figure 5.5, the simulated power function is still better than for many other ordering methods. For example, it can be also seen in Figure 5.5 that destroying an inherent order by ordering the regression vectors randomly is a very bad idea. For the "Grid" data set and the "Spiral" data set nearly all simulated power values are less than or equal to α which leads to very bad power functions.

Figure 5.7 shows the results for the scalarization based methods. As in the previous subsection the results of the ordering according to a weighted sum and according to an orthogonal projection are not shown because they are identical to an ordering according to the median values. The figure shows the expected result: The ordering according to the median value and according to only one component of each regression vector have a direction with low power values and all other areas of the alternative hypothesis have rather great power values. Although the number of points with power less than or equal to α is smaller for the "Random" data set than for the other two data sets, the areas with rather small power values are approximately the same size for all three data sets. The transition from low to great power values is more sudden for the "Grid" data set and the "Spiral" data set because there is less variance in the data sets than in the "Random" data set. The euclidean norm method performs very poorly for the "Random" data set and the "Grid" data set. Especially the power values for the "Grid" data set are nearly always less than or equal to α which shows that the obtained order in this data set is rather messy (for this see also Figure 3.6 on page 37). For the "Spiral" data the euclidean norm works perfectly, at least it finds the inherent order of the data set. As mentioned before, unfortunately finding this inherent order leads to a power function which is not completely satisfying.

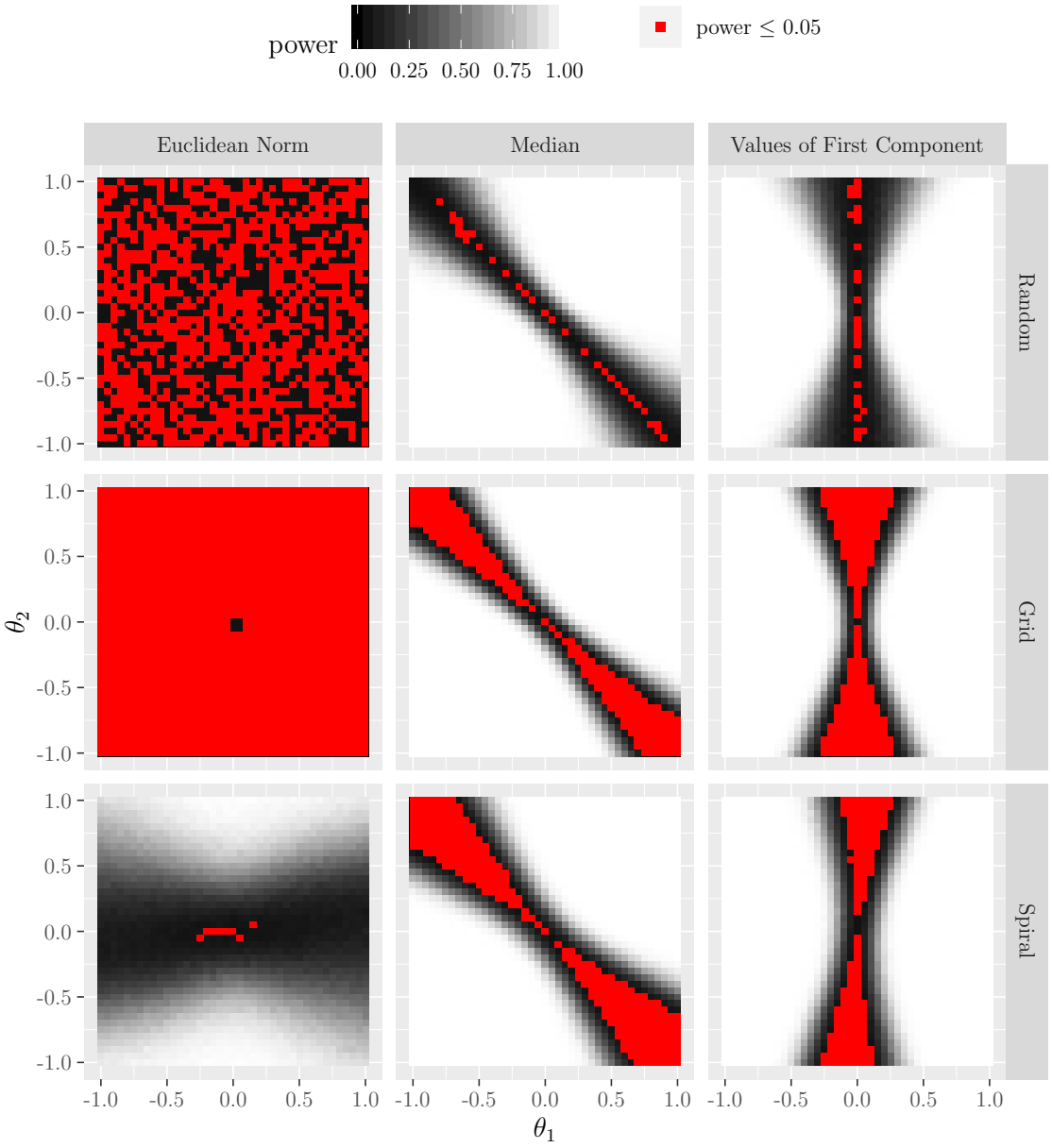


Figure 5.7: Simulated power functions of the scalarization based ordering methods presented in Section 3.2 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. All data sets consist of $N = 100$ data points. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The orderings according to a weighted sum and projecting all vectors orthogonal on a line are not plotted here because the results are identical to calculating the median when setting all weights to an equal value or all entries in the direction vector have equal values, respectively.

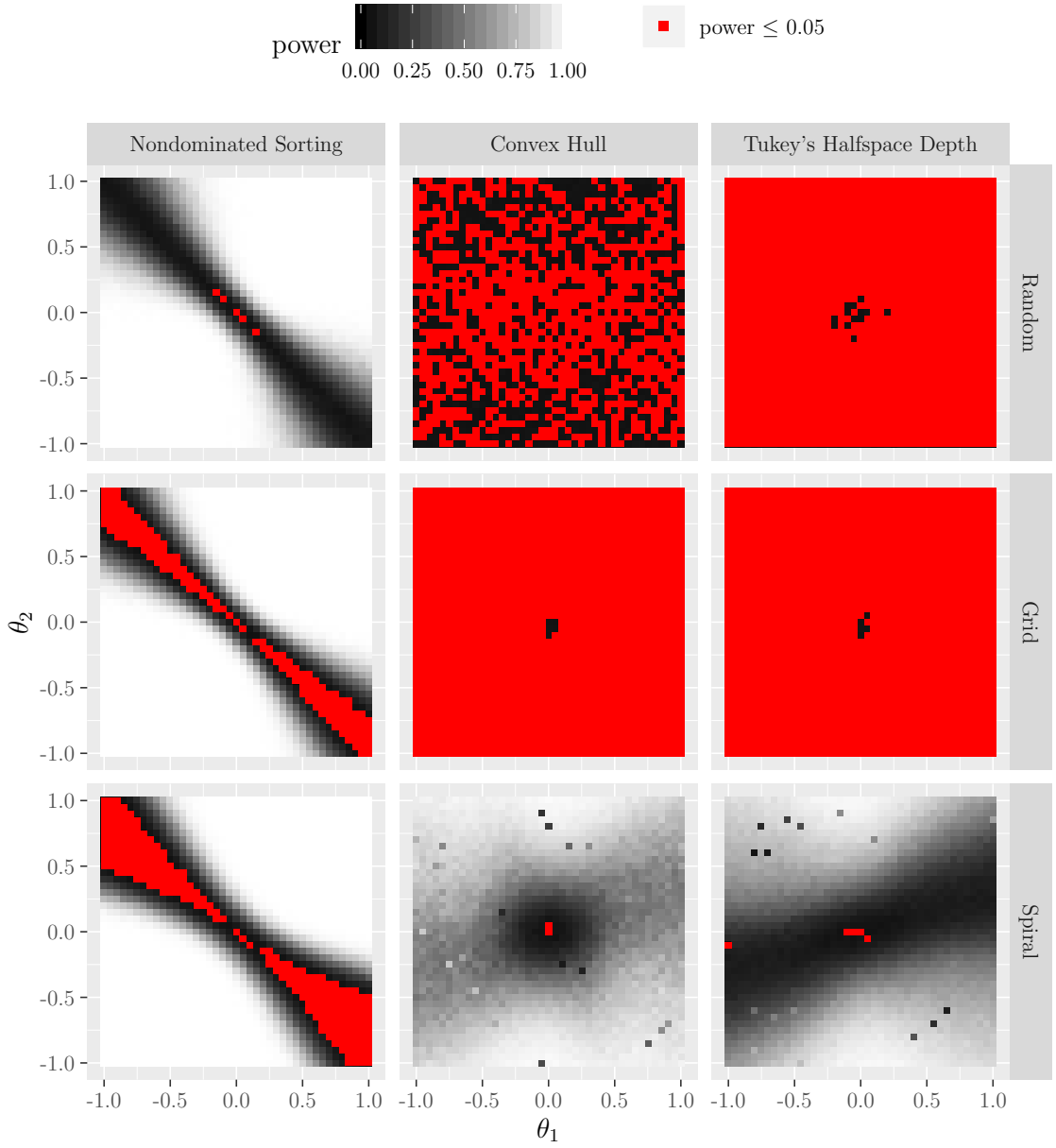


Figure 5.8: Simulated power functions of the ordering methods based on partial sorting presented in Section 3.3 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. All data sets consist of $N = 100$ data points. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The regression vectors with the same rank of the nondominated sorting method are ordered according to their appearance in the data set.

In Figure 5.8 the simulated power functions for the ordering methods based on partial sorting are shown. It can be seen that the nondominated sorting method performs comparatively well. Of course it has low power in the orthogonal direction of the "direction of ordering", but in all other directions the power values are quite satisfying. The power functions of the orderings based on convex hulls and on Tukey's halfspace depth are very bad for the "Random" data set and the "Grid" data set. As mentioned before this is because of the "mess" in the orders. The power functions for the "Spiral" data set are much better. Here, the structure of the data set is an advantage for these two ordering methods because these methods order the data from the edges to the center of a data set and so there is rather little mess in the obtained orders. Interestingly, for both ordering methods there are some values in the power functions of the "Spiral" data which are much lower than the values next to them. This can be explained by the fact that both methods need rather complicated algorithms for calculating their results which may not converge in single cases. It seems that this was the case for single values here which is the reason for the clearly lower power values at some random values. Furthermore, it can be seen that the convex hull method seems to perform a little bit better than the halfspace depth method. But overall, both methods cannot be recommended to use when applying the sign depth test to some data. Also, the nondominated sorting method cannot be fully recommended, especially because its power functions are not better than some of the scalarization based methods but the computational runtime for computing the order based on a nondominated sorting is much larger than the runtime of all scalarization based methods, see for example Table C.1 starting on page 221.

A better performance is achieved by the distance based methods. Its simulated power functions can be found in Figure 5.9. It can be seen that all three methods achieve satisfying power functions for the "Random" data set and the "Grid" data set. The best performance is achieved by the Shortest Hamiltonian Path method. Its approximation and the hierarchical clustering method perform slightly worse, but are overall very satisfying. The results on the "Spiral" data set are quite interesting. It is no surprise that the Shortest Hamiltonian Path method finds the inherent order in the data set and so the power function is not fully satisfying. Also the nearest neighbor approximation has a similar power function. It is quite surprising that this power function is slightly different than the power function of the exact method because for this data situation the nearest neighbor algorithm should always find the inherent order in the data set. In some rare simulation runs this was obviously not the case which is quite interesting and cannot be explained so far. But the majority

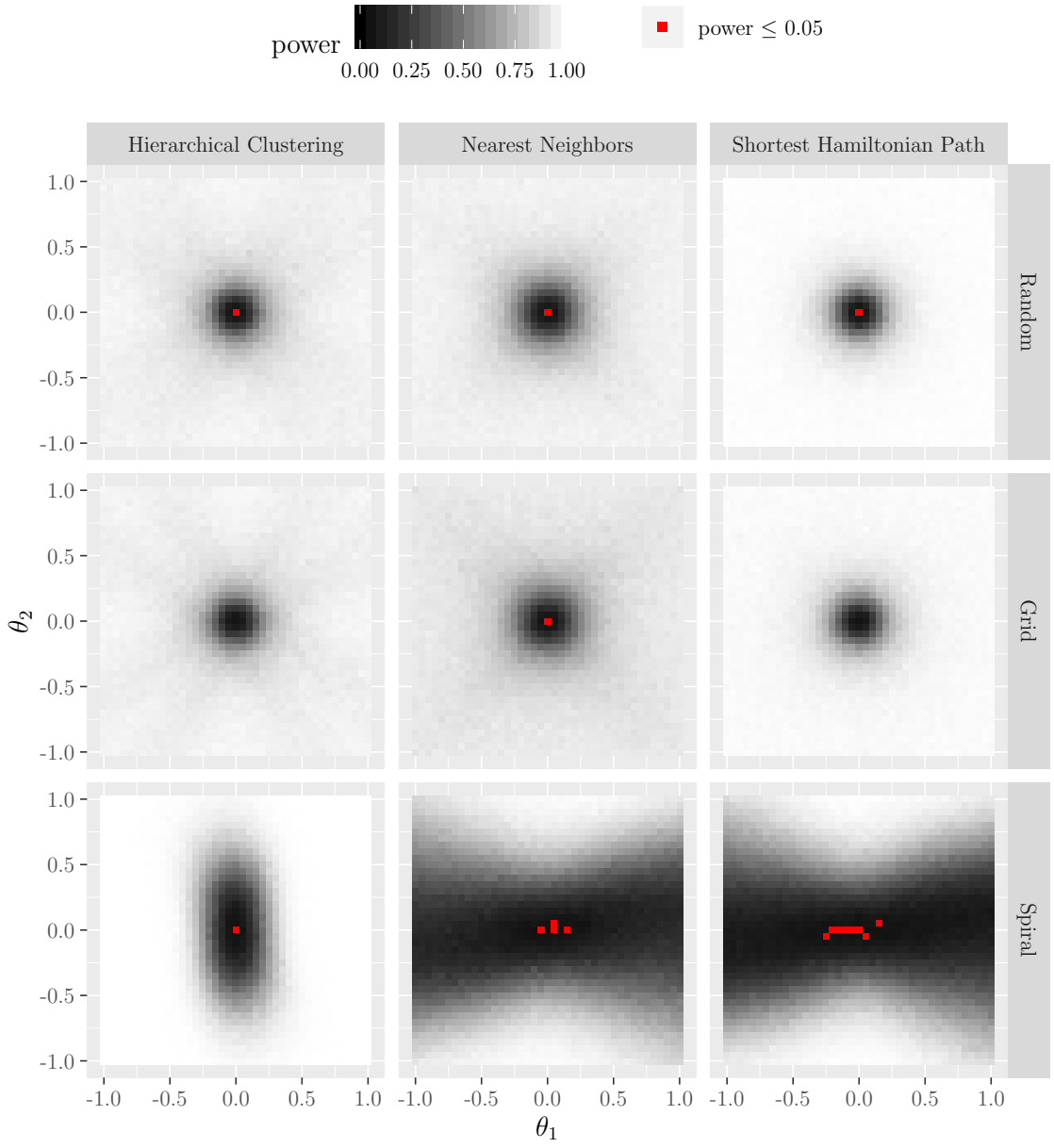


Figure 5.9: Simulated power functions of the distance based ordering methods presented in Section 3.4 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. All data sets consist of $N = 100$ data points. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). For the hierarchical clustering method a complete linkage was chosen to order the regression vectors.

of the simulation runs has found the inherent order and so the power functions of both methods are nearly identical. The power function of the hierarchical clustering method is very interesting. For the clustering a complete linkage is used. As it can be seen in Figure 3.34 on page 79, this method does not find the inherent order of the data set. But, since there is rather little mess in the order, the simulated power function is really satisfying. This order avoids the above mentioned problem with the few sign changes in the ordered residual vector. So, as it can be seen here, finding the inherent order in the data leads not necessarily to the best performance (although the problem with the inherent order in this data set can be solved by increasing the number of data points or the parameter K of the sign depth, as mentioned above).

As a conclusion for this subsection, it can be said that the choice of the data set can have an effect on the performance of the sign depth test. The ordering methods which perform quite well (i.e. the distance based methods) are barely affected. However, the performance on the "Spiral" data set is somehow special because here finding the inherent order only leads to good performances when increasing the number of data points and/or the value of the parameter K of the sign depth. Regarding the ordering methods based on a partial sorting, only the performance of the nondominated sorting method is not much affected by the data set, whereas the convex hull method and the halfspace depth method show different behavior for each of the three data sets. But overall, all three methods cannot be recommended because they have areas with low power. The same holds for the scalarization based methods. While an ordering according to the median values and according to only one component of the regression vectors perform similar for all three data sets, an ordering according to the euclidean norm only leads to satisfying results for the "Spiral" data set. The performance of the naive ordering methods is quite bad and especially the performance of an ordering according to the order the regression vectors appear in the data set depends heavily on the data set. So, overall, only the distance based methods can be recommended to use when applying the sign depth test.

5.2.3 Effect of the Error Distribution on the Results

Next, the effect of the error distribution on the simulated power functions will get analyzed. For this, the entries of the error vector $\boldsymbol{e}$ in the simulations will follow three different distributions: A normal distribution $\mathcal{N}(\mu, \sigma^2)$ with parameters $\mu = 0$ and $\sigma^2 = 0.2^2 = 0.04$, a Cauchy distribution $Cau(s, t)$ with parameters $s = 0$ and

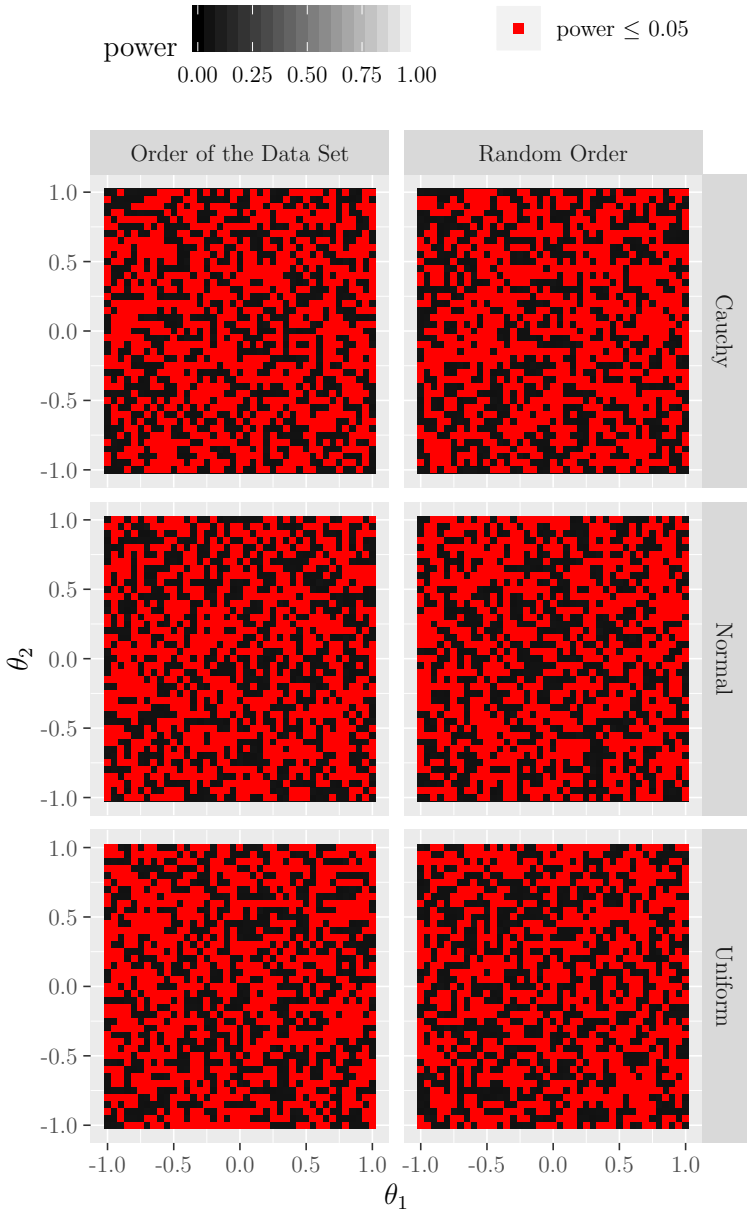


Figure 5.10: Simulated power functions of the naive ordering methods presented in Section 3.1 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

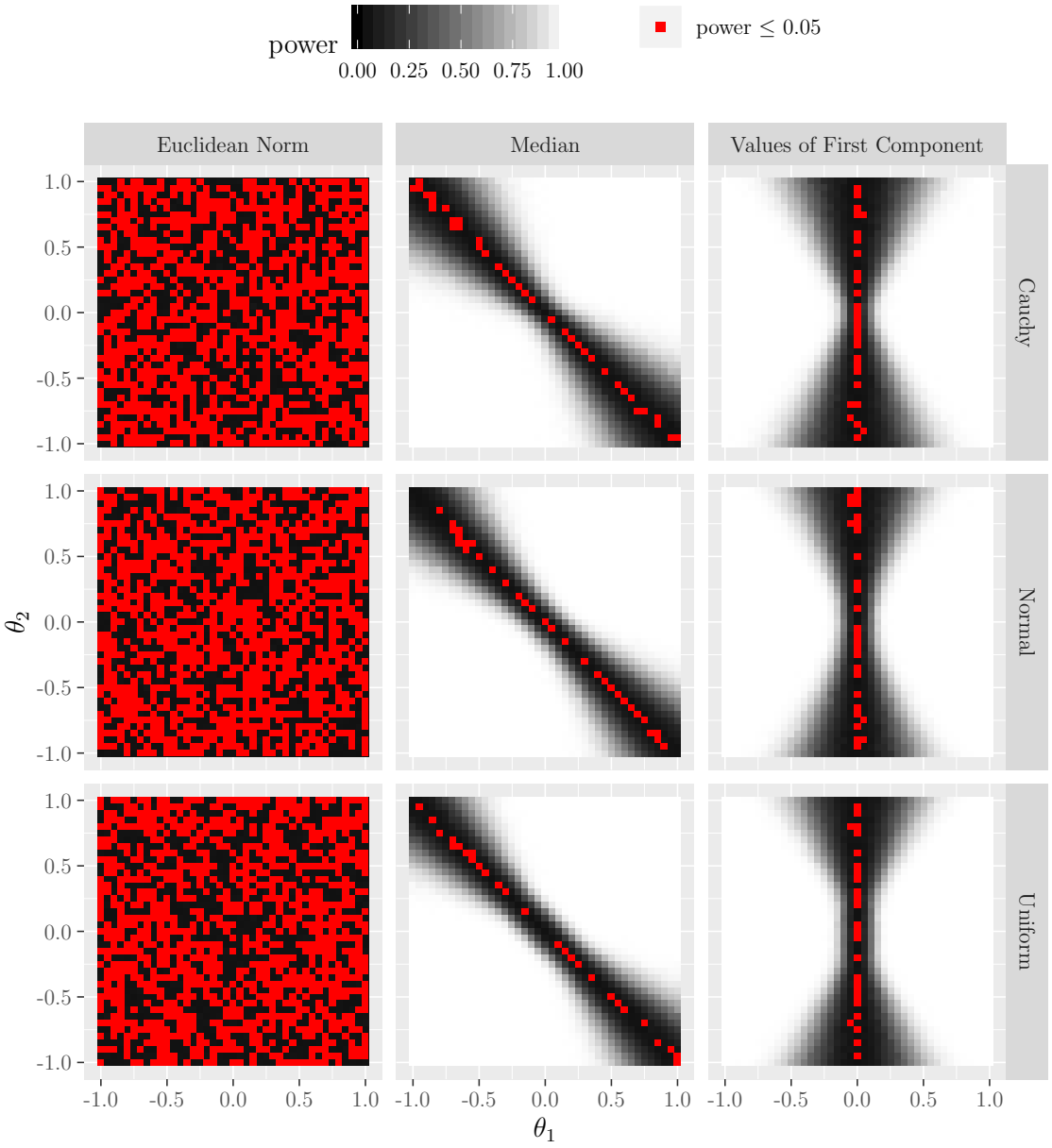


Figure 5.11: Simulated power functions of the scalarization based ordering methods presented in Section 3.2 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The orderings according to a weighted sum and projecting all regression vectors orthogonal on a line are not plotted here because the results are identical to calculating the median when setting all weights to an equal value or all entries in the direction vector have equal values, respectively.

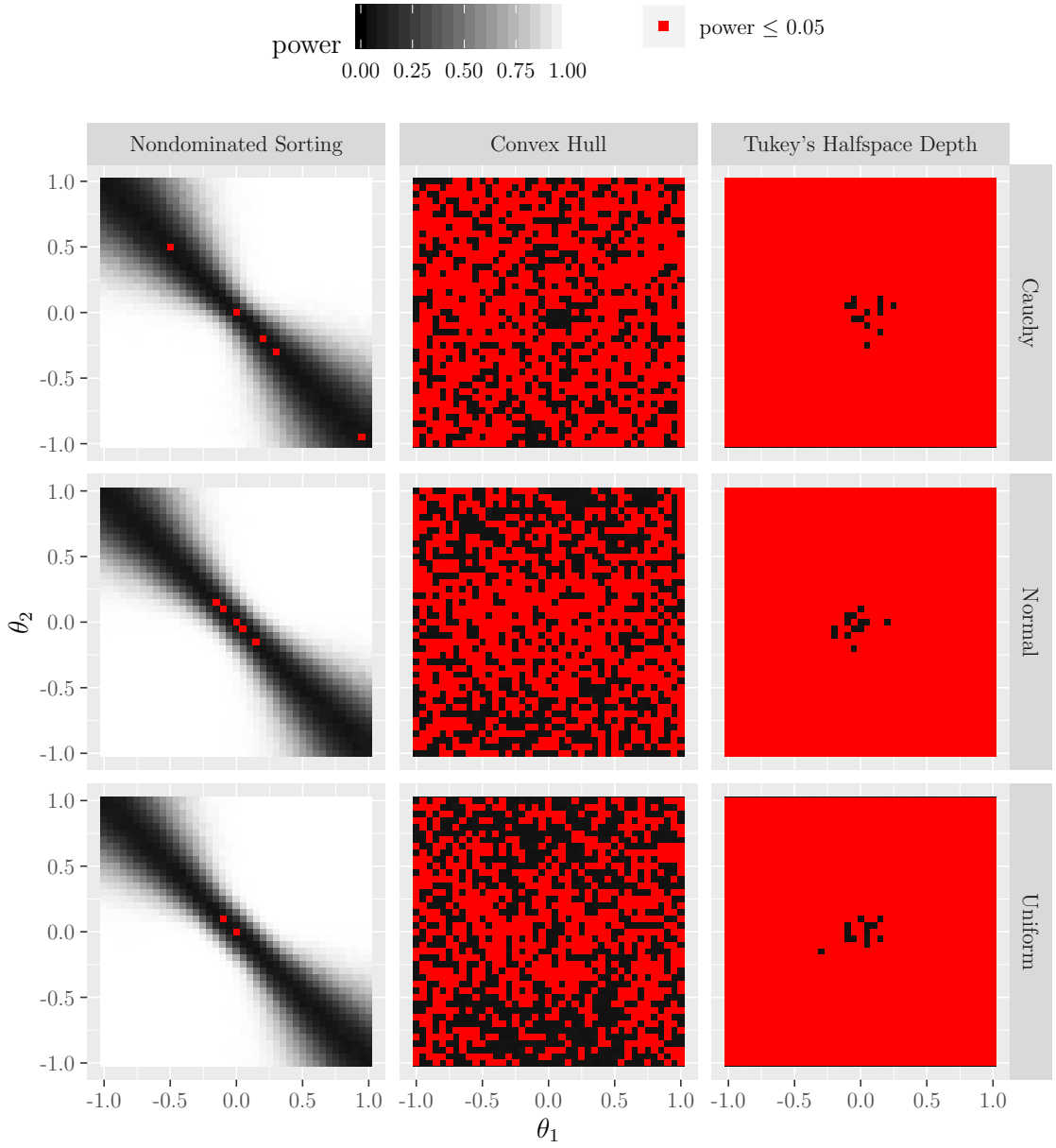


Figure 5.12: Simulated power functions of the ordering methods based on partial sorting presented in Section 3.3 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The regression vectors with the same rank of the nondominated sorting method are ordered according to their appearance in the data set, which is a random order here.

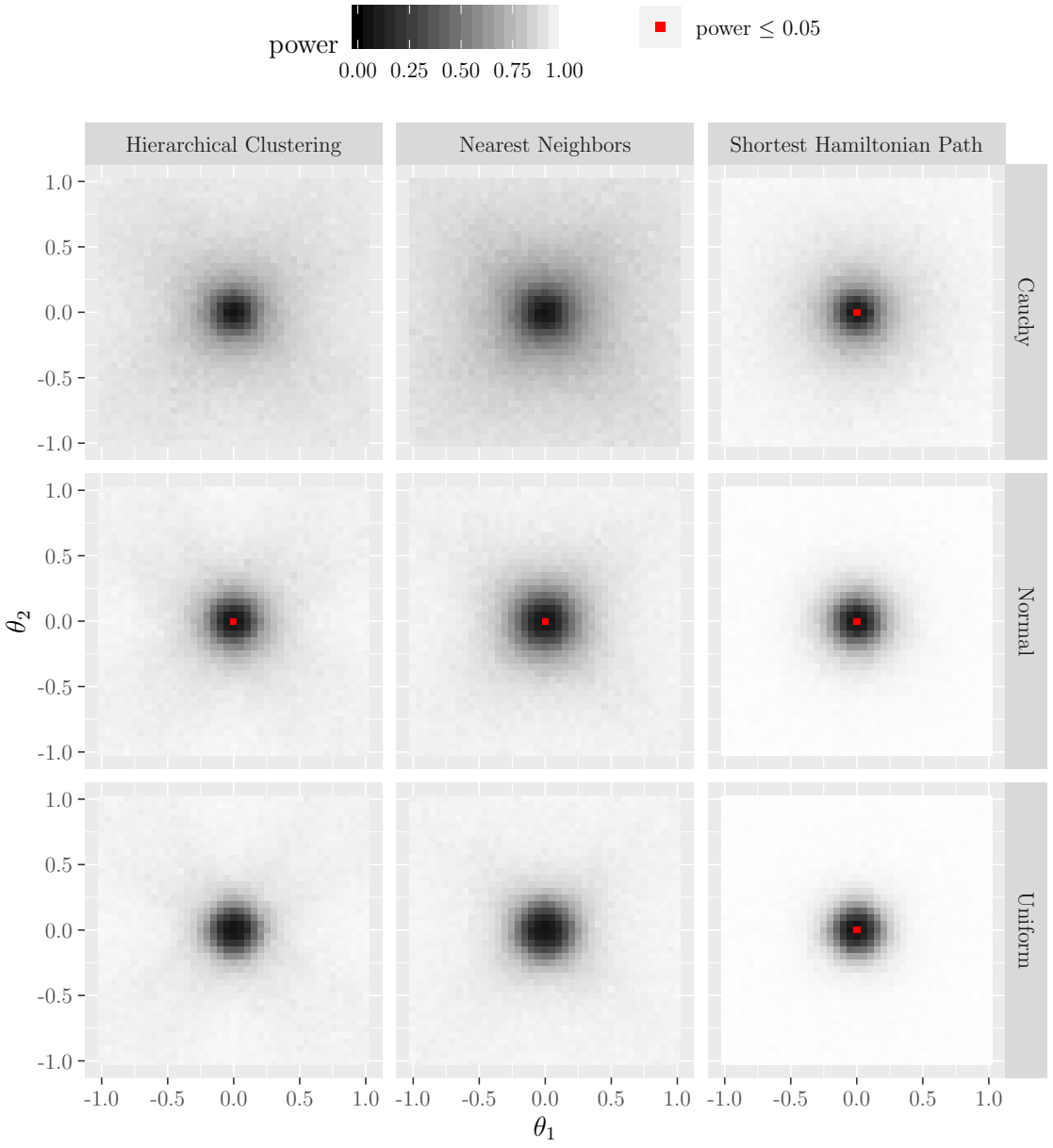


Figure 5.13: Simulated power functions of the distance based ordering methods presented in Section 3.4 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). For the hierarchical clustering method a complete linkage was chosen to order the regression vectors.

$t = 0.2 \cdot u_{0.75}$ and a uniform distribution $\mathcal{U}(a, b)$ with parameters $a = -0.2 \cdot 2 \cdot u_{0.75}$ and $b = 0.2 \cdot 2 \cdot u_{0.75}$, where $u_{0.75}$ denotes the 75%-quantile of the standardnormal distribution. All entries in the vector $\boldsymbol{e}$ are stochastically independent of each other. The parameters of all three distributions are set so that the distributions have median zero and the same interquartile range. This choice shall increase the comparability of the power values and the simulated power functions.

The Figures 5.10 - 5.13 show the simulated power functions on the "Random" data set with parameter $K = 3$ and $N = 100$ data points for all ordering methods. Since the sign depth test only depends on the signs of the residuals and not on the values itself, in Figures 5.10 - 5.13 no real differences between the power functions with the different error distributions can be seen. Small differences are due to simulation stochastics.

Overall, these simulated power functions show the same behavior and have the same characteristics as described in the previous subsections: The naive ordering methods including the ordering according to the values of the euclidean norm perform quite poorly, the scalarization based methods including the nondominated sorting method have a direction with low power, the remaining ordering methods based on partial sorting are quite bad and the distance based methods perform very well. Again, the ordering method based on the exact solution of the Shortest Hamiltonian Path problem seems to perform best. It is slightly better than the ordering methods based on the nearest neighbor approximation and on the hierarchical clustering.

Although this result could be expected, these power functions clearly state that the error distribution does not have an effect on the power of the sign depth test which is a big advantage of the sign depth test in contrast to other robust and non-robust tests.

5.2.4 Effect of the Parameter K on the Results

The parameter K of the sign depth and the sign depth test may have a crucial effect on the power of the sign depth test. As seen before for the "Spiral" data set in Figure 5.6 on page 115 and Figure C.2 on page 229, the parameter K can change the characteristic of the power function. Also, in Section 5.5 it will be shown that the classical sign test is much worse than the sign depth test, where the classical sign

test is equivalent to the sign depth test with parameter $K = 2$ which also shows that there has to be an effect of this parameter on the power values of the sign depth test.

In the following, the simulated power functions of the different ordering methods with $K \in \{3, 4, 5\}$ will get analyzed. For this, the power functions on the "Random" data set with $N = 100$ data points and normally distributed errors will be shown.

Figure 5.14 shows the simulated power functions for the naive ordering methods. As it can be seen, no differences between the power functions are visible. So, in this case, the parameter K of the sign depth test seems to have no effect on the power of the test. But since it could be seen in the previous subsections that a rather messy order of the regression vectors leads to bad power functions, it seems reasonable that this behavior cannot be changed by the length of the tuples which are used for calculating the sign depth.

The same holds for the euclidean norm method in Figure 5.15. But for both other scalarization based ordering methods an effect can be seen: The areas with low power are wider for $K = 4$ than for $K = 3$ and $K = 5$. The reason for this phenomenon is not finally known so far, but Leckey et al. (2020) showed that the sign depth has different characteristics for odd and even values of K. In this situation, odd values of K seem to perform better than even values of K. Unfortunately, there is no second even value of K to compare the results for $K = 4$ with, but when looking at the results for $K = 3$ and $K = 5$, it seems that increasing K has no big effect on the results, neither positive nor negative. The direction with low power will remain for all reasonable values of K and increasing the number of data points will have more positive effect on the size of this area with low power than changing the value of K.

The same phenomenon can be nicely seen for the nondominated sorting method in Figure 5.16 since this ordering method behaves similarly to most of the scalarization based methods. For the convex hull method, an improvement of the power function can be seen when increasing K. But since this improvement is on a very low level, it is regarded as non-relevant here when looking for "good" power functions. The simulated power function for $K = 5$ is still very bad, only the number of simulated points where the power is less than or equal to α has decreased. Interestingly, the power function of the halfspace depth method seems to be worse for $K = 4$ than for the other values of K like most of the scalarization based methods and the nondominated sorting method, although the ordering method is quite similar to an

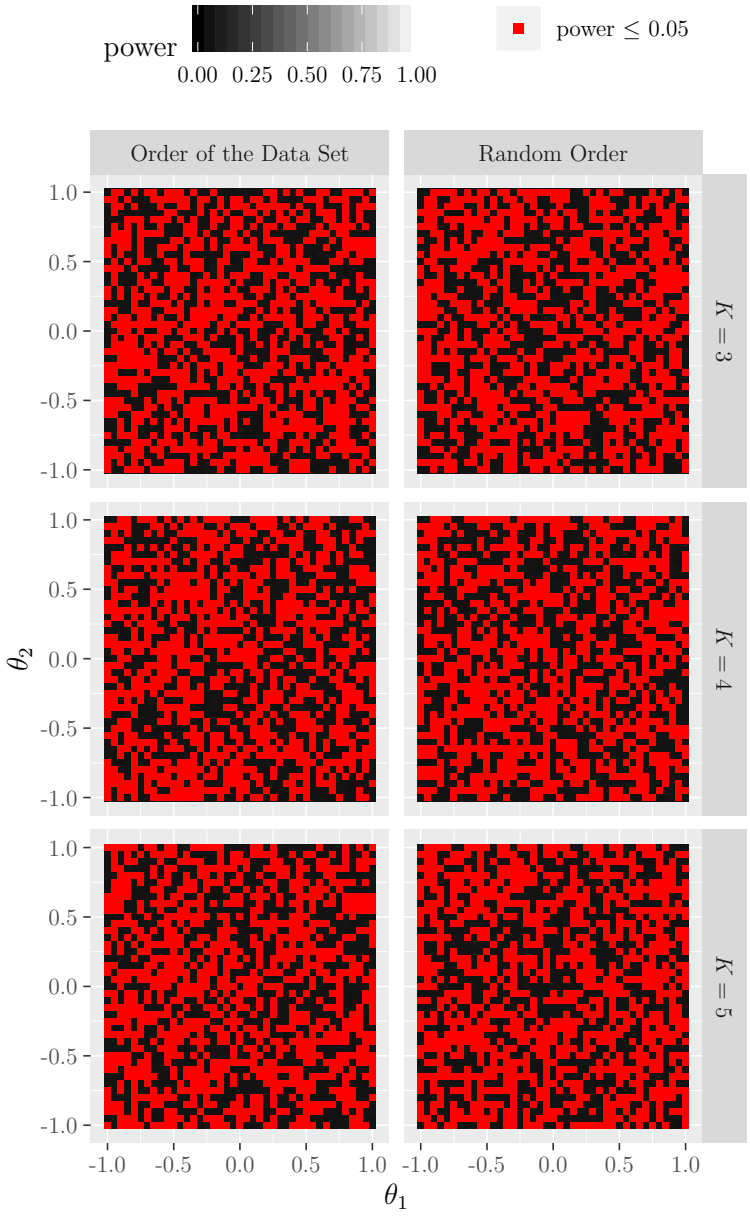


Figure 5.14: Simulated power functions of the naive ordering methods presented in Section 3.1 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

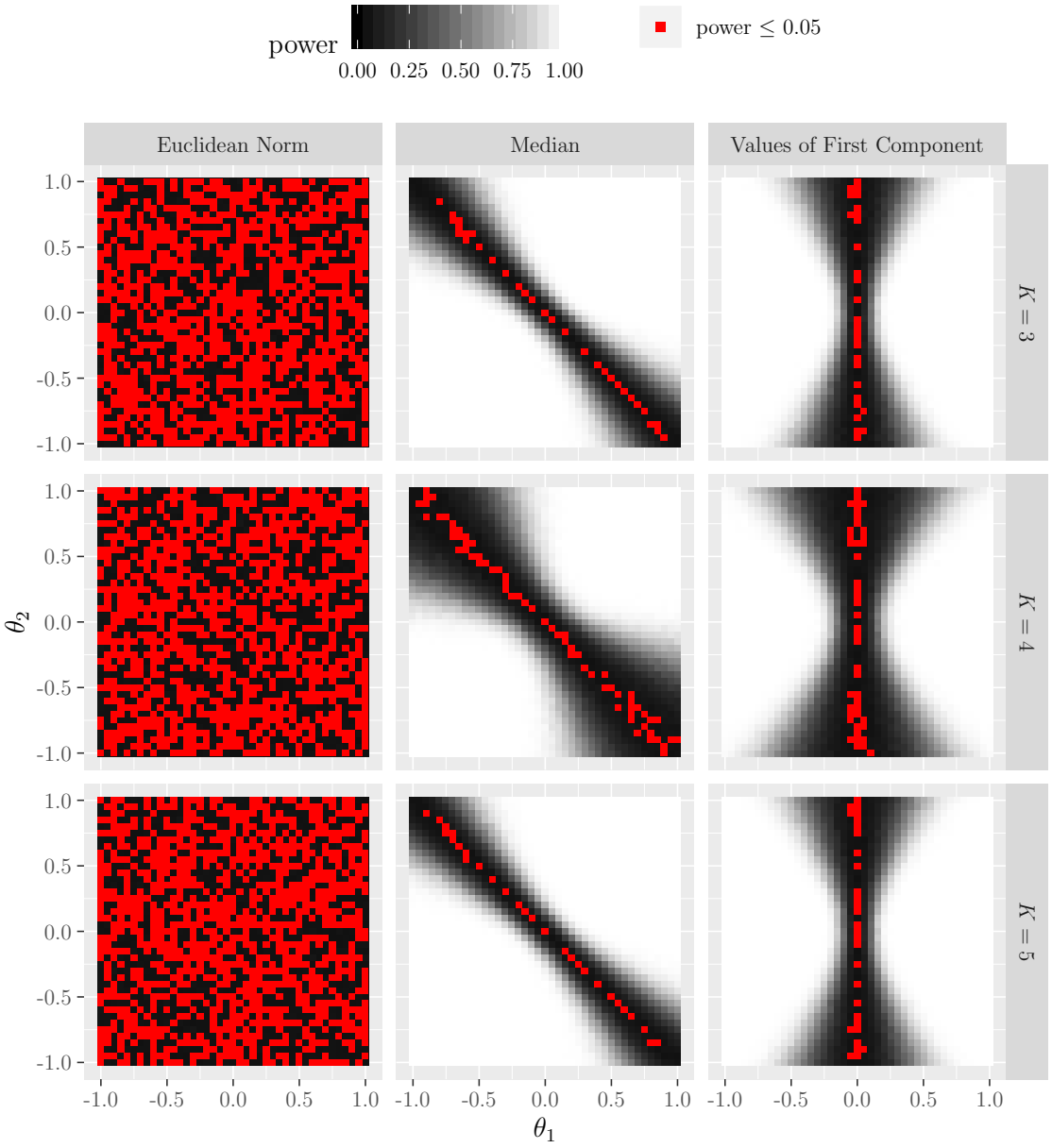


Figure 5.15: Simulated power functions of the scalarization based ordering methods presented in Section 3.2 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The orderings according to a weighted sum and projecting all vectors orthogonal on a line are not shown because the results are identical to calculating the median when setting all weights to an equal value or all entries in the direction vector have equal values, respectively.

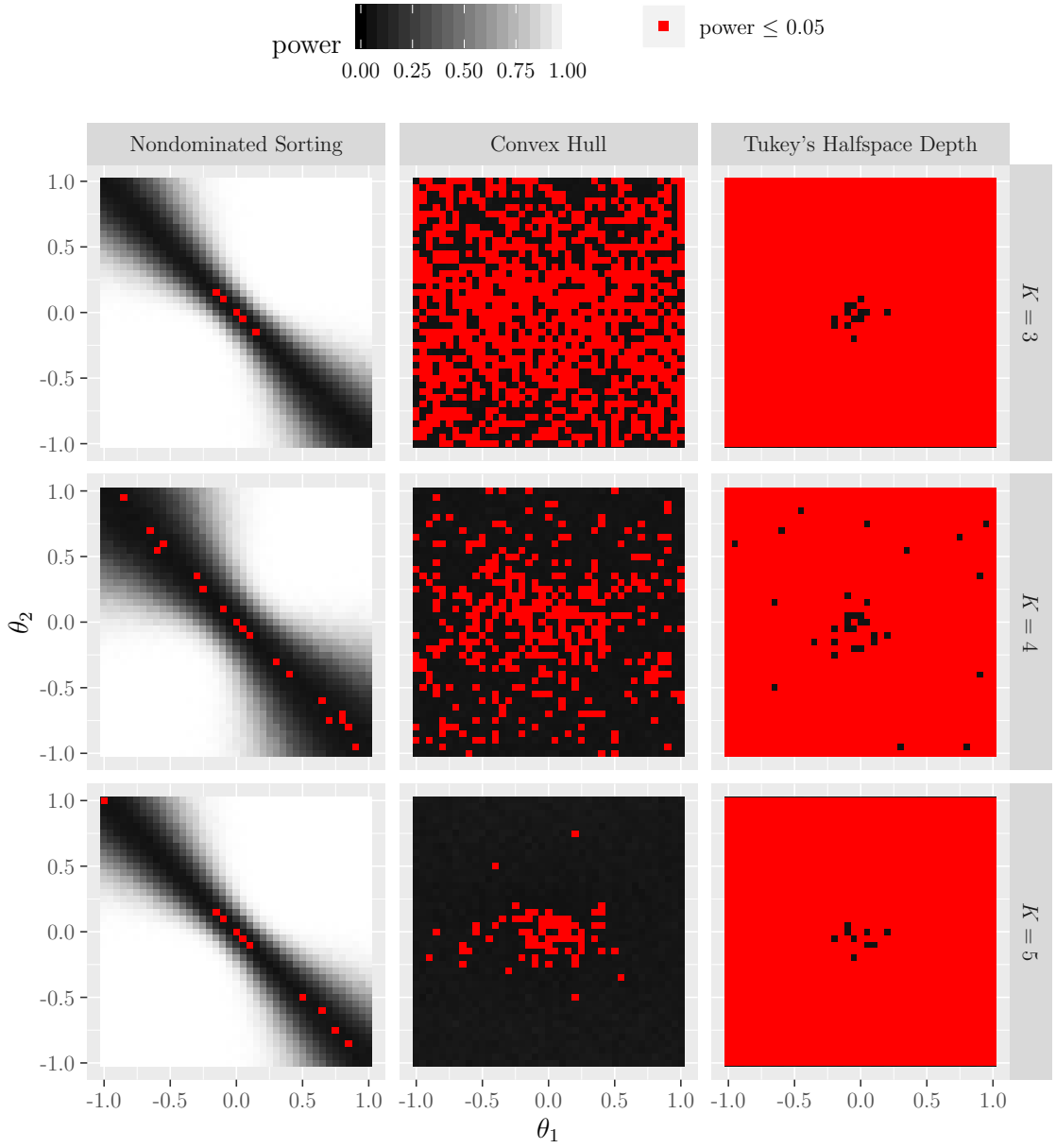


Figure 5.16: Simulated power functions of the ordering methods based on partial sorting presented in Section 3.3 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). The regression vectors with the same rank of the nondominated sorting method are ordered according to their appearance in the data set, which is a random order here.

ordering according to convex hulls. But since this ordering method is even worse than an ordering according to convex hulls, this result also is not relevant here.

Figure 5.17 shows the simulated power functions for the distance based methods. Here, it can be seen that the power functions get better when K increases. Especially, the hierarchical clustering method and the nearest neighbor approximation get better when increasing K from 3 to 4. The Shortest Hamiltonian Path method performs best and is very good already for $K = 3$, although its power also increases for greater values of K. In general, all of these simulated power functions are very satisfying.

As a conclusion it can be said that the parameter K has an effect on the power of the sign depth test. But the general behavior of the test is not changed by this parameter: if the test performs badly/well for small values of K, its performance is also bad/good for larger values. There seem to be three groups of ordering methods which are affected differently: The first group consists of ordering methods which are apparently not or not much affected by the parameter K. These are both naive ordering methods and the ordering according to the euclidean norm. The second group consists of ordering methods which are better for odd values of K than for even values. These are all scalarization based methods (except the ordering according to the euclidean norm), the nondominated sorting method and the halfspace depth method. But, whereas the scalarization based methods and the nondominated sorting method have low power only in one direction, the power functions of the halfspace depth method are very bad. The last group consists of ordering methods whose performance increases for larger values of K. These are all distance based ordering methods and the ordering according to convex hulls. But only the power functions of the distance based methods are really satisfying whereas the performance of the sign depth test when using convex hulls for ordering is very poor. Probably, also these ordering methods behave differently for odd and even values of K, but this effect is not as pronounced as for the scalarization based methods and cannot be seen in the simulated power functions. Overall, only the distance based ordering methods can be recommended to use and for them larger values of K perform better, but also the performance for $K = 3$ is satisfying. In former times when calculating the sign depth had time complexity $\mathcal{O}(N^K)$, it would be totally sufficient to use $K = 3$ to save computational runtime and get good results nevertheless, at least for models with a relatively low number of parameters. Today, when calculating the sign depth can be done in linear time complexity in N, larger values of K can also be used and should be used since the performance increases for larger values of K.

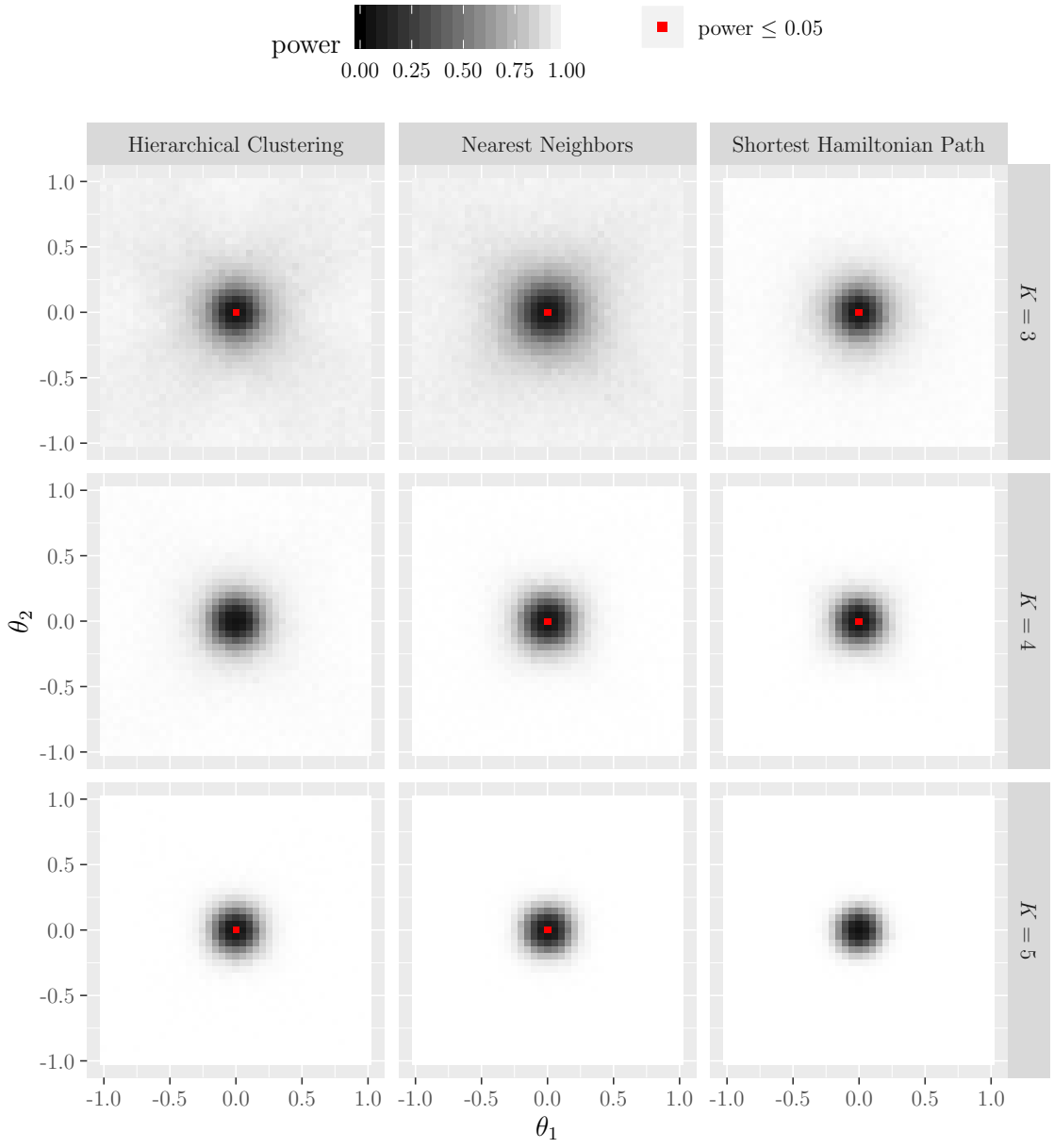


Figure 5.17: Simulated power functions of the distance based ordering methods presented in Section 3.4 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). For the hierarchical clustering method a complete linkage was chosen to order the regression vectors.

5.2.5 Effect of Hyper-Parameters of the Ordering Methods on the Results

At last, the effect of any hyper-parameters of the ordering methods will be shown. Not all ordering methods have hyper-parameters which is an advantage of them, but for those who have some, the question arises which hyper-parameters are best or does the choice of the hyper-parameters has an effect at all? In general, most choices of hyper-parameters will be best for some (data) situation, so here no final answer about the best hyper-parameters can be given, but it is looked at a specific situation to get an impression of the effect of the specific hyper-parameter. For this subsection, the simulated power functions obtained by the 3-sign depth test on the "Random" data set with $N = 100$ data points and normally distributed error vector $\boldsymbol{e}$ are considered.

The naive ordering methods do not have any hyper-parameters. An ordering according to the appearance of the regression vectors in the data set is not affected by any parameters and taking a random order is also not. Of course, the result of the random order is affected by the specific algorithm which is used and the starting point (i.e. seed) of the algorithm, but these are no real hyper-parameters which can be optimized.

Most scalarization based methods have hyper-parameters. Solely the ordering according to the median of each regression vector has none (neglecting the fact that there are different possibilities to calculate the median of an even number of values which will have no real effect on the power of the sign depth test). The ordering according to a vector norm of the regression vectors is based on the parameter p of the norm. In all previous subsections p was set to 2, i.e. the euclidean norm was calculated. But every other positive value for p would also be possible. Figure 5.18 shows the simulated power functions for $p = 1$ (Manhattan norm), $p = 2$ (euclidean norm) and $p = \infty$ (maximum norm). As it can be seen, no differences between the power functions are visible, so the value of this hyper-parameter seems to have no effect. Indeed, the effect is also very small when shifting the regression vectors to only positive values (as shown in Figure C.1 on page 228) because then there would be a "direction of ordering" for all norms which would lead to a direction with low power in the power function independent of the value of p.

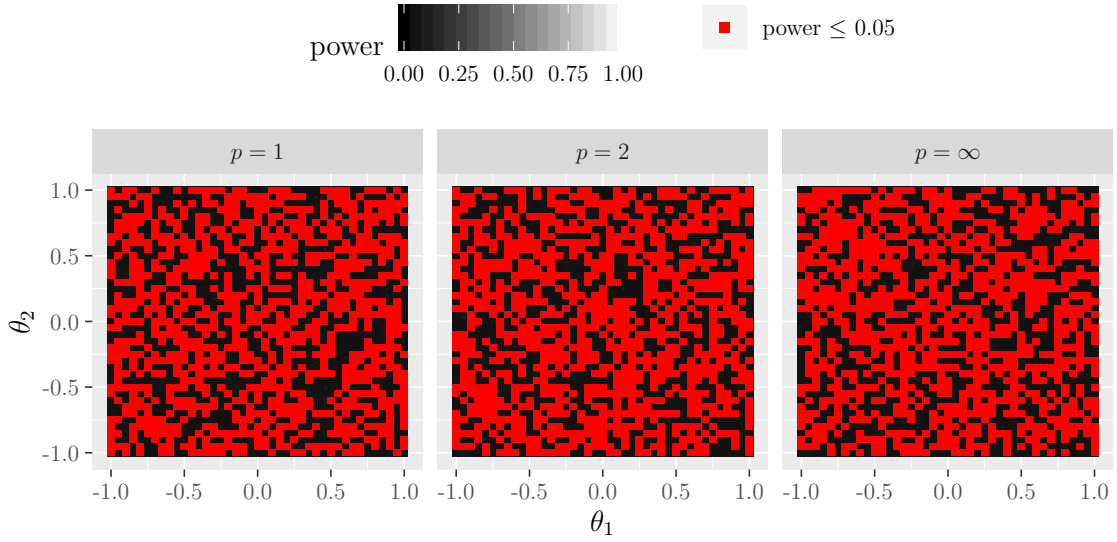


Figure 5.18: Simulated power functions of the ordering according to different vector norms for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

In contrast, of course the choice of the component of the regression vector whose values are ordered when ordering only to one component of the regression vector has an effect on simulated the power function of the sign depth test. This can be seen in Figure 5.19. By changing the component whose values are ordered, the "direction of ordering" changes and so the direction with low power values in the power function is different. But, of course, when the values of all components have the same characteristics, like here, it does not matter which component is chosen. Here, in both cases the area with low power values has the same size and in general, when rotating one of the power functions, both simulated power functions would look very similar (small differences are due to simulation stochastics). On the other hand, when the characteristics of the components of the regression vectors are different, for example when there are outliers in some components, the choice of the component for ordering may have more effect. But since this effect is data dependent, no general rule for choosing the component for ordering can be given.

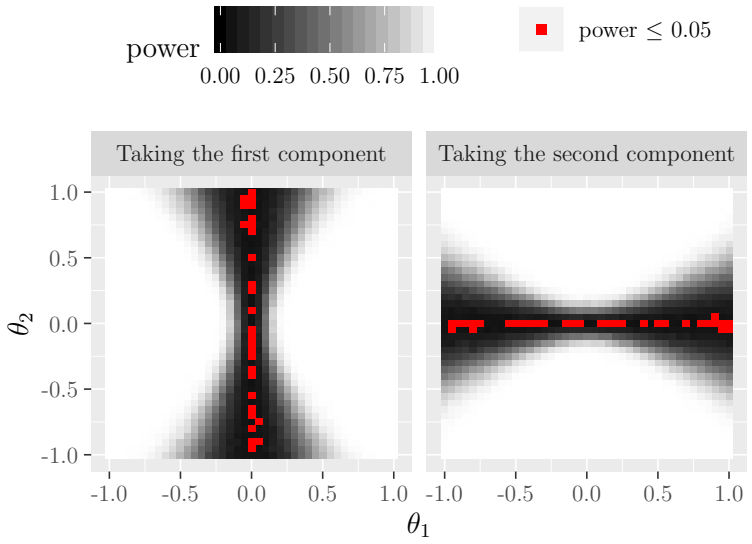


Figure 5.19: Simulated power functions of the ordering according to different components of the regression vectors for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

The next figure, Figure 5.20, shows the simulated power functions when using an orthogonal projection on different lines for ordering. The figure mentions the different direction vectors $\boldsymbol{v}$ of the line to project on, but not the location vectors $\boldsymbol{u}$ because (as shown in Subsection 3.2.5) the location vector has no effect on the ordering. The vectors $\boldsymbol{v}$ could also be regarded as weight vectors for an ordering according to a weighted sum when neglecting the fact that negative weights are not allowed there, so that the middle plot in Figure 5.20 could not be obtained by using the weighted sum method. The figure shows the expected result: Each power function has a direction with low power which lays orthogonal to the "direction of ordering" which is here also the orthogonal direction to the line on which the regression vectors are projected on. This behavior cannot be prevented when using this ordering method. There is no parameter value which performs better than others, so there is no general recommendation which parameter value to use for this ordering method. As mentioned before for the ordering according to only one component of each regression vector, the choice of the line to project on should depend on the characteristics of the underlying data. If there are outliers in some components, it could be a good

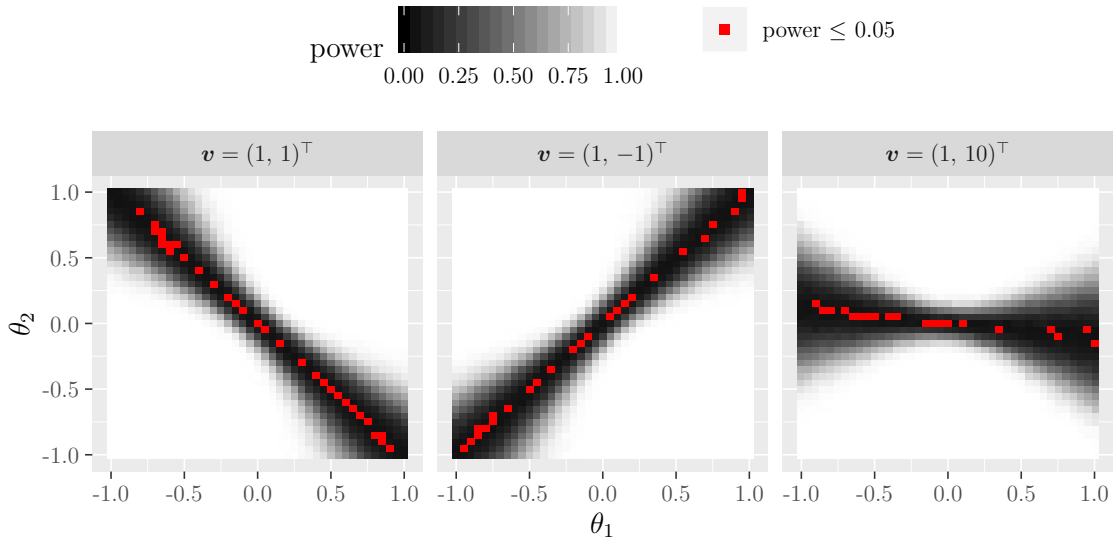


Figure 5.20: Simulated power functions of the ordering according to projections on different lines for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The location vector of the lines is always $\boldsymbol{u} = (0, 0)^\top$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

idea to choose a line which gives the respective components no or only little influence on the ordering process.

From the ordering methods based on partial sorting only the nondominated sorting method has a hyper-parameter. Both other methods do not have hyper-parameters because the regression vectors which have the same rank will get ordered with a Traveling Salesman algorithm. Of course, it could be discussed whether this is the only meaningful way and also whether the Traveling Salesman algorithm itself has hyper-parameters or not (see below), but for our purpose we assume that there are no hyper-parameters for the convex hull method and the ordering on the basis of Tukey's halfspace depth. But the nondominated sorting method has a hyper-parameter for ordering the regression vectors which got the same rank in the partial sorting process. This hyper-parameter describes an arbitrary other ordering method for multidimensional data. In Figure 5.21 the power functions obtained with two different tie-breaking methods are shown. In the left plot an ordering according to the appearance of the regression vectors in the data set is done. Since the data set is the

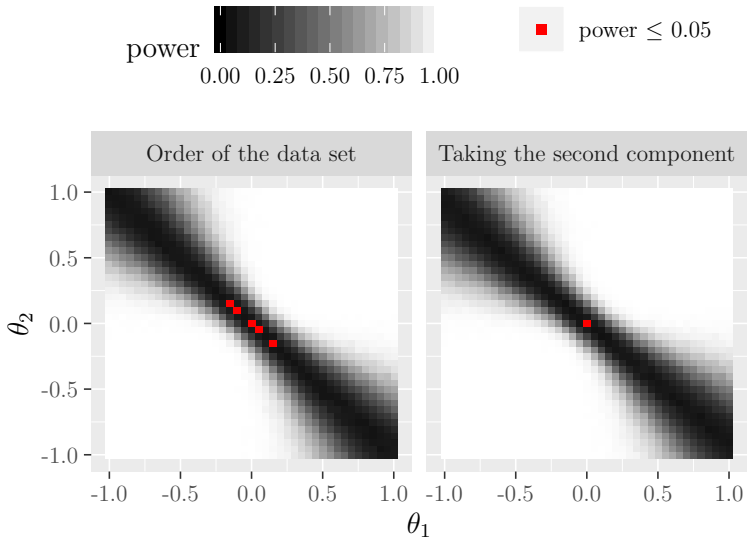


Figure 5.21: Simulated power functions of the ordering according to a nondominated sorting for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The tie-breaking for regression vectors with the same rank is done in two different ways. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

"Random" data set, this describes a random ordering of the regression vectors with the same rank. In the right plot, an ordering according to only a single component of the regression vectors is done. As seen above, the choice of the component does not matter. Here, the second component was chosen. Other tie-breaking methods are also possible of course, but these two seemed the most meaningful choices, at least for this two-dimensional case. When the number of dimensions is larger, also an ordering according to the solution of the Shortest Hamiltonian Path problem could be reasonable as long as there are not too many regression vectors with the same rank because of the possibly large computational runtime of this ordering method. Figure 5.21 shows that there are no big differences between both power functions. But in detail, it can be seen that the usage of the random ordering for tie-breaking is slightly worse than using one component of each regression vector for ordering. At least the power is less than or equal to α only at H_0 when using one component for ordering whereas this is the case for some more points when using the random order. So, it seems that in detail the tie-breaking hyper-parameter of the nondominated

sorting method is relevant for the power of the sign depth test: An ordering which leads to less ”mess” in the ordering is preferable.

From the distance based ordering methods only the hierarchical clustering method has a hyper-parameter when neglecting the fact that also the choice of the distance measure could be a possible hyper-parameter. Since in this thesis only the euclidean distance is considered, the distance measure is not considered as a hyper-parameter here. The distance measure may become of more interest when looking at models with not only metric regressors, but also with ordinal or nominal regressors since then the euclidean distance is no longer a possible choice. When neglecting the distance measure, the Shortest Hamiltonian Path method and its approximation only have technical hyper-parameters like a precision parameter for the TSP-solver ”Concorde”. But a hierarchical clustering always depends on the choice of the linkage function. Three possible linkage functions are described in Subsection 3.4.3 and also shown in Figure 5.22: A single linkage, an average linkage and a complete linkage. It can be

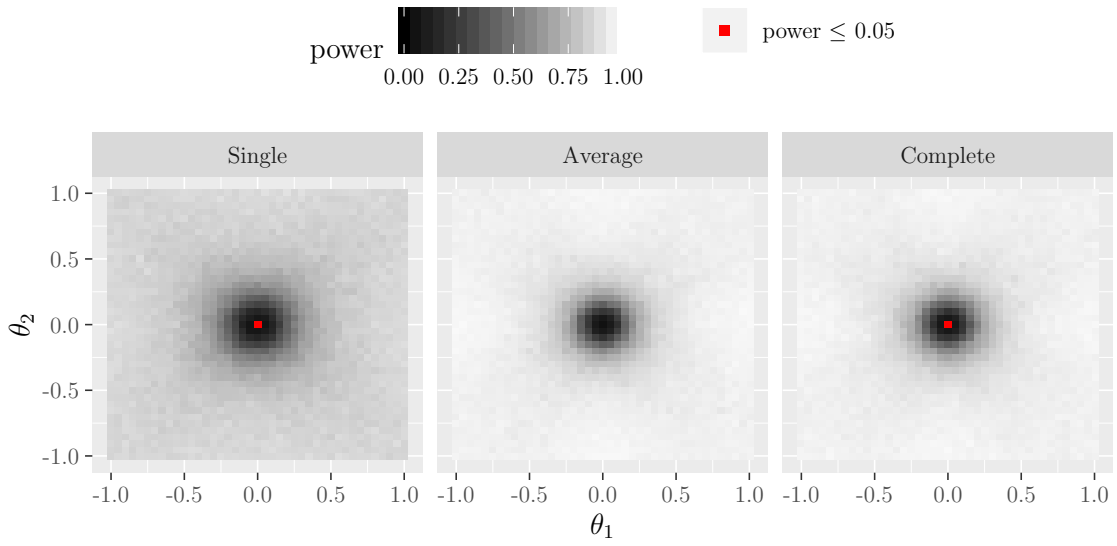


Figure 5.22: Simulated power functions of the ordering according to a hierarchical clustering with different linkage functions for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 3. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

seen that the choice of the linkage function has an effect on the power of the sign depth test here. The power function of the ordering with single linkage is slightly worse than the other two power functions. In Figure 3.34 on page 79 it can be seen that the single linkage leads to more ”mess” in the order than the other two linkage functions which is the explanation for the slightly worse power values here. Since this is a general behavior of an ordering when using the single linkage function for a hierarchical clustering, it should be recommended to use one of the other two linkage functions.

All in all, it can be said that hyper-parameters only have very little effect on the performance of the sign depth test. In most cases, hyper-parameter either have no visible effect or they are only changing the direction with low power for the ordering methods which have such a direction. A small effect could be seen for the tie-breaking method of the nondominated sorting. Here, an ordering method should be use which produces as little ”mess” as possible in the resulted ordering. The same holds for the choice of the linkage function in the hierarchical clustering. Here, a single linkage leads to slightly worse power values than using different linkage functions. So, overall, it was shown in this subsection that the sign depth test can be used without having to worry too much about hyper-parameters. Especially, because ordering methods which lead to very good results of the sign depth test do not have real hyper-parameters at all, like an ordering according to the solution of the Shortest Hamiltonian Path problem or its approximation.

5.2.6 Further Analysis and Summary of the Results

The previous subsections have led to much understanding about the different ordering methods and their behavior. In this subsection, some more results will be shown and a summary of all results so far will be given.

As a general result, it can be said that the ordering method has a crucial effect on the power of the sign depth test. Also, the number of data points in the data set and the parameter K of the sign depth and the sign depth test have some effect whereas the error distribution, the structure of the underlying data set and possible hyper-parameters of the ordering methods have no or only little effect on the power of the test. Because of this, the different ordering methods will get analyzed once again relating the numbers of data points and the parameter K. For this, the power

functions of all ordering methods on the "Random" data set and with normally distributed errors are simulated once again. This time, the power functions are simulated only on the interval $\boldsymbol{\theta} \in \{-0.4, -0.38, -0.36, \ldots, 0.36, 0.38, 0.4\}^2$, so only in a small area around the null-hypothesis $H_0 : \boldsymbol{\theta} = \mathbf{0}$. So, we have $41^2 = 1\,681$ different parameter values of $\boldsymbol{\theta}$, from which $1\,680$ are from the alternative hypothesis $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$. The power functions are simulated for all $N \in [10, 100] \cap \mathbb{N}$ and for $K \in \{3, 4, 5\}$. An ideal power function would have power less than or equal to $\alpha = 0.05$ at H_0 and larger power values in H_1 which go to the value of one very quickly. While in the previous subsections it could be seen that obtaining a power value of about α at H_0 is no problem for every ordering method, the power in H_1 does not always reach values greater than α. Because of this, in the following it is looked at the percentage of simulated power values in H_1 which have a power of at least $\alpha = 0.05$ for all ordering methods and all values of N and K. In addition, it is also looked at the percentages of power values in H_1 which are at least 0.5 and 0.95.

Figure 5.23 shows these results. In this figure, the results of the naive ordering methods are displayed as greenish points, the results of the scalarization based methods are bluish (where the color of the ordering according to the median, according to a weighted sum and according to an orthogonal projection is the same), the results of the ordering methods based on partial sorting are yellowish and the results of the distance based methods are displayed as reddish points.

In general, for being a satisfying test all power values in H_1 should be larger than $\alpha = 0.05$. At least the percentage of values which are at least 0.05 should increase when the number of data points in the data set N gets larger. As it can be seen in the left column of Figure 5.23, this is the case for most of the ordering methods, but not for all. The naive ordering methods and the ordering according to the values of the euclidean norm have independently of N and K (nearly) always around half of the power values greater than or equal to 0.05. Only for very small N (smaller than approximately 25), the percentage is even smaller. This behavior could also be seen in the visualizations of the power functions in the previous subsections where it was visible that for these ordering methods all power values are about 0.05 and it is random whether they are slightly less or greater than α. Also, the values of the convex hull method and the ordering according to the values of Tukey's halfspace depth show an undesired behavior: The percentage of values greater or equal than 0.05 decreases for larger N. This behavior was also visible in Figure 5.3 on page 111, but here it can be seen more clearly. In general, the percentage of values greater or

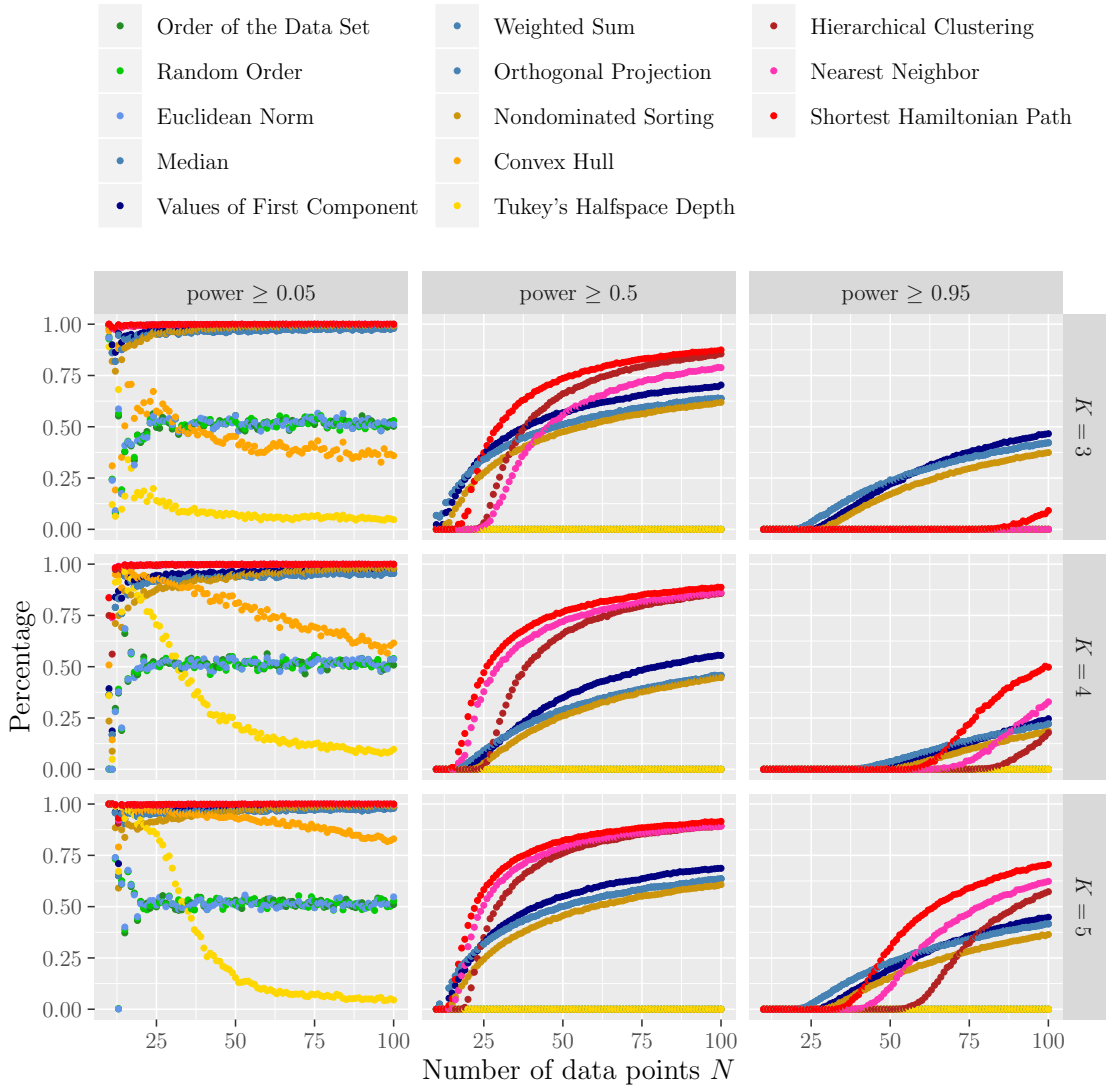


Figure 5.23: Percentage of power values in H_1 of all ordering methods which are at least 0.05, 0.5 and 0.95. The tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The power is simulated on the interval $\boldsymbol{\theta} \in \{-0.4, -0.38, -0.36, \ldots, 0.36, 0.38, 0.4\}^2$. The underlying data set consists of N random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution.

equal than 0.05 is larger for the convex hull method than for the ordering according to the halfspace depth which was also visible in the previous subsections. But now, it can be seen that the decrease of the power values is much bigger for increasing N for the ordering according to the halfspace depth than for the convex hull method. For the convex hull method the decrease is rather slow and for larger values of K the percentage of values greater or equal than 0.05 is nearly one. On the other

hand, for the ordering according to the halfspace depth the values decrease very fast, although for very small N and $K \in \{4, 5\}$ the percentage of values greater or equal than 0.05 is quite large. But, all in all, the behavior of these two methods is highly undesired. The third ordering method based on partial sorting, the nondominated sorting method, behaves similar to the scalarization based methods (except the above mentioned ordering according to the euclidean norm). These methods have percentages which are nearly one. The percentages will never be exactly one because the power is less than or equal to α on a specific line, but except these few values all power values are at least 0.05. The distance based methods are even better than the scalarization based methods. Here, the values are most times exactly one, so all power values in the area of the alternative hypothesis are at least 0.05. This holds for all three distance based methods and all values of N and K. So, overall, the left column of Figure 5.23 shows that the distance based methods are best, but also most scalarization based methods and the ordering according to the nondominated sorting have quite satisfying performance.

In the middle row of Figure 5.23 the percentages of power values which are at least 0.5 are visualized. The first thing one notices is that only the distance based methods, the scalarization based methods (except the ordering according the values of the euclidean norm) and the nondominated sorting method have values which are greater than zero. All methods which have already performed poorly in the left column of Figure 5.23 cannot produce any power value which is at least 0.5, for no value of N and no value of K. Furthermore, it can be seen that the three distance based methods behave similarly and the scalarization based methods including the nondominated sorting method also behave similarly. For the scalarization based methods including the nondominated sorting method it can be nicely seen that they behave differently for $K = 4$ than for $K = 3$ and $K = 5$. But, it always holds that an ordering according to only one component of all regression vectors is slightly better than an ordering according to the median, according to a weighted sum and according to an orthogonal projection. The nondominated sorting method is always the worst of these methods. But indeed, the ordering according to only one component is not better than the other scalarization based methods. It seems like this because the direction with low power for this ordering method is parallel to a coordinate axis and for the other methods it is diagonal to the coordinate axes and so when simulating the power functions on a square there are less points with low power for the ordering according to only one component than for the other methods. But the slightly lower values for the nondominated sorting method are an important result because they show that

this method produces slightly wider areas with low power than the scalarization based methods. In general, the distance based methods perform best and especially, the ordering according to the exact solution of the Shortest Hamiltonian Path problem is the best of the three methods. Interestingly, for $K = 3$ the hierarchical clustering method performs better than the nearest neighbor approximation whereas it is the other way round for $K = 4$ and $K = 5$. Generally, the scalarization based methods including the nondominated sorting method as well as the distance based methods have some power values which are at least 0.5 already for very small numbers of data points, i.e. less than 25 data points. This holds especially for $K = 5$, but also for $K = 3$ and $K = 4$.

The right column of Figure 5.23 shows the percentages of simulated points with a power of at least 0.95. In general, most findings from the middle column also hold for this column: Only the scalarization based methods (except the ordering according to the euclidean norm), the nondominated sorting method and the distance based methods have some power values which are at least 0.95. In addition, for the scalarization based methods including the nondominated sorting method $K = 4$ is worse than $K = 3$ and $K = 5$, whereas for the distance based methods larger values of K are better than smaller values. Of course, for reaching a power of at least 0.95 more data points are needed than for reaching a power of at least 0.5. The scalarization based methods including the nondominated sorting method need approximately 25 data points for $K = 3$ and $K = 5$ and about 50 data points for $K = 4$ whereas the distance based methods need approximately 100 data points for $K = 3$, 75 data points for $K = 4$ and 50 data points for $K = 5$. It is remarkable that the scalarization based methods do need less data points for reaching some points with power of at least 0.95 than the distance based methods. But when N increases or a larger interval of $\boldsymbol{\theta}$ is looked at, at some point the distance based methods will be always better than all other methods because the distance based methods have no directions with low power.

As a conclusion from this and all previous subsections it can be said that the distance based methods are by far the best choice when ordering multidimensional values for applying the sign depth test to some data. The naive ordering methods, the ordering according to the values of the euclidean norm of each regression vector, the ordering on the basis of convex hulls and the ordering on the basis of Tukey's halfspace depth perform very poorly and should not be used at all. All other scalarization based methods have some advantages and some areas with very high power already for small

numbers of data points, but since there is always a direction in the power function with power values about α, these methods cannot be fully recommended. The nondominated sorting method is slightly worse than the scalarization based methods. In addition, in Subsection 3.3.1 and Figure 3.17 on page 50 it is explained why this ordering method cannot be used for data in high dimensions and Figure 3.20 on page 55 and Table C.1 in the appendix show that this ordering method has very large computational runtimes. So, overall this ordering method and in general all ordering methods based on partial sorting should not be used for applying the sign depth test. The best choices when applying the sign depth test to some multidimensional data are the distance based methods. All three methods have performed really well in the simulations. Here, the best was using the exact solution of the Shortest Hamiltonian Path problem. But this ordering method has a very large time complexity and also the empirical runtimes of this ordering method are rather large, see for example Table C.1 in the appendix. Also the runtimes of the approximate solution of the Shortest Hamiltonian Path problem are large and indeed often larger than the runtimes of the exact solution. Because of this, using the approximate solver of this problem is not always recommendable. The hierarchical clustering method has performed slightly worse in most cases than the methods based on the shortest path problem, but its empirical runtime is very small. So, also this ordering method can be fully recommended. In general, the distance based methods have many advantages, for this see also Table 3.1 on page 82. The data points can be linearly transformed without changing the obtained order and the ordering methods can also be used when having ordinal or nominal components in the regression vectors. So, as a general recommendation from the results so far it can be said that for small and medium numbers of data points the exact solution of the Shortest Hamiltonian Path problem should be used for ordering the data and for large numbers of data points a hierarchical clustering (with average or complete linkage) should be used.

Because of these results, in the following only the distance based methods will be used. In addition, it is focused on the ”Random” data set and normally distributed errors because the results have shown that both things have only little effect on the power of the sign depth test.

5.3 Results of the Distance Based Methods for Other Models

In the following, the power of the sign depth test is analyzed for some more models: Models with intercept, models with interactions and models with non-linear terms. In addition, in Subsection 5.3.4 high-dimensional models with up to 80 dimensions will be considered. Because the results on the simple model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ in the previous section have shown that only the distance based ordering methods perform really satisfyingly, from now on only these methods are considered.

5.3.1 Linear Models with Intercept

Of course, linear models with intercept are of interest. Probably they are more relevant in the everyday life of a statistician than models without intercept. Because of this, in this subsection it is looked at the two models $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ and $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$, where $\boldsymbol{e}$ is normally distributed with the same parameters as in the previous section, i.e. the components of $\boldsymbol{e}$ are independent and identically distributed with $\mu = 0$ and $\sigma^2 = 0.2^2 = 0.04$. The underlying data set consists of N random regression vectors in the interval $[-1, 1]^2$ and $[-1, 1]^3$, respectively.

Model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$

At first, it is looked at the linear model with two regressors and an intercept, i.e. $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. Because the visualizations in the previous section have shown that the area with low power is rather small for the distance based methods, here the power functions are simulated only on the interval $\boldsymbol{\theta} \in [-0.5, 0.5]^3$ to save computational runtime. Figure 5.24 shows the results of the simulated power functions for θ_1, θ_2 and an extract of θ_0. The four-dimensional power functions are visualized three-dimensionally (two-dimensions for θ_1 and θ_2 plus the color for the power values) for five different values of θ_0. Since it could be seen in Subsection 5.2.4 that $K = 5$ performs better than $K = 3$ and $K = 4$ for the distance based ordering methods, here the power functions are plotted for $K = 5$. Furthermore, the underlying data set consists of $N = 100$ random regression vectors in $[-1, 1]^2$. It can be seen in

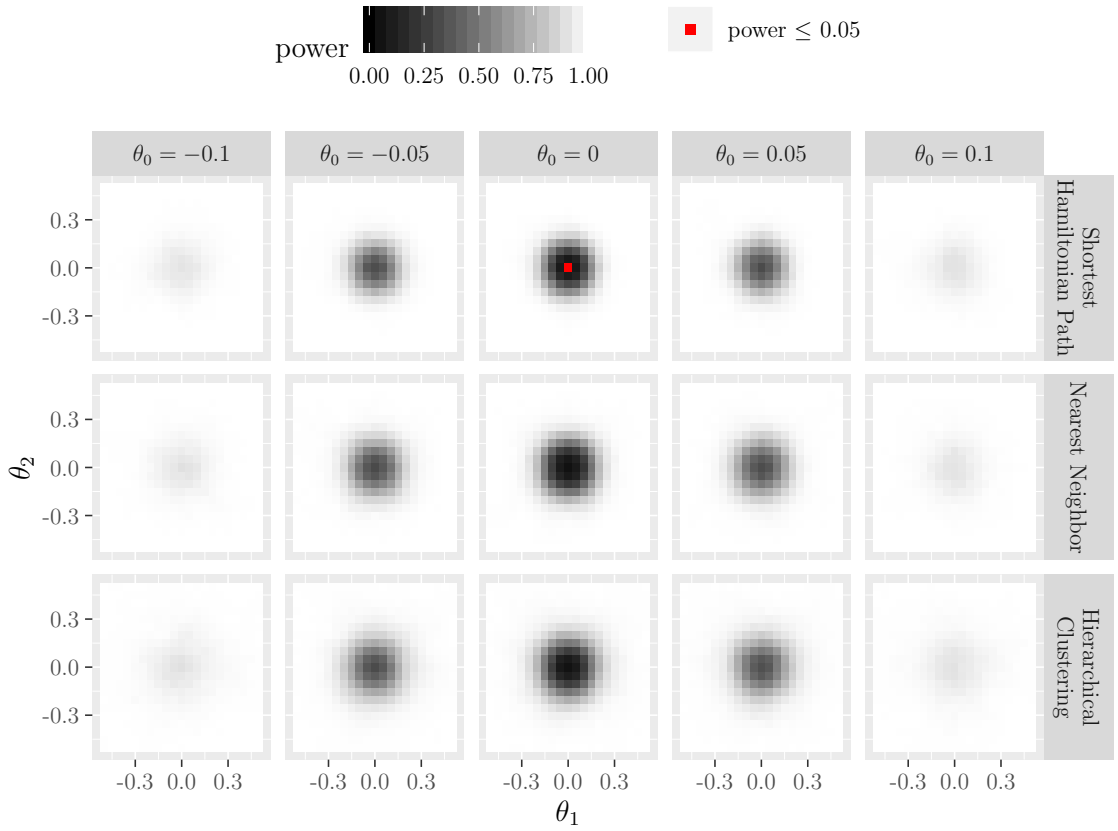


Figure 5.24: Simulated power functions of the distance based ordering methods presented in Section 3.4 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). For the hierarchical clustering method a complete linkage was chosen to order the regression vectors.

Figure 5.24 that the power functions of all three methods are very satisfying. Indeed, when θ_0 is zero, the power functions (of course) look the same as in the previous section. But only small changes in θ_0 lead to large power values. So, the test seems to be very sensitive regarding deviations in the intercept.

All in all, no big differences are visible between the three ordering methods in Figure 5.24. For analyzing whether there are differences and also analyzing the effect of the number of data points and the parameter K, a similar graphic is made as in Subsection 5.2.6. Figure 5.25 shows the percentages of power values in

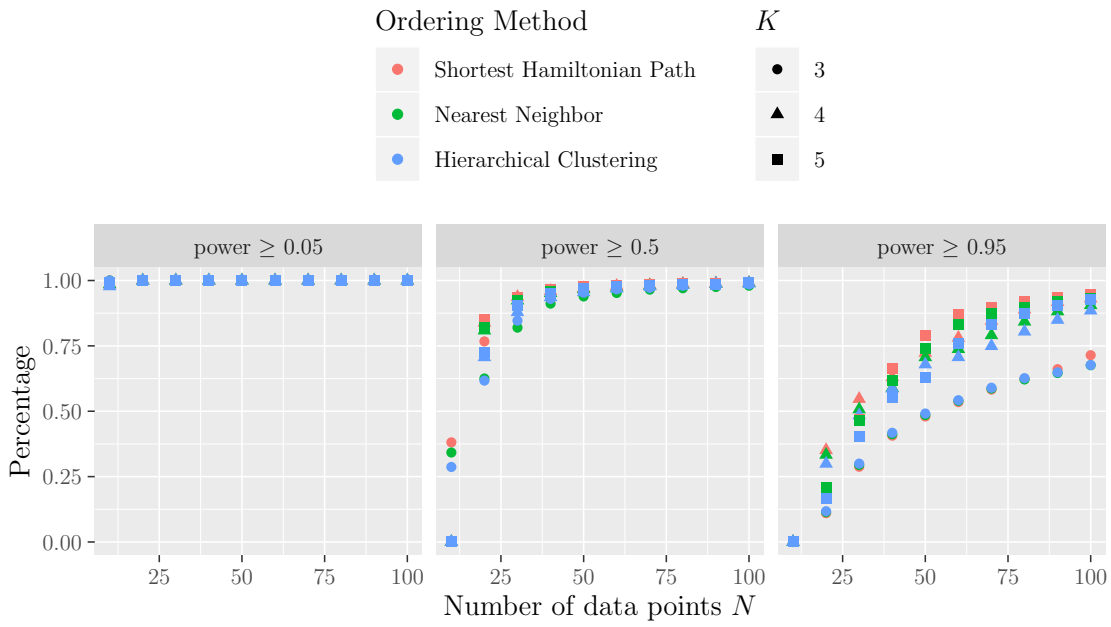


Figure 5.25: Percentages of power values in H_1 of all distance based ordering methods which are at least 0.05, 0.5 and 0.95. The tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The power is simulated on the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^3$. The underlying data set consists of N random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution.

$H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ which are at least 0.05, 0.5 and 0.95 for different numbers of data points ($N \in \{10, 20, \ldots, 100\}$) and $K \in \{3, 4, 5\}$. It can be nicely seen that (nearly) all power values in H_1 are greater than or equal to 0.05, independent of the number of data points and the value of K. Furthermore, for all three values of K also nearly all power values have a value of at least 0.5 when N is not too small. For $N = 10$, interestingly only the power functions simulated with parameter $K = 3$ have some values which are at least 0.5 whereas for $K = 4$ and $K = 5$ no simulated power values are greater than or equal to 0.5. This is caused by the fact that N has to be sufficiently larger than K to be able to reject the null-hypothesis in the sign depth test because otherwise too few K-tuples exist and the necessary α-quantile of the distribution of the K-sign depth is identical to the minimum. Obviously, for $K = 4$ and $K = 5$ more than $N = 10$ data points are necessary to be able to reject the null-hypothesis. But already for $N = 20$ all ordering methods and all values of K lead to power functions where more than half of the power values are at least 0.5. And for all numbers of data points $N \geq 20$, $K = 5$ is best and $K = 3$ is worst, but for

sufficient large N ($N \geq 50$ approximately) this is not relevant anymore because then (nearly) all power values are greater than or equal to 0.5, independent of the value of K and the ordering method. In general, it can be seen that the hierarchical clustering method performs worse than the methods based on the Shortest Hamiltonian Path, where the exact solution is always better than the approximate one. This holds also for the percentages of power values which are at least 0.95. In this plot, it can also be seen that $K = 3$ is much worse than $K = 4$ and $K = 5$ (but $K = 3$ still leads to satisfying power functions). Interestingly, for $N = 20$ and $N = 30$, $K = 4$ is better than $K = 5$. It seems that there is a trade-off between the number of data points, the value K and the achieved power: The fewer data points one has, the better the small values of K perform, but on the other hand, the larger the power values in H_1 should be, the better are larger values of K. Overall, all three ordering methods perform really well. For $K \in \{4, 5\}$ and $N = 100$ nearly all power values in the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^3$ (excluding $H_0 : \boldsymbol{\theta} = \mathbf{0}$) are 0.95 or larger.

Model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$

Next, the same analysis is made for a linear model with one regressor more, i.e. $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$. The parameters of the simulation are the same as before: Normally distributed errors and random regression vectors in $[-1, 1]^3$. At first it is looked at the simulated power functions when setting the parameter K to 5 and having $N = 100$ data points. Since the power functions plots are now five-dimensional ($\boldsymbol{\theta} \in \mathbb{R}^4$ plus the power value), again only an extract, displayed as some layers of the power functions, can be shown. It was decided to show the power functions for $\theta_0 \in \{-0.1, -0.05, 0, 0.05, 0.1\}$ and $\theta_1 \in \{-0.2, -0.1, 0, 0.1, 0.2\}$ because Figure 5.24 has already shown that the sign depth test is very sensitive regarding deviations in the intercept, but for getting nicely visible differences in other components of $\boldsymbol{\theta}$ slightly larger deviations from H_0 are necessary. For θ_2 and θ_3 all power values in the range from -0.5 to 0.5 are shown.

The Figures 5.26, 5.27 and 5.28 show these simulated power functions for the exact and approximate ordering methods based on the Shortest Hamiltonian Path and the hierarchical clustering method, respectively. It can be seen that all three ordering methods perform really well. Again, the best results were obtained by ordering the regression vectors according to the exact solution of the Shortest Hamiltonian Path.

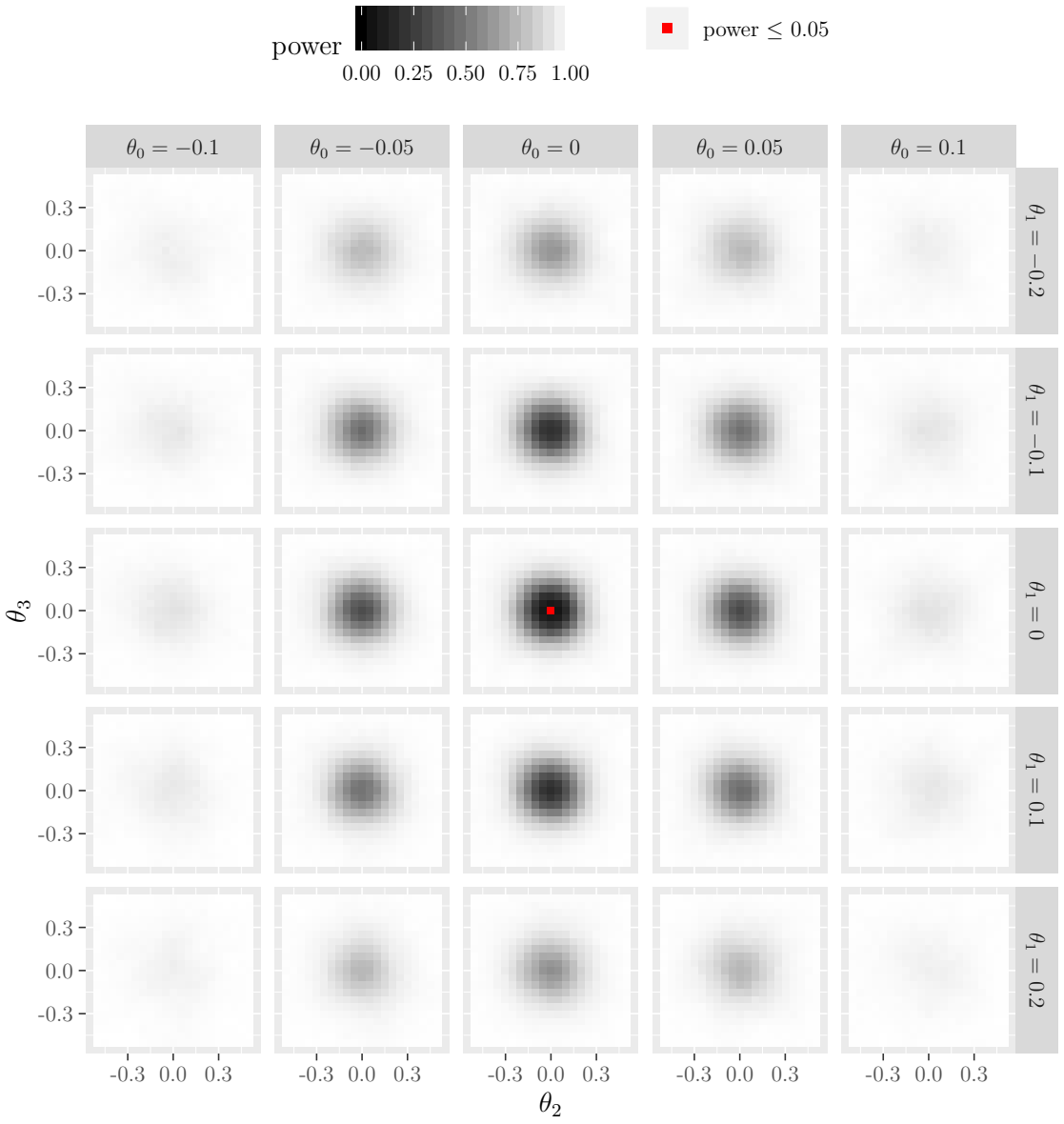


Figure 5.26: Simulated power functions of the ordering method based on the Shortest Hamiltonian Path for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$, $\boldsymbol{x}_{\bullet 2}$ and $\boldsymbol{x}_{\bullet 3}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

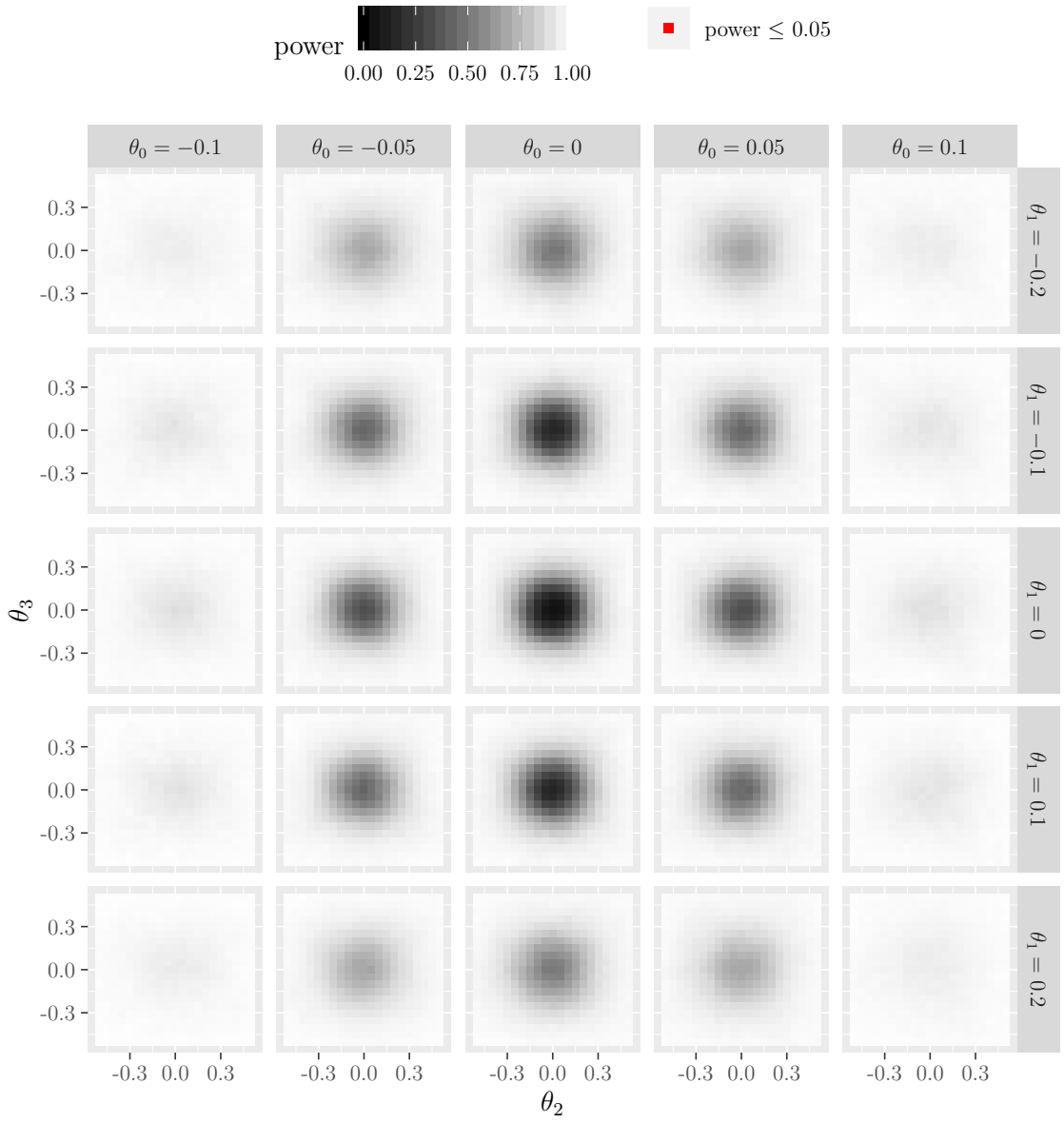


Figure 5.27: Simulated power functions of the ordering method based on the nearest neighbor approximation for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$, $\boldsymbol{x}_{\bullet 2}$ and $\boldsymbol{x}_{\bullet 3}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

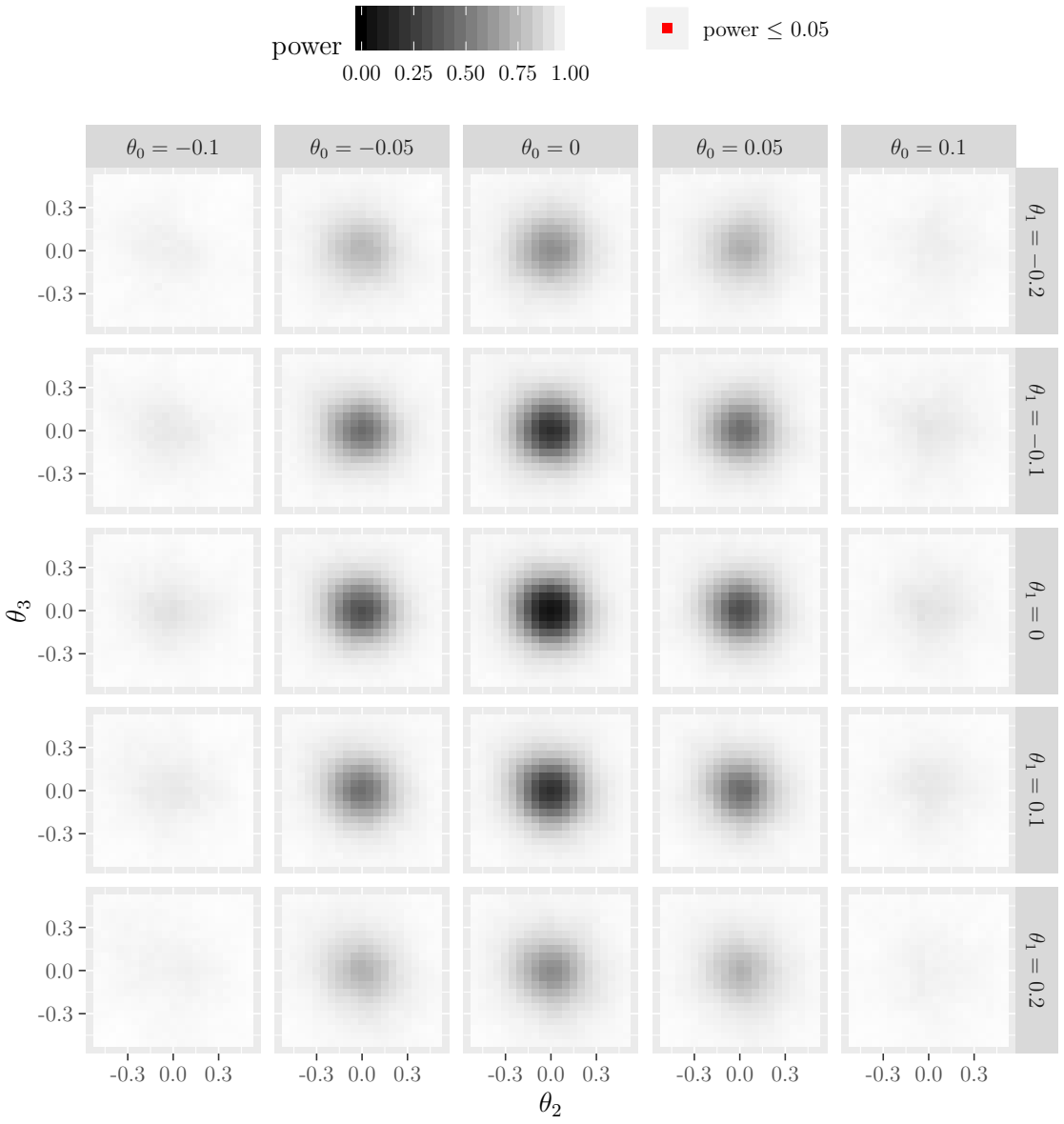


Figure 5.28: Simulated power functions of the ordering method based on a hierarchical clustering with complete linkage for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$, $\boldsymbol{x}_{\bullet 2}$ and $\boldsymbol{x}_{\bullet 3}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

The simulated power is less than or equal to $\alpha = 0.05$ only when $\boldsymbol{\theta}$ is zero, i.e. at H_0, and anywhere else the power is larger than α. The power converges in every direction to one and it can nicely be seen that this convergence is faster for θ_0 than for the other components of $\boldsymbol{\theta}$. This phenomenon can easily be explained: When increasing the intercept, all residuals get smaller by the same value and possibly many residuals which were positive are now negative. And the change of only some residuals from positive to negative (or vice versa) can have crucial effect on the sign depth test. On the other hand, changing the value of any other component of $\boldsymbol{\theta}$, i.e. changing a slope of the model, affects the residuals differently: Some will get larger and some will get smaller and so the effect on the sign depth test is different. The results of the approximate solution of the Shortest Hamiltonian Path and the hierarchical clustering are similar to those of the exact solution of the Shortest Hamiltonian Path, but the areas with low power are slightly larger for these two methods which means that they perform slightly worse.

This can also be seen when looking at Figure 5.29 which shows the percentages of power values for this model which are at least 0.05, 0.5 and 0.95. This figure looks very similar to Figure 5.25 on page 147, which has shown these percentages for a model with one regressor less. As in Figure 5.25 it can be seen in Figure 5.29 that (nearly) all power values in H_1 have a value of at least 0.05, independent of the number of data points, the parameter K and the ordering method. Also as seen before, the percentages of power values which are at least 0.5 are larger for $K = 3$ than for $K = 4$ and $K = 5$ when having only 10 data points. A difference here in comparison to Figure 5.25 is the smaller percentage of power values which are at least 0.5 for the nearest neighbor method when K is set to 3. But for sufficient large value of N this number also converges to one. In addition, when having only 20 or 30 data points, $K = 4$ has a larger percentage of values which are 0.95 or greater than $K = 3$ and $K = 5$ for all ordering methods. But as it was visible before, overall the ordering method based on the exact solution of the Shortest Hamiltonian Path performs best and for sufficient large values of N always $K = 5$ is best.

Overall, it has become clear in this subsection that the sign depth test performs very well for this type of linear regression models when using the distance based ordering methods. Since this type of linear regression models, models with an intercept and some linear regressors, is very much used in practice, the results of this subsection are very valuable. They have shown that the sign depth test is a good alternative to other robust and non-robust tests in case of standard multiple regression.

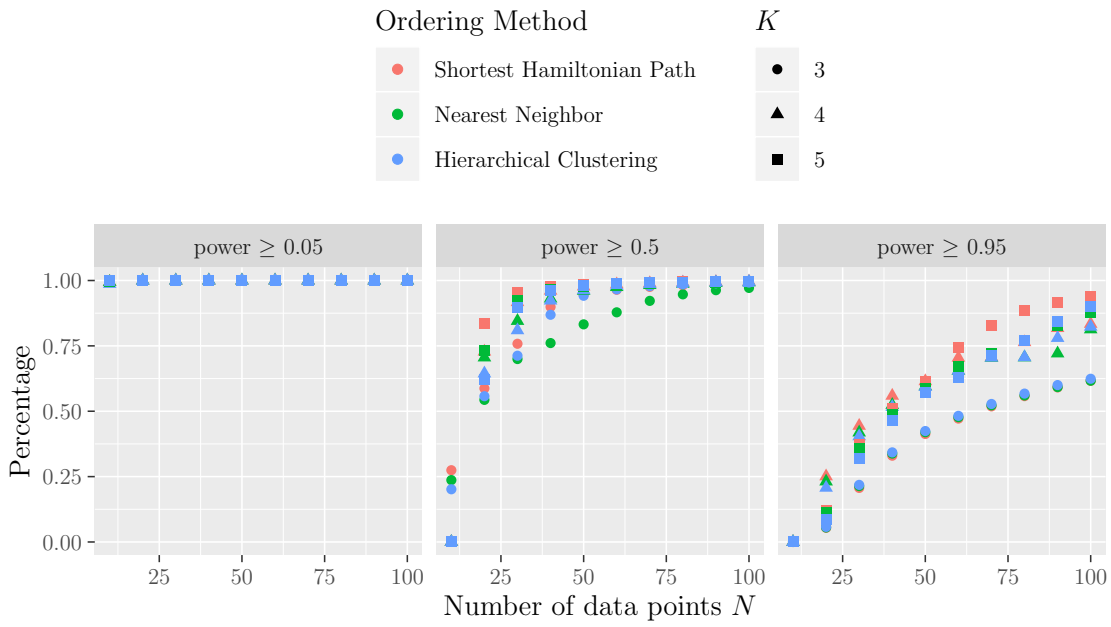


Figure 5.29: Percentages of power values in H_1 of all distance based ordering methods which are at least 0.05, 0.5 and 0.95. The tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 3} + \boldsymbol{e}$. The power is simulated on the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^4$. The underlying data set consists of N random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$, $\boldsymbol{x}_{\bullet 2}$ and $\boldsymbol{x}_{\bullet 3}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution.

5.3.2 Linear Models with Interactions

Next, it is looked at the power of the sign depth test when having interactions in the model. For this, two models will get analyzed in this subsection: One without intercept, i.e. $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$, and one with intercept, i.e. $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$.

Model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$

An extract of the simulated power functions for the model without intercept for $\theta_3 \in \{-0.5, -0.25, 0, 0.25, 0.5\}$ can be found in Figure 5.30. It can be seen that all three ordering methods perform satisfyingly, although in the direction of the parameter of the interaction (θ_3) values which are quite far away from H_0 are needed for getting large power values. So, very small deviations from the null-hypothesis may not be detected in this case. As in the previous subsection, the ordering method

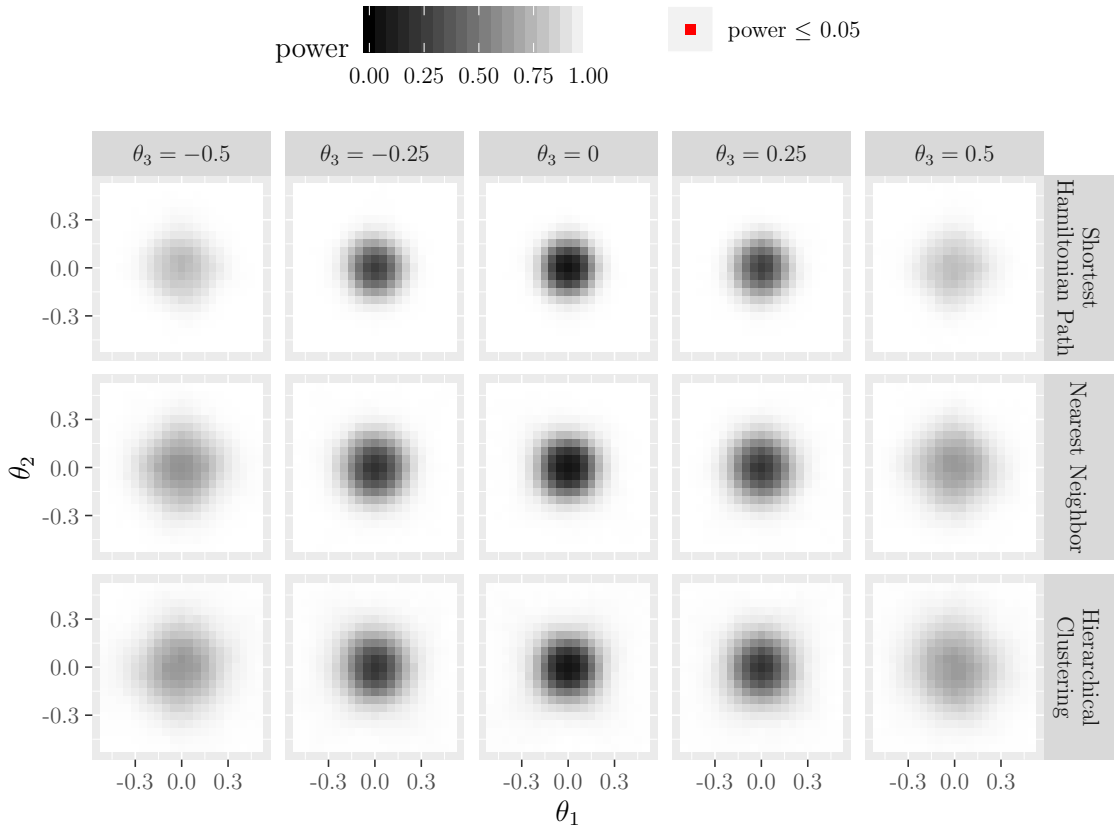


Figure 5.30: Simulated power functions of the distance based ordering methods presented in Section 3.4 for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0). For the hierarchical clustering method a complete linkage was chosen to order the regression vectors.

based on the exact solution of the Shortest Hamiltonian Path problem seems to perform best. Its area with low power is the smallest of the three methods. And the nearest neighbor method seems to have smaller areas with low power than the hierarchical clustering method.

This visual impression can be verified when looking at Figure 5.31. It shows the percentages of power values in the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^3$ which are at least 0.05, 0.5 and 0.95. It can be seen that this figure looks different

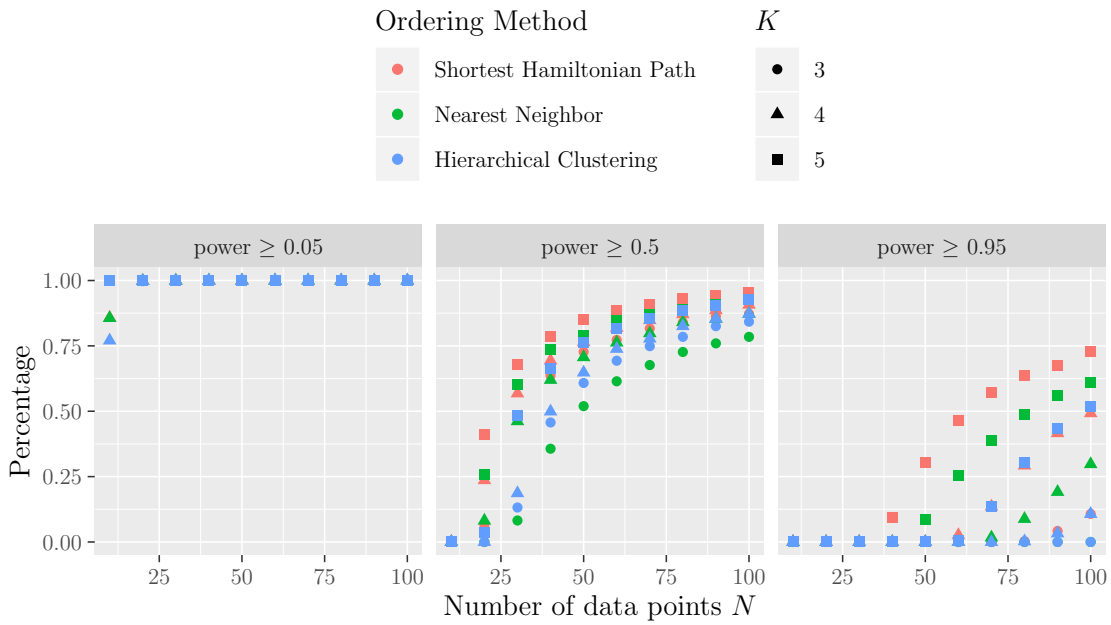


Figure 5.31: Percentages of power values in H_1 of all distance based ordering methods which are at least 0.05, 0.5 and 0.95. The tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The power is simulated on the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^3$. The underlying data set consists of N random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution.

than the figures from the previous subsection where no interaction was modeled in the simulations. Here, there is much more deviation between the different values. When looking at the left plot it can be seen that for $K = 4$ and $N = 10$ data points not all values have at least a value of 0.05 (the value of the Shortest Hamiltonian Path method is nearly the same as the nearest neighbor method and therefore cannot be seen). In Subsection 5.2.4 it could be seen that the sign depth test can behave differently for odd and even values of K, but this was mostly the case for the scalarization based methods and not for the distance based methods. In addition, this phenomenon can only be seen in this specific case when N is 10 and it is looked at the percentage of values which is at least 0.05. For larger values of N or larger power values, always $K = 5$ is best and $K = 3$ is worst. In addition, always the ordering method based on the exact solution of the Shortest Hamiltonian Path problem is best. This holds for the percentage of values which is at least 0.5 as well as for the percentage of values which is at least 0.95. But it can be seen that the values are smaller than the values in the previous subsection. While in the previous subsection

nearly all values of the Shortest Hamiltonian Path method were at least 0.95 (and so also greater than or equal to 0.5) for sufficient large values of N, here the percentages are clearly smaller than one. This can be explained by the above mentioned fact that quite large deviations from H_0 in the direction of the parameter of the interaction are needed to obtain great power values. Of course, the power values also get greater when N gets larger. For getting at least some power values which are greater than or equal to 0.5, here $N = 20$ data points are sufficient, for getting power values which are at least 0.95 $N = 40$ data points are needed for $K = 5$, $N = 60$ data points for $K = 4$ and $N = 90$ data points for $K = 3$. For the other two distance based ordering methods, it is not clear which one is better. When looking at the percentages of values which are at least 0.5 or 0.95, at least for $K = 3$ the hierarchical clustering is better than the nearest neighbors approximation. On the other hand, for $K = 4$ and $K = 5$ the nearest neighbor method is better than the hierarchical clustering. But both methods are clearly worse than the Shortest Hamiltonian Path method. All in all, it can be said that interactions in a model make it harder for the sign depth test to reach great power values than it is for models without interactions. But the results are satisfying, nevertheless. The power functions of all methods have satisfying characteristics with areas with low power only near the null-hypothesis and great power values far away from the null-hypothesis.

Model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$

In the following, an intercept will be added to the model used before, which leads to the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. Extracts of the simulated power functions for this model can be found in Figures 5.32, 5.33 and 5.34. The power functions are visualized for $\theta_0 \in \{-0.1, -0.05, 0, 0.05, 0.1\}$ and $\theta_3 \in \{-0.5, -0.25, 0, 0.25, 0.5\}$. The values for θ_1 and θ_2 are shown in the complete interval from -0.5 to 0.5. These figures show the expected behavior: In the direction of the parameter of the interaction (θ_3) relatively large deviations from H_0 are needed to get great power values, whereas this is not the case in the direction of the intercept (θ_0). For the intercept, small shifts from H_0 are sufficient to get great power values. This behavior could also be seen in the previous subsection and of course, it also holds here in a model with an interaction. Furthermore, again it can be seen that the ordering method based on the exact solution of the Shortest Hamiltonian Path problem seems to perform best.

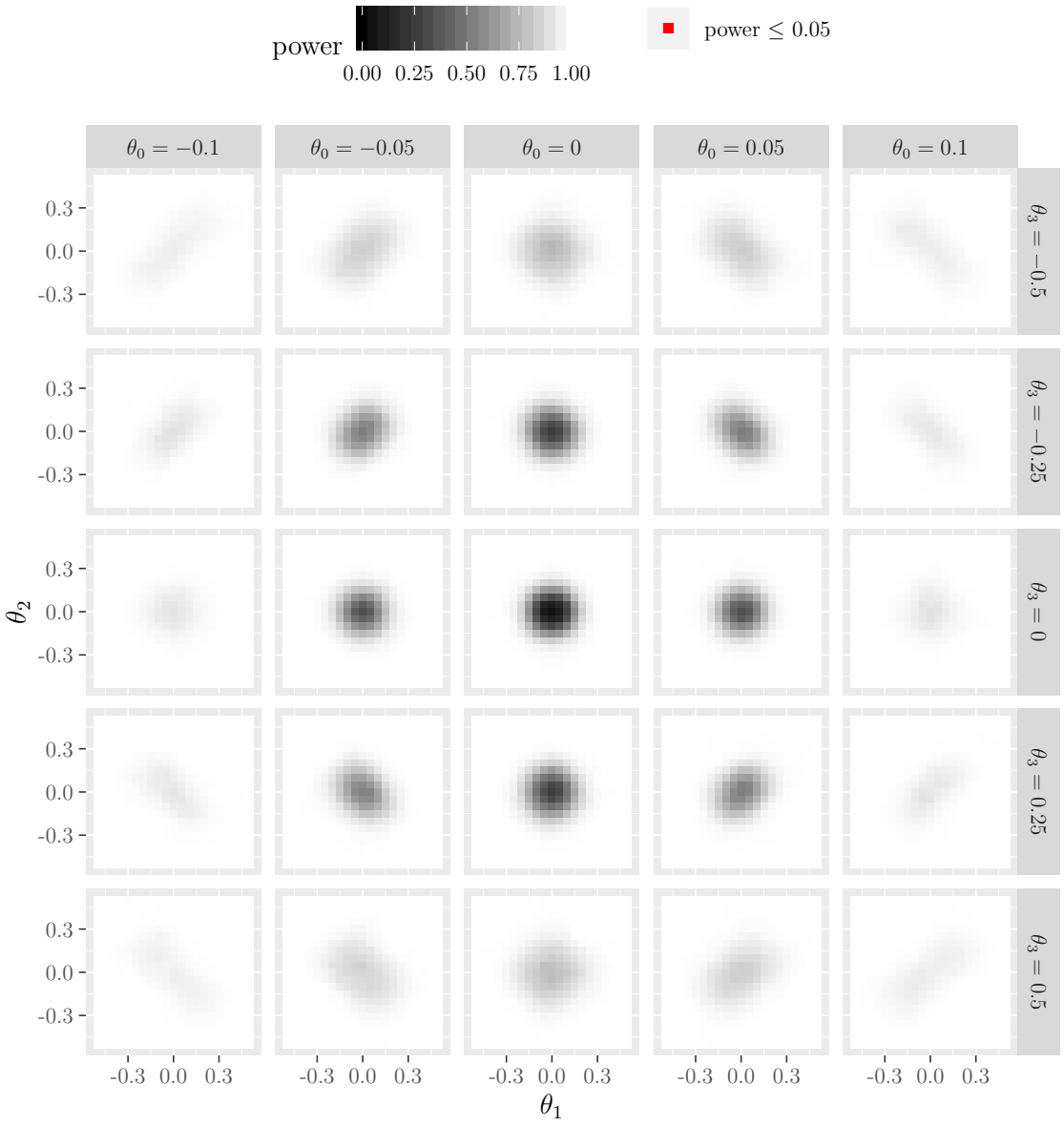


Figure 5.32: Simulated power functions of the ordering method based on the Shortest Hamiltonian Path for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

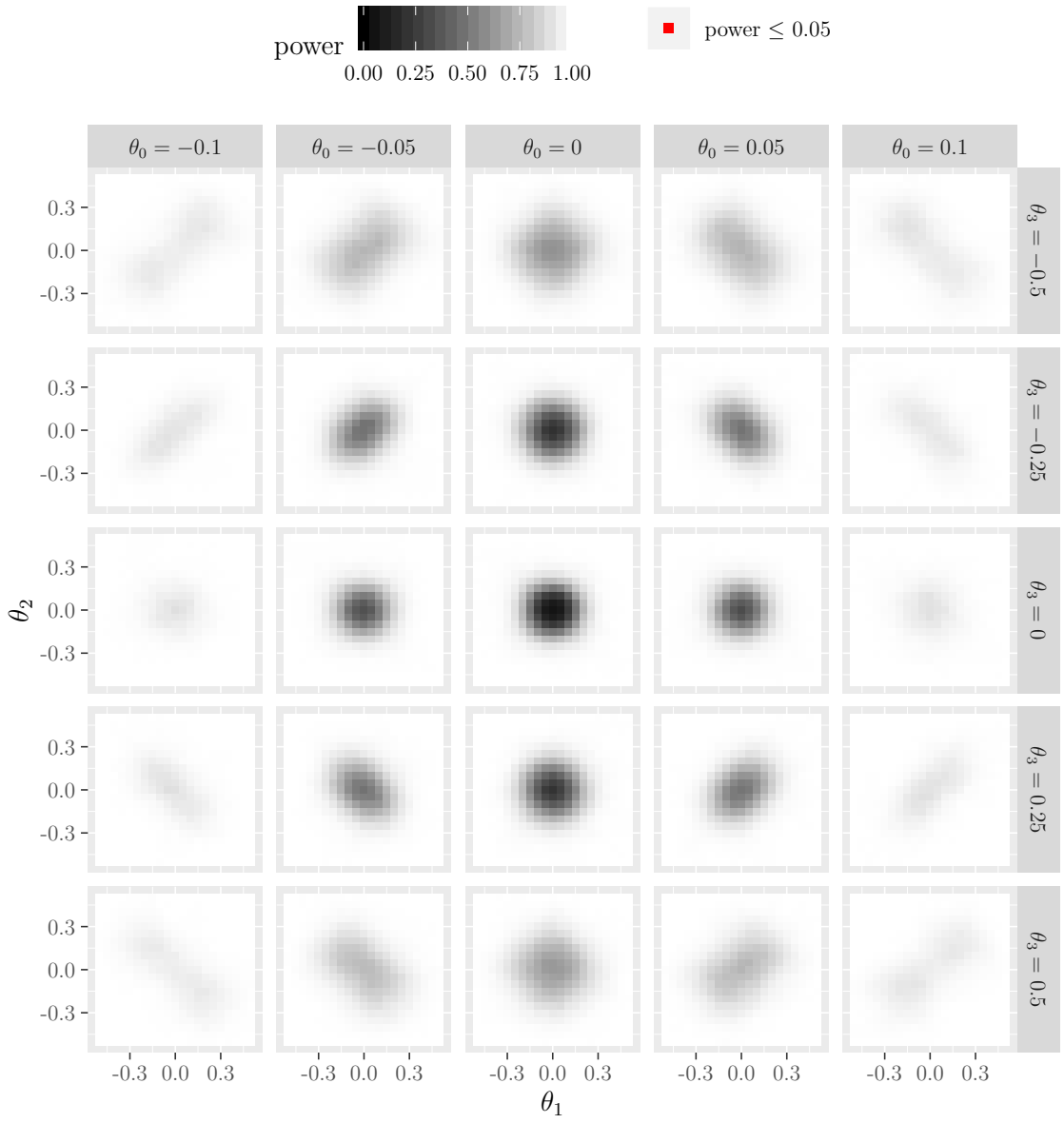


Figure 5.33: Simulated power functions of the ordering method based on the nearest neighbor approximation for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

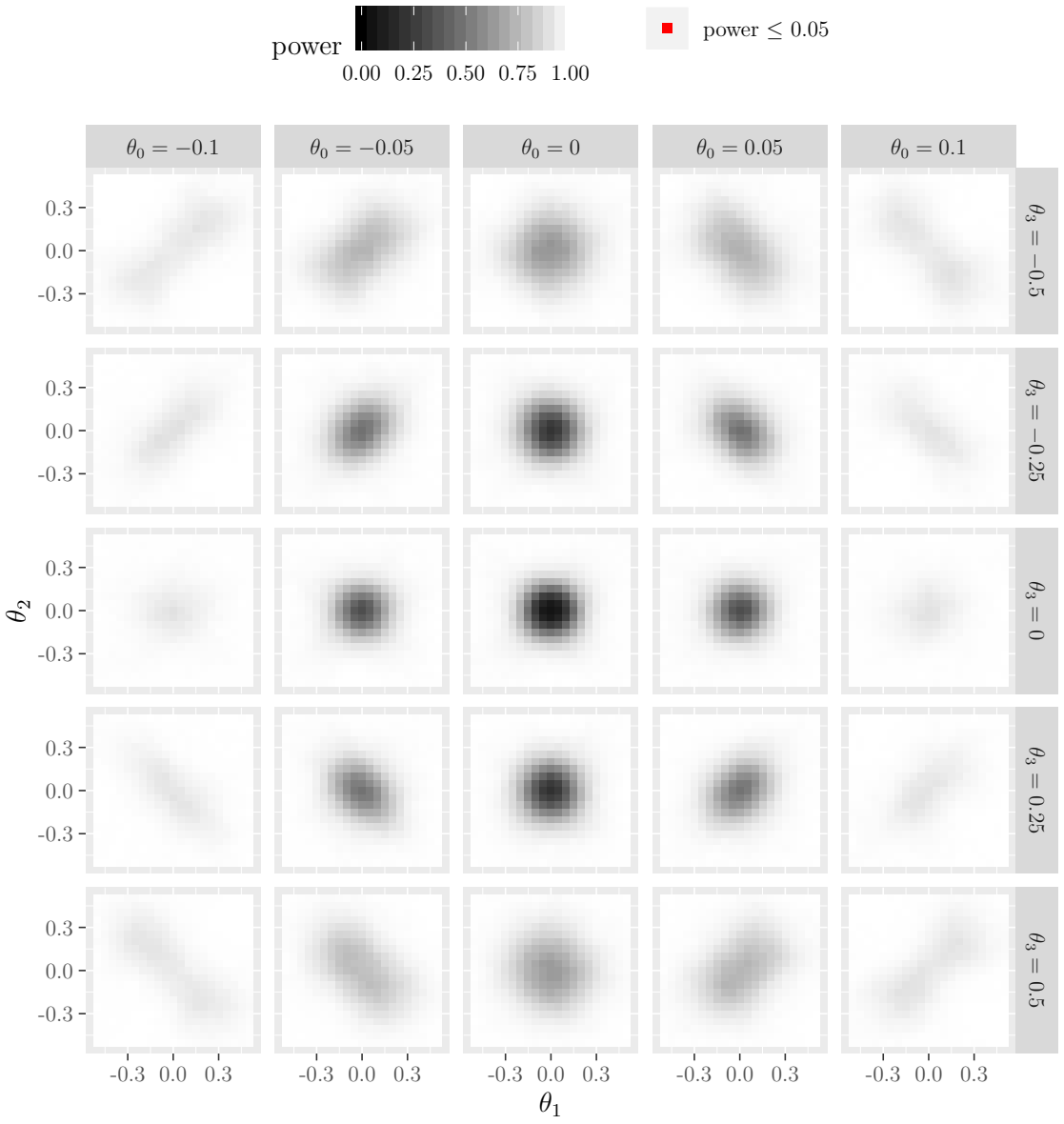


Figure 5.34: Simulated power functions of the ordering method based on a hierarchical clustering with complete linkage for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

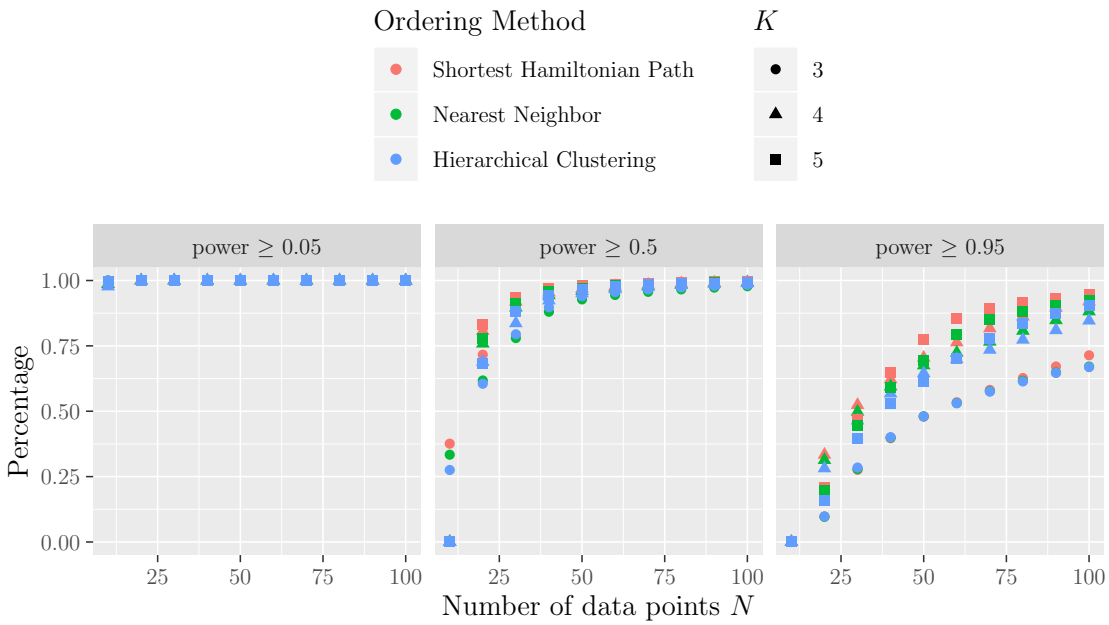


Figure 5.35: Percentages of power values in H_1 of all distance based ordering methods which are at least 0.05, 0.5 and 0.95. The tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The power is simulated on the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^4$. The underlying data set consists of N random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution.

This can also be seen when looking at Figure 5.35. Although the differences between the ordering methods are not as large as they were for the model without intercept. Having an intercept in the model, for which small deviations from H_0 are sufficient for getting large power values, of course has a crucial effect on the percentages. But overall, the result stays the same: The Shortest Hamiltonian Path method with parameter $K = 5$ performs best. All methods and all values of K lead to power functions where (nearly) all power values in H_1 have values of at least 0.05. As in the previous subsection, it can be seen that for very small N, $K = 3$ performs better than $K = 4$ and $K = 5$ when looking at the percentages of power values which are at least 0.5. And when looking at the percentages of values which are at least 0.95, $K = 4$ is best for $N = 20$ and $N = 30$. In this plot it can also be seen very well that overall $K = 3$ is worse than $K = 4$ and $K = 5$. Its values are clearly smaller than the other values.

Overall, this subsection has shown that the sign depth test performs satisfyingly when having interactions in the model. Although for getting large power values in

the directions of the parameters of the interactions quite large deviations from H_0 and/or large numbers of data points are necessary. The best results always were obtained when using the exact solution of the Shortest Hamiltonian Path problem for ordering.

5.3.3 Quadratic Regression

Next, the power of the sign depth test in case of polynomial regression is analyzed. Here, it is focused on quadratic regression, i.e. it is looked at the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. Adding an intercept to that model would lead to the same phenomenon as in the previous subsections: In the direction of the parameter of the intercept the power values are quite large already for small deviations from H_0. Because of this and because the parameter vector $\boldsymbol{\theta}$ is already four-dimensional in this model, the intercept is omitted here.

The simulated power functions of the sign depth test applied with the three distance based ordering methods can be found in Figures 5.36, 5.37 and 5.38. It can be seen that these power functions look a little different to the before seen power functions. In these figures the power functions are only shown for $\theta_3, \theta_4 \in \{-0.5, -0.25, 0, 0.25, 0.5\}$ which are the two parameters for the quadratic terms. Interestingly, the power values of all three simulated power functions are lower when $\theta_3 = -\theta_4$ holds. This can be explained by the fact that these parameters cancel each other out in this specific situation. Here, everything is symmetrically simulated: The model is symmetric in $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ and the data points in the data set consist of random values in the interval $[-1, 1]$, which means that they are also relatively symmetric around zero. And when θ_3 and θ_4 are canceling each other out, it is hard for every test to detect deviations from the null-hypothesis if θ_1 and θ_2 have also values in the area of the null-hypothesis. This phenomenon of lower power values can also be seen in a weakened way when $|\theta_3 + \theta_4| = 0.25$ because then the deviation between θ_3 and $-\theta_4$ is relatively small. In general, it can be seen in Figures 5.36, 5.37 and 5.38 that all three power functions are quite satisfying, although the best power function again is obtained when using the exact solution of the Shortest Hamiltonian Path problem for ordering. Especially, the hierarchical clustering method has larger areas with lower power values than the two other methods. But when $|\theta_3 + \theta_4| > 0.25$, all three methods produce very large power values, independently of θ_1 and θ_2 which is a very good result.

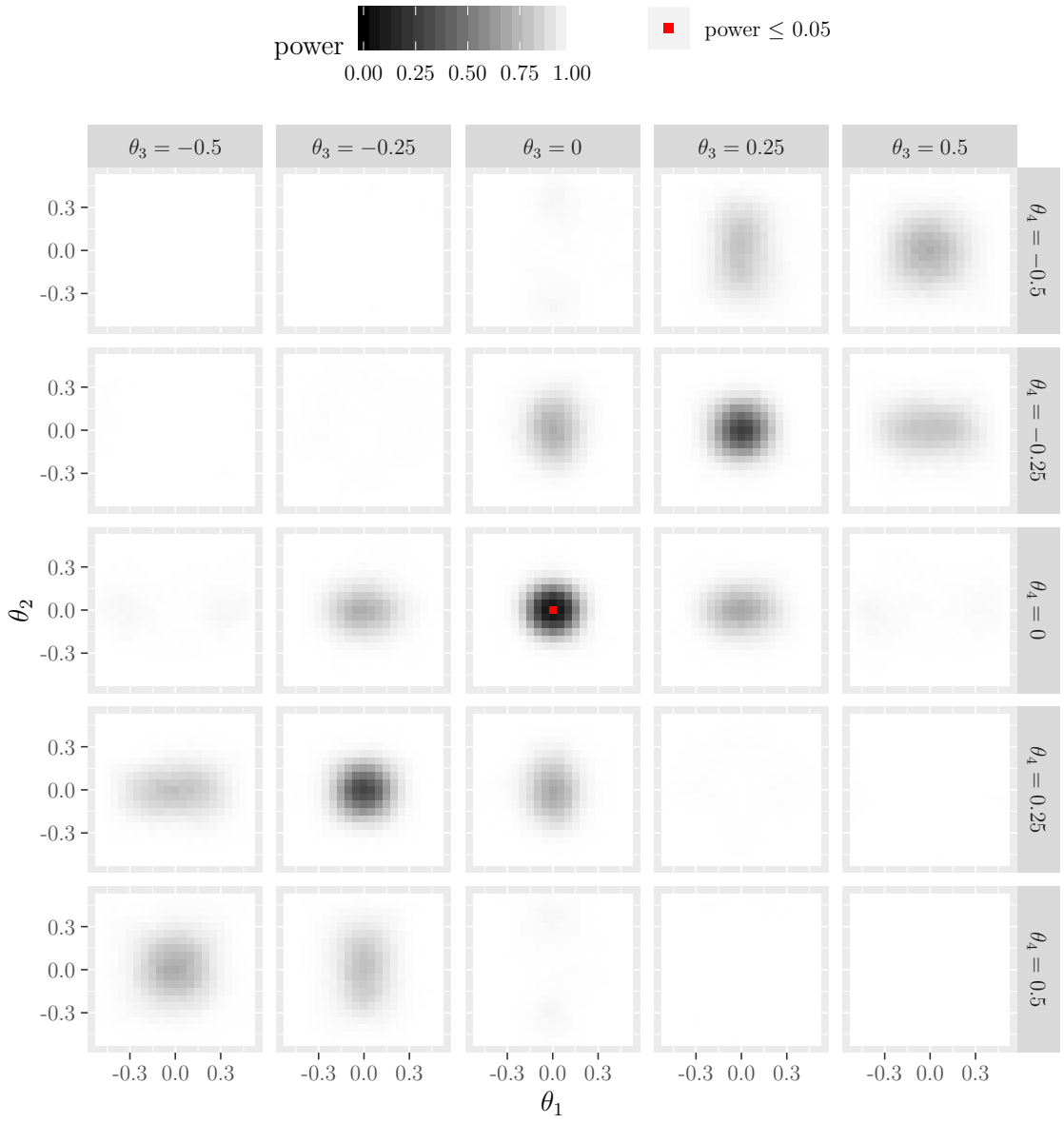


Figure 5.36: Simulated power functions of the ordering method based on the Shortest Hamiltonian Path for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

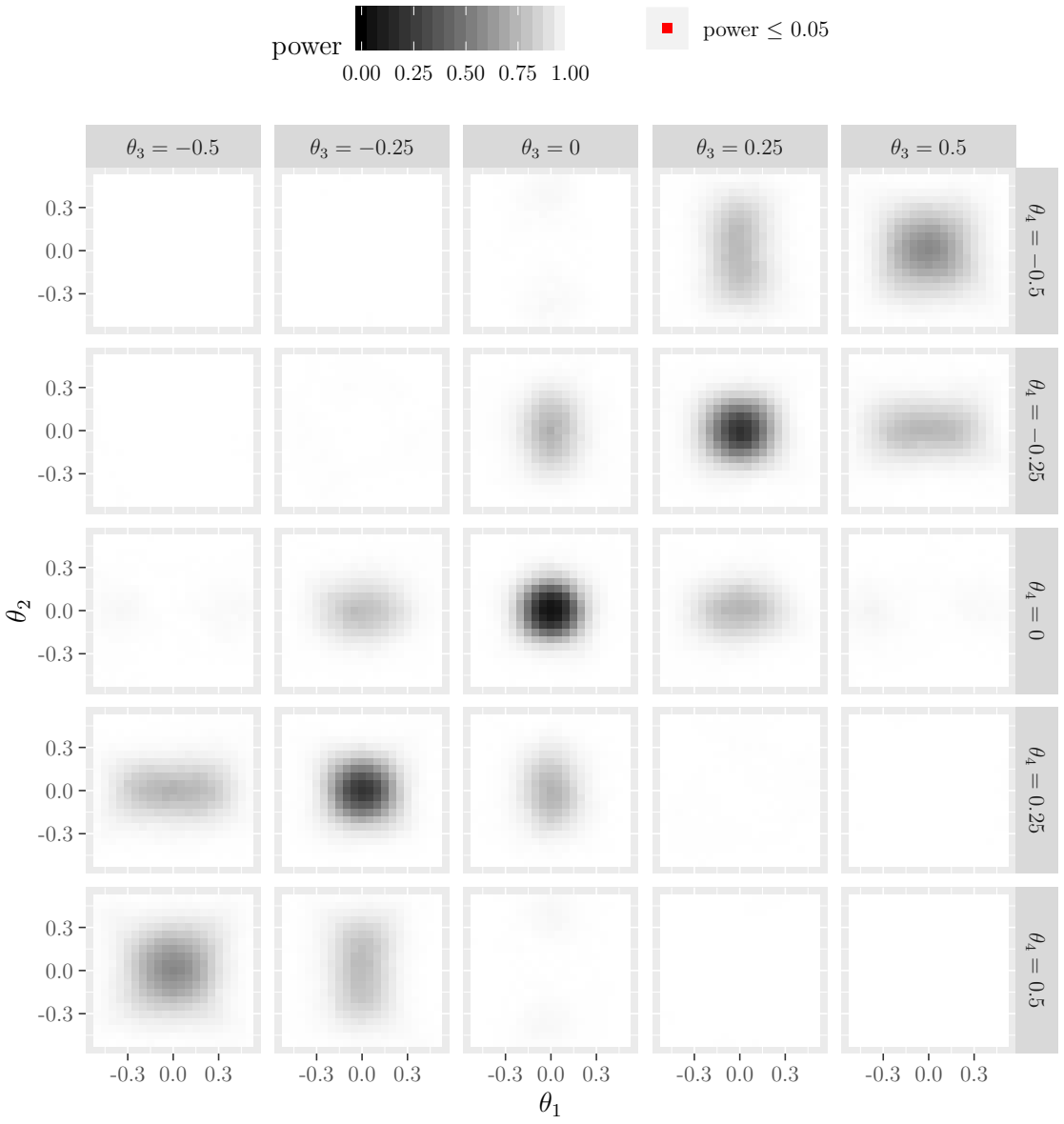


Figure 5.37: Simulated power functions of the ordering method based on the nearest neighbor approximation for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

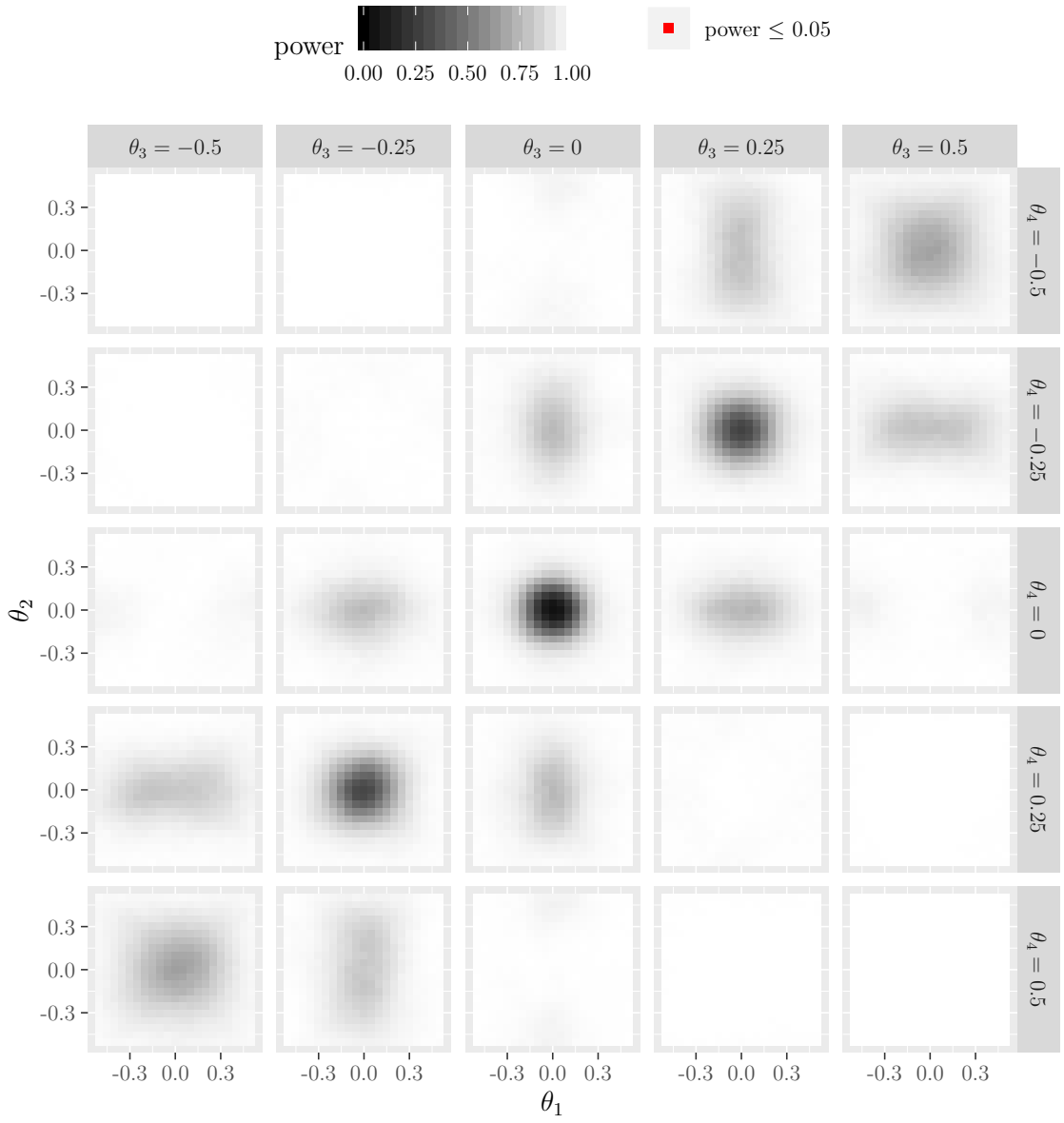


Figure 5.38: Simulated power functions of the ordering method based on a hierarchical clustering with complete linkage for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

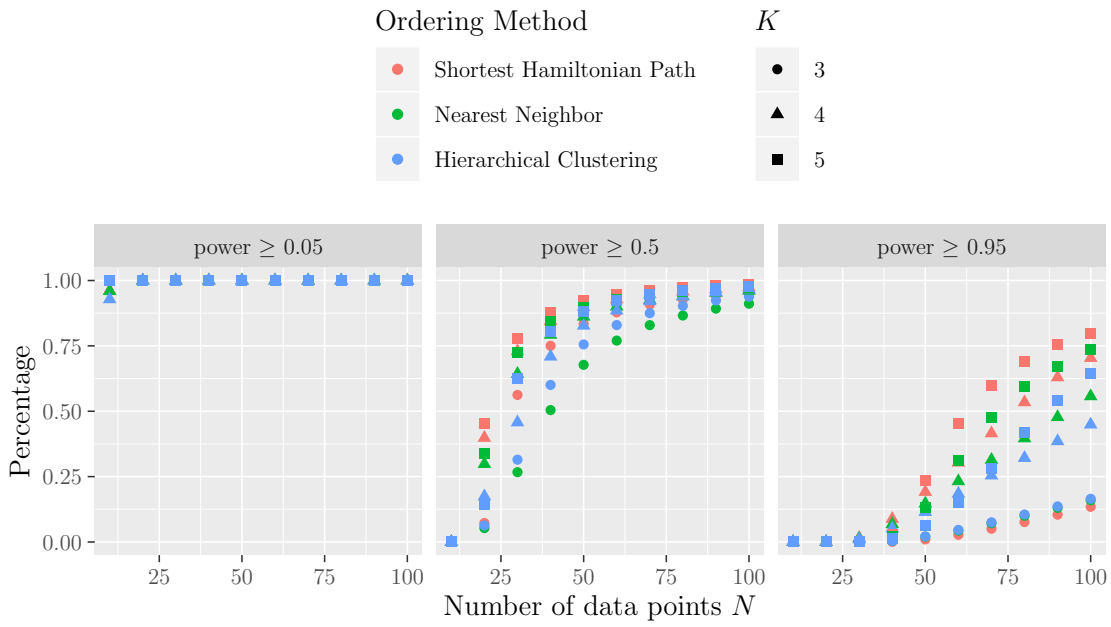


Figure 5.39: Percentages of power values in H_1 of all distance based ordering methods which are at least 0.05, 0.5 and 0.95. The tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The power is simulated on the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^4$. The underlying data set consists of N random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution.

In Figure 5.39 the percentages of power values which are at least 0.05, 0.5 and 0.95 are shown for this model. This figure is quite similar to Figure 5.31 on page 155 which has shown the percentages for the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. Like there, here no intercept is modeled which leads to similar characteristics of both figures. For example, it can be again seen that for $N = 10$ data points and parameter $K = 4$ not all power values are at least 0.05 which is the case for all other values of N and K. And for $N = 10$ no power values are greater than or equal to 0.5, independently of the ordering method and the value of K. But for $N \geq 20$, all ordering methods and all values of K have some power values which are at least 0.5. The percentages are increasing for all ordering methods and all values of K until for $N = 100$ all percentages are near the value of one, which means that nearly all power values in H_1 have a value of at least 0.5. This is different to Figure 5.31. Also, it is a large difference that all ordering methods and all values of K are able to produce power values of 0.95 or greater for sufficient large N. In Figure 5.39 it can be seen that for $N \geq 50$ this is reached. Interestingly, this plot shows that for $K = 3$ the

hierarchical clustering method has larger percentages of power values which are at least 0.95 than the other two methods. For $K = 4$ and $K = 5$ this is obviously not the case. Here, again the exact solution of the Shortest Hamiltonian Path is clearly the best and the hierarchical clustering method is the worst method of these three methods. And again it can be seen that $K = 5$ is better than $K = 4$ which is again better than $K = 3$.

Overall, the results of the simulated power functions are really satisfying. Applying the sign depth test to models with quadratic terms is possible and leads to good results. In general, already small deviations from the null-hypothesis are sufficient to get large power values of the test, especially in the directions of the quadratic terms, unless some parameters cancel each other out. In this case the power of the test is rather low, especially for small deviations from the null-hypothesis, but in any case it is larger than α.

5.3.4 High Dimensional Linear Models

So far, models with two-, three-, and four-dimensional parameter vector $\boldsymbol{\theta}$ were analyzed. In this subsection it will be looked at models with much more regressors, especially for looking whether in this case the sign depth test has the same power as in models with less regressors and whether rather small values of K are still sufficient for getting great power values. It is expected that small values of K lead to rather bad results in the case of high dimensional models and that the value of K should have at least the same magnitude as D for getting great power values in the alternative hypothesis.

For answering this question in this subsection the model $\boldsymbol{y} = \sum_{d=1}^{D} \theta_d \boldsymbol{x}_{\bullet d} + \boldsymbol{e}$ is considered with $D \in \{10, 20, 40, 80\}$ and $N = 100$ random regression vectors in the interval $[-1, 1]^D$ each. For K the usual values 3, 4 and 5 are used and additionally $K = 11$ and $K = 21$ will be considered for having values which have the same magnitude as D in two of four cases. Considering larger values of K for having also values with the same magnitude for $D = 40$ and $D = 80$ is unfortunately not possible due to three reasons. Firstly, for these large values of K the values of the sign depth are so small that there may occur numerical problems and instability when computing the sign depth and the needed quantile of the sign depth test which leads to results that are not reliable. Secondly, as written in Chapter 4, an exact

implementation of the sign depth in linear runtime was available at the moment of the simulation execution only for $K \leq 5$ and computing the exact sign depth for larger K had time complexity of $\mathcal{O}(\binom{N}{K})$ which is too much to calculate the sign depth for the test and especially for simulating the needed quantile. Of course, for this the asymptotic implementation of Dennis Malcherczyk which is also implemented in the `R`-package `GSignTest` could be used (which was made for $K = 11$ and $K = 21$), but is it not clear how large the deviations from the exact values are for such large values of K. This would again lead to non-reliable results. And thirdly, the sign depth and sign depth test cannot work when N is too small in comparison to K: When N is not much larger than K, the required α-quantile for the sign depth test is the same as the minimum of the respective distribution and so the test cannot work. As a rule of thumb, the number of data points N should be at least two to three times as large as the value of K. Of course, for avoiding the third reason, the number of data points could be enlarged for these simulations, but this would have no effect on the first and second reason. Because of this, in this subsection only $K \in \{3, 4, 5, 11, 21\}$ is considered.

Figures 5.40 and 5.41 show extracts of the simulated power functions for the four high-dimensional models and the five values of K. The sign depths for the tests were computed exactly for $K \in \{3, 4, 5\}$ and asymptotically for $K \in \{11, 21\}$. Figure 5.40 shows the power functions for $\theta_1 \in \{-1, -0.98, -0.96, ..., 0.96, 0.98, 1\}$, all other components of $\boldsymbol{\theta}$ are set to zero and Figure 5.41 shows the power functions for $\theta_1 = \ldots = \theta_D = \gamma$ with $\gamma \in \{-1, -0.98, -0.96, ..., 0.96, 0.98, 1\}$. Due to the high dimensionality it was not possible to compute and show the whole power functions and because of this it was decided to focus on these two aspects of the power functions only. The first aspect shows the behavior of the power functions when only one component of $\boldsymbol{\theta}$ is different from the null-hypothesis $H_0 : \boldsymbol{\theta} = \mathbf{0}$. Because of the symmetry of the model in the components of $\boldsymbol{\theta}$, the power functions would look the same when choosing any other component than the shown θ_1 for the x-axis of the plots. The second aspect shows the behavior of the power functions when varying all components of $\boldsymbol{\theta}$ by the same factor, i.e. it describes the behavior on the diagonal line through the D-dimensional hyperspace.

The two Figures 5.40 and 5.41 show very interesting results. Although all power functions seem to have a power of about α at H_0 and larger power values in the area of H_1, there are great differences between the single power functions. Interestingly, for $K \in \{3, 4, 5\}$ for all four models the hierarchical clustering ordering method is

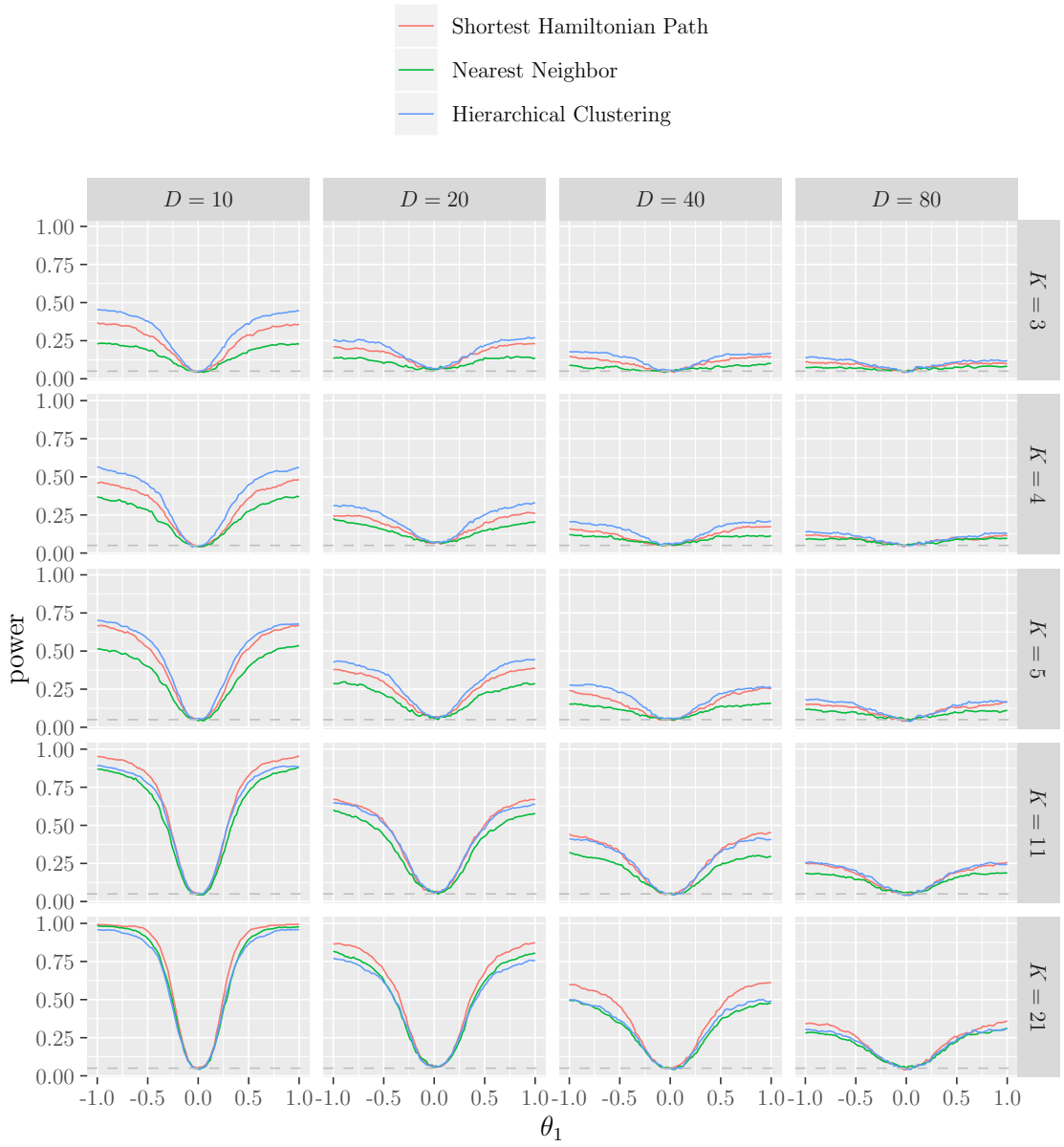


Figure 5.40: Extracts of the simulated power functions of the sign depth test for the model $\boldsymbol{y} = \sum_{d=1}^{D} \theta_d \boldsymbol{x}_{\bullet d} + \boldsymbol{e}$, where the vector $\boldsymbol{e}$ has a normal distribution. The hypothesis $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ is tested. Here, the power functions are only shown for $\theta_1 \in \{-1, -0.98, -0.96, ..., 0.96, 0.98, 1\}$, all other values of $\boldsymbol{\theta}$ are zero. For the sign depth test the distance based ordering methods presented in Section 3.4 are used, where for the hierarchical clustering method a complete linkage was chosen. The sign depths were computed exactly for $K \in \{3, 4, 5\}$ and asymptotically for $K \in \{11, 21\}$. The gray dashed line shows the level of the test $\alpha = 0.05$.

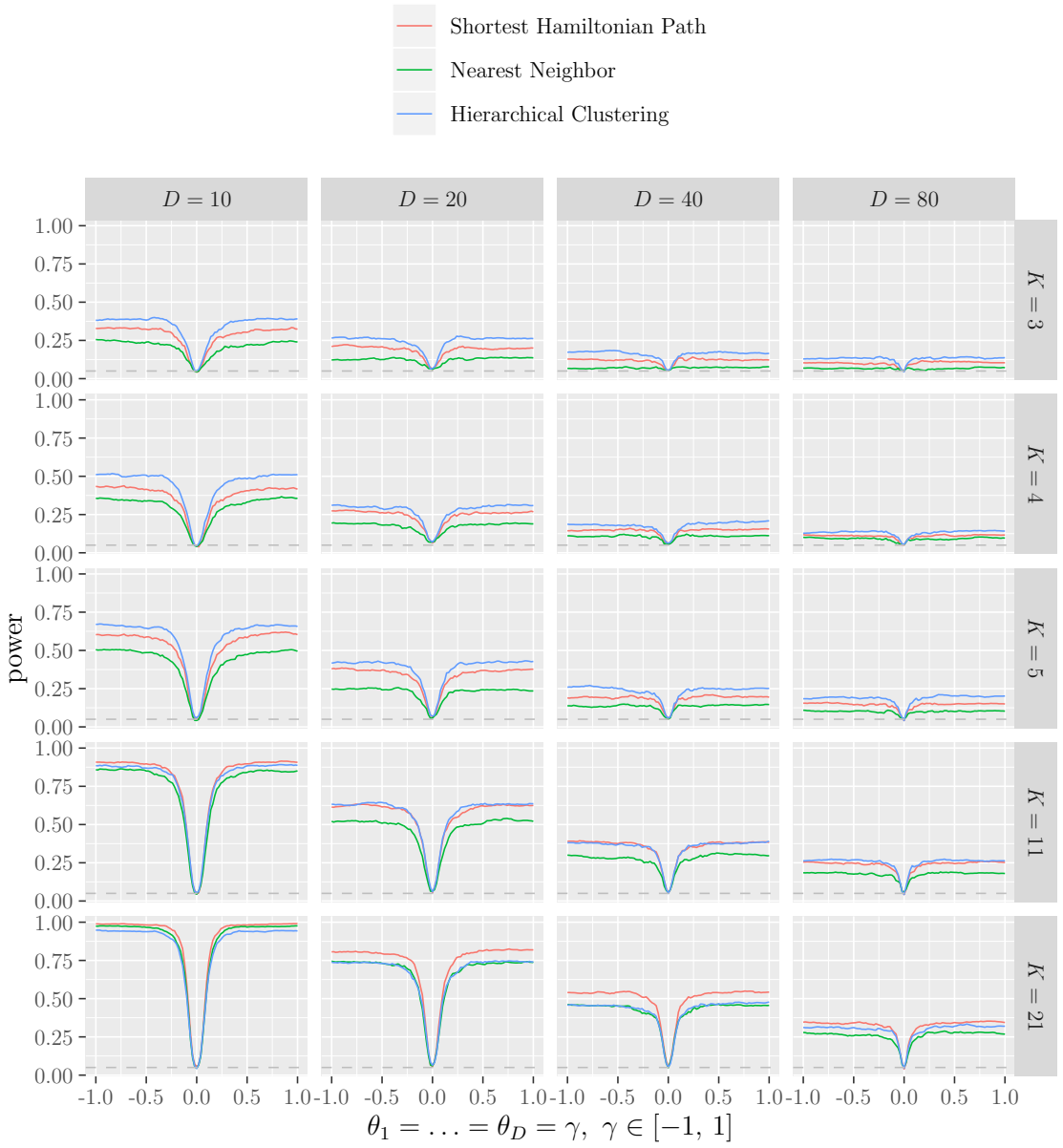


Figure 5.41: Extracts of the simulated power functions of the sign depth test for the model $\boldsymbol{y} = \sum_{d=1}^{D} \theta_d \boldsymbol{x}_{\bullet d} + \boldsymbol{e}$, where the vector $\boldsymbol{e}$ has a normal distribution. The hypothesis $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ is tested. Here, the power functions are only shown for $\theta_1 = \ldots = \theta_D = \gamma$ with $\gamma \in \{-1, -0.98, -0.96, ..., 0.96, 0.98, 1\}$. For the sign depth test the distance based ordering methods presented in Section 3.4 are used, where for the hierarchical clustering method a complete linkage was chosen. The sign depths were computed exactly for $K \in \{3, 4, 5\}$ and asymptotically for $K \in \{11, 21\}$. The gray dashed line shows the level of the test $\alpha = 0.05$.

better than the exact solution of the Shortest Hamiltonian Path problem. But this is not the case for $K \in \{11, 21\}$ where again the Shortest Hamiltonian Path is better than the hierarchical clustering. Furthermore, it can be seen that the sign depth test has quite low power when K is smaller than D. In contrast, the power values are much better when K and D have the same magnitude and the power values are very good when K is larger than D, i.e. $K = 21$ and $D = 10$. In addition, it can be seen that the behavior for the first aspect is slightly different than for the second aspect: While for the first aspect in Figure 5.40 the power values seem to increase a little bit slower, for the second aspect the increase of the power values in the area of H_1 is quite steep. But the power values then remain at values which are much smaller than one if K is smaller than D. The value on which the power values remain is smaller the larger the difference $D - K$ is. Especially, for $D = 80$ and $K = 3$, the power values are not much larger than $\alpha = 0.05$ anymore.

These results have clearly shown that the value of K has a crucial effect on the power of the sign depth test. Especially, K should not be smaller than the number of dimensions D, but at least be at the same magnitude. The best would be to have a value of K which is larger than D. But, one has to think about the fact that due to numerical issues the sign depth and its distribution cannot be calculated for arbitrary K, but only for small and medium values at the moment. Also, it is not possible to choose a value of K which is not sufficiently smaller than N because then the required quantile of the distribution of the sign depth is the same as the minimum value of this distribution. So, when having a high-dimensional model for which only values of K are possible which are smaller than D, the sign depth test may have no large power. In addition, the results in this subsection have shown that in the case where K is smaller than D a hierarchical clustering as ordering method for the sign depth test performs best. But as soon as K has at least the same magnitude as D, the exact solution of the Shortest Hamiltonian Path should be used for ordering the regression vectors.

5.4 Choice of Vectors Used for Ordering

Especially in the context of the considered models in Section 5.3 one question arises: Which components of the regression vectors should be used for ordering the data? Only the so-called design vectors $(x_{n1}, x_{n2})^\top$, $n = 1, \ldots, N$ or the complete regression

vectors, e.g. $(1,\, x_{n1},\, x_{n2},\, x_{n1}x_{n2})^\top$ for the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}\boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ or $(x_{n1},\, x_{n2},\, x_{n1}^2,\, x_{n2}^2)^\top$ for the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$?

In this thesis, the second possibility was chosen: Always the complete regression vectors were used for multidimensional ordering. There are several reasons for this decision: Firstly, this way of ordering uses all available information about the data and the model and not only the information provided by the design vectors. Secondly, it is easier and less error-prone to implement the sign depth test in `R` when using the complete regression vectors for ordering. As described in Algorithm 4.8 on page 98 and in Section 4.4 the function `depth.test()` in the package `GSignTest` can be used with a model description (i.e. a `formula`) and a data set (i.e. a `data.frame`) as input. The complete design matrix (which contains row-wise the regression vectors) can then be easily extracted with the help of the function `model.matrix()` whereas it would be more difficult to distinguish between components which belong to the design vectors and other components in this case. And thirdly, the best performing ordering method, the Shortest Hamiltonian Path, has smaller empirical runtimes when having higher dimensional data, see Figure 3.27 on page 69. Especially, in the two-dimensional case the runtimes are quite large. Because of this it seems advantageous to use all components of the regression vectors for ordering.

To show exemplary the differences of both choices of used vectors for ordering, in the following the polynomial model with quadratic terms from Subsection 5.3.3 is used, i.e. $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. Figure 5.42 shows the different ordering variants for the three data sets used in Chapter 3 (LHS-data, Grid-data and Spiral-data) when using the exact solution of the Shortest Hamiltonian Path problem for ordering, which is the best performing ordering method. In the upper row the design vectors $(x_{n1},\, x_{n2})^\top,\ n = 1, \ldots, N$ are ordered and in the lower row the regression vectors $(x_{n1},\, x_{n2},\, x_{n1}^2,\, x_{n2}^2)^\top,\ n = 1, \ldots, N$ are ordered. It can be seen that the obtained orders differ slightly for the LHS-data and the Grid-data, although both ordering variants lead to orders which have no real "disorder" in the obtained orderings. The inherent order of the Spiral-data is obtained in both cases. The previous sections have shown that the sign depth test performs well as long as there is not too much "mess" in the ordering. Obviously, this is the case for both ordering variants. So, although the orders may differ slightly (dependent on the data set) and so the results of the sign depth test may differ slightly, it can be expected that both ordering variants will lead to satisfying results.

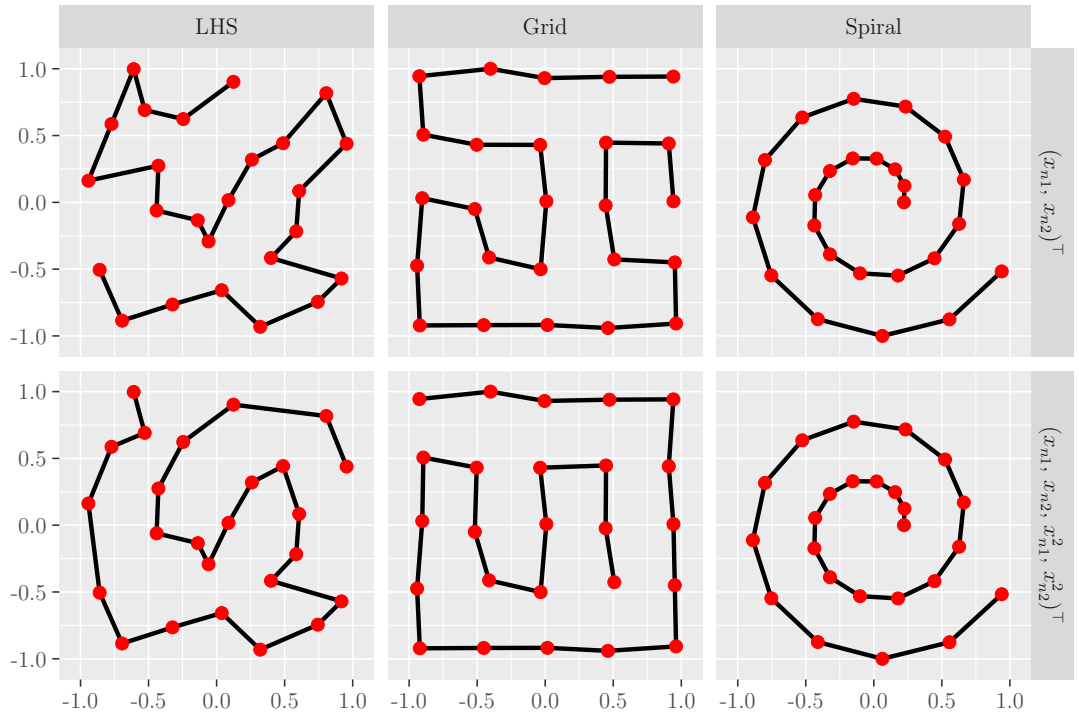


Figure 5.42: Visualization of the obtained orders when using a Shortest Hamiltonian Path for ordering. In the upper row the data points are ordered two-dimensionally according to the plotted values. In the lower row the data points are ordered four-dimensionally according to the plotted values as well as their squared values.

This expected result can be seen when looking at Figure 5.43. This figure shows the percentages of power values which are at least 0.05, 0.5 and 0.95 for the quadratic regression model when using the distance based ordering methods and the sign depth test with parameter $K = 5$. It can be seen that both ordering variants, ordering the design vectors as well as ordering the regression vectors, lead to satisfying results which are very similar. Both variants have always a power of at least 0.05 in H_1 and both variants need at least 20 data points for having power values which are at least 0.5 and 40 data points for having power values which are at least 0.95. Especially, the values for an ordering according to the exact solution of the Shortest Hamiltonian Path problem are very similar, whereas there is slightly more difference for the nearest neighbor method and the hierarchical clustering method. All results show that ordering the design vectors leads to slightly better results than ordering the regression vectors, but when increasing the number of data points, the differences

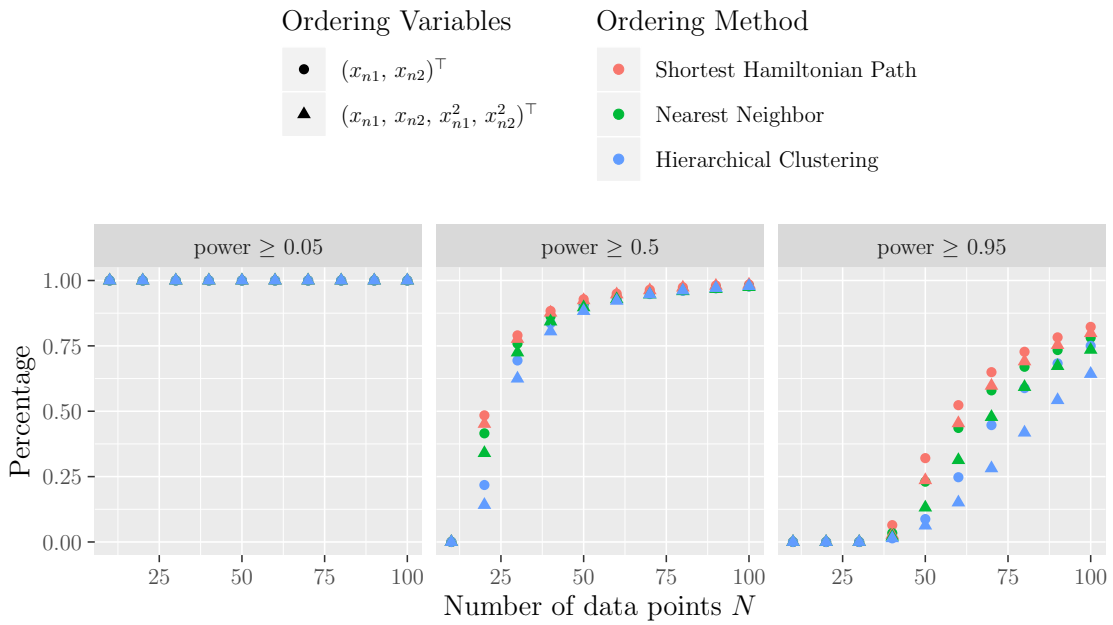


Figure 5.43: Percentages of power values in H_1 of all distance based ordering methods which are at least 0.05, 0.5 and 0.95. The tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The power is simulated on the interval $\boldsymbol{\theta} \in \{-0.4, -0.35, -0.3, \ldots, 0.3, 0.35, 0.4\}^4$. The underlying data set consists of N random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a normal distribution. The parameter K of the sign depth test is set to 5.

get smaller. It can be assumed that this behavior will be on-going for $N > 100$ and the differences will be (nearly) disappear.

So, as an result it can be said that ordering the design vectors instead of the regression vectors may have led to slightly better results of the power functions in this thesis, but when considering also the above mentioned advantages of ordering the regression vectors, it seems acceptable making it this way.

5.5 Comparison to Other Tests for Multiple Regression

An important thing to do is comparing new methods to established ones. Of course, this is also done in this thesis. As seen in the previous sections, the best ordering method when applying the sign depth test in the context of multiple regression is an ordering according to the solution of a Shortest Hamiltonian Path problem.

Because of this, in this section this method is compared with some classical robust and non-robust tests for testing parameters in (multiple) regression problems.

For comparison with the sign depth test three established tests were chosen: The classical sign test, the F-test when using the parameters of an ordinary least-squares regression for estimation and a Wald test when using the parameters of a robust model estimated via MM-estimation for estimation. Especially, it is of interest whether and in which situations the sign depth test performs better than the F-test and the classical sign test and whether the sign depth test can reach a power level similar to the robust Wald test. The F-test is the optimal test for testing parameters in a linear regression model when having normally distributed errors under H_0. Its definition can be found in Theorem A.1 on page 213. As it can be seen there, the F-test needs an estimated parameter vector $\hat{\boldsymbol{\theta}}$. Here, this parameter vector is estimated via ordinary least-squares regression. The sign test is a non-parametric alternative whose definition can be found in Theorem A.2 on page 213. In contrast to the F-test, the sign test has no requirements regarding the error distribution, except of having errors with median zero. Furthermore, like for the sign depth test, no explicit estimator $\hat{\boldsymbol{\theta}}$ is needed because the tests are solely based on the residuals under H_0. In addition, it can be easily shown that the sign test is equivalent to the sign depth test with parameter $K = 2$. As a third method for comparison, a robust Wald test is chosen. Like the F-test, the Wald test is based on an estimator $\hat{\boldsymbol{\theta}}$. In Section 2.3 the MM-estimation for regression is described. An often used variant of this MM-estimation is implemented in the function `lmRob()` in the `R`-package `robust`. Because of this, this method will be used here and the estimated parameters will be used for the Wald test which is defined in Theorem 2.1 on page 14.

At first, the performance of these three methods and the sign depth test with ordering according to the solution of the Shortest Hamiltonian Path problem will be shown for the simplest model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ which was also analyzed in Section 5.2. Since Subsection 5.2.4 has shown that the sign depth test performs best for the distance based ordering methods when using parameter $K = 5$, in this subsection always this parameter is used for comparison. Here, it is looked at the performance of these comparison methods for different numbers of data points, different underlying data sets and different error distributions of the vector $\boldsymbol{e}$. Again, the simulation setting of Section 5.2 is used, see Section 5.1 for details. Furthermore, as before, the tested hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$.

Effect of the error distribution

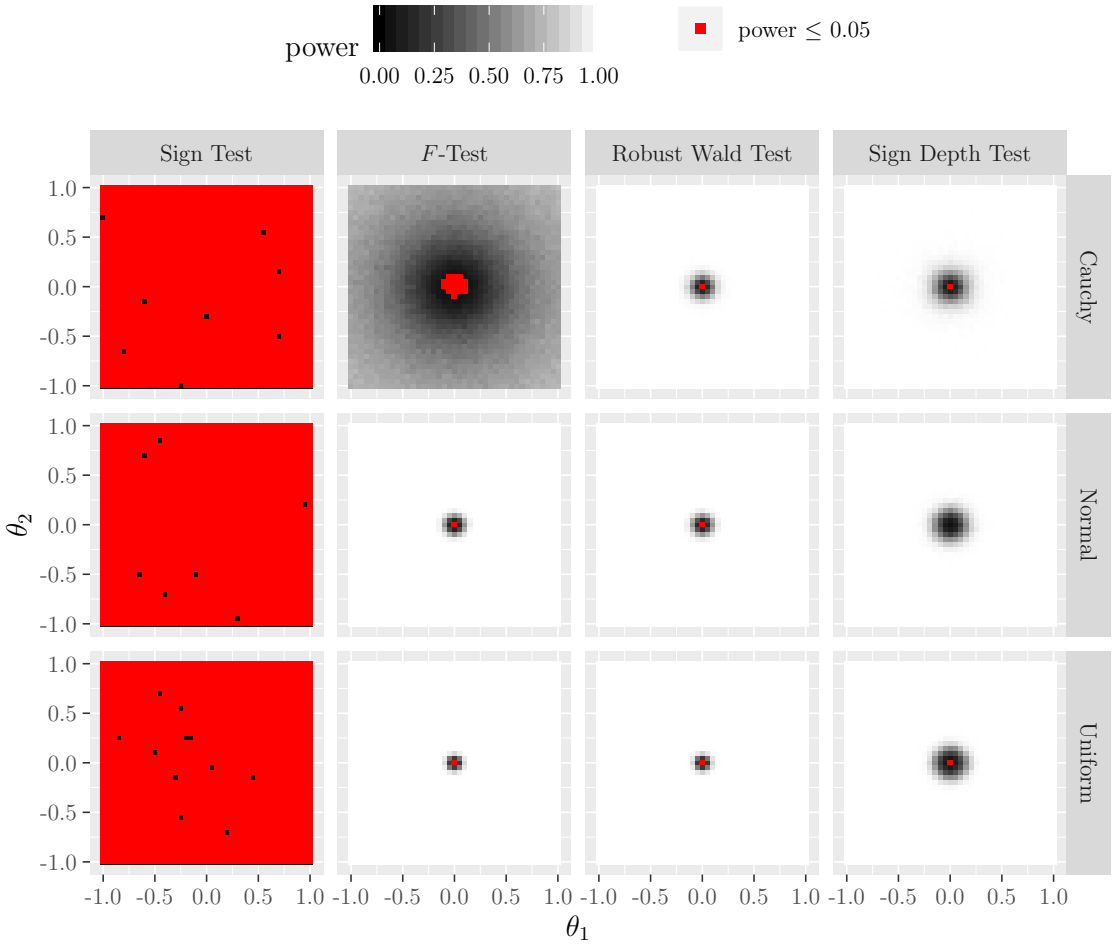


Figure 5.44: Simulated power functions of the comparison methods for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The sign depth test is conducted with an ordering according to the exact solution of the Shortest Hamiltonian Path problem and parameter $K = 5$. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

Figure 5.44 shows the results of the three comparison methods and the sign depth test when using an ordering according to the solution of the Shortest Hamiltonian Path problem for different error distributions of the vector $\boldsymbol{e}$. It can be seen that two of the three comparison methods perform satisfyingly. Of course, the F-test is best when having normally distributed errors. But also the robust Wald test performs

very well. In contrast to the F-test, this test is also a very powerful test when having Cauchy distributed errors. On the other hand, the classical sign test performs very poorly. Nearly all simulated power values are less than or equal to 0.05. But this behavior can easily be explained: In this data situation always about half of the residuals are negative and half are positive because no intercept is modeled and the regression vectors are randomly drawn from the interval $[-1, 1]^2$. And since the sign test only is counting the number of positive residuals and checks whether this number is about half of the number of data points, in this model and this specific data situation the sign test can nearly never reject the null-hypothesis. Because the sign test is equivalent to the sign depth test when using parameter $K = 2$, these results show once more that the power of the sign depth test is smaller for smaller values of K and that $K = 2$ should not be used at all, but at least $K = 3$ is necessary to obtain satisfying results. All in all, the sign depth test performs much better than the sign test and the F-test when having Cauchy distributed errors. When having normally distributed errors or uniformly distributed errors, the F-test and the robust Wald test perform slightly better than the sign depth test, but overall the sign depth test and the robust Wald test perform satisfying in all situations.

Effect of the number of data points

Next, the effect of the number of data points is analyzed. For this, as in Subsection 5.2.1 the power functions are simulated with $N = 25$ data points as well as with $N = 100$ data points. Because it is clear that the F-test is always best when having normally distributed errors, in Figure 5.45 and all following figures the tests are simulated with Cauchy distributed errors to show the case where the normality assumption is violated. But it should not be forgotten that the F-test is always the best test in the special case of normally distributed errors. This figure shows the expected results: The sign test performs badly and also the F-test has relatively low power values because of the Cauchy distributed errors, but the sign depth test and the robust Wald test perform very well. Of course, the power values in H_1 are larger when having more data points, but also for $N = 25$ both tests have satisfying results. The robust Wald test seems to be a little bit better for $N = 25$ whereas there are no big differences for $N = 100$.

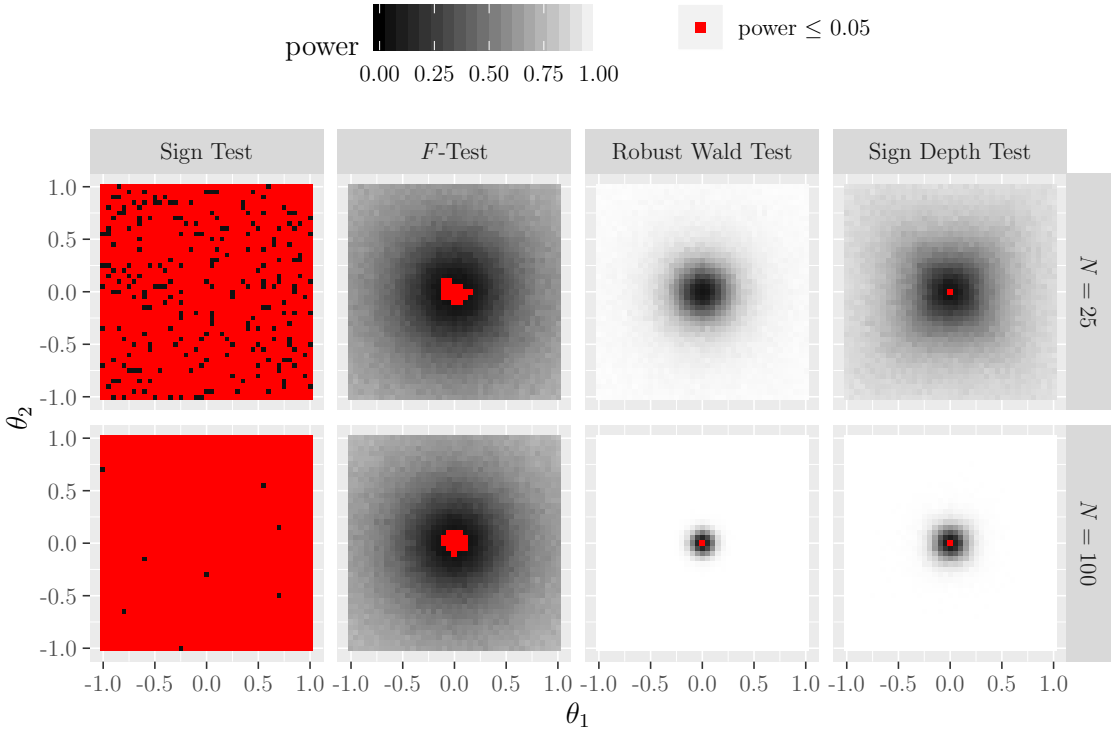


Figure 5.45: Simulated power functions of the comparison methods for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N \in \{25, 100\}$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The sign depth test is conducted with an ordering according to the exact solution of the Shortest Hamiltonian Path problem and parameter $K = 5$. The error distribution of the vector $\boldsymbol{e}$ is a Cauchy distribution. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

Effect of the data set

The next figure, Figure 5.46, shows the simulated power functions of the comparison methods and the sign depth test for different underlying data sets. The data sets are the same as in Subsection 5.2.2: The "Random" data set, the "Grid" data set and the "Spiral" data set. Again, it can be seen that the sign test performs poorly and also the F-test is quite bad when having Cauchy distributed errors. On the other hand, the sign depth test and the robust Wald test perform very well. The sign depth test is slightly worse than the Wald test on the "Spiral" data set, but in the other situations there are no big differences visible. The problems of the sign

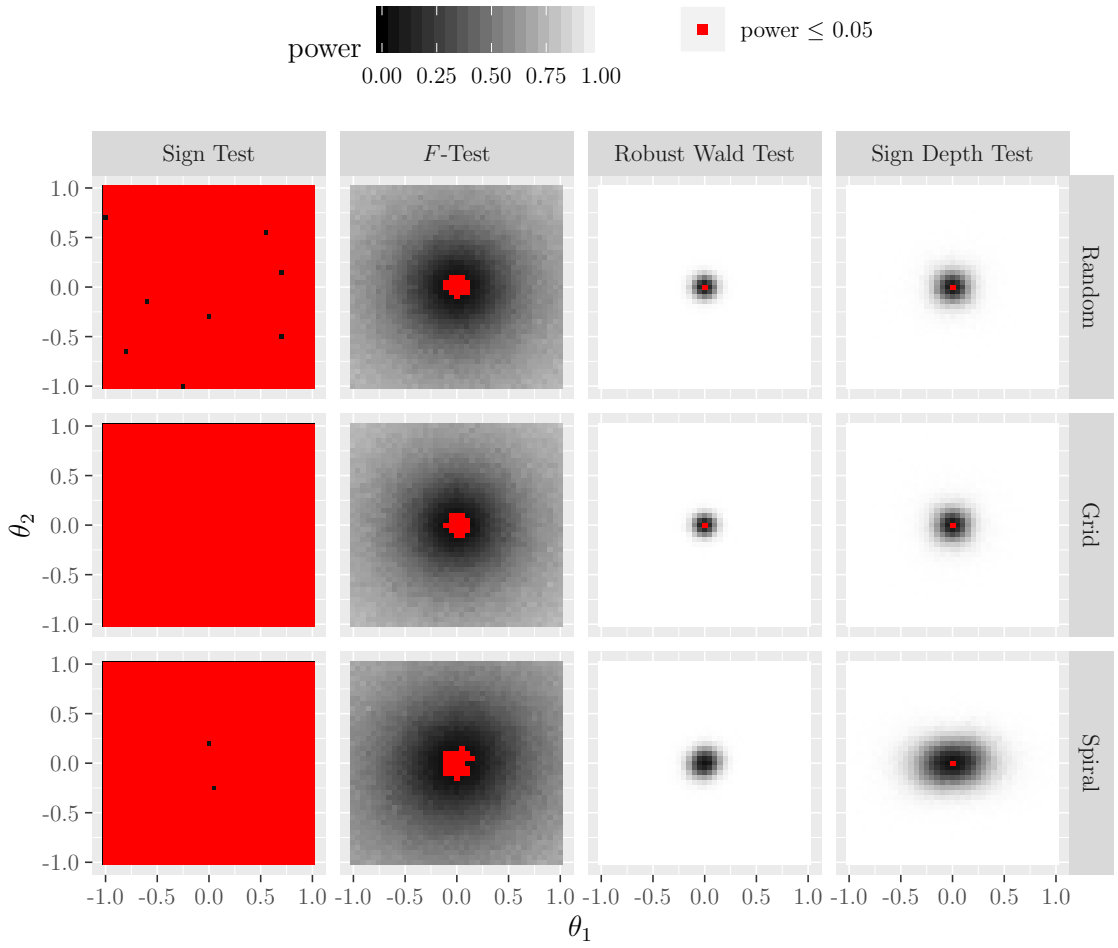


Figure 5.46: Simulated power functions of the comparison methods for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. All data sets consist of $N = 100$ data points. The sign depth test is conducted with an ordering according to the exact solution of the Shortest Hamiltonian Path problem and parameter $K = 5$. The error distribution of the vector $\boldsymbol{e}$ is a Cauchy distribution. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

depth test on the "Spiral" data were analyzed in Subsection 5.2.2. Obviously the robust Wald test and also the other tests are not affected by these problems because they do not depend on an ordering of the data set.

So, as an interim conclusion it can be said that the sign depth test is competitive with the established tests. It is much better than the classical sign test and since it is not affected by the distribution of the errors also often better than the F-test.

Of course, when having normally distributed errors, the F-test is always the best test. In all considered situations the sign depth test performs about as good as the robust Wald test. But is should not be forgotten that the Wald test and the F-test depend on estimated parameter vectors $\hat{\boldsymbol{\theta}}$ which is not the case for the sign test and the sign depth test. During the fitting process numerical instability can occur or the optimization algorithms needed for the MM-regression may not converge. Especially, the latter case happened several times during the simulations. So, the dependance of these tests of a fitted regression model is a large disadvantage in contrast to the sign test and the sign depth test which only need the residuals under H_0.

In the following, the sign depth test is also compared to the established tests for some of the other models analyzed in Section 5.3. It is of interest how the established tests perform on models with intercept, with interaction, with quadratic regressors and in high dimensions in contrast to the sign depth test.

Model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$

When adding an intercept to the model, i.e. $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$, the simulated power functions are visualized in Figure 5.47. It can be seen that all four tests have greater power values when θ_0 is not zero which is the desired behavior since the null-hypothesis is $H_0 : \boldsymbol{\theta} = \mathbf{0}$. Also, it can be seen that the sign test and the F-test perform rather badly, whereas the sign depth test and the robust Wald test perform very well. In the case of the F-test, the rather bad performance is of course caused by the Cauchy distributed errors which were used for simulating the power functions. Because of this the power is rather low and there is a quite large area of values around H_0 which has power values of α or less. The performance of the sign test would also be bad for normally distributed errors, but this test behaves differently than the F-test. While the F-test has low power values in all directions of the power function, the sign test is able to detect at least changes in the intercept. Its power function has areas with larger power values when θ_0 is not zero. On the other hand, there seem to be no big differences between the sign depth test and the robust Wald test, both methods perform very satisfyingly. So, also for linear models with intercept the sign depth test is competitive with the established methods.

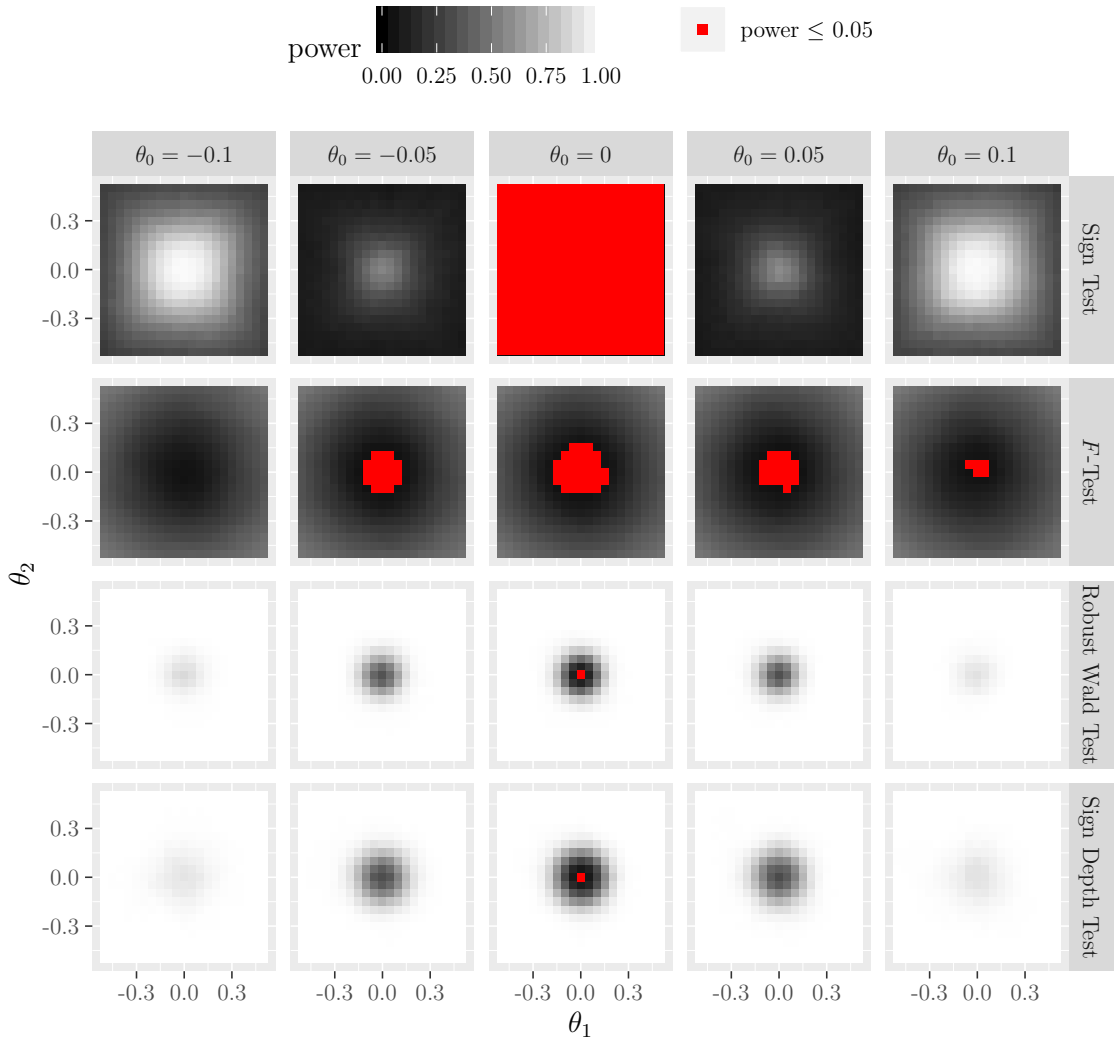


Figure 5.47: Simulated power functions of the comparison methods for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a Cauchy distribution. The sign depth test is conducted with an ordering according to the exact solution of the Shortest Hamiltonian Path problem and parameter $K = 5$. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

Model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$

Next, it is looked at the model with an interaction and without intercept, i.e. $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The simulated power functions of the sign depth

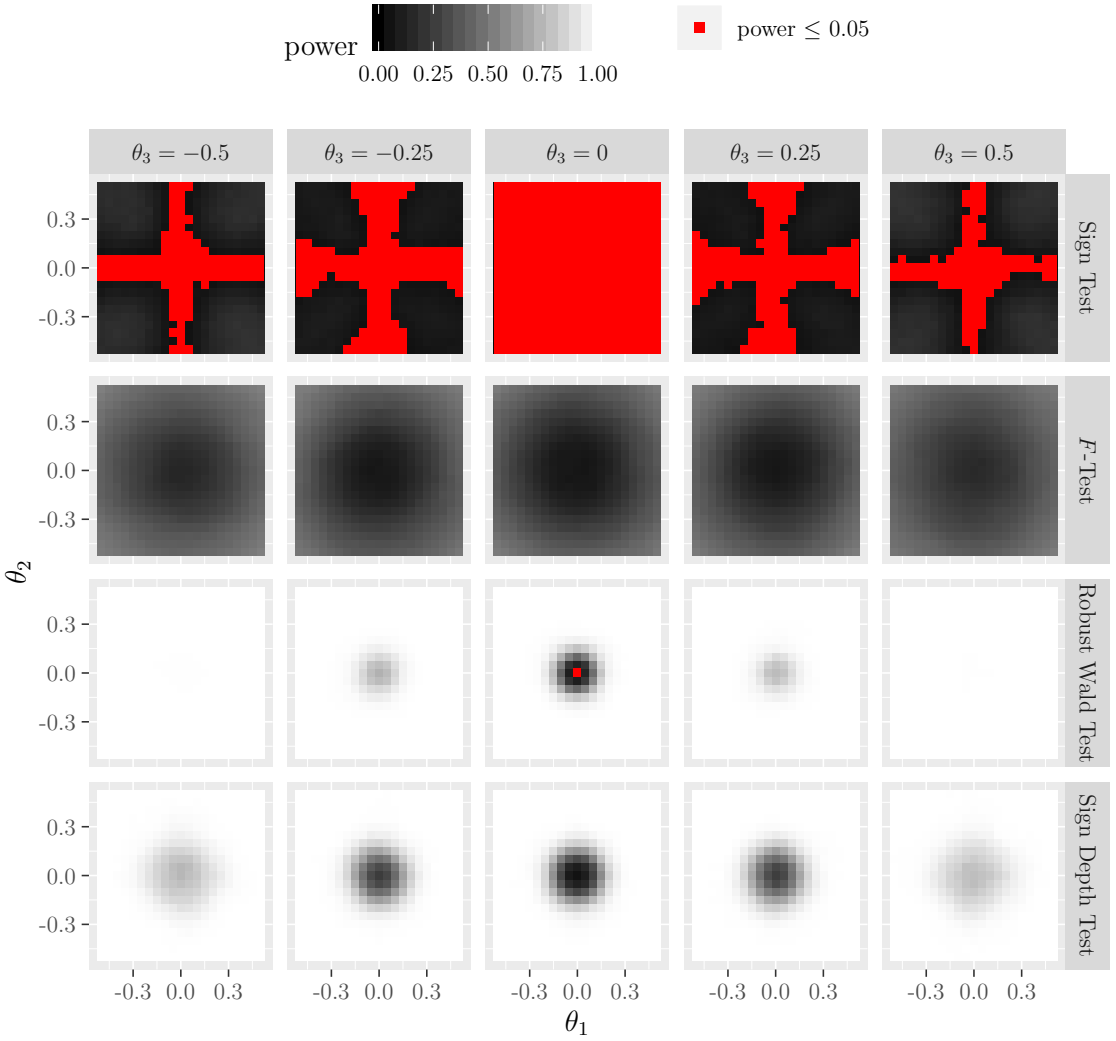


Figure 5.48: Simulated power functions of the comparison methods for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a Cauchy distribution. The sign depth test is conducted with an ordering according to the exact solution of the Shortest Hamiltonian Path problem and parameter $K = 5$. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

test with an ordering according to the exact solution of the Shortest Hamiltonian Path problem and the three comparison methods are shown in Figure 5.48. It can be seen that again the sign test and the F-test perform rather badly, whereas the sign depth test and the robust Wald test perform really well. Here, the robust Wald test is better than the sign depth test, although the performance of the sign depth test is also satisfying. As mentioned in Subsection 5.3.2, the sign depth test needs relatively large deviations from H_0 in the direction of the parameter of the interaction for getting great power values. This seems to be different for the robust Wald test. But although the sign depth test obviously is not the best test in this situation, its performance is quite good, so that one gets satisfying results when applying it in such situations.

Model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$

Next, it is looked at the model with quadratic terms from Subsection 5.3.3, i.e. $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The simulated power functions of the three comparison methods can be found in Figures 5.49, 5.50 and 5.51. The simulated power function of the sign depth test when using the exact solution of the Shortest Hamiltonian Path problem for ordering can be found in Subsection 5.3.3 in Figure 5.36 on page 162. In contrast to the three comparison methods in this section, the sign depth test in Subsection 5.3.3 was simulated with normally distributed errors, whereas here the power functions are simulated with Cauchy distributed errors, but as mentioned before the error distribution has no effect on the performance of the sign depth test as long as the median of the errors is zero. So, there is no problem in comparing the methods with each other.

In Figure 5.49 it can be seen that the sign test has clearly lower power values when $|\theta_3 + \theta_4|$ is small, like it is also the case for the sign depth test. But in contrast to the sign depth test the sign test has clearly lower power and the power function is not only at H_0 less than or equal to α. Overall, the results of the sign test are not satisfying because of the large areas with low power values. The same holds for the F-test shown in Figure 5.50. Here, no simulated power values are less than or equal to α, but all power values are rather small, also for large values of $|\theta_3 + \theta_4|$. The simulated power function of the robust Wald test in Figure 5.51 looks much better. Here, the area with low power is very small, also when $|\theta_3 + \theta_4|$ is small or even zero. So, here the robust Wald test is clearly the best test, also better than the sign depth

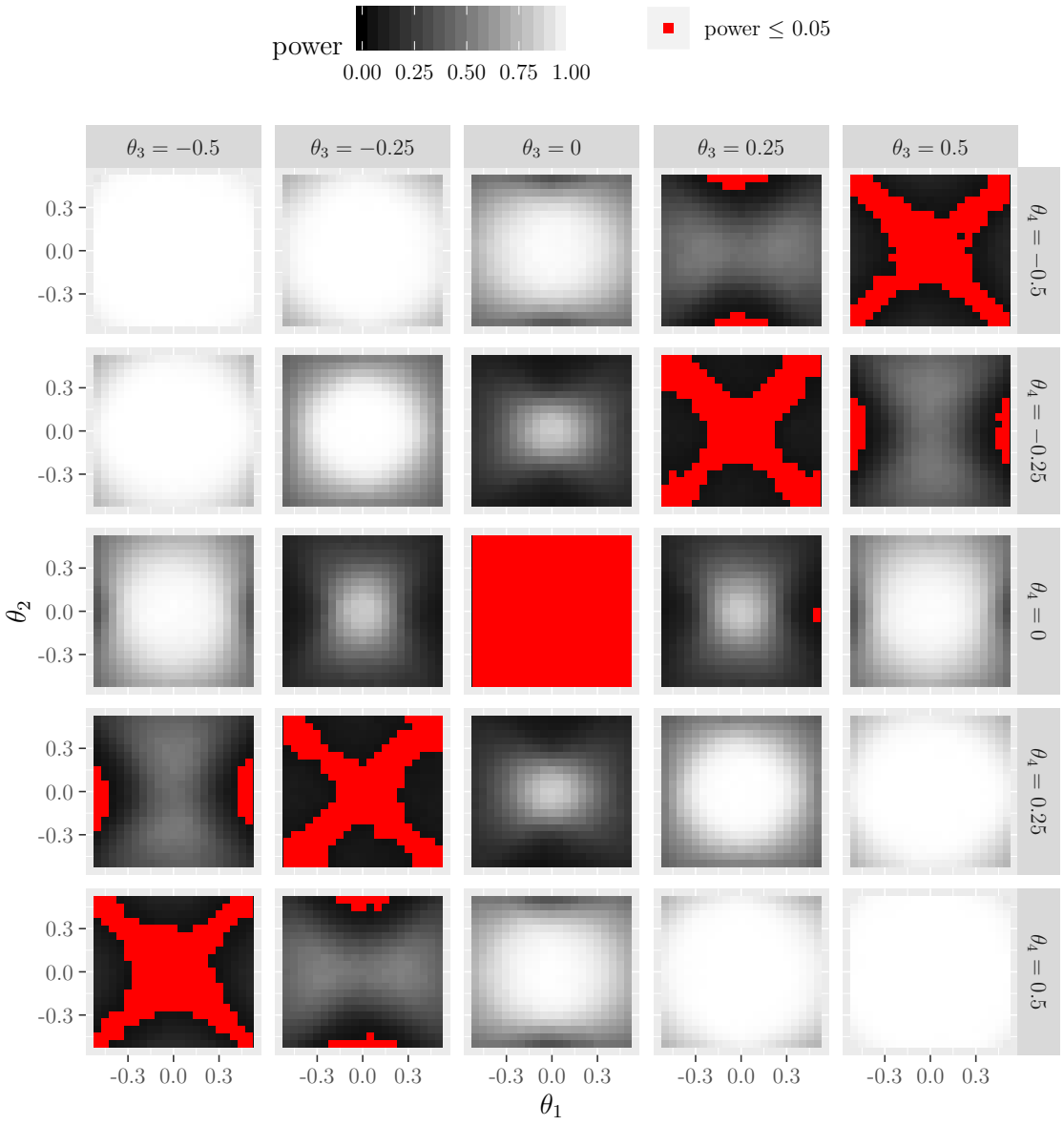


Figure 5.49: Simulated power functions of the sign test for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a Cauchy distribution. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

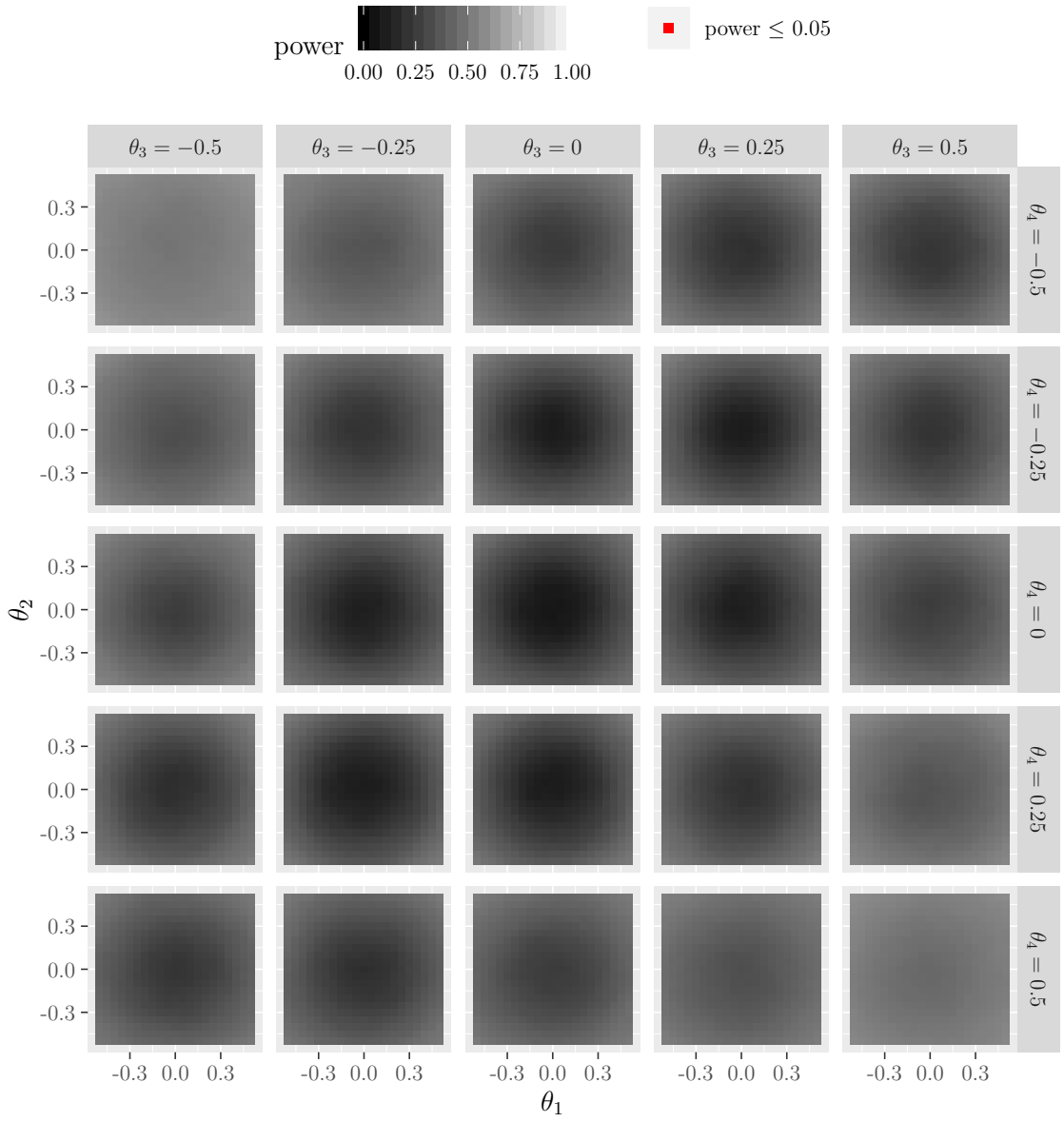


Figure 5.50: Simulated power functions of the F-test for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a Cauchy distribution. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

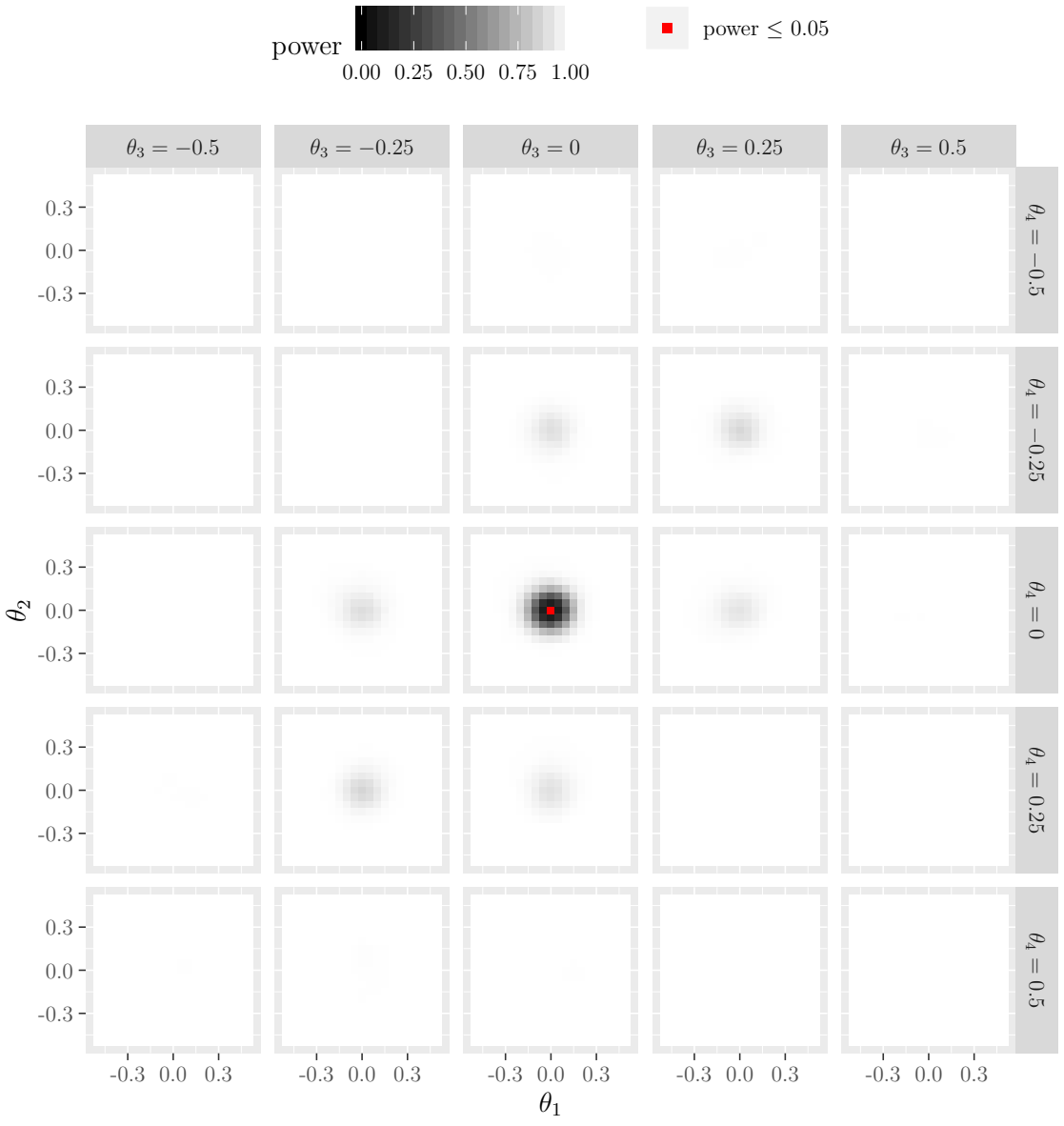


Figure 5.51: Simulated power functions of the robust Wald test for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ and the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$. The underlying data set consists of $N = 100$ random values in the range $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$. The error distribution of the vector $\boldsymbol{e}$ is a Cauchy distribution. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

test. But both tests, the sign depth test and the robust Wald test are satisfying at the end.

Model $\boldsymbol{y} = \sum_{d=1}^{D} \theta_d \boldsymbol{x}_{\bullet d} + \boldsymbol{e}$

At last, it is looked at the high-dimensional models of Subsection 5.3.4. Here, the 10-, 20-, 40- and 80-dimensional linear models are tested with all three comparison methods as well as the sign depth test with $K = 5$ and $K = 21$. As in Subsection 5.3.4 explained for $K = 21$ only an asymptotic implementation of the sign depth is available. In addition, these power functions are here simulated only for the same two aspects as in Subsection 5.3.4: Firstly, when varying only θ_1 and all other components of $\boldsymbol{\theta}$ are zero, and secondly the power on the diagonal line through the D-dimensional hyperspace is considered. Furthermore, the results are simulated for these models with normally distributed errors as well as with Cauchy distributed errors to show the effect of the error distribution to the tests when plotting the power functions two-dimensional and not three-dimensional as always before.

Figures 5.52 and 5.53 show the extracts of the simulated power functions for both considered aspects. These figures show several interesting things. Firstly, the sign depth test performs better for $K = 21$ than for $K = 5$ which is no surprise since in Subsection 5.3.4 it was pointed out that K should have at least the same magnitude as D to reach good results of the sign depth test. Secondly, the robust Wald test performs satisfyingly for $D = 10$ and $D = 20$, but for $D = 40$ the level is not maintained, i.e. the power values are much larger than $\alpha = 0.05$ at H_0 and for $D = 80$ the robust Wald test cannot be carried out at all. Indeed, it is not the Wald test itself which causes the problems, but the underlying MM-estimation. The `R`-function `lmRob()` always throws an error when trying to calculate the estimator for $D = 80$ dimensions and $N = 100$ data points which says that internally a matrix cannot be inverted because of numerical singularity. In contrast to this, the sign depth tests remains computable for such large number of dimensions, although the value of K has to be smaller than D and so the power of the sign depth test is not very good. Of course, the F-test performs best when having normally distributed errors, but also for Cauchy distributed errors its performance is best for very large D in the second aspect (although near the null-hypothesis the sign depth test is better). For the first aspect shown in Figure 5.52, the sign depth test is better than the F-test. The classical sign test performs poorly independent from the number

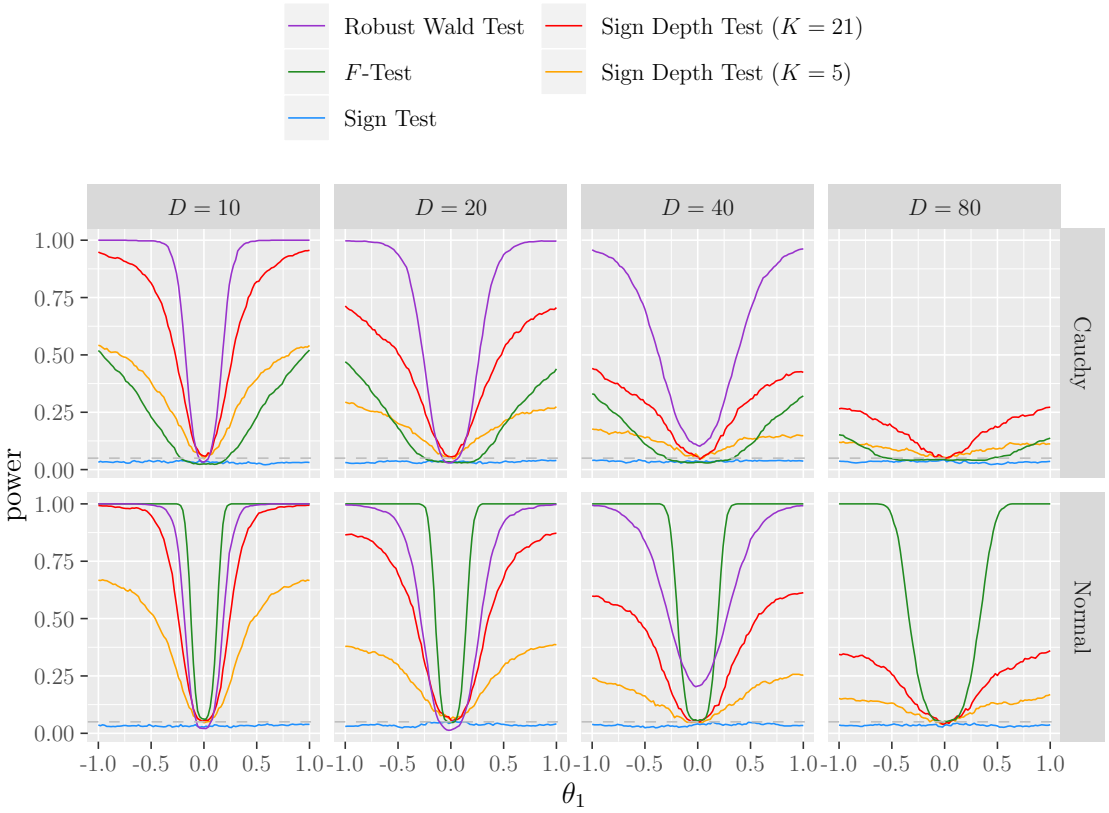


Figure 5.52: Extracts of the simulated power functions of the comparison methods for the model $\boldsymbol{y} = \sum_{d=1}^{D} \theta_d \boldsymbol{x}_{\bullet d} + \boldsymbol{e}$. Here, the power functions are only shown for $\theta_1 \in \{-1, -0.98, -0.96, ..., 0.96, 0.98, 1\}$, all other values of $\boldsymbol{\theta}$ are zero. The sign depth test is conducted with an ordering according to the exact solution of the Shortest Hamiltonian Path problem. The gray dashed line shows the level of the test $\alpha = 0.05$.

of dimensions. Its power is always about 0.05. Furthermore, the case $D = 10$ and $K = 21$ shows that the sign depth test can keep up with the robust Wald test and the F-test (for normally distributed errors) when K is sufficient large in comparison to D.

So, overall it can be said that in the case of high-dimensional models the sign depth test may not be optimal when D is larger than K, but also the robust Wald test has problems with the high-dimensionality. While the sign depth test can be computed for high dimensions and relatively few data points, this is not possible for the robust Wald test. In addition, its power values are larger than α at H_0 for $D = 40$ which

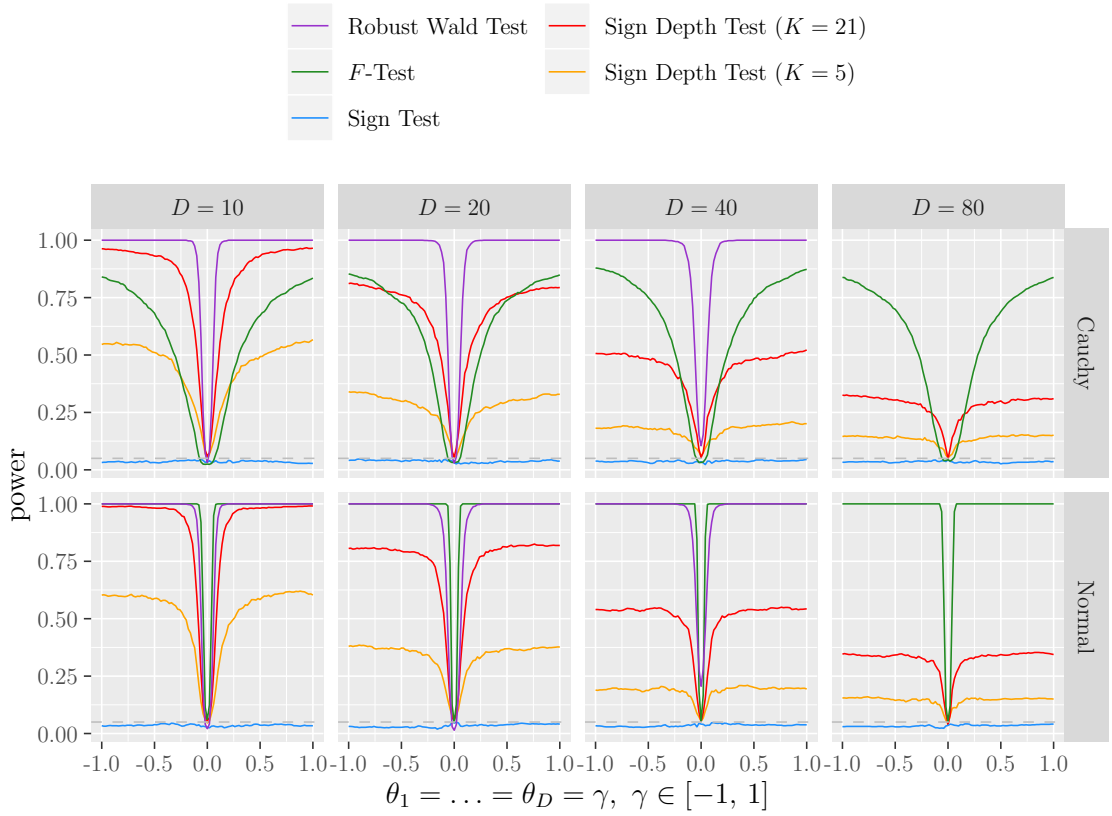


Figure 5.53: Extracts of the simulated power functions of the comparison methods for the model $\boldsymbol{y} = \sum_{d=1}^{D} \theta_d \boldsymbol{x}_{\bullet d} + \boldsymbol{e}$. Here, the power functions are only shown for $\theta_1 = \ldots = \theta_D = \gamma$ with $\gamma \in \{-1, -0.98, -0.96, ..., 0.96, 0.98, 1\}$. The sign depth test is conducted with an ordering according to the exact solution of the Shortest Hamiltonian Path problem. The gray dashed line shows the level of the test $\alpha = 0.05$.

is not the case for any other of the considered tests. Again, of course the F-test is best for normally distributed errors. Also, this test has no problems with the high dimensions. The sign test performs badly for all numbers of dimensions.

As a conclusion, it can be said that the sign depth test is always better than the classical sign test in the situations considered here. This is no surprise since the sign test is equivalent to the sign depth test when choosing $K = 2$ and this chapter has shown that in general greater values of K are better than smaller ones. The sign depth test is also better than the F-test when the assumption of normally distributed errors is violated. For normally distributed errors, of course the F-test is

the best test one can choose. But in the context of possibly non-normally distributed errors, the sign depth test performs clearly better than the F-test. The robust Wald test via MM-estimation has performed similar to the sign depth test and in some situations it was better than the sign depth test, especially when having interactions or polynomial regressors in the model. But the Wald test has a major disadvantages the sign depth test has not: The Wald test needs the assumption of asymptotically normally distributed errors under H_0. This is achieved here by using MM-estimation, but when this assumption is violated or the optimization(s) needed for fitting the MM-estimator do not converge, this test can fail. In addition, for high dimensions the MM-estimation needed for the robust Wald test cannot be computed anymore. The sign depth test can be used more universally: Its only assumption is that the median of the errors is zero under H_0. Furthermore, the sign depth test is easy to understand and to implement. It has shown no numerical problems whereas for the Wald test a parameter $\hat{\boldsymbol{\theta}}$ and an associated covariance matrix have to be estimated and in case of the covariance matrix also inverted and the inversion of matrices can be numerically challenging.

So, although the Wald test has proven to be a very good test in the considered situations, also the sign depth test performs very well. For having an easier to understand and implement test with less computational and numerical challenges which can also be used more universally than a Wald test, the sign depth test has slightly less power in some situations and needs some more computational runtime. All in all, both tests can be used in the context of multiple regression, each with individual advantages and disadvantages.

5.6 Sign Depth Test for Model Checks

So far, the sign depth test and the comparison methods were used only for testing parameters in a known model. In real life, often the regressors of a linear model are not known and the correct model class has to be found first before a testing on parameters is possible at all. For this, techniques for model and variable selection are known. Most methods are based on testing whether a fitted model describes the data sufficiently or not. For this, next to some established tests like the Wald test, also the sign depth test can be used.

In this section, the performance of the sign depth test in contrast to the robust Wald test will be shown and analyzed in the case of testing for model checks. For this, data will be simulated from two different models: $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ and $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$ with $N = 100$ random regression vectors in the interval $[-1, 1]$ for $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ each. But the tests will be testing only for a significant influence of θ_1 and θ_2, i.e. $H_0 : \boldsymbol{\theta} = \boldsymbol{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \boldsymbol{0}$ for $\boldsymbol{\theta} = (\theta_1, \theta_2)^\top$. So, a possible influence of θ_3 (and θ_4) is "forgotten" here. This will be done in dependance of the value(s) of θ_3 (and θ_4) and the power of the sign depth test and the robust Wald test will be compared. i.e. it is looked how large the "forgotten" parameter(s) have to be to detect this with the respective tests. For the sign depth test an ordering according to the exact solution of the Shortest Hamiltonian Path problem is used and the value K of the sign depth is set to 5.

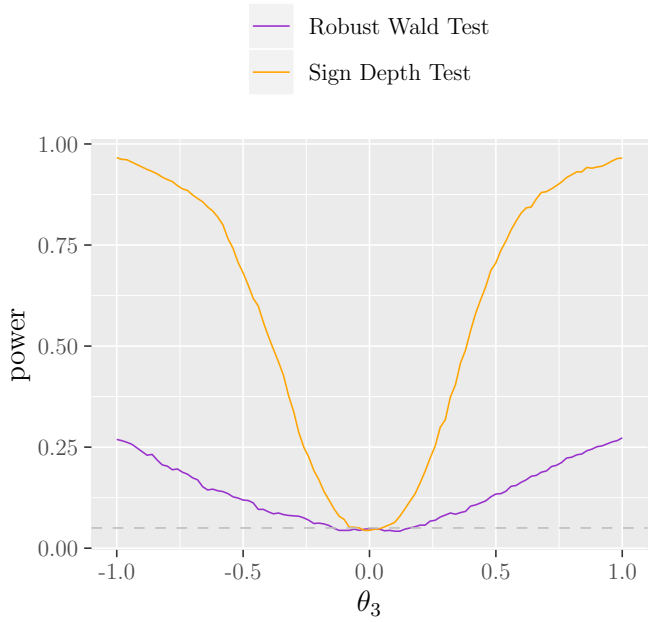


Figure 5.54: Power of the sign depth test and the robust Wald test when testing on a significant influence of θ_1 and θ_2 in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$, but the data is generated from the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$, where $\boldsymbol{e}$ is normally distributed and $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ consist of $N = 100$ random values in the range $[-1, 1]$ each. For the sign depth test an ordering according to the exact solution of the Shortest Hamiltonian Path problem is chosen and K is set to 5. The dashed gray line denotes the level of the test, i.e. $\alpha = 0.05$.

Figure 5.54 shows the power of the sign depth test and the robust Wald test when the data is generated from the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1} \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$ and it is tested only on a significant influence of θ_1 and θ_2 in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. It can be seen that the sign depth test in this situation performs much better than the robust Wald test. Although both tests have power values which are greater than or equal to $\alpha = 0.05$ for most values of θ_3 except $\theta_3 = 0$, the power values for the sign depth test are always much greater than the power values of the robust Wald test. This phenomenon can be easily explained by the fact that the sign depth test does not need to compute an estimator for the model, but only needs the parameter $\boldsymbol{\theta}_0$ from the null-hypothesis for testing the model. In contrast, the robust Wald test needs to compute an estimator first, here obtained by an MM-regression, which leads to less power of the test because the estimation process leads to models which are considered as fitting for the Wald test as long as the parameter θ_3 differs not too much from zero. So, in this case of testing for model checks and "forgetting" an interaction in the model the sign depth test is clearly preferable to the robust Wald test.

The same tests are then also carried out for a second model check. Again, it is tested on a significant influence of θ_1 and θ_2 in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$, but this time the data is generated from the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$, so here two parameters θ_3 and θ_4 were "forgotten". The power of these tests in dependance of the values of θ_3 and θ_4 can be found in Figure 5.55. As before, it can be seen that the performance of the sign depth test is much better than the performance of the robust Wald test. In general, the power values are much greater and also the area with power values of α or less is much smaller for the sign depth test. As seen in Subsection 5.3.3, the power of the sign depth test in this model is lower on a line where θ_3 and θ_4 cancel each other out. But apart from that line the power of the sign depth test is quite large, whereas the power of the robust Wald test is nearly everywhere quite low. So, as before, this has shown that testing for model checks can be better carried out with the sign depth test than with the established robust Wald test.

This section has shown that the sign depth test has very good performance when making model checks. In contrast to all previous sections where the robust Wald test was on the same performance level like the sign depth test or even better, here we have a clear advantage of the sign depth test.

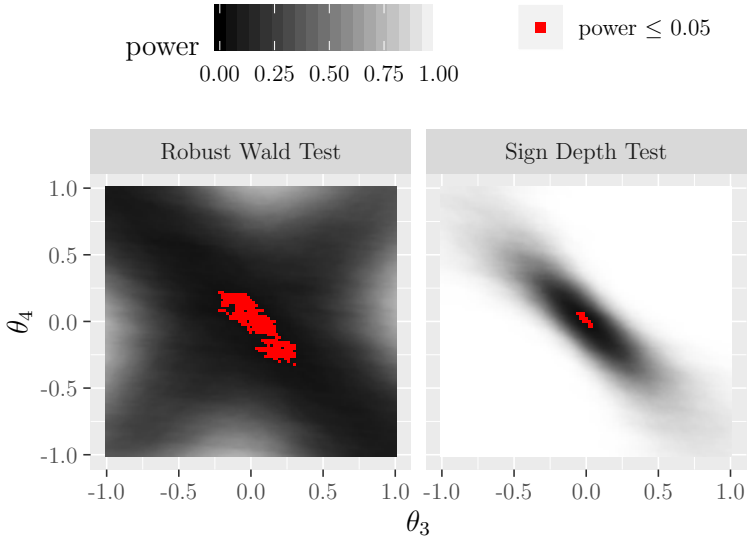


Figure 5.55: Power of the sign depth test and the robust Wald test when testing on a significant influence of θ_1 and θ_2 in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$, but the data is generated from the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \theta_3 \boldsymbol{x}_{\bullet 1}^2 + \theta_4 \boldsymbol{x}_{\bullet 2}^2 + \boldsymbol{e}$, where $\boldsymbol{e}$ is normally distributed and $\boldsymbol{x}_{\bullet 1}$ and $\boldsymbol{x}_{\bullet 2}$ consist of $N = 100$ random values in the range $[-1, 1]$ each. For the sign depth test an ordering according to the exact solution of the Shortest Hamiltonian Path problem is chosen and K is set to 5. The red squares denote a simulated power of 0.05 $(= \alpha)$ or less.

5.7 Application of the Sign Depth Test on Data from a Bridge Monitoring

Finally, in this section, an application of the sign depth test is shown. So far, the sign depth test was analyzed on simulated data with testing hypotheses which had only little practical usage. Now, the sign depth test will be used in a setting which is more relevant: A multiple regression model is fit to some data and the goodness of the fit should be found out, i.e. the hypothesis $H_0 : \boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ vs. $H_1 : \boldsymbol{\theta} \neq \hat{\boldsymbol{\theta}}$ is considered where $\hat{\boldsymbol{\theta}}$ describes the estimator of the fitted model. As seen in the previous section for such a testing for model checks the sign depth test is a very good choice. For this, at first the application is described and afterwards two different models are fit and the fits are tested on their goodness.

As part of the *SFB 823: Statistical modelling of nonlinear dynamic processes* the project *B5: Statistical Methods for Damage Processes under Cyclic Load* deals with fatigue damage processes in concrete structures and especially with data obtained

from a bridge monitoring. The monitored highway bridge in Bochum has showed significant deficits of fatigue strength and several cracks in the concrete with widths up to 0.5 mm in a routine bridge inspection in 2016. Because of this, a monitoring of the cracks was installed in June 2016 and has run until the demolition of the bridge in October 2017 (first roadway) and November 2018 (second roadway), respectively. Overall, 16 cracks were monitored to detect a potential increase of crack widths as early as possible since an increase of crack widths may indicate a damage due to fatigue in the cracked areas. For each of the 16 monitored cracks, every two seconds the width of the crack was measured. But the crack widths are not constant over time or slowly increasing but are affected by several things: Firstly, the temperature. The warmer the air (and so the bridge) is, the larger is the crack. Secondly, the traffic. When heavy vehicles, like the tram or trucks, pass the bridge the respective crack widths change rapidly. And thirdly, some anomalous sequences with large variance

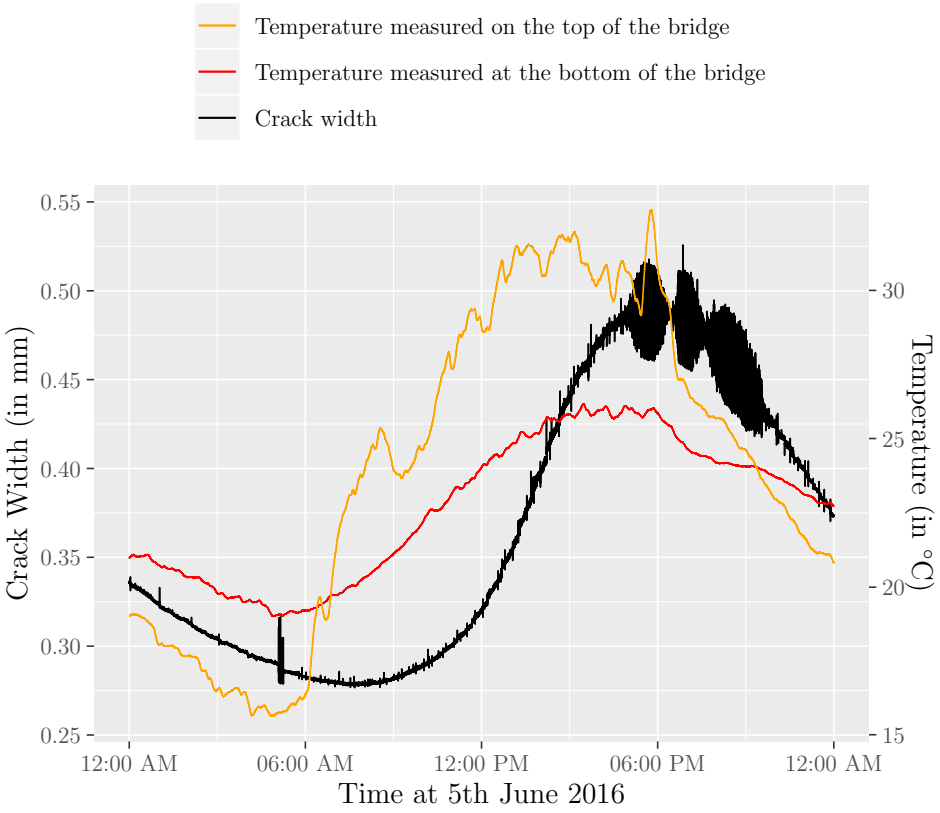


Figure 5.56: Extract of the crack width measurements for the largest crack (called WWN2) at 5th June 2016. In addition, two temperature measurements are shown, one measured on the top of the bridge and one measures at the bottom of the bridge.

in the measured crack widths whose reason is unknown, but probably were caused by technical issues of the measuring devices. All three influence factors can be nicely seen in Figure 5.56, where the crack widths of the largest crack (called *WWN2*) are shown at 5th June 2016. Additionally, two temperatures are displayed. One temperature is measured at the top of the bridge, which means that this measurement is highly affect by the weather, especially by the sun. The second temperature is measured at the bottom of the bridge, which means that this measuring device is always in the shade and so is not as much affected by the weather and the sun as the first temperature measurement. In Figure 5.56 the dependance of the crack width from the temperature can be nicely seen, although the crack width reacts with a delay on changes in the temperature. Also, a longer time period with these anomalous sequences in the crack width measurement can be seen in the evening. And the traffic causes many small peaks in the measurement. The two large peaks in the early morning were caused by a heavy test load which was conducted at this day by using a mobile crane of 48 tons.

Further description of the monitoring data and several approaches how to eliminate the anomalous sequences from the data can be found in Abbas et al. (2019). The complete data of the monitoring can be found on the website of the SFB 823[1].

As mentioned before, the major aim of this bridge monitoring was to detect significant increases in the crack widths. But such an increase would be clearly smaller than the normal range the crack width has over day and also smaller than the peak a single tram or truck causes. Because of this, such a permanent change in the width of the crack is very difficult to detect and could not be done timely to the measurements, but has to be done now after completion of the monitoring. In the last years since the monitoring started in 2016, many approaches were taken to model the crack widths. For this, the temperature measurements are of high importance since they caused the large changes in the widths of the cracks. For the other two influencing factors, the traffic and the anomalous sequences, a simple smoothing of the data is sufficient to eliminate these effects.

For finding a good model for the crack width in dependance of the temperature the sign depth test can be used. As mentioned above, it can decide whether a model fits the data or not. Although we have a multiple regression problem here, the monitoring data has an inherent order since it is a time-series, so the sign depth test could be

[1] `https://www.statistik.tu-dortmund.de/1938.html`

applied without ordering the regression vectors. But Section 5.2 has shown that using the sign depth test with an ordering according to only one component of the regression vectors (here: the time) may lead to worse results of the test than when using all components of the regression vectors for an ordering of the data points. So, the research question for this section is: Does an ordering according to the exact solution of the Shortest Hamiltonian Path problem lead to different results of the sign depth test than using the inherent order of the data in the context of finding a good description of the crack width of the bridge monitoring? If yes, how does this effect look like?

To answer this question, some preprocessing of the data is necessary. It was decided to use the largest crack WWN2 here for the analysis and to neglect the temperature measurement on the top of the bridge because it is too much affected by the weather. Furthermore, the crack values and the value of the temperature measurements at the bottom of the bridge were smoothed by taking the mean of the values for intervals of 15 minutes each. So, for every day $24 \cdot 4 = 96$ observations of the crack width, the temperature and the time are available. Here, the values of exactly one year, from 1st June 2016 to 31st May 2017, are used. Furthermore, not one big model shall be fitted, but several smaller models to have several values for the goodness of fit which can be compared. Because of this, it was decided to fit a model for every time of the day separately, so that at the end 96 models are fitted, each with up to 365 data points. As regressors of the models the date (transformed to numeric values), the crack width 24 hours ago, the temperature and an intercept were chosen. Taking also the crack width 24 hours ago in consideration seems meaningful since we are here in the context of time-series where the actual crack width may depend on the previous value of the crack width. For the temperature not the values of the respective point in time were used but the temperature four hours ago, since the bridge reacts with a delay of several hours to changes in the temperature and a short analysis has shown that four hours seem to be a good estimator for the delay. In addition, the mean of the temperature at the bottom of the bridge of the last week was added to the model. This regressor shall describe the level of the temperatures and so the season of the year. So, overall the models look like

$$\begin{aligned}\text{Crack width } &= \theta_0 + \theta_1 \cdot \text{ Crack width 24 hours ago } + \theta_2 \cdot \text{ Date } + \\ &\quad \theta_3 \cdot \text{ Temperature four hours ago } + \\ &\quad \theta_4 \cdot \text{ Mean of temperatures of the last seven days } + \boldsymbol{e}.\end{aligned}$$

Since the crack widths and the temperatures have different scales, it was decided to transform the values of all regressors and the crack width by dividing all values through the standard deviation of the respective regressor to avoid numerical instability in the fitting process. The fitting process itself is done with two different methods: An ordinary least-squares-regression and an MM-regression made by the function `lmRob()` from the `R`-package `robust`. The latter one was chosen because although the data was smoothed there are some outliers in the data and especially the crack widths have different variances in the different seasons of the year. The residuals of the fitted model are then tested with the sign depth test for the hypothesis H_0 : the model fits the data vs. H_1 : the model fits not the data. Firstly, the residuals are ordered according to their inherent order, the date. And secondly, the residuals are ordered according to the order obtained by the Shortest Hamiltonian Path when using all regressors (the intercept, the crack width 24 hours ago, the date, the temperature four hours ago and the mean of temperatures of the last seven days) for ordering.

Figure 5.57 shows the p-values of the sign depth tests. Every point denotes a pair of p-values for a model with observations from a specific time of every day. So, overall for each model type, 96 pairs of p-values are available. The first thing that can be noticed is that the robust MM-regression fits the data much better than the ordinary least-squares regression. For the ordinary least-squares regression, all p-values are very small and clearly smaller than 0.05 which means that the models are regarded as non-fitting models when testing to the level $\alpha = 0.05$ independently from the ordering method. On the other hand, the p-values of the MM-regression are much larger and many of them are greater than 0.05 so that many models are regarded as fitting models. This shows that in general an ordinary least-squares-regression is less suitable for fitting the monitoring data than robust approaches. Next, it is remarkable that most of the points are located above the bisecting line which means that the fitting is regarded as better when the regression vectors are ordered according to the solution of the Shortest Hamiltonian Path problem than when ordered according to their inherent order. This is a very interesting result: It shows that the ordering has an effect on the result of the sign depth test at all when applying to real data. Additionally, it can be seen in Figure 5.57 that some p-values are smaller than 0.05 when ordering according to the date, but greater than 0.05 when ordering according to the Shortest Hamiltonian Path. However, the reverse case occurs only once. Since it has been shown in Section 5.2 that an ordering according to the solution of the Shortest Hamiltonian Path problem leads to larger power of the sign depth test, using such an ordering although when having an inherent order in the data seems

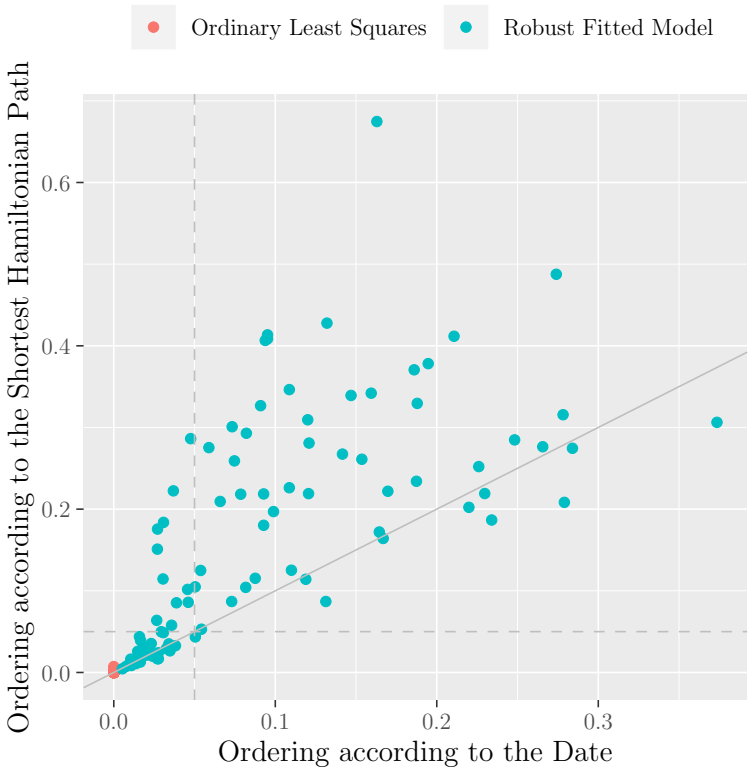


Figure 5.57: p-values of the sign depth test with $K = 3$ for fitted ordinary least-squares models as well as for robust fitted models via MM-regression. Every point describes a pair of p-values for a fitted model at a specific time of the day. The solid gray line describes the bisecting line which denotes which p-values are smaller and which larger when using the Shortest Hamiltonian Path for ordering instead of the inherent order of the data. The dashed gray lines denote the level $\alpha = 0.05$ of the sign depth test.

meaningful. Of course, this example cannot be generalized to all data situations and in every application it has to be thought again about sensible ways of ordering the regression vectors. But this thesis has shown that using the sign depth test in combination with an ordering according to the exact solution of the Shortest Hamiltonian Path problem is a good idea.

For better comprehension of the data situation, four of the overall 96 fitted models are shown as examples in the appendix in Figure C.3 on page 230. In addition, here in Figure 5.58 the regression coefficients of the 96 robust fitted models are shown. It can be seen that all coefficients are relatively constant over time, so that all fitted models are similar which is a nice result. Furthermore, it is very interesting that the coefficient of the date is always positive because this means that all models predict

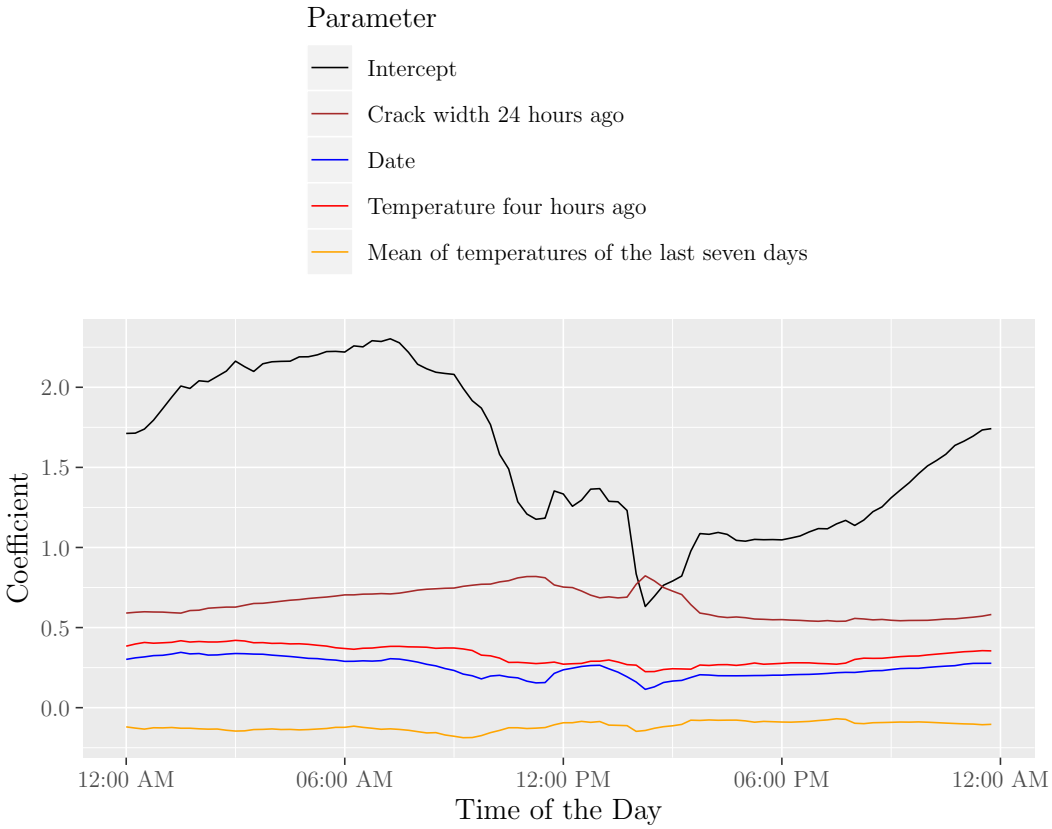


Figure 5.58: Coefficients of the 96 fitted models via MM-regression.

the crack width to grow over time. This is an indicator for the progressive damage of the (in the meantime demolished) bridge.

Chapter 6

Conclusion and Outlook

In this thesis much understanding and many results were achieved about the sign depth test in the context of multiple linear regression. This chapter summarizes the results of all chapter once again and gives an outlook about open problems and future questions in this field.

6.1 Conclusion

The aim of this thesis was to make the sign depth test applicable for multiple regression. The sign depth test describes a statistical test for testing on parameters of regression models by counting the number of K-tuples in the vector of residuals which have alternating signs. In contrast to other tests for this situation, the sign depth test needs less assumptions. While for example the F-test needs the assumption of normally distributed errors, the sign depth test only needs the assumption of the median of the errors being zero. Because of this, the sign depth test can be used more generally and especially in the context of robust regression. But the sign depth test had one big disadvantage: In the past, it could not be applied in the context of multiple regression because the sign depth and the sign depth test depend on the ordering of the residuals and while the residuals in simple regression follow an inherent order this is often not the case in the context of multiple regression. This thesis has found a suitable way to apply the sign depth test also in this situation.

In Chapter 2 the sign depth and sign depth test were described in the context of robust regression. In addition, the problem of the sign depth test in the context of

multiple regression was described in detail and the aim of this thesis was presented. For this, at first some classical robust estimators and tests for linear regression models were described, like the L_1-regression, the LTS-regression and the M-, S- and MM-regression. Afterwards, the field of data depths was introduced. In this context, also the sign depth and the sign depth test were defined and its characteristics were described, see Definition 2.3 on page 16 and Theorem 2.2 on page 19. Furthermore, in Section 2.7 an example was provided about the problem of the sign depth test in the context of multiple regression and how crucial the effect of different orderings of the residuals on the sign depth test may be. At the end of this chapter the aim of this thesis was described and several research questions for this thesis were formulated. These questions were answered in Chapters 3, 4 and 5 and will now be summarized in this chapter.

The first research question in Section 2.8 deals with the possibilities of ordering multidimensional data. It is asked which methods exist and which characteristics each method has. Especially, it is of interest which data types can be ordered (Question 1(a)), whether the orderings are affected by linear transformations of the data points (Question 1(b)), whether the methods preserve an inherent order in the data if there is one (Question 1(c)), how large the theoretical and empirical runtimes of the different ordering methods are (Question 1(d)) and whether there are further advantages or disadvantages of the ordering methods (Question 1(e)). All these questions were answered in Chapter 3 where overall 13 different ordering methods for multidimensional data were described. In particular, reference is made to Table 3.1 on page 82 which summarized the characteristics of the different ordering methods.

But to answer the questions here in a bit more detail, at first it has to be said that the 13 found ordering methods can be assigned to four groups whose methods have similar characteristics: Two naive ordering methods, five scalarization based ordering methods, three ordering methods based on partial sorting and three distance based ordering methods.

The naive ordering methods were presented in Section 3.1. These methods describe an ordering according to the appearance of the regression vectors in the data set and on the other hand a random ordering of the regression vectors. Both methods have the advantage that the orderings do not depend on the values of the regression vectors itself, but only on the index numbers of the regression vectors and because of this, all data types (including nominal and ordinal data) can be ordered and the orderings are not affected by (linear) transformations of the data points. Furthermore, the

theoretical and empirical runtimes of these ordering methods are very small. On the other hand, only the ordering according to the appearance of the regression vectors in the data set preserves an inherent order in the data and this is also only the case if the regression vectors are already ordered according to that inherent order in the data set. If this is not the case or a random order is chosen, a possible inherent order in the data set is not preserved.

The scalarization based methods were described in Section 3.2. Its way to order multidimensional regression vectors is to scalarize each regression vector and order the regression vectors according to these scalarized values. Two of the described methods are only special cases of a third one: Taking only the values of one component of each regression vector for ordering and taking a weighted sum of the values of each regression vector can both also be expressed as an ordering according to an orthogonal projection on a line with different parameters of the line. Also the fourth method, an ordering according to the median value of each regression vector, is a special case of the projection method in the two-dimensional case. Only the fifth method, an ordering according to the values of a vector norm of each regression vector, is really different to the other methods. All five methods have in common that they need metric values for ordering. Nominal or ordinal components in the regression vectors are not allowed, with the only exception that when using only the values of one component for ordering the data types of the other components do not matter. The results of all scalarization based methods except the ordering according to the values of a vector norm are invariant to linear transformations of the data points. Possibly the obtained orders may be reversed after linear transformation of the data points, but this does not matter for the sign depth and the sign depth test. On the other hand, when ordering according to the values of a vector norm the data points can only be transformed multiplicatively without changing the order, but not additively. And this ordering method has a further disadvantage to the others: When having only one-dimensional data the inherent order is not preserved. When having an inherent order in higher dimensional data, all scalarization based methods do not preserve this order necessarily. An advantage of all scalarization based methods is their rather small computational runtime, theoretical as well as empirical. Although the runtimes are larger than those of the naive ordering methods, all runtimes depend only linearly on the number of data points and possibly on the number of dimensions.

The ordering methods based on partial sorting of the regression vectors are described in Section 3.3. These methods order the regression vectors multivariately. Because

this only leads to a partial order of the regression vectors, in a second step the regression vectors with the same rank in the partial sorting process have to be ordered somehow. In this thesis, three ordering methods based on partial sorting were described: A partial sorting on the basis of a nondominated sorting, a partial sorting based on convex hulls and a partial sorting based on the values of Tukey's halfspace depth. At the beginning of Section 3.3 a large disadvantage of all three methods is described: These methods can only be applied in relatively few dimensions. When the data is higher dimensional the partial sorting does not work anymore because in this case (nearly) all regression vectors get the same rank. This behavior is visualized in Figure 3.17 on page 50. Another disadvantage of these methods is their complexity. While the naive ordering methods and the scalarization based ordering methods were easily understandable, this is not necessarily the case for the methods based on partial sorting. In addition, like the scalarization based methods, these ordering methods need metric data, nominal or ordinal components in the regression vectors are not possible. Furthermore, the theoretical time complexities as well as the empirical computational runtimes of these ordering methods are very large, for the ordering method based on convex hulls and the ordering according to the values of the halfspace depth up to exponential in the number of data points in the worst case. But these two methods have the advantage that the data points can be transformed linearly without changing the resulting order. This is not the case for the nondominated sorting method: Here, when transforming the data points multiplicatively the order may change. On the other hand, the nondominated sorting method is the only method of these three which preserves an inherent one-dimensional order. But when having an inherent order in higher dimensions, all three methods may fail to preserve this order.

The last group of presented ordering methods is the group of distance based methods presented in Section 3.4. The aim of these methods is to order regression vectors with a small pairwise distance near to each other and regression vectors with a large pairwise distance far away from each other. This easily understandable concept is used by three different ordering methods: Firstly, by searching for the exact solution of the Shortest Hamiltonian Path problem, secondly, by an approximation of it and thirdly, by using the order of a dendrogram obtained by a hierarchical clustering process. The approximation of the Shortest Hamiltonian Path problem is considered here because the computational time complexity of searching for the exact solution is exponential in the number of data points in the worst case which is a big disadvantage of this ordering method. Also, the other two ordering methods have

relatively large time complexities. The approximation of the Shortest Hamiltonian Path is cubic in the number of data points and the hierarchical clustering quadratic. But while the empirical runtimes of both methods based on the Shortest Hamiltonian Path problem are also relatively large (and indeed have the same magnitude), the empirical runtimes of the hierarchical clustering are quite small. But besides of the relatively large computational time complexities, there exist only advantages of these methods. Firstly, the obtained orders of these methods are not affected by linear transformations of the data points because the methods do not base on the values of the regression vectors itself, but only on the pairwise distances. Secondly, this is also the cause why these ordering methods can be used also for non-metric data or data where single components are non-metric. As long as a meaningful distance measure exists and the pairwise distances can be calculated all of these methods are applicable. Furthermore, these methods usually preserve inherent orders in the data. Especially, in the one-dimensional case an inherent order will be preserved for sure.

The second research question in Section 2.8 deals with the implementation of the sign depth, the sign depth test and the ordering methods in `R`. It is asked how these things can be efficiently implemented and how these implementations can be made usable for other people.

This research question is answered in Chapter 4, where the developed software for this thesis was described. The most challenging task in this regard was implementing the sign depth since for calculating the sign depth $\binom{N}{K}$ summands have to be calculated. Although in the meantime Dennis Malcherczyk and Kevin Leckey have found a way to compute an (asymptotic) version of the sign depth in linear time complexity in the number of data points, for this thesis three different implementations of the sign depth were developed before calculating the sign depth in linear time complexity was possible. These three versions were described in Section 4.1. Although none of these implementations is able to compute the sign depth in less time complexity than $\mathcal{O}(\binom{N}{K})$, the empirical runtimes strongly decrease from version one to three, see Figure 4.3 on page 95. This speed up was obtained by two things: Firstly, clever summarization of the summands of the sign depth and secondly, using the programming language `C++` instead of `R`.

In contrast to implementing the sign depth implementing the sign depth test and the ordering methods was quite easy. These implementations were described in Sections 4.2 and 4.3. The sign depth test was implemented in two ways: Either with a vector of residuals as input or with a model description and parameters for

testing as input. For both ways of course the distribution of the sign depth has to be known for getting the necessary quantile for the sign depth test. These quantiles were simulated for $N \leq 100$ and $K \leq 5$ for this purpose. Regarding the implementation of the ordering methods, the only challenging thing in this regard was to connect the TSP-solver *Concorde* to `R` which is needed for computing the exact solution of the Shortest Hamiltonian Path problem. Unfortunately, every user has to do this on his/her own when wanting to use the respective ordering method, this could not be made globally. The twelve other ordering methods could be easily implemented or extracted from other `R`-packages.

The last part of the second research question deals with the possibilities to make the implementations available for other people. For this, in Section 4.4 the `R`-package `GSignTest` is described which was written during this thesis and contains three implementations of the sign depth (the best one described in Section 4.1, the asymptotic one in linear time complexity as well as the block-implementation from Dennis Malcherczyk), the sign depth test, all described ordering methods as well as the F-test and the classical sign test. For making this package available for other people this package was uploaded to GitHub where everybody can download it for free.

The third research question of Section 2.8 deals with the power of the sign depth test in dependance of the different ordering methods. It is of interest which ordering method performs best, where the performance is measured by looking at and analyzing simulated power functions. Especially, the effect of different numbers of data points (Question 3(a)), different values of the parameter K of the sign depth (Question 3(b)), different underlying data sets (Question 3(c)), different error distributions in the simulated data sets (Question 3(d)), different values of possible hyper-parameters of the ordering methods (Question 3(e)) and different linear models (Question 3(f)) is of interest.

The research questions 3(a) - 3(e) were answered in Section 5.2. In this section, power functions were simulated for testing on $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. In general, the simulated power functions have shown that the underlying data set, the error distribution in the data and the hyper-parameters of the ordering methods have no or only little effect on the power of the sign depth test, independent of the ordering method. Only the hierarchical clustering method is affected a little bit more by its hyper-parameter which describes the linkage function. But Subsections 5.2.1 and 5.2.4 showed that the power of the sign depth test depends

on the number of data points and the value of the parameter K of the sign depth. While for K in general larger values are better than smaller ones and odd values are better than even values, for the number of data points of course the power of the sign depth test should increase for larger values. But indeed, this was not the case for all of the ordering methods. For the naive ordering methods and the ordering according to a vector norm the power is always about α, independent from the number of data points and for the methods based on convex hulls and on Tukey's halfspace depth the power is even decreasing when increasing the number of data points. The eight other ordering methods have performed better. The four remaining scalarization based methods and the ordering methods based on a nondominated sorting all produce power functions which have a direction with low power in it. Although the power in all other directions is very good this behavior is not desirable. The three distance based ordering methods instead have performed completely satisfying: Its power is low only in or near the area of H_0 and otherwise quite great. In all considered situations the ordering based on the exact solution of the Shortest Hamiltonian Path problem performed best whereas the two other distance based ordering methods were slightly worse. A good overview of the power of the sign depth test applied with the different ordering methods in dependance of the number of data points and the value of K can be found in Figure 5.23 on page 141. As a conclusion from this section it can be said that the sign depth test is better the less "mess" there is in the ordering of the regression vectors. Because of this, only the distance based ordering methods shall be used when applying the sign depth test in the context of multiple regression. In particular, if possible, the exact solution of the Shortest Hamiltonian Path problem shall be used because it leads to the best performance of the sign depth test.

The research question 3(f) was answered in Section 5.3. In this section the distance based ordering methods were applied on different linear models: Models with intercept, models with interactions, a quadratic regression model and really high dimensional models. The results were in general the same as in the section before: The distance based ordering methods lead to satisfying simulated power functions, where the best power is achieved when using the exact solution of the Shortest Hamiltonian Path problem for ordering. So, the sign depth test can be used independently of the characteristics of the regressors in the linear model. But, it was noticed that the power in the direction of parameters of interactions or quadratic terms is slightly smaller than in other directions. Especially, in the direction of the parameter of an intercept the power is very great, so that already small deviations from the true

intercept are noticed by the sign depth test whereas deviations in interactions or quadratic terms have to be larger to be noticed by the test. Subsection 5.3.4 has shown further interesting results regarding different linear models when using the sign depth test. In this subsection very high dimensional models were considered with up to 80 dimensions. It has been shown that for these large number of dimensions relatively small values of K are not sufficient anymore to get a good power of the sign depth test. Instead, the value of K should have at least the same magnitude as the number of dimensions and really great power values are only achieved if K is larger than the number of dimensions.

The fourth research question in Section 2.8 deals with the question how the sign depth test performs in contrast to other tests on the parameters of linear regression models. For this, three tests should be considered: The classical sign test, the F-test and a robust Wald test.

This question was answered in the Sections 5.5 and 5.6. In Section 5.5 the three tests were compared with the sign depth test in some situations of Sections 5.2 and 5.3, i.e. for different data and model situations. For comparison in this section always the exact solution of the Shortest Hamiltonian Path problem was used for ordering the regression vectors and the parameter K of the sign depth was set to 5 because the previous sections have shown that in this case the sign depth test performs best. In general, the classical sign test has performed very badly in all situations and the F-test of course cannot be beaten when having normally distributed errors, but performs rather badly when having non-normally distributed errors. In contrast, the robust Wald test performed very good in all situations and often (slightly) better than the sign depth test which had also satisfying results in all situations. The only exception were models with a very large number of dimensions. While the sign depth test worked in these situations (although the power of the test was rather bad), the robust Wald test could not be carried out anymore due to numerical reasons. Already for a medium number of dimensions, the robust Wald test had not maintained the level of the test anymore which was never the case for the sign depth test. Also in Section 5.6 the robust Wald test performed quite badly. In this section the sign depth test was compared to the robust Wald test in the situation of model checks, i.e. when one or more regressors were not seen in the data. Here, the sign depth test performed much better than the robust Wald test because the sign depth test does not depend on an estimation of the parameter vector.

The last research question of Section 2.8 asked how the sign depth test performs on real data. For answering this, in Section 5.7 the sign depth test was applied to data from a bridge monitoring. It was examined if the crack width of a crack in this bridge could be sufficiently modeled by the temperature, the date and the previous crack width. For this, both an ordinary least-squares model and a robust regression model were used and the fits of the models were compared with the sign depth test. As seen in Section 5.6 before, the sign depth test performs really satisfying when using it for such model checks. Although this data can be regarded as time-series with an inherent order in it, it was also looked at the effect of a multidimensional ordering of the regression vectors considering all regressors. Here, it was shown that an ordering according to the exact solution of the Shortest Hamiltonian Path problem leads to changes in the p-values of the sign depth test in contrast to using the inherent order, but the general result of the test remained the same: The poorly fitting ordinary least-squares models were again seen as poorly fitting, whereas the robust models were mostly considered as good fitting, independent of whether the regression vectors were ordered according to the exact solution of the Shortest Hamiltonian Path problem or according to the inherent order. So, in this case, the performance of the sign depth test was really satisfying.

To summarize the obtained results in this thesis it can be said that the goal making the sign depth test applicable for multiple regression is fulfilled. For using the sign depth test in this situation, an ordering of the regression vectors is necessary. For this, always methods based on pairwise distances of the regression vectors should be used. If possible, the regression vectors should be ordered according to the exact solution of the Shortest Hamiltonian Path problem. Although this method has a large computational time complexity, the empirical runtimes of ordering several hundreds of regression vectors are not too large. The approximate solution of the Shortest Hamiltonian Path problem should only be used for ordering when the exact solver is not available due to technical reasons because its performance is worse than the performance of the exact solution and an advantage in the empirical runtimes could not be seen for up to 1 000 data points. If a speed up of the sign depth test is needed or a data set with a really large number of data points is used, the hierarchical clustering method should be used for ordering because its empirical runtimes are very low and although its performance is worse than the one of the exact solution of the Shortest Hamiltonian Path problem it is still better than all other considered ordering methods.

All in all, the sign depth test is competitive to other tests for multiple regression. Although its performance is sometimes slightly worse than those of a robust Wald test or, in case of normally distributed errors, the F-test, it has many advantages like its general usability and its easy comprehensibility. Furthermore, in the case of model checks it outperforms the robust Wald test clearly.

6.2 Outlook

This thesis has led to much understanding about the sign depth test in the context of multiple regression. But although this thesis has answered many questions, some questions are still open and there is much more potential for research in the field of sign depths and sign depth tests in the context of multiple regression as well as in different contexts. This section will give a short overview over open questions and possible future research topics.

In this thesis a large number of simulations was carried out. Although they cover a wide range of possible situations, there can be done much more in the future. In the following, ten possible future research questions are formulated.

Firstly, the simulations in this thesis were limited to up to only 100 data points in the data sets. There is not much doubt that the sign depth test would perform similarly when having much more data points, but to be sure and also for analyzing the empirical runtimes of the sign depth test and the ordering methods in this context, simulations with larger numbers of data points could be carried out. Of course, for this at first quantiles of the distribution of the K-sign depth have to be simulated for the respective number of data points.

Secondly, it could be seen especially in Subsection 5.2.4 that the sign depth test seems to perform differently when choosing odd or even values for K. The reason for this behavior is still unknown and an interesting research question. In addition, it is not cleared why for example the power of the sign depth test is more affected by this when using scalarization based ordering methods than when using distance based ordering methods.

Thirdly, by using Cauchy distributed errors in this thesis only the case of outliers in the response variable was considered. The case of outliers in the regressors (i.e. leverage points) was neglected. But this case is of big interest in this context,

especially because leverage points do also affect the ordering of the regression vectors. Here, the ordering methods could be analyzed regarding their robustness and the effect on the power of the sign depth test could be analyzed in this situation.

Fourthly, this thesis has shown that the sign depth test performs best when using distance based methods for ordering the regression vectors. Three different methods for this were proposed. But of course, there may be more possible methods. Especially, in the context of approximate solutions of the Shortest Hamiltonian Path problem there exist more approaches than the here used nearest neighbor approach, see for example Rego et al. (2011). Since the results in this thesis have shown that the nearest neighbor approach has quite large empirical runtimes and its performance is clearly worse than the performance of the exact solution, it is of great interest whether some other approximations perform better and have smaller empirical runtimes.

Fifthly, Section 5.4 has shown that there may be small differences in the performance of the sign depth test when using the design vectors for ordering instead of the regression vectors. This effect seems to be very small, but was only analyzed exemplary for one model. Here, some more research about this effect may be useful.

Sixthly, for the simulations in this thesis only metric regressors were considered. As stated in Section 3.4, the distance based ordering methods could also be used for non-metric data if pairwise distances between the regression vectors could be still calculated. This is possible when using for example the so-called *Gower's distance* which is based on Gower's coefficient (Gower, 1971) and implemented for example in the function `daisy()` in the `R`-package `cluster` (Maechler et al., 2019b).

Seventhly, in this thesis the sign depth test always was used for testing on a fixed parameter vector of a linear model. Often, not the complete parameter vector is of interest, but only the influence of some components. Also in this case the sign depth test can be used for testing, but its performance in such situations, especially in contrast to other (robust) tests, has not been examined in detail so far.

Eighthly, the results of Subsection 5.3.4 and Section 5.6 were very interesting and promising. But in the field of testing high-dimensional models and performing model checks many more things can be examined. For example, in the context of high-dimensional models the following questions arise: How much has the value of K be larger than the number of dimensions to obtain satisfying results? Which is the largest possible value of K before getting numerical instability of the sign depth

values? Does this value also depend on the number of data points? And in the context of performing model checks with the help of the sign depth test it can be asked whether this test performs in all situations as well as seen in Section 5.6 or if there are situations where the test fails.

Ninthly, in this thesis the sign depth test was compared with the classical sign test, the F-test and a robust Wald test. Of course, in literature there exist more (robust) tests for testing on parameters in linear models, for example the τ-test (see for example Section 7.2 of Hampel et al. (1986)). For getting a better impression of the performance of the sign depth test, its performance should be compared to more tests in the future.

Tenthly, the application of the sign depth test is not limited to linear regression. Kustosz et al. (2016a) and Kustosz et al. (2016b) have used the sign depth test for example for time-series, where because of the inherent order in the data no problem with ordering regression vectors occurred. But for other model classes there may be no inherent order of multidimensional regression vectors. For example, there is the big field of generalized linear models, for which the performance of the sign depth test could also be analyzed in many different situations.

Of course, the research about sign depths and sign depth tests is not limited to the mainly simulative approach as in this thesis. Also, theoretically many things about the behavior and the characteristics of sign depths can be analyzed: consistency of the tests, structure of the distributions of the sign depths including quantiles and extremes, cases of failure of the test and much more. In addition, also the time complexity of the calculation of the sign depths is a current research topic. For this, several approaches exist, which were developed mainly by Dennis Malcherczyk.

It can be seen that there are many more interesting research questions in the context of sign depths and sign depth tests. Hopefully, many of them can be answered in the future. I would be glad by being part of it!

Acknowledgment

The author gratefully acknowledge support from the Collaborative Research Center "Statistical Modelling of Nonlinear Dynamic Processes" (SFB 823, B5) of the German Research Foundation (DFG).

The author gratefully acknowledge the computing time provided on the Linux HPC cluster at Technical University Dortmund (LiDO3), partially funded in the course of the Large-Scale Equipment Initiative by the German Research Foundation (DFG) as project 271512359.

Appendix A

Theorems and Algorithms

Theorem A.1. ***The F-test for ordinary least-squares regression***

Let $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{e}$ be a linear model with $rank(\boldsymbol{X}) = D < N$ and $\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$. The hypothesis $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ vs. $H_1 : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0$ can be rejected to the level $\alpha \in (0,\, 1)$ if

$$\frac{N-D}{D} \frac{(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)^\top (\boldsymbol{X}^\top \boldsymbol{X})(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)}{\hat{\boldsymbol{e}}(\hat{\boldsymbol{\theta}})^\top \hat{\boldsymbol{e}}(\hat{\boldsymbol{\theta}})} > F_{D,N-D,1-\alpha},$$

where $F_{D,N-D,1-\alpha}$ denotes the $(1-\alpha)$-quantile of the F-distribution with D and $N-D$ degrees of freedom.

Theorem A.2. ***The sign test***

Let $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{e}$ be a linear model and N_+ the number of positive residuals under the null-hypothesis, i.e. $N_+ = \#\{\hat{\boldsymbol{e}}(\boldsymbol{\theta}_0) > 0\}$. The hypothesis $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$ vs. $H_1 : \boldsymbol{\theta} \neq \boldsymbol{\theta}_0$ can be rejected to the level $\alpha \in (0,\, 1)$ if

$$N_+ \notin [q_{N,0.5,\alpha/2},\, q_{N,0.5,1-\alpha/2}],$$

where $q_{N,0.5,\alpha}$ denotes the α-quantile of the Binomial distribution with parameters N and 0.5.

Algorithm A.1 Naive algorithm for computing the simple regression model $\boldsymbol{y} = \theta_0 + \theta_1 \boldsymbol{x}_{\bullet 1} + \boldsymbol{e}$ with maximal K-sign depth.

Input: Data set $\boldsymbol{Z} \in \mathbb{R}^{N \times 3}$, where one column has constant value 1, one column consists of the values of the regressor $\boldsymbol{x}_{\bullet 1}$ and the last column consists of the values of the response variable $\boldsymbol{y}$; Parameter K of the sign depth

Output: Parameter vector $\hat{\boldsymbol{\theta}} = (\theta_0, \theta_1)^\top$, which leads the the maximal K-sign depth

1: Compute all $\binom{N}{2}$ combinations of values $\boldsymbol{z}_{n_1}$ and $\boldsymbol{z}_{n_2}$, $n_1, n_2 = 1, \ldots, N$.
2: $max\_sign\_depth = 0$
3: **for all** combinations **do**
4: $\quad \theta_0 = y_{n_1} - x_{n_1} \cdot \frac{y_{n_2} - y_{n_1}}{x_{n_2} - x_{n_1}}$
5: $\quad \theta_1 = \frac{y_{n_2} - y_{n_1}}{x_{n_2} - x_{n_1}}$
6: $\quad$ Compute residuals $\hat{\boldsymbol{e}}(\boldsymbol{\theta}) = \boldsymbol{y} - \theta_0 - \theta_1 \boldsymbol{x}_{\bullet 1}$.
7: $\quad$ Compute the K-sign depth $current\_depth$ of the residuals
8: $\quad$ **if** $current\_depth > max\_sign\_depth$ **then**
9: $\quad\quad max\_sign\_depth = current\_depth$
10: $\quad\quad$ Cache the values of θ_0 and θ_1
11: $\quad$ **end if**
12: **end for**
13: **return** The last cached values of $\hat{\boldsymbol{\theta}} = (\theta_0, \theta_1)^\top$

Appendix B

Further Implementations

In this chapter, further implementations are shown. The codes are written in `C++` with the help of the `R`-package `Rcpp` (Eddelbuettel and Balamuta, 2017).

```
// signC: helper-function: the sign-function
// input: double x: the value to determine the sign of
// output: int: 1, if x is positive, 0 if x is 0,
//              -1, if x is negative

int signC(double x) {
  if (x > 0) {
    return 1;
  } else if (x == 0) {
    return 0;
  } else {
    return -1;
  }
}
```

Code B.1: A simple sign function which is needed in Code B.2 and Code B.6.

```
// calcSwitchSign: helper-function: determines if a vector has
//                 alternating signs
// input: NumericVector v: The underlying vector
//        IntegerVector ind: the indices one wants to look at
//                           (ind has length K)
// output: double: 1 if v[ind] has alternating signs, 0 otherwise

double calcSwitchSign(NumericVector v, IntegerVector ind) {
  int K = ind.size();
  double counter = 1;
  int tmp = signC(v[ind[0]]);
  for(int i = 1; i < K; i++) {
    if(signC(v[ind[i]]) == tmp) {
      counter = 0;
      break;
    } else {
      tmp *= -1;
    }
  }
  return counter;
}
```

Code B.2: Function for checking whether a vector has alternating signs. This function is needed in Code B.4. A pseudocode of this algorithm is given in Algorithm 4.3 on page 89.

```
// calcNextVec: helper-function: calculates the "next" combination
//              of values given an old combination
//              (Ordering is like in the R-function combn())
// input: IntegerVector o: an "old" index-vector
//        int N: the maximal possible value in the output
//        int K: the length of o
// output: IntegerVector: output has length K.
//                        The "next" index-vector

IntegerVector calcNextVec(IntegerVector o, int N, int K) {

  IntegerVector n(K);

  // Calculate the position for increasing
  int j = K - 1;
  while(o[j] == N - K + j)
    j--;

  // All values before j stay the same
  for(int i = 0; i < j; i++)
    n[i] = o[i];

  // Increase the value at j
  n[j] = o[j] + 1;

  // Increase all following values by one
  for(int i = j + 1; i < K; i++)
    n[i] = n[i - 1] + 1;

  return n;
}
```

Code B.3: Computing the ”next” index vector on the basis of an old one. See Algorithm 4.2 on page 88.

```
// calcDepth: calculates the K-sign depth of a given vector of
//            residuals.
//
// Input: NumericVector e: the residuals
//        int K: the K of the sign depth
// Output: double: the calculated K-sign depth of e.

double calcDepth(NumericVector e, int K) {

  int N = e.size();
  if(N < K)
    stop("Cannot calculate Depth when K is larger than the numbers
       of residuals");

  double num = 0;
  double denom = Rf_choose(N, K);

  // Create the "first" index vector and check whether it has
  // alternating signs
  IntegerVector o(K);
  for(int i = 0; i < K; i++)
    o[i] = i;
  num = calcSwitchSign(e, o);

  // Calculate the numerator of the K-sign depth
  IntegerVector n(K);
  for(int i = 1; i < denom; i++) {
    n = calcNextVec(o, N, K);
    num += calcSwitchSign(e, n);
    o = n;
  }

  return num / denom;
}
```

Code B.4: Computation of the K-sign depth with iteratively computing the index combinations. A pseudocode of this algorithm is given in Algorithm 4.4 on page 90.

```
// recFun: Recursive function used in calcFastDepth() for computing
//         the sign depth
// Input: IntegerVector r: Vector of signs
//        int K: length of considered tuple
//        int j: index
//        int SignOld: sign of previous residual
// Output: int: number of alternating K tuples starting at r[j]

int recFun(IntegerVector r, int K, int j, int signOld) {
  if (r[j] == signOld) return 0;
  if (K == 1) return 1;
  int N = r.size();
  int result = 0;
  for (int i = j + 1; i < n - K + 2; i++)
    result += recFun(r, K - 1, i, r[j]);
  return result;
}
```

Code B.5: C++-implementation of `recFun()` described in Algorithm 4.5 on page 92.

```
// calcFastDepth: calculates the K-sign depth of a given
//                vector of residuals.
//
// Input: NumericVector e: the residuals
//        int K: the K of the sign depth
// Output: double: the calculated K-sign depth of e.

double calcFastDepth(NumericVector e, int K) {
  int N = res.size();
  if(N < K)
    stop("Cannot calculate Depth when K is larger than the numbers
        of residuals");
  IntegerVector r = signC(e);
  int num = 0;
  for(int i = 0; i < N - K + 1; i++) {
    num += recFun(r, K, i, 0);
  }
  return num / Rf_choose(N, K);
}
```

Code B.6: Recursive implementation of the K-sign depth, see Algorithm 4.6 on page 92.

Appendix C

Further Results and Graphics

C.1 Further Results and Graphics of Chapter 3

Method	D	$N = 200$	$N = 400$	$N = 600$	$N = 800$	$N = 1\,000$
	1	0.0003	0.0003	0.0003	0.0003	0.0003
	2	0.0003	0.0003	0.0003	0.0004	0.0003
	3	0.0003	0.0003	0.0004	0.0004	0.0004
Order	4	0.0003	0.0004	0.0004	0.0004	0.0004
of the	10	0.0004	0.0004	0.0004	0.0005	0.0005
Data Set	20	0.0005	0.0005	0.0006	0.0006	0.0006
	50	0.0008	0.0008	0.0009	0.0010	0.0010
	100	0.0012	0.0013	0.0015	0.0016	0.0017
	1	0.0003	0.0003	0.0003	0.0003	0.0003
	2	0.0004	0.0004	0.0003	0.0004	0.0004
	3	0.0004	0.0004	0.0004	0.0004	0.0004
Random	4	0.0004	0.0004	0.0004	0.0004	0.0004
Order	10	0.0004	0.0004	0.0004	0.0005	0.0005
	20	0.0005	0.0005	0.0006	0.0006	0.0006
	50	0.0008	0.0009	0.0009	0.0010	0.0011
	100	0.0012	0.0013	0.0015	0.0016	0.0017

Method	D	$N = 200$	$N = 400$	$N = 600$	$N = 800$	$N = 1000$
	1	0.0010	0.0015	0.0019	0.0023	0.0027
	2	0.0010	0.0015	0.0020	0.0024	0.0028
	3	0.0011	0.0015	0.0020	0.0025	0.0029
Norm	4	0.0011	0.0016	0.0020	0.0025	0.0030
	10	0.0012	0.0018	0.0022	0.0029	0.0034
	20	0.0016	0.0023	0.0029	0.0036	0.0043
	50	0.0021	0.0030	0.0039	0.0048	0.0055
	100	0.0029	0.0042	0.0053	0.0065	0.0078
	1	0.0061	0.0118	0.0173	0.0229	0.0287
	2	0.0078	0.0150	0.0222	0.0294	0.0373
	3	0.0064	0.0120	0.0178	0.0235	0.0291
Median	4	0.0078	0.0150	0.0225	0.0296	0.0376
	10	0.0081	0.0156	0.0230	0.0307	0.0384
	20	0.0087	0.0165	0.0244	0.0323	0.0403
	50	0.0098	0.0182	0.0267	0.0350	0.0442
	100	0.0112	0.0206	0.0298	0.0398	0.0500
	1	0.0004	0.0004	0.0004	0.0005	0.0005
	2	0.0005	0.0005	0.0005	0.0005	0.0006
	3	0.0005	0.0005	0.0005	0.0006	0.0006
Taking	4	0.0005	0.0005	0.0005	0.0006	0.0006
only One	10	0.0005	0.0006	0.0006	0.0006	0.0007
Component	20	0.0007	0.0007	0.0008	0.0008	0.0009
	50	0.0010	0.0011	0.0012	0.0012	0.0013
	100	0.0015	0.0016	0.0018	0.0020	0.0021
	1	0.0005	0.0005	0.0006	0.0006	0.0006
	2	0.0006	0.0007	0.0006	0.0007	0.0007
	3	0.0006	0.0006	0.0007	0.0007	0.0007
Weighted	4	0.0007	0.0007	0.0007	0.0008	0.0008
Sum	10	0.0007	0.0008	0.0009	0.0009	0.0010
	20	0.0010	0.0011	0.0012	0.0012	0.0013
	50	0.0015	0.0017	0.0020	0.0021	0.0024
	100	0.0025	0.0029	0.0033	0.0037	0.0042

Method	D	$N = 200$	$N = 400$	$N = 600$	$N = 800$	$N = 1000$
	1	0.0012	0.0016	0.0020	0.0024	0.0029
	2	0.0013	0.0018	0.0023	0.0027	0.0031
	3	0.0013	0.0018	0.0024	0.0028	0.0034
Orthogonal	4	0.0013	0.0019	0.0023	0.0029	0.0034
Projection	10	0.0015	0.0021	0.0026	0.0034	0.0040
	20	0.0019	0.0028	0.0035	0.0042	0.0049
	50	0.0027	0.0037	0.0047	0.0058	0.0069
	100	0.0039	0.0054	0.0070	0.0086	0.0102
	1	0.8324	3.3718	7.6705	13.9339	22.1543
	2	3.9227	15.9386	35.5911	63.4982	98.8578
	3	4.5597	18.5137	41.0351	73.4129	114.7499
Nondominated	4	5.2563	21.1785	47.7421	84.2942	131.2823
Sorting	10	9.5994	38.2405	85.7279	153.4971	240.7873
	20	17.0616	68.4228	152.5466	273.1385	425.7929
	50	39.4030	157.2094	353.3438	626.3493	979.9437
	100	77.2364	307.7078	688.9082	1228.7646	1915.7058
	1	0.2899	0.8211	1.6626	2.8087	4.2244
	2	0.4803	1.0849	2.0093	2.9377	4.2019
	3	0.4699	1.3248	2.8403	4.6583	6.3406
Convex	4	0.5027	1.6008	3.0990	4.7090	6.6732
Hull	10	144.1640	665.3777	1474.4174	2531.3699	3687.5023
	20	341.3908	340.2331	343.0663	351.5732	359.3921
	50	378.3224	377.7972	387.1958	389.9978	395.6217
	100					
	1	5.1465	19.2338	41.6567	73.9024	117.9331
	2	5.6109	19.9908	43.0630	75.0212	113.8194
	3	5.6199	20.4481	42.9085	76.8782	119.1387
Halfspace	4	5.8156	19.9264	42.3194	76.8346	114.0558
Depth	10	6.2754	22.0333	48.7999	82.2124	126.2264
	20	6.8064	23.1262	51.7690	84.3729	132.3699
	50	8.4549	26.6148	55.2507	92.5085	138.5278
	100	11.3093	32.5389	64.4420	107.1481	156.0145

Method	D	$N = 200$	$N = 400$	$N = 600$	$N = 800$	$N = 1000$
	1	0.2901	0.8708	1.7870	2.9416	4.2861
	2	0.7693	5.9061	15.7736	49.4663	83.8316
	3	0.9230	3.7431	7.5536	12.7607	21.0504
Shortest	4	0.8121	3.0953	5.8133	8.8990	14.4815
Hamiltonian	10	0.8106	2.9934	5.1815	7.5932	13.3471
Path	20	0.9165	3.0365	5.5343	9.2464	13.7624
	50	1.2011	3.4868	5.7618	11.8676	17.3258
	100	1.0107	3.5993	7.6992	12.5747	17.2181
	1	0.6557	4.1341	13.8947	36.7756	70.5320
	2	0.6501	4.2284	14.2549	36.5267	68.2966
	3	0.6504	4.1579	14.0209	36.0181	66.6280
Nearest	4	0.6655	4.2079	13.7341	36.2438	68.5275
Neighbor	10	0.6536	4.1425	13.7903	36.5060	68.7059
Heuristic	20	0.6658	4.2924	13.7143	36.2407	66.8821
	50	0.6789	4.2368	14.1209	36.1265	69.2928
	100	0.6693	4.3258	14.0899	37.5369	67.1088
	1	0.0020	0.0063	0.0133	0.0233	0.0377
	2	0.0024	0.0072	0.0151	0.0278	0.0433
	3	0.0025	0.0079	0.0169	0.0308	0.0478
Hierarchical	4	0.0027	0.0087	0.0192	0.0339	0.0529
Clustering	10	0.0039	0.0133	0.0298	0.0512	0.0819
	20	0.0062	0.0214	0.0473	0.0847	0.1304
	50	0.0119	0.0435	0.0962	0.1716	0.2678
	100	0.0214	0.0803	0.1790	0.3171	0.4949

Table C.1: Median runtime (in seconds) of the ordering methods presented in Chapter 3.

Proof C.1. ***Time complexity of the Quickhull algorithm (Algorithms 3.2 and 3.3)***

a) ***Average Case (i.e. the groups are divided in equal sized parts):***

Let $T_A(N)$ the function of the average time complexity of the procedure FindHull *dependent of the number of data points N and let c_i, $i \in \mathbb{N}$ be positive constant values.*

If $N = 1$, the time complexity of FindHull is constant, i.e. $T_A(1) = c_1$.

For $N > 1$, lines 1 to 3 of Algorithm 3.3 have constant time complexity c_1 and lines 4 to 6 have linear time complexity $c_2 \cdot N$. When the data set is split in two groups with $N/2$ data points each, lines 7 and 8 have time complexity $T_A(\frac{N}{2})$ each. For all $k \in \mathbb{N}$ with $k \leq \log_2(N)$, this leads to a time complexity of FindHull of

$$\begin{aligned}
T_A(N) &= 2 \cdot T_A\left(\frac{N}{2}\right) + c_2 N + c_1 \\
&= 2 \cdot \left(2 \cdot T_A\left(\frac{N}{2^2}\right) + c_2 \frac{N}{2} + c_1\right) + c_2 N + c_1 \\
&= 2^2 \cdot T_A\left(\frac{N}{2^2}\right) + 2c_2 N + 2c_1 + c_1 \\
&= 2^2 \cdot \left(2 \cdot T_A\left(\frac{N}{2^3}\right) + c_2 \frac{N}{2^2} + c_1\right) + 2c_2 N + 2c_1 + c_1 \\
&= 2^3 \cdot T_A\left(\frac{N}{2^3}\right) + 3c_2 N + 2^2 c_1 + 2c_1 + c_1 \\
&= \dots \\
&= 2^k \cdot T_A\left(\frac{N}{2^k}\right) + kc_2 N + c_1 \sum_{i=0}^{k-1} 2^i \\
&= 2^k \cdot T_A\left(\frac{N}{2^k}\right) + kc_2 N + c_1 \cdot (2^k - 1)
\end{aligned}$$

For $k = \log_2(N)$, the time complexity is

$$\begin{aligned}
T_A(N) &= 2^{\log_2(N)} \cdot T_A\left(\frac{N}{2^{\log_2(N)}}\right) + \log_2(N) c_2 N + c_1 \cdot (2^{\log_2(N)} - 1) \\
&= N \cdot T(1) + c_2 N \log_2(N) + c_1(N-1) \\
&= c_1 N + c_1(N-1) + c_2 N \log_2(N) \\
&= c_1(2N-1) + c_2 N \log_2(N)
\end{aligned}$$

The average time complexity of Quickhull composes of the linear time complexity of the first two lines of Algorithm 3.2 and two calls of FindHull with $N/2$ data points each. This leads to an overall time complexity $T_A^Q(N)$ of

$$\begin{aligned}
T_A^Q(N) &= c_3 N + 2 \cdot T_A\left(\frac{N}{2}\right) \\
&= c_3 N + 2 \cdot \left(c_1\left(2\frac{N}{2} - 1\right) + c_2 \frac{N}{2} \log_2\left(\frac{N}{2}\right)\right) \\
&= c_3 N + c_1(2N-2) + c_2 N(\log_2(N) - 1)
\end{aligned}$$

So, the overall average time complexity of Quickhull is $\mathcal{O}(N \log(N))$.

b) ***Worst Case (i.e. in every step all data points are assigned to one group):***

Let $T_W(N)$ *the function of the worst case time complexity of the procedure* FindHull *dependent of the number of data points* N *and let* c_i, $i \in \mathbb{N}$ *be positive constant values.*

If $N = 1$, *the time complexity of FindHull is constant, i.e.* $T_W(1) = c_1$.

For $N > 1$, *lines 1 to 3 of Algorithm 3.3 have constant time complexity* c_1 *and lines 4 to 6 have linear time complexity* $c_2 \cdot N$. *When the data set is split in two groups where no data points are in the one group and the other* $N - 1$ *data points are in the other group, w.l.o.g. line 7 has constant time complexity* c_1 *and line 8 has time complexity* $T_W(N-1)$. *For all* $k \in \mathbb{N}$ *with* $k \leq N - 1$, *this leads to a time complexity of FindHull of*

$$\begin{aligned}
T_W(N) &= T_W(N-1) + c_2N + 2c_1 \\
&= T_W(N-2) + c_2(N-1) + 2c_1 + c_2N + 2c_1 \\
&= T_W(N-2) + c_2(N-1) + c_2N + 2 \cdot 2c_1 \\
&= T_W(N-3) + c_2(N-2) + 2c_1 + c_2(N-1) + c_2N + 2 \cdot 2c_1 \\
&= T_W(N-3) + c_2(N-2) + c_2(N-1) + c_2N + 3 \cdot 2c_1 \\
&= \dots \\
&= T_W(N-k) + c_2 \sum_{i=0}^{k-1} (N-i) + 2kc_1 \\
&= T_W(N-k) + c_2 \left(Nk - \sum_{i=0}^{k-1} i \right) + 2kc_1 \\
&= T_W(N-k) + c_2 \left(Nk - \frac{(k-1)k}{2} \right) + 2kc_1
\end{aligned}$$

For $k = N - 1$, *the time complexity is*

$$\begin{aligned}
T_W(N) &= T_W(1) + c_2 \left(N(N-1) - \frac{(N-2)(N-1)}{2} \right) + 2(N-1)c_1 \\
&= c_1 + c_2 \left(N^2 - N - \frac{N^2}{2} + \frac{3N}{2} - 1 \right) + 2(N-1)c_1 \\
&= (2N-1)c_1 + c_2 \left(\frac{N^2}{2} + \frac{N}{2} - 1 \right)
\end{aligned}$$

The worst case time complexity of Quickhull composes of the linear time complexity of the first two lines of Algorithm 3.2, one call of FindHull with constant

time complexity and one call of FindHull with $N-1$ data points. This leads to an overall time complexity $T_W^Q(N)$ of

$$\begin{aligned} T_W^Q(N) &= c_3 N + c_1 + T_W(N-1) \\ &= c_3 N + c_1 + (2(N-1)-1)c_1 + c_2 \left(\frac{(N-1)^2}{2} + \frac{N-1}{2} - 1 \right) \\ &= c_3 N + 2(N-1)c_1 + c_2 \left(\frac{(N-1)^2}{2} + \frac{N-1}{2} - 1 \right) \end{aligned}$$

So, the overall worst case time complexity of Quickhull is $\mathcal{O}(N^2)$.

C.2 Further Results and Graphics of Chapter 4

K	N	primitive `R`	iterative `C++`	recursive `C++`
$K=3$	20	0.0058	0.0002	0.0001
	40	0.0552	0.0017	0.0002
	60	0.1832	0.0067	0.0005
	80	0.4378	0.0219	0.0012
	100	0.8586	0.0382	0.0023
$K=4$	20	0.0263	0.0009	0.0001
	40	0.4918	0.0238	0.0008
	60	2.6177	0.1045	0.0037
	80	8.5366	0.3329	0.0159
	100	21.2139	0.8291	0.0383
$K=5$	20	0.0893	0.0028	0.0001
	40	3.6025	0.1414	0.0027
	60	30.0921	1.1639	0.0293
	80	132.1187	5.1218	0.1113
	100	415.9200	15.9729	0.3331

Table C.2: Median empirical runtime (in seconds) of the three algorithms presented in Section 4.1. For visualization of these values see Figure 4.3 on page 95.

C.3 Further Results and Graphics of Chapter 5

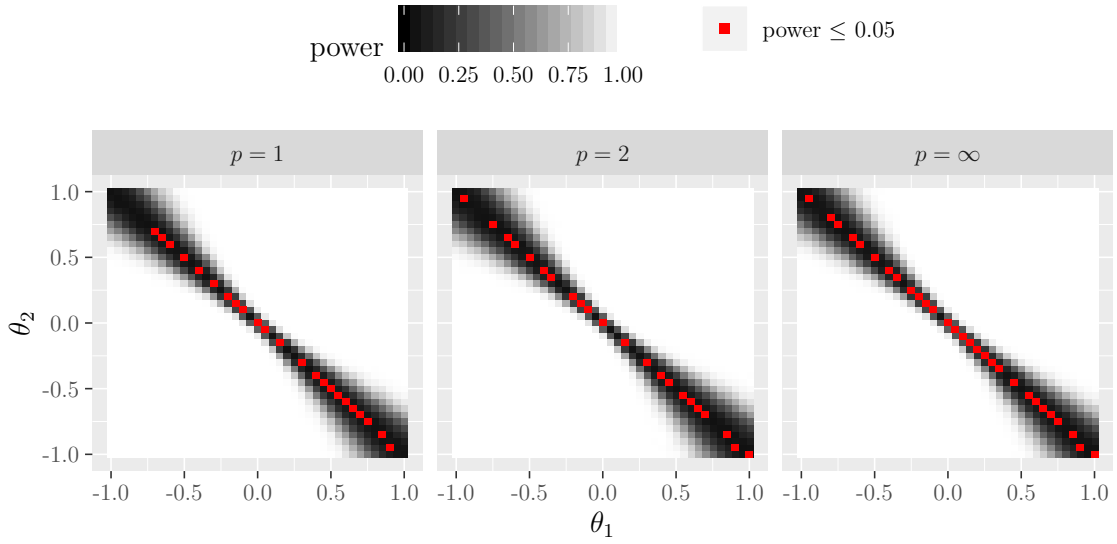


Figure C.1: Simulated power function when applying the sign depth test to a data set with random regression vectors in the range $[0, 2]^2$ and using different vector norms for ordering. Here, the hypothesis $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ is tested in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$. The power function is simulated with $N = 100$ data points, normally distributed errors and parameter $K = 3$ of the sign depth test. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

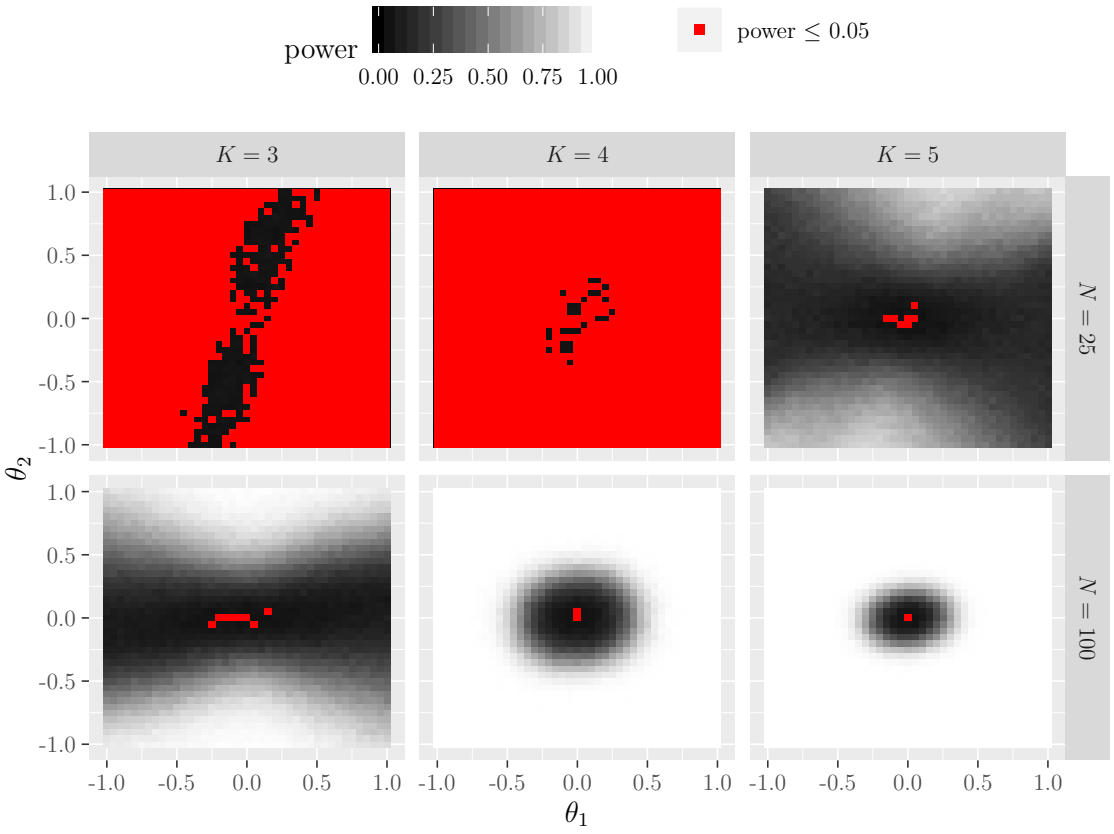


Figure C.2: Simulated power functions for the test $H_0 : \boldsymbol{\theta} = \mathbf{0}$ vs. $H_1 : \boldsymbol{\theta} \neq \mathbf{0}$ in the model $\boldsymbol{y} = \theta_1 \boldsymbol{x}_{\bullet 1} + \theta_2 \boldsymbol{x}_{\bullet 2} + \boldsymbol{e}$, where the vector $\boldsymbol{e}$ follows a normal distribution. The underlying data set is the "Spiral" data set and the regression vectors are ordered according to their inherent order. The red squares denote a simulated power of 0.05 ($= \alpha$) or less which should occur only when $\boldsymbol{\theta}$ is zero (H_0).

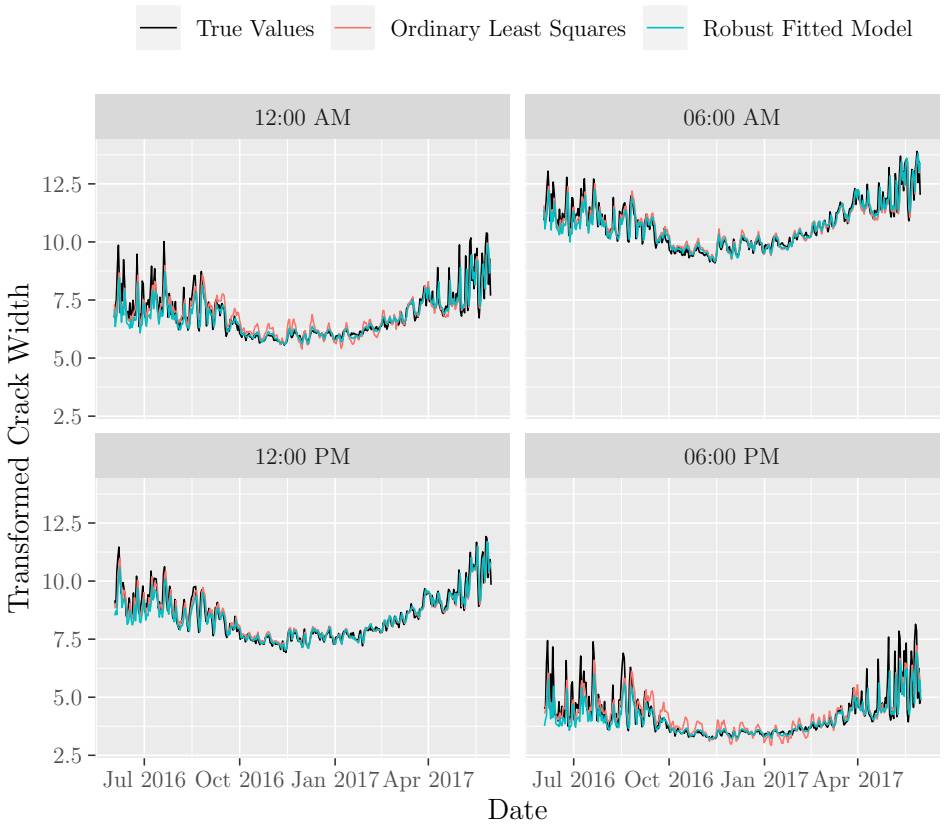


Figure C.3: Four of the overall 96 fitted models on the monitoring data for the times 12am, 6am, 12pm and 6pm.

Bibliography

Abbas, S., Fried, R., Heinrich, J., Horn, M., Jakubzik, M., Kohlenbach, J., Maurer, R., Michels, A., and Müller, C. H. (2019). "Detection of Anomalous Sequences in Crack Data of a Bridge Monitoring". In: *Applications in Statistical Computing: From Music Data Analysis to Industrial Quality Improvement*. Springer International Publishing, pp. 251–269.

Adler, D. and Murdoch, D. (2020). *rgl: 3D Visualization Using OpenGL*. R package version 0.100.54. URL: `https://CRAN.R-project.org/package=rgl`.

Agostinelli, C. (2018). "Local half-region depth for functional data". In: *Journal of Multivariate Analysis* 163, pp. 67–79.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2001). "TSP Cuts Which Do Not Conform to the Template Paradigm". In: *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 261–303.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2004). *Concorde TSP Solver*. URL: `www.tsp.gatech.edu`.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2006). *The Traveling Salesman Problem*. Applied Mathematics. Princton University Press.

Applegate, D., Cook, W., Dash, S., and Mevenkamp, M. (2003). *QSopt Linear Programming Solver*. URL: `http://www.math.uwaterloo.ca/~bico/qsopt/index.html`.

Auguie, B. (2017). *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R package version 2.3. URL: `https://CRAN.R-project.org/package=gridExtra`.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). "The Quickhull Algorithm for Convex Hulls". In: *ACM Trans. Math. Softw.* 22.4, pp. 469–483.

Beaton, A. E. and Tukey, J. W. (1974). "The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data". In: *Technometrics* 16.2, pp. 147–185.

Bischl, B., Lang, M., Bossek, J., Horn, D., Richter, J., and Surmann, D. (2017). *BBmisc: Miscellaneous Helper Functions for B. Bischl*. R package version 1.11. URL: `https://CRAN.R-project.org/package=BBmisc`.

Blum, M., Floyd, R. W., Pratt, V. R., Rivest, R. L., and Tarjan, R. E. (1973). "Time Bounds for Selection". In: *JCSS* 7, pp. 448–461.

Büning, H. (1991). *Robuste und adaptive Tests*. Berlin: de Gruyter.

Carnell, R. (2019). *lhs: Latin Hypercube Samples*. R package version 1.0.1. URL: `https://CRAN.R-project.org/package=lhs`.

Claeskens, G., Hubert, M., Slaets, L., and Vakili, K. (2014). "Multivariate functional halfspace depth". In: *Journal of the American Statistical Association* 109.505, pp. 411–423.

Cuesta-Albertos, J. and Nieto-Reyes, A. (2008). "The random Tukey depth". In: *Computational Statistics & Data Analysis* 52.11, pp. 4979–4988.

Dahl, D. B., Scott, D., Roosen, C., Magnusson, A., and Swinton, J. (2019). *xtable: Export Tables to LaTeX or HTML*. R package version 1.8-4. URL: `https://CRAN.R-project.org/package=xtable`.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). "Solution of a large-scale traveling salesman problem". In: *Operations Research* 2.4, pp. 393–410.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II". In: *Parallel Problem Solving from Nature PPSN VI*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 849–858.

Dodge, Y. (2008). "Least Absolute Deviation Regression". In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, pp. 299–302.

Durstenfeld, R. (1964). "Algorithm 235: Random Permutation". In: *Commun. ACM* 7.7, pp. 420–421.

Eddelbuettel, D. and Balamuta, J. J. (2017). "Extending `R` with `C++`: A Brief Introduction to `Rcpp`". In: *PeerJ Preprints* 5, e3188v1.

Emmerich, M. T. and Deutz, A. H. (2018). "A tutorial on multiobjective optimization: fundamentals, and evolutionary methods". In: *Natural Computing* 17, pp. 585–609.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2003). *Bayesian Data Analysis, Second Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

Gower, J. C. (1971). "A General Coefficient of Similarity and Some of Its Properties". In: *Biometrics* 27.4, pp. 857–871.

Habel, K., Grasman, R., Gramacy, R. B., Mozharovskyi, P., and Sterratt, D. C. (2019). *geometry: Mesh Generation and Surface Tessellation*. R package version 0.4.5. URL: `https://CRAN.R-project.org/package=geometry`.

Hahsler, M. and Hornik, K. (2019). *TSP: Traveling Salesperson Problem (TSP)*. R package version 1.1-7. URL: `https://CRAN.R-project.org/package=TSP`.

Hampel, F., Ronchetti, E., Rousseeuw, P., and Stahel, W. (1986). *Robust Statistics: The Approach Based on Influence Functions*. New York: Wiley.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer.

Held, M. and Karp, R. (1962). "A Dynamic Programming Approach to Sequencing Problems". In: *Journal of the Society for Industrial and Applied Mathematics* 10.1, pp. 196–210.

Horn, M. (2021). *GSignTest: Robust Tests for Regression-Parameters via Sign Depth*. R package version 1.0.8. URL: `https://github.com/melaniehorn/GSignTest`.

Hu, Y., Wang, Y., Wu, Y., Li, Q., and Hou, C. (2011). "Generalized mahalanobis depth in the reproducing kernel Hilbert space." In: *Statistical Papers* 52.3, pp. 511–522.

Huber, P. J. (1964). "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1, pp. 73–101.

Knuth, D. E. (1976). "Big Omicron and big Omega and big Theta". In: *SIGACT News*, pp. 18–24.

Knuth, D. E. (1998). *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.

Koller, M. and Stahel, W. A. (2011). "Sharpening Wald-type inference in robust regression for small samples". In: *Computational Statistics & Data Analysis* 55.8, pp. 2504–2515.

Kustosz, C., Leucht, A., and Müller, C. H. (2016a). "Tests based on simplicial depth for AR(1) models with explosion". In: *Journal of Time Series Analysis* 37.6, pp. 763–784.

Kustosz, C., Müller, C. H., and Wendler, M. (2016b). "Simplified simplicial depth for regression and autoregressive growth processes". In: *Journal of Statistical Planning and Inference* 173, pp. 125–146.

Lang, M. (2017). "checkmate: Fast Argument Checks for Defensive R Programming". In: *The R Journal* 9.1, pp. 437–445.

Lang, M., Bischl, B., and Surmann, D. (2017). "batchtools: Tools for R to work on batch systems". In: *The Journal of Open Source Software* 2.10, p. 135.

Lange, K. L., Little, R. J. A., and Taylor, J. M. G. (1989). "Robust Statistical Modeling Using the t Distribution". In: *Journal of the American Statistical Association* 84.408, pp. 881–896.

Laporte, G. (1992). "The Traveling Salesman Problem: An overview of exact and approximate algorithms". In: *European Journal of Operational Research* 59, pp. 231–247.

Leckey, K., Malcherczyk, D., and Müller, C. H. (2020). *Powerful generalized sign tests based on sign depth*. Preprint available at `https://eldorado.tu-dortmund.de/bitstream/2003/39099/1/DP_1220_SFB823_Leckey_Malcherczyk_M%c3%bcller.pdf`.

Leisch, F. and Dimitriadou, E. (2010). *mlbench: Machine Learning Benchmark Problems*. R package version 2.1-1.

Liu, R. (1990). "On a notion of data depth based on random simplices". In: *Annals of Statistics* 18, pp. 405–414.

López-Pintado, S. and Romo, J. (2009). "On the concept of depth for functional data." In: *Journal of the American Statistical Association* 104.486, pp. 718–734.

Maechler, M., Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Verbeke, T., Koller, M., Conceicao, E. L. T., and Anna di Palma, M. (2019a). *robustbase: Basic Robust Statistics*. R package version 0.93-5. URL: `http://robustbase.r-forge.r-project.org/`.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2019b). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.1.0.

Malcherczyk, D. (2021+). *K-sign depth: Asymptotic distribution, efficient computation and applications*. Dissertation in preparation.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Maronna, R., Martin, D., and Yohai, V. (2006). *Robust Statistics: Theory and Methods*. John Wiley & Sons.

Mersmann, O. (2019). *microbenchmark: Accurate Timing Functions*. R package version 1.4-7. URL: `https://CRAN.R-project.org/package=microbenchmark`.

Mosler, K. (2002). *Multivariate dispersion, central regions and depth. The lift zonoid approach*. Vol. 165. New York, NY: Springer, p. 291.

Müller, C. H. (2005). "Depth estimators and tests based on the likelihood principle with application to regression". In: *Journal of Multivariate Analysis* 95, pp. 153–181.

Pokotylo, O., Mozharovskyi, P., and Dyckerhoff, R. (2019). "Depth and Depth-Based Classification with R Package ddalpha". In: *Journal of Statistical Software* 91.5, pp. 1–46.

Preparata, F. P. and Shamos, M. I. (1990). *Computational geometry*. Texts and monographs in computer science. New York: Springer.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: `https://www.R-project.org/`.

Rego, C., Gamboa, D., Glover, F., and Osterman, C. (2011). "Traveling salesman problem heuristics: Leading methods, implementations and latest advances". In: *European Journal of Operational Research* 211.3, pp. 427–441.

Rockafellar, R. T. (1970). *Convex analysis*. Vol. 28. Princeton mathematical series. Princton University Press.

Rosenkrantz, D., Stearns, R. E., and II, P. M. L. (1977). "An Analysis of Several Heuristics for the Traveling Salesman Problem". In: *SIAM Journal on Computing* 6.3, pp. 563–581.

Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust Regression and Outlier Detection*. USA: John Wiley & Sons, Inc.

Rousseeuw, P. and Yohai, V. (1984). "Robust Regression by Means of S-Estimators". In: *Robust and Nonlinear Time Series Analysis*. New York, NY: Springer US, pp. 256–272.

Rousseeuw, P. and Driessen, K. (1999). "A Fast Algorithm for the Minimum Covariance Determinant Estimator". In: *Technometrics* 41, pp. 212–223.

Rousseeuw, P. and Hubert, M. (1999). "Regression Depth". In: *Journal of the American Statistical Association* 94, pp. 388–402.

Segaert, P., Hubert, M., Rousseeuw, P., and Raymaekers, J. (2020). *mrfDepth: Depth Measures in Multivariate, Regression and Functional Settings*. R package version 1.0.12. URL: `https://CRAN.R-project.org/package=mrfDepth`.

Sharpsteen, C. and Bracken, C. (2019). *tikzDevice: R Graphics Output in LaTeX Format*. R package version 0.12.3. URL: `https://CRAN.R-project.org/package=tikzDevice`.

Stein, M. (1987). "Large Sample Properties of Simulations Using Latin Hypercube Sampling". In: *Technometrics* 29, pp. 143–151.

Struyf, A. J. and Rousseeuw, P. J. (1999). "Halfspace Depth and Regression Depth Characterize the Empirical Distribution". In: *Journal of Multivariate Analysis* 69.1, pp. 135–153.

TIBCO Software Inc. (2010). *TIBCO Spotfire S+ 8.2 Robust Library User's Guide*. URL: `https://www.msi.co.jp/splus/download/pdf/robust.pdf`.

Tsou, C.-S. (2013). *nsga2R: Elitist Non-dominated Sorting Genetic Algorithm based on R*. R package version 1.0. URL: `https://CRAN.R-project.org/package=nsga2R`.

Tukey, J. W. (1975). "Mathematics and the picturing of data". In: *Proceedings of the International Congress of Mathematicians (Vancouver, BC, 1974), Volume 2*. Montréal, Québec, Canada: Canadian Mathematical Congress, pp. 523–531.

Van Aelst, S., Rousseeuw, P. J., Hubert, M., and Struyf, A. (2002). "The Deepest Regression Method". In: *Journal of Multivariate Analysis* 81.1, pp. 138–166.

Wang, J., Zamar, R., Marazzi, A., Yohai, V., Salibian-Barrera, M., Maronna, R., Zivot, E., Rocke, D., Martin, D., Maechler, M., and Konis., K. (2019). *robust: Port of the S+ "Robust Library"*. R package version 0.4-18.2. URL: `https://CRAN.R-project.org/package=robust`.

Wickham, H. (2011). "testthat: Get Started with Testing". In: *The R Journal* 3, pp. 5–10.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H., Danenberg, P., Csárdi, G., and Eugster, M. (2019a). *roxygen2: In-Line Documentation for R*. R package version 7.0.2. URL: `https://CRAN.R-project.org/package=roxygen2`.

Wickham, H., Hester, J., and Chang, W. (2019b). *devtools: Tools to Make Developing R Packages Easier*. R package version 2.2.1. URL: `https://CRAN.R-project.org/package=devtools`.

Wickham, H. and Seidel, D. (2019). *scales: Scale Functions for Visualization*. R package version 1.1.0. URL: https://CRAN.R-project.org/package=scales.

Wilke, C. O. (2019). *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'*. R package version 1.0.0. URL: https://CRAN.R-project.org/package=cowplot.

Yohai, V. J. and Zamar, R. H. (1998). "Optimal Locally Robust M-estimates of Regression". In: *Journal of Statistical Planning and Inference* 64, pp. 309–323.

Yohai, V. J., Stahel, W. A., and Zamar, R. H. (1991). "A Procedure for Robust Estimation and Inference in Linear Regression". In: *Directions in Robust Statistics and Diagnostics*. New York, NY: Springer New York, pp. 365–374.

Yohai, V. (1987). "High Breakdown-Point and High Efficiency Robust Estimates for Regression". In: *The Annals of Statistics* 15.2, pp. 642–656.

Declaration on Oath

Horn, Melanie	147305
Surname, First name	Matriculation ID

I hereby declare that this dissertation on

Sign Depth for Parameter Tests in Multiple Regression

is solely my original work. I have used only the sources and materials indicated and have not received any unauthorized assistance from others. All quotations from other works as well as paraphrases or summaries of other works have been identified as such and properly acknowledged in the dissertation.

This dissertation or parts thereof have not been submitted to an educational institution in Germany or abroad as part of an examination or degree qualification.

I hereby certify that the information provided in this declaration is true and correct.

I fully understand the meaning of this declaration on oath as well as the criminal penalties for submitting a false or incomplete statement.

I hereby confirm that to the best of my knowledge the above statements are true, correct and complete.

Dortmund, March 23, 2021	M. Horn
Location, Date	Signature