PRO GRADU -THESIS

Matti Rytkönen

On multivariate regression using spatial signs and ranks

UNIVERSITY OF TAMPERE
School of Information Sciences
Statistics

February 2015

University of Tampere

School of Information Sciences

RYTKÖNEN, MATTI: On multivariate regression using spatial signs and ranks

Pro gradu -thesis, 39 p., 12 appendix p.

Statistics

February 2015

Abstract

Regular least squares regression can fail when using data with non-normally distributed residuals. This thesis examines multivariate regression methods using spatial signs and ranks as an alternative to least squares regression.

The computational aspects of spatial sign and rank regression were considered. The methods rely on an iterative algorithm, which can fail in certain conditions. Some options to prevent this are tested. Additionally, the algorithm can take a considerable amount of time to calculate, especially in the case of spatial ranks. A faster implementation using the C++ programming language is presented and compared to the original functions available in R.

Then the performance of those methods using a finite sample was compared to asymptotic results in a simulation study. The different methods were used in two different testing problems. In testing problem one if the whole matrix of explaining variables has no effect, and in testing problem two for a split model. When using residuals from the normal distribution, least squares regression was found to be more effective in detecting if the regression co-efficients were different from zero. However, with t-distributed residuals, spatial sign and rank methods appear to be more useful.

Keywords: multivariate linear regression, spatial median, simulation study, power

Contents

Notation and Symbols

$\mathbf{v}$ A vector
$\mathbf{V}$ A matrix
N_p p-variate normal distribution
$t_{p,\nu}$ p-variate t-distribution with ν degrees of freedom
χ^2_k chi-squared distribution with k degrees of freedom
$\mathbf{Y}_{n\times p} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)'$ a $n \times p$ matrix of response variables
$\mathbf{X}_{n\times q} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)'$ a $n \times q$ matrix of explaining variables
$\mathbf{T}(\mathbf{y})$ General score function
$\mathbf{L}(\mathbf{y})$ Optimal location score function
$\mathbf{U}(\mathbf{y})$ Spatial sign function
$\mathbf{R}(\mathbf{y})$ Spatial rank function
$\mathbf{Q}(\mathbf{y})$ Spatial signed-rank function
ave Average
cov Covariance matrix
tr Trace
$||\cdot||$ Euclidean norm
vec Column-wise vectorization of matrix
$\otimes$ Kronecker product

1 Introduction

Regression is based on the idea that the values of one or more variables, which are called response variables, are a function of the values of certain explaining variables. The relationship between the variables can have different forms, for example linear.

A regression model is a way of expressing the values of the response variables through the use of the explaining variables and regression coefficients. Since the coefficients are not known to begin with, they must be estimated by some means.

There are several different methods of calculating regression estimates. A very well known method of fitting is least squares regression, where the objective is to minimize the squared sum of the residuals.

Least squares is one of the most popular methods in use. However, this method can have problems, since it has many assumptions. For example, if the residuals are not normally distributed, the least squares estimate is not the maximum likelihood estimator. In addition, it is highly affected by outliers in the data. As a result other, more robust methods have been suggested.

Sign- and rank-based methods produce estimates which can achieve robustness. In this thesis we will consider L_1-regression methods for multivariate regression and show results of simulation studies comparing different approaches.

In Chapter 2 we will present spatial signs and ranks. The theory behind L_1-regression is presented in Chapter 3 and the two testing problems are detailed in Chapter 4. In Chapter 5 we will consider the computational aspects of the methods, and in Chapter 6 we will present the findings of the conducted simulation studies. Finally in Chapter 7 we will present the conclusions from the thesis.

2 Spatial Signs and Ranks

The results of this chapter are based on Oja (2010, Chapter 4).

In the univariate case, signs and ranks are based on ordering the data. Let us consider an univariate data $\mathbf{Y} = y_1, \ldots, y_n$. The univariate sign function is

$$U(y) = \begin{cases} +1, & \text{if y} > 0 \\ 0, & \text{if y} = 0 \\ -1, & \text{if y} < 0 \end{cases}$$

The sign of the observation y_i is then $U(y_i)$. Similarly, the centered rank is then $\mathbf{ave}_j\{U(y_i - y_j)\}$.

In the multivariate case there is no simple way to order the data points. However, with L_1 functions it is possible to use the idea of signs and ranks in a multivariate set-up. Let $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)'$ be an $n \times p$ data matrix with n observations and p variables. Then the multivariate spatial sign, the multivariate spatial rank and the multivariate spatial signed-rank functions $\mathbf{U}(\mathbf{y})$, $\mathbf{R}(\mathbf{y})$ and $\mathbf{Q}(\mathbf{y})$ may be defined as

$$\begin{aligned} \mathbf{U}(\mathbf{y}) &= \begin{cases} ||\mathbf{y}||^{-1}\mathbf{y}, & \text{if } \mathbf{y} \neq \mathbf{0} \\ \mathbf{0}, & \text{if } \mathbf{y} = \mathbf{0} \end{cases}, \\ \mathbf{R}_{\mathbf{Y}}(\mathbf{y}) = \mathbf{R}(\mathbf{y}) &= \mathbf{ave}\{\mathbf{U}(\mathbf{y} - \mathbf{y}_i)\}, \text{ and} \\ \mathbf{Q}(\mathbf{y}) &= \frac{1}{2}[\mathbf{R}_{\mathbf{Y}}(\mathbf{y}) + \mathbf{R}_{-\mathbf{Y}}(\mathbf{y})]. \end{aligned}$$

Where $||\cdot||$ is the Euclidean norm and $\mathbf{R}_{-\mathbf{Y}}(\mathbf{y}) = -\mathbf{R}_{\mathbf{Y}}(-\mathbf{y})$. In practical applications, we can call these functions score functions $\mathbf{T}(\mathbf{y})$ which give us individual scores $\mathbf{T}_i = \mathbf{T}(\mathbf{y}_i), i = 1, \ldots, n$. Therefore we have the spatial sign score function $\mathbf{U}(\mathbf{y})$, the spatial rank score function $\mathbf{R}(\mathbf{y})$ and the spatial signed-rank score function $\mathbf{Q}(\mathbf{y})$. In addition we can define the identity score function $\mathbf{T}(\mathbf{y}) = \mathbf{y}$.

In testing we will first transform the original values

$$\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)' \rightarrow \mathbf{T} = (\mathbf{T}_1, \ldots, \mathbf{T}_n)' = (\mathbf{T}(\mathbf{y}_1), \ldots, \mathbf{T}(\mathbf{y}_n))'.$$

Generally it is useful to center and standardize the scores for testing and estimation. This can be achieved through the following ways.

Outer centering of the scores:

$$\mathbf{T}_i \to \hat{\mathbf{T}}_i = \mathbf{T}_i - \bar{\mathbf{T}}.$$

Outer standardization of the scores:

$$\mathbf{T}_i \to \hat{\mathbf{T}}_i = \mathbf{cov}(\mathbf{T})^{-1/2}\mathbf{T}_i.$$

Outer centering and standardization of the scores:

$$\mathbf{T}_i \to \hat{\mathbf{T}}_i = \mathbf{cov}(\mathbf{T})^{-1/2}(\mathbf{T}_i - \bar{\mathbf{T}}).$$

However, using outer standardization does not guarantee that the test using these scores will be affine invariant. This can be achieved by the following method.

Inner centering of the scores: Find a vector $\mathbf{M}$ so that if $\hat{\mathbf{T}}_i = \mathbf{T}(\mathbf{y}_i - \mathbf{M})$, then

$$\mathbf{ave}\{\hat{\mathbf{T}}_i\} = 0.$$

Then transform

$$\mathbf{T}_i \to \hat{\mathbf{T}}_i = \mathbf{T}(\mathbf{y}_i - \mathbf{M}).$$

Inner standardization of the scores: Find a matrix $\mathbf{S}^{-1/2}$ so that if $\hat{\mathbf{T}}_i = \mathbf{T}(\mathbf{S}^{-1/2}\mathbf{y}_i)$, then

$$p \cdot \mathbf{ave}\{\hat{\mathbf{T}}_i\hat{\mathbf{T}}_i'\} = \mathbf{ave}\{\hat{\mathbf{T}}_i'\hat{\mathbf{T}}_i\}\mathbf{I}_p.$$

After that transform

$$\mathbf{T}_i \to \hat{\mathbf{T}}_i = \mathbf{T}(\mathbf{S}^{-1/2}\mathbf{y}_i).$$

Inner centering and standardization of the scores: Find a vector $\mathbf{M}$ and a matrix $\mathbf{S}^{-1/2}$ so that if $\hat{\mathbf{T}}_i = \mathbf{T}(\mathbf{S}^{-1/2}(\mathbf{y}_i - \mathbf{M}))$, then

$$\mathbf{ave}\{\hat{\mathbf{T}}_i\} = 0, \text{ and}$$
$$p \cdot \mathbf{ave}\{\hat{\mathbf{T}}_i\hat{\mathbf{T}}_i'\} = \mathbf{ave}\{\hat{\mathbf{T}}_i'\hat{\mathbf{T}}_i\}\mathbf{I}_p.$$

Finally transform

$$\mathbf{T}_i \to \hat{\mathbf{T}}_i = \mathbf{T}(\mathbf{S}^{-1/2}(\mathbf{y}_i - \mathbf{M})).$$

Here $\mathbf{M} = \mathbf{M}(\mathbf{Y})$ is a location statistic and $\mathbf{S} = \mathbf{S}(\mathbf{Y})$ is the scatter statistic of the score function $\mathbf{T}(\mathbf{y})$ used. The matrix $\mathbf{S}^{-1/2}$ is also assumed to be symmetric.

A two-dimensional visualization of the different spatial scores is presented in Figure 2.1 for a randomly generated data. The spatial signs are found on the unit circle, while the spatial ranks and signed ranks indicate the distance of the data point from the center as well as the direction.

There are several location and shape estimators related to the different spatial scores. For outer standardized spatial signs, the corresponding estimate is the L_1- or spatial median. It is a popular robust estimate of multivariate location.

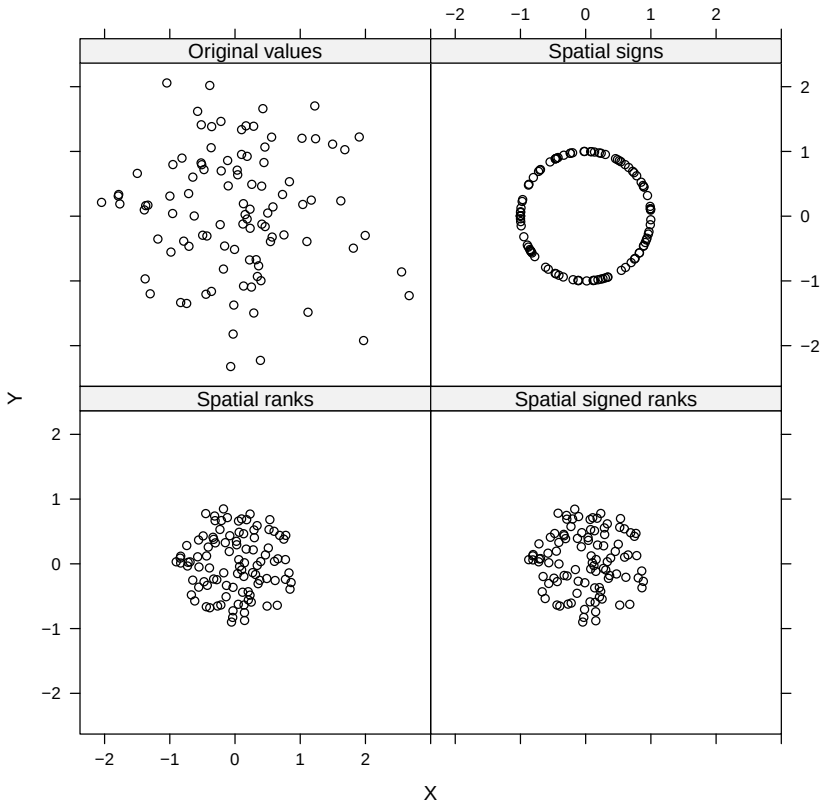

Figure 2.1. Example of spatial scores for a randomly generated two-dimensional data.

The spatial median can be thought of as the center point of the data structure, with the lowest total distance to each point. For a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}'$ the spatial median $\hat{\boldsymbol{\mu}}$ is defined as

$$\hat{\boldsymbol{\mu}}(\mathbf{X}) = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{i=1}^{n} ||\mathbf{x}_i - \boldsymbol{\mu}||$$

If the data points are not all on the same line, the solution is unique (Milasevic & Ducharme 1987).

Several algorithms to solve the spatial median have been proposed. One of these is the Weiszfeld algorithm, where the spatial median $\boldsymbol{\mu}$ can be found using the iteration

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \frac{\mathbf{ave}\{U(\mathbf{y}_i - \boldsymbol{\mu}\}}{\mathbf{ave}\{||\mathbf{y}_i - \boldsymbol{\mu}||^{-1}\}}$$

which converges to the solution.

The spatial median is not affine equivariant, but it is location and orthogonally equivariant. An affine equivariant median can be found by simultaneously estimating Taylor's shape matrix, see Hettmansperger & Randles (2002). The

obtained location and scatter estimate is called the Hettmansperger-Randles (HR) estimate.

The Weiszfeld algorithm does not always converge to the spatial median. It is easy to see from the equation that if the algorithm lands on a data point, it gets stuck even if that is not the solution. Vardi & Zhang (1999) have presented a modified Weiszfeld algorithm that will always converge to the correct solution by assigning weights in the event the algorithm lands on a data point.

There are also several other algorithms available for computing the spatial median. For a comparison of these, see Fritz, Filzmoser & Croux (2012).

For spatial ranks, the corresponding estimate is the spatial Hodges-Lehmann estimate. See Hodges & Lehmann (1963) and Lehmann (2006).

For more on the tests and estimates based on multivariate sign and rank methods, see for example Choi & Marden (1997) and Möttönen & Oja (1995).

3 L1-Regression

The results of this chapter are based on (Oja 2010, Chapter 13).

3.1 General assumptions

Let us consider the linear regression model

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{Y}$ is an $n \times p$ matrix of n observed values of q response variables. $\mathbf{X}$ is an $n \times q$ matrix consisting of an intercept and $q-1$ explaining variables, $\boldsymbol{\beta}$ an $q \times p$ matrix of regression coefficients and $\boldsymbol{\epsilon}$ and $n \times p$ matrix of residuals.

We assume that $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n)'$ is a random sample from a p-variate distribution, and that it is "centered" at the origin. We will also assume in this chapter the following.

Assumption 1.

$$\frac{1}{n}\mathbf{X}'\mathbf{X} \to \mathbf{D} \quad \text{and} \quad \frac{\max_{1\leq i\leq n}\{\mathbf{x}_i'\mathbf{C}'\mathbf{C}\mathbf{x}_i\}}{\sum_{i=1}^{n}\{\mathbf{x}_i'\mathbf{C}'\mathbf{C}\mathbf{x}_i\}} \to 0$$

for some positive definite $q \times q$ matrix $\mathbf{D}$ and for all $p \times q$ matrices $\mathbf{C}$ with positive rank.

In general, we wish to estimate the unknown matrix $\boldsymbol{\beta}$. The estimate based on a score function $\mathbf{T}$ will often solve the equation

$$\mathbf{T}(\hat{\boldsymbol{\beta}}')\mathbf{X} = \mathbf{0}.$$

Of particular interest are two testing problems.

1. Testing the hypothesis that $\boldsymbol{\beta} = \mathbf{0}$.
2. In a model $\mathbf{Y} = \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \boldsymbol{\epsilon}$, testing the hypothesis that $\boldsymbol{\beta}_2 = \mathbf{0}$.

In testing problem one, our hypothesis is that the explaining variables have no effect on the response variables. Additionally we test if the mean of the response variables is zero since the intercept is included in $\mathbf{X}$.

In testing problem two, our linear model is formulated so that $\mathbf{X} = (\mathbf{X}_1\, \mathbf{X}_2)$ and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1\, \boldsymbol{\beta}_2)$. Our hypothesis is that the explaining variables in $\mathbf{X}_2$ have no effect on the response variables.

3.2 Multivariate linear L_2 regression

We can use the identical score function $\mathbf{T}(\mathbf{y} = \mathbf{y})$ to estimate $\boldsymbol{\beta}$. We will thus assume that $\boldsymbol{\epsilon}$ is a random sample from a distribution with

$$\mathbf{E}(\boldsymbol{\epsilon}_i) = \mathbf{0} \quad \text{and} \quad \mathbf{cov}(\boldsymbol{\epsilon}_i) = \boldsymbol{\Sigma}.$$

We therefore assume that second moments exist and that $\mathbf{X}'\mathbf{X}$ is a full-rank matrix. We also assume that the explaining variables are fixed and satisfy Assumption 1.

In L_2 estimation, the estimate $\hat{\boldsymbol{\beta}}$ solves the equation

$$(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}})'\mathbf{X} = \mathbf{0}.$$

If $\mathbf{X}$ has full rank, then the solution is simply

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y},$$

which is called the least squares (LS) estimate. This estimate has the following equivariance properties.

1. Regression equivariance:

   $\hat{\boldsymbol{\beta}}(\mathbf{X}, \mathbf{XH} + \mathbf{Y}) = \hat{\boldsymbol{\beta}}(\mathbf{X}, \mathbf{Y}) + \mathbf{H}$ for all full-rank matrices $\mathbf{H}_{q\times p}$.

2. $\mathbf{Y}$ equivariance:

   $\hat{\boldsymbol{\beta}}(\mathbf{X}, \mathbf{YW}) = \hat{\boldsymbol{\beta}}(\mathbf{X}, \mathbf{Y})\mathbf{W}$ for all full-rank matrices $\mathbf{W}_{p\times p}$.

3. $\mathbf{X}$ equivariance:

   $\hat{\boldsymbol{\beta}}(\mathbf{XV}, \mathbf{Y}) = \mathbf{V}^{-1}\hat{\boldsymbol{\beta}}(\mathbf{X}, \mathbf{Y})$ for all full-rank matrices $\mathbf{V}_{q\times q}$.

3.3 L_1 regression based on spatial signs

Next we will assume in our linear regression model

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

that $\boldsymbol{\epsilon}$ is a random sample with the spatial median at the origin. In other words,

$$\mathbf{E}(\mathbf{U}(\boldsymbol{\epsilon}_i)) = \mathbf{0}.$$

We will again assume that $\mathbf{X}'\mathbf{X}$ is a full-rank matrix and that is satisfies Assumption 1.

Using L_1 estimation based on the spatial sign score, the estimate $\hat{\boldsymbol{\beta}}$ then often solves

$$\mathbf{U}(\hat{\boldsymbol{\beta}})'\mathbf{X} = \mathbf{0},$$

where

$$\mathbf{U}_i(\boldsymbol{\beta}) = \mathbf{U}(\mathbf{y}_i - \boldsymbol{\beta}'\mathbf{x}_i),\ i = 1, \ldots, n \quad \text{and} \quad \mathbf{U}(\boldsymbol{\beta}) = (\mathbf{U}_1(\boldsymbol{\beta}), \ldots, \mathbf{U}_n(\boldsymbol{\beta}))'.$$

The estimate is also called the least absolute deviation (LAD) estimate.

The solution does not have a closed form but may be calculated using the following two-step iterative algorithm.

1. $\mathbf{e}_i \leftarrow \mathbf{y}_i - \boldsymbol{\beta}'\mathbf{x}_i, \quad i = 1, \ldots, n.$
2. $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + [\mathbf{ave}\{|\mathbf{e}_i|^{-1}\mathbf{x}_i\mathbf{x}_i'\}]^{-1}\mathbf{ave}\{\mathbf{x}_i\mathbf{U}(\mathbf{e}_i)'\}.$

The estimate is regression and $\mathbf{X}$ equivariant, but not $\mathbf{Y}$ equivariant. A fully equivariant estimate can be achieved using the following transformation retransformation technique (Oja 2010, p. 191). We will use an algorithm with three steps, first updating the residuals, then $\boldsymbol{\beta}$ and finally the residual scatter matrix $\mathbf{S}$.

1. $\mathbf{e}_i \leftarrow \mathbf{S}^{-1/2}(\mathbf{y}_i - \boldsymbol{\beta}'\mathbf{x}_i), \quad i = 1, \ldots, n.$
2. $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + [\mathbf{ave}\{||\mathbf{e}_i||^{-1}\mathbf{x}_i\mathbf{x}_i'\}]^{-1}\mathbf{ave}\{\mathbf{x}_i\mathbf{U}(\mathbf{e}_i)'\}\mathbf{S}^{1/2}.$
3. $\mathbf{S} \leftarrow p\mathbf{S}^{1/2}\mathbf{ave}\{\mathbf{U}(\mathbf{e}_i)\mathbf{U}(\mathbf{e}_i)'\}\mathbf{S}^{1/2}.$

After each iteration t, the difference of the Euclidean norms of the regression coefficients $\boldsymbol{\beta}_t$ and $\boldsymbol{\beta}_{t-1}$ is calculated. The algorithm is determined to have converged when $||\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}|| < \epsilon$, where ϵ is a tolerance limit.

3.4 L_1 regression based on spatial ranks

In this section we will use the spatial rank function $\mathbf{R}(\mathbf{y})$ as the score function. Since spatial ranks are invariant under location shifts, the intercept vector must be estimated separately. We now use the model

$$\mathbf{Y} = \mathbf{1}_n\boldsymbol{\mu}' + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{X}$ is the matrix of actual explaining variables, $\boldsymbol{\mu}$ is the intercept and $\boldsymbol{\epsilon}$ is a random sample from a continuous distribution. The design matrix $(\mathbf{1}_n\,\mathbf{X})$ satisfies Assumption 1, since

$$\frac{1}{n}\mathbf{X}'(\mathbf{I}_n - \mathbf{P}_{\mathbf{1}_n})\mathbf{X} \to \mathbf{D}_0 \quad \text{as} \quad n \to \infty.$$

In the following, we will write

$$\mathbf{y}_{ij} = \mathbf{y}_j - \mathbf{y}_i, \quad \mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i \quad \text{and} \quad \boldsymbol{\epsilon}_{ij} = \boldsymbol{\epsilon}_j - \boldsymbol{\epsilon}_i,$$

for $i, j = 1, \ldots, n$.

The estimate $\hat{\boldsymbol{\beta}}$ for the rank test solves

$$\mathbf{ave}\{\mathbf{U}_{ij}(\boldsymbol{\beta})\mathbf{x}_{ij}'\} = \mathbf{0},$$

where

$$\mathbf{U}_{ij}(\boldsymbol{\beta}) = \mathbf{U}(\mathbf{y}_{ij} - (\boldsymbol{\beta}'\mathbf{x}_{ij})), \quad i, j = 1, \ldots, n.$$

The solution is found similarly to LAD regression, but by replacing response and explaining variables with pairwise differences of response and explaining variables. The two-step iterative algorithm is then as follows.

1. $\mathbf{e}_{ij} \leftarrow \mathbf{y}_{ij} - \boldsymbol{\beta}'\mathbf{x}_{ij}, \quad i \neq j.$
2. $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + [\mathbf{ave}\{||\mathbf{e}_{ij}||^{-1}\mathbf{x}_{ij}\mathbf{x}'_{ij}\}]^{-1}\mathbf{ave}\{\mathbf{x}_{ij}\mathbf{U}(\mathbf{e}_{ij})'\}.$

This estimate can be called a rank-based estimate since the estimating equation can be written as

$$R(\hat{\boldsymbol{\beta}})'\mathbf{X} = \mathbf{X}.$$

Like the LAD estimate, this estimate is regression and $\mathbf{X}$ equivariant, but not $\mathbf{Y}$ equivariant. The transformation re-transformation procedure can again be done in the same fashion as in the previous case (Oja 2010, p. 195). The three-step algorithm will then be as follows.

1. $\mathbf{e}_{ij} \leftarrow \mathbf{S}^{-1/2}(\mathbf{y}_{ij} - \boldsymbol{\beta}'\mathbf{x}_{ij}), \quad i = 1, \ldots, n.$
2. $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + [\mathbf{ave}\{||\mathbf{e}_{ij}||^{-1}\mathbf{x}_{ij}\mathbf{x}'_{ij}\}]^{-1}\mathbf{ave}\{\mathbf{x}_{ij}\mathbf{U}(\mathbf{e}_{ij})'\}\mathbf{S}^{1/2}.$
3. $\mathbf{S} \leftarrow p\mathbf{S}^{1/2}\mathbf{ave}\{\mathbf{U}(\mathbf{e}_{ij})\mathbf{U}(\mathbf{e}_{ij})'\}\mathbf{S}^{1/2}.$

We define convergence similarly to the previous case. The algorithm has converged if after iteration t, $||\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}|| < \epsilon$.

4 Two Testing Problems

4.1 Testing problem one

Let us again consider the linear regression model

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{Y}$ is an $n \times p$ matrix of n observed values of q response variables, $\mathbf{X}$ an $n \times q$ matrix of n observed values of q explaining variables, $\boldsymbol{\beta}$ an $q \times p$ matrix of regression coefficients and $\boldsymbol{\epsilon}$ and $n \times p$ matrix of residuals.

In testing problem one we will consider what happens under the hypotheses

$$H_0 : \boldsymbol{\beta} = \mathbf{0} \text{ vs. } H_n : \boldsymbol{\beta} = n^{-1/2}\boldsymbol{\delta},$$

where $\boldsymbol{\delta}$ is a $q \times p$ matrix which gives the direction of an alternate sequence.

Under the null hypothesis the distribution of our test statistic for the chosen score function $\mathbf{T}(\mathbf{y})$ using outer standardization is (Oja 2010, p. 184)

$$Q^2 = Q^2(\mathbf{X}, \mathbf{Y}) = n \cdot tr((\mathbf{T}'\mathbf{P}_\mathbf{X}\mathbf{T})(\mathbf{T}'\mathbf{T})^{-1} \rightarrow_d \chi^2_{pq}.$$

Under an alternate hypothesis, the distribution of Q^2 is a non-central chi-square distribution with pq degrees of freedom and non-centrality parameter

$$vec(\boldsymbol{\delta}')'(\mathbf{D} \otimes (\mathbf{A}\mathbf{B}^{-1}\mathbf{A}))vec(\boldsymbol{\delta}') = tr(\boldsymbol{\delta}'\mathbf{D}\boldsymbol{\delta})(\mathbf{A}\mathbf{B}^{-1}\mathbf{A})).$$

Here $\mathbf{A} = \mathbf{E}\{\mathbf{T}(\boldsymbol{\epsilon}_i)\mathbf{L}(\boldsymbol{\epsilon}_i)'\}$ and $\mathbf{B} = \mathbf{E}\{\mathbf{T}(\boldsymbol{\epsilon}_i)\mathbf{T}(\boldsymbol{\epsilon}_i)'\}$ where $\mathbf{L}(\mathbf{y})$ is the optimal multivariate score function, *vec* is the column-wise vectorization of a matrix and $\otimes$ is the Kronecker product.

The distribution of the test statistic at the true value of $\boldsymbol{\beta}$ can be approximated by a non-central chi-square distribution with pq degrees of freedom and a non-centrality parameter

$$tr((\boldsymbol{\Delta}'\mathbf{P}_\mathbf{X}\boldsymbol{\Delta})(\mathbf{A}\mathbf{B}^{-1}\mathbf{A})),$$

where $\boldsymbol{\Delta} = \mathbf{X}\boldsymbol{\beta}$.

For identity scores, the non-centrality parameter can further be simplified as

$$tr((\boldsymbol{\delta}'\boldsymbol{\delta})(\boldsymbol{\Sigma}^{-1})),$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the residual term.

Our interest is to find out how the different regression methods perform with varying set-ups. We will set the parameters so that we know what the theoretical results should be, and then compare those with the data gathered from simulation studies.

We will calculate the theoretical powers for L_2 regression under the normal model by using the distribution and quantile functions of the non-central chi-square distribution. The power for any one case is $P(1 - Q(0.95; nc; \nu))$, where nc and ν are the non-centrality parameter and the degrees of freedom, respectively.

For the simulation study in this thesis, we will consider a specific design for $\boldsymbol{\delta}$ which allows us to easily calculate the final regression co-efficients. The structure of $\boldsymbol{\delta}$ is presented in Chapter 6.

4.2 Testing problem two

In the second testing problem we examine in the linear regression model

$$\mathbf{Y} = \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \boldsymbol{\epsilon},$$

where $\mathbf{Y}$ is an $n \times p$ matrix of n observed values of q response variables, $\mathbf{X}_1$ and $\mathbf{X}_2$ are respectively an $n \times q_1$ and an $n \times q_2$ matrices of n observed values of q_1 and q_2 explaining variables, $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are $q_1 \times p$ and $q_2 \times p$ matrices of regression coefficients and $\boldsymbol{\epsilon}$ and $n \times p$ matrix of residuals.

We are interested in the hypotheses

$$H_0 : \boldsymbol{\beta}_2 = \mathbf{0} \text{ vs. } H_n : \boldsymbol{\beta}_2 = n^{-1/2}\boldsymbol{\delta},$$

where $\boldsymbol{\delta}$ is again a $q_2 \times p$ matrix which gives the direction of an alternate sequence. The test statistic is constructed by finding centered scores

$$\hat{\mathbf{T}} = \mathbf{T}(\hat{\boldsymbol{\beta}}_1, \mathbf{0})$$

so that $\hat{\mathbf{T}}'\mathbf{X}_1 = \mathbf{0}$. We will also write $\hat{\mathbf{X}}_2 = (\mathbf{I}_n - \mathbf{P}_{\mathbf{X}_1})\mathbf{X}_2$. The test statistic will then be (Oja 2010, p. 186)

$$Q^2 = Q^2(\mathbf{X}_1, \mathbf{X}_2, \mathbf{Y}) = n \cdot tr((\hat{\mathbf{T}}'\mathbf{P}_{\hat{\mathbf{X}}_2}\hat{\mathbf{T}}(\hat{\mathbf{T}}'\hat{\mathbf{T}})^{-1}) \rightarrow_d \chi^2_{pq_2}.$$

We call this the Score test statistic.

Under an alternate hypothesis, the distribution of the test statistic at the true value of $\boldsymbol{\beta}$ can again approximated by a non-central chi-square distribution with pq_2 degrees of freedom and a non-centrality parameter

$$tr((\hat{\boldsymbol{\Delta}}'\mathbf{P}_{\mathbf{X}_2}\hat{\boldsymbol{\Delta}})(\mathbf{A}\mathbf{B}^{-1}\mathbf{A})),$$

where $\hat{\boldsymbol{\Delta}} = (\mathbf{I}_n - \mathbf{P}_{\mathbf{X}_1})\boldsymbol{\Delta}$. We will use similar methods as in the first testing problem to calculate our regression co-efficients using the non-central chi-square distribution.

For the second testing problem we can also calculate the Wald test statistic,

$$n\, vec(\hat{\boldsymbol{\beta}}_2)' \widehat{\mathbf{cov}(\hat{\boldsymbol{\beta}}_2)}^{-1} vec(\hat{\boldsymbol{\beta}}_2)',$$

where

$$\widehat{\mathbf{cov}(\hat{\boldsymbol{\beta}}_2)} = (\frac{1}{n}\hat{\mathbf{X}}_2'\hat{\mathbf{X}}_2)' \otimes (\hat{\mathbf{A}}^{-1}\hat{\mathbf{B}}\hat{\mathbf{A}}^{-1}),$$

which is asymptotically equivalent with Q^2. These statistics were also compared to each other in the simulation study.

5 Computational Aspects

5.1 Residual method

The modified Weiszfeld algorithm is not usable in the regression case, because the weights cannot be easily defined. Because of that, alternative methods of handling the problem of small residuals need to be considered.

Three different ways of handling the residuals were compared in addition to a general optimizer function. The best method of those was chosen to be used in the next chapter.

The criteria for selecting the best methods were simple. We would primarily test how prone to errors the different methods were. In addition, we want to find the correct answer, see the method converge quickly and be computationally fast.

The Weiszfeld algorithm can fail if the norm of the estimated residuals is too small. In the iterative algorithms presented in Chapter 3, the terms $||\mathbf{e}_i||^{-1}$ and $||\mathbf{e}_{ij}||^{-1}$ are used. If any of the norms are close to zero, this can obviously lead to problems as they are used in the denominator.

We define the limit for the norms to be γ. We tried three different methods to manage cases where the norms of the residuals are too small: Replace, Delete and Ignore.

In the Replace method, we would replace all norms of residuals smaller than γ with γ. Here

$$||\mathbf{e}_i||^* = \begin{cases} \gamma, & \text{if } ||\mathbf{e}_i|| < \gamma \\ ||\mathbf{e}_i||, & \text{otherwise} \end{cases}$$

In regression using spatial ranks the method works similarly for $||\mathbf{e}_{ij}||$.

In the Delete method, we would temporarily delete all observations where $||\mathbf{e}_i|| < \gamma$. We will then use this reduced data-set to calculate the regression co-efficients, and then use the whole data again for the next iteration. Here

$$\mathbf{x}_{i,t} = \begin{cases} \text{removed}, & \text{if } ||\mathbf{e}_{i,t}|| < \gamma \\ \mathbf{x}_i, & \text{otherwise} \end{cases}$$

And likewise for the corresponding $\mathbf{y}_i$ and $\mathbf{e}_i$.

In the Ignore method, we tested what happens if the residuals were unchanged. No transformation would be done to the data regardless of the values of the residuals given by the model.

Using these residual methods three different outcomes for the estimation of regression coefficients exist:

1. The algorithm estimates $\boldsymbol{\beta}$ successfully.
2. The algorithm is interrupted by a computational error.
3. The algorithm fails to convergence within the set iteration limit.

In the latter two cases we do not achieve a reliable estimate for $\boldsymbol{\beta}$ as we do not meet our convergence criteria.

All three methods were tested with the exact same random data-sets to ensure comparability of the results. The parameters for the test were n=100, p=3, q=5, and the maximum amount of iterations allowed to find the result was 1,000. The process was repeated 10,000 times using outer and inner signs estimation to get a representative sample.

Table 5.1. Performance of the different residual methods.

	Outer Signs		Inner Signs	
Method	No convergence	Error	No convergence	Error
Replace	0	0	0	0
Delete	122	0	0	205
Ignore	0	9	6	33

The results for the latter two outcomes are presented in table 5.1. The replacement method always estimated $\boldsymbol{\beta}$ correctly in this test. The other two methods proved to be more unreliable as could not arrive at the solution each time.

The deletion method was found to have a special problem. In addition to errors, it was found that there is a possibility that the function will be stuck in an infinite loop.

In figure 5.1 the convergence of the deletion method over the last 100 iterations is presented for one specific data set. At first the difference $||\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}||$ decreases as expected. Before the convergence criteria is fulfilled however, the difference increases sharply for one iteration before decreasing again. This continues for several cycles until the maximum iteration limit is reached.

The case was retested with a maximum iteration of 10,000 and the function still failed to converge. Thus the deletion method was not considered to be used in further applications.

Ultimately, the replacement method proved the most robust of those tested. It did not fail in any of the cases with the selected sample. Therefore it was chosen to be used in the main simulation study.

5.2 Programming considerations

One implementation of the spatial regression methods used in this thesis are available in the *MNM*-package (Nordhausen & Oja 2011) for the R language

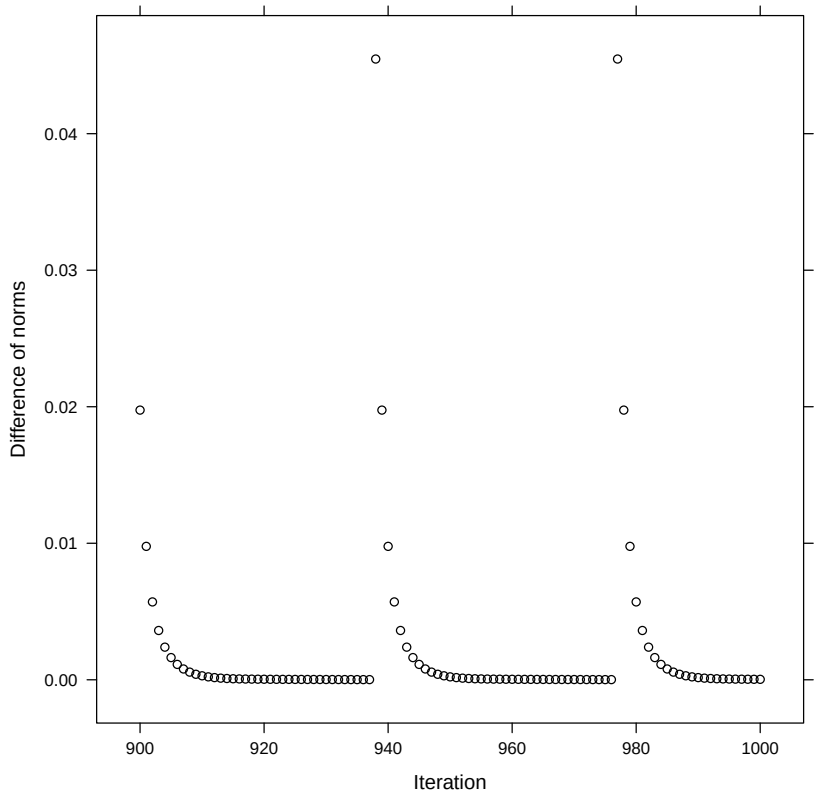

Figure 5.1. Convergence example using the Delete method.

(R Core Team 2012). The functions in the package are written using the R language. However, R is an interpreted programming language. This means that commands are executed directly by the program, in contrast to compiled languages.

For operations with large objects or many iterations this can result in considerably lengthy run times. Since multivariate regression can fill both of these conditions, alternatives to the implementations of the functions in the *MNM*-package were considered.

One compiled language that is easy to use with R is C++ through the *Rcpp*-package (Eddelbuettel & Francois 2011). The package allows transformations of R objects to C++ objects and vice versa. This enables integrating a C++ -function directly within R code. For more literature on *Rcpp*, see also Eddelbuettel (2013).

Base C++ does not contain a matrix class or many of the functions necessary for multivariate regression. However, several linear algebra libraries are available which provide these functionalities.

For this thesis, the open source Armadillo library (Sanderson 2010) was chosen for implementing the code in C++. This has an additional benefit as the R package *RcppArmadillo* (Eddelbuettel & Sanderson 2014) enables the

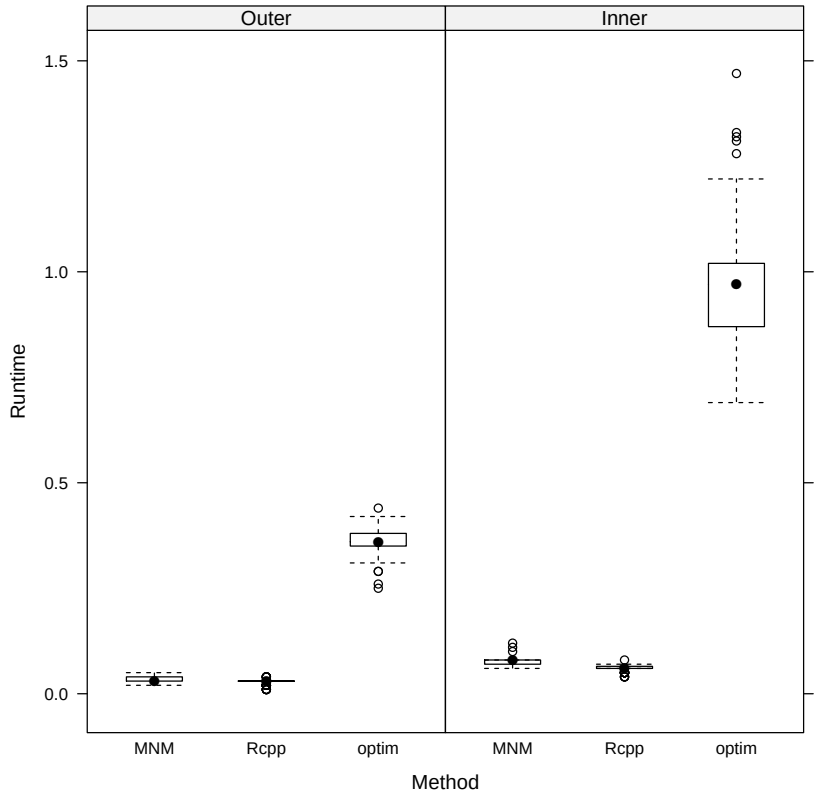

Figure 5.2. Convergence time in seconds for different methods for spatial signs regression with $\mathbf{Y}_{100\times 3}$ and $\mathbf{X}_{100\times 5}$.

use of Armadillo specific objects in R.

Since loops can be especially time-consuming in R, the iterative process of finding regression co-efficients was rewritten in C++. With regression using spatial ranks, calculating the matrices of pairwise differences was also implemented in C++ code, since the dimensions of those matrices can get very large.

In addition to R and C++ based solutions, an optimizer based on the R *optim* function was tested. The solutions were judged based on computation time and accuracy of results.

Figure 5.2 presents the times each implementation took to converge using spatial signs with 10,000 repetitions. It can be clearly seen that *optim* performs very poorly compared to the other functions. However, it should be noted that for *optim* there was no need to consider the case of small residuals.

All of the three methods were found to give practically the same results. In Figure 5.3 the differences in the solutions given by the functions and the true regression co-efficients are shown. Due to its slow convergence speed, and the fact that the results are not significantly different between the methods, an *optim* based solution was rejected.

An additional test of computation speeds was performed between R and

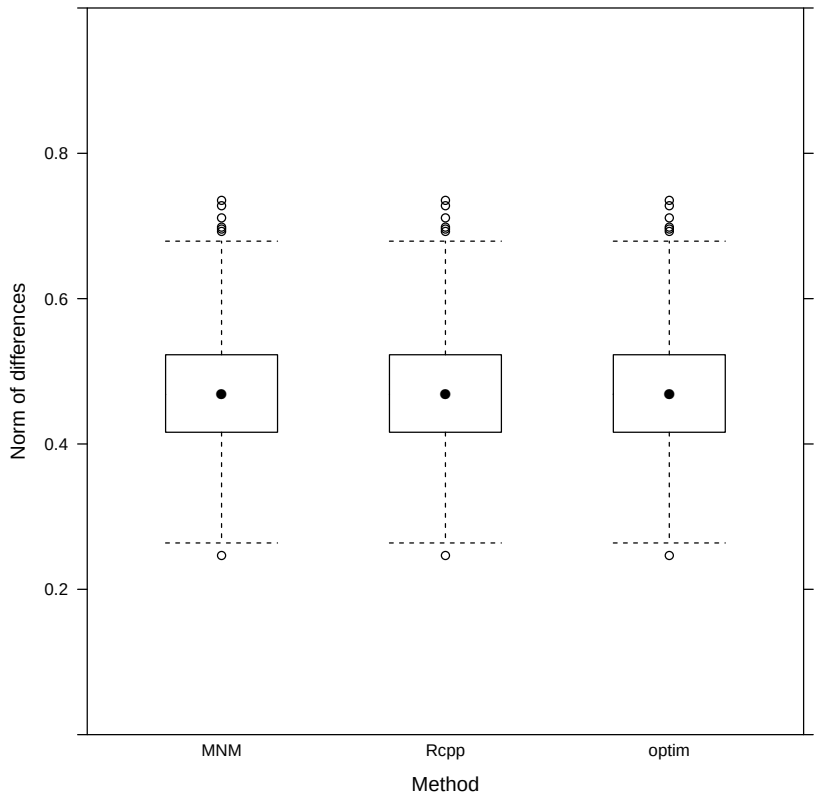

Figure 5.3. Difference of solutions compare to true values of co-efficients in outer spatial signs regression with $\mathbf{Y}_{100\times 3}$ and $\mathbf{X}_{100\times 5}$.

C++ implementations with regression using inner ranks. The results are presented in Figure 5.4. Here it can be clearly seen that the C++ based code computes faster than plain R. The estimated regression co-efficients are practically identical in this case as well.

Therefore a C++ based solution was chosen to be used in the main simulation study, due to increased computation speed compared to the other alternatives that were considered.

The code for the main C++–classes for spatial signs and ranks regression, as well as the *optim* based function for regression based on outer standardized spatial signs are included in Appendices A and B.

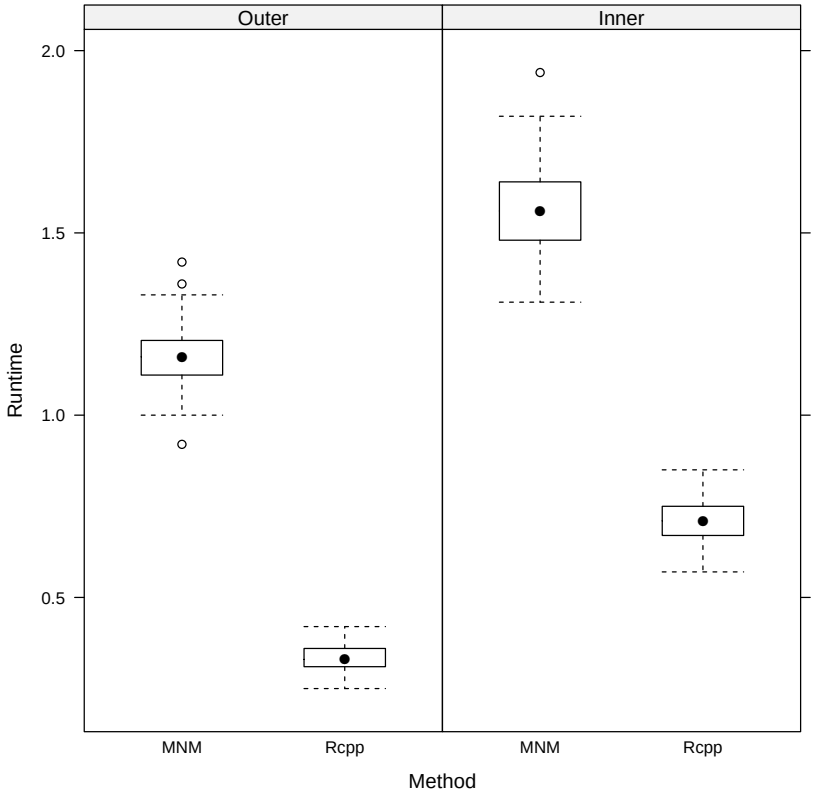

Figure 5.4. Convergence time in seconds for different methods for spatial rank regression with $\mathbf{Y}_{400\times5}$ and $\mathbf{X}_{400\times6}$.

6 Simulations

6.1 Testing problem one

In our simulation, we will consider the case $\boldsymbol{\delta} = d\mathbf{1}_p'\mathbf{1}_q$, where d is a non-negative constant. We will calculate d so that using $\boldsymbol{\delta}$ we obtain theoretical powers of 0.2, 0.4, 0.6 and 0.8 for the alternative sequences under the normal model when using identity scores.

The values of d for each used power and degree of freedom are given in Appendix C. As the values depend only on the covariance matrix of the residual and not on $\mathbf{cov}(\mathbf{Y})$, the same values are used for Settings 1 and 3, and Settings 2 and 4 (see below).

In the testing problem, $\mathbf{X}$ will consist of an intercept term and a $q-1$ -variate normal distribution with expected value $\mathbf{0}_{q-1}$ and variance $\mathbf{I}_{q-1}$. We will then generate an appropriate error term and calculate $\boldsymbol{\beta}$ for the desired power. Then we will form

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$
$$\mathbf{Y}^* = \mathbf{W}\mathbf{Y},$$

where $\mathbf{W}$ is a weight matrix to set $\mathbf{cov}(\mathbf{Y})$.

The simulation was done under multiple different scenarios. Four different values for n, three for p and five for q are used as follows:

$$n = \{50, 100, 200, 400\}$$
$$p = \{2, 3, 5\}$$
$$q = \{1, 2, 3, 5, 10\}$$

The simulation was run for all 15 possible combinations of p and q.

Two different covariance matrices will be used for both $\boldsymbol{\varepsilon}$ and $\mathbf{W}$. We will also generate $\boldsymbol{\varepsilon}$ from the multivariate N_p and multivariate $t_{p,3}$ distributions. This will give us a total of 240 different combinations over all the parameters.

The covariance matrices used for $\boldsymbol{\varepsilon}$ and $\mathbf{W}$ are the Identity matrix $\mathbf{I}_p$ and the diagonal matrix $\mathbf{D}_p$, where $\text{diag}(\mathbf{D}) = (1, 0.01, \ldots, 0.01)$. We will perform the simulation with all four possible combinations of the matrices. This will give us the following cases for the final covariance matrices of $\mathbf{Y}^*$ and $\boldsymbol{\varepsilon}$.

$$\text{Setting 1: } \boldsymbol{\Sigma}_{\mathbf{Y}^*} = \mathbf{I}, \quad \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} = \mathbf{I}$$
$$\text{Setting 2: } \boldsymbol{\Sigma}_{\mathbf{Y}^*} = \mathbf{I}, \quad \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} = \mathbf{D}$$
$$\text{Setting 3: } \boldsymbol{\Sigma}_{\mathbf{Y}^*} = \mathbf{D}, \quad \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} = \mathbf{I}$$
$$\text{Setting 4: } \boldsymbol{\Sigma}_{\mathbf{Y}^*} = \mathbf{D}, \quad \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} = \mathbf{D}.$$

For the t-distribution, $\mathbf{\Sigma}_\varepsilon$ was further divided by $\frac{\nu}{\nu-2}$.

Six different regression methods were used to estimate the regression model from the created data sets. The models were estimated using inner and outer standardized sign and rank methods.

In addition two different L_2 methods were used for comparison. The first was the R *lm* function which uses QR decomposition, for more information see Chambers & Hastie (1992, Chapter 4) and Wilkinson & Rogers (1973). The second was the identity score option used by the *MNM*-package *mv.l1lm* function, see Nordhausen & Oja (2011). The functions produce the same estimates for $\boldsymbol{\beta}$, but estimate the covariance matrix differently.

In each case a thousand different data-sets were created. From each model and repetition the p-value for all coefficients being zero was calculated, and from the whole the ratio of p-values under 0.05.

This represents the models where H_0 was rejected. As d increases, the L_2 estimates should follow the theoretical power in the normal case. We are thus interested in how the spatial sign and rank methods compare to this baseline.

For each test case, we will draw power curves representing how the different methods performed for the selected variables. We will show some results here, and summarize the rest. In many cases the results are quite similar and thus will only be noted here.

In the figures the used regression methods are abbreviated as follows: LS1 – *lm* function, LS2 – *mv.l1lm* function identity scores, OS – outer signs, IS – inner signs, OR – outer ranks and IR – inner ranks. The different covariance settings will also be named as presented previously.

All random data-sets in the simulation study were generated using functions in the *mvtnorm*-package. See Genz & Bretz (2009).

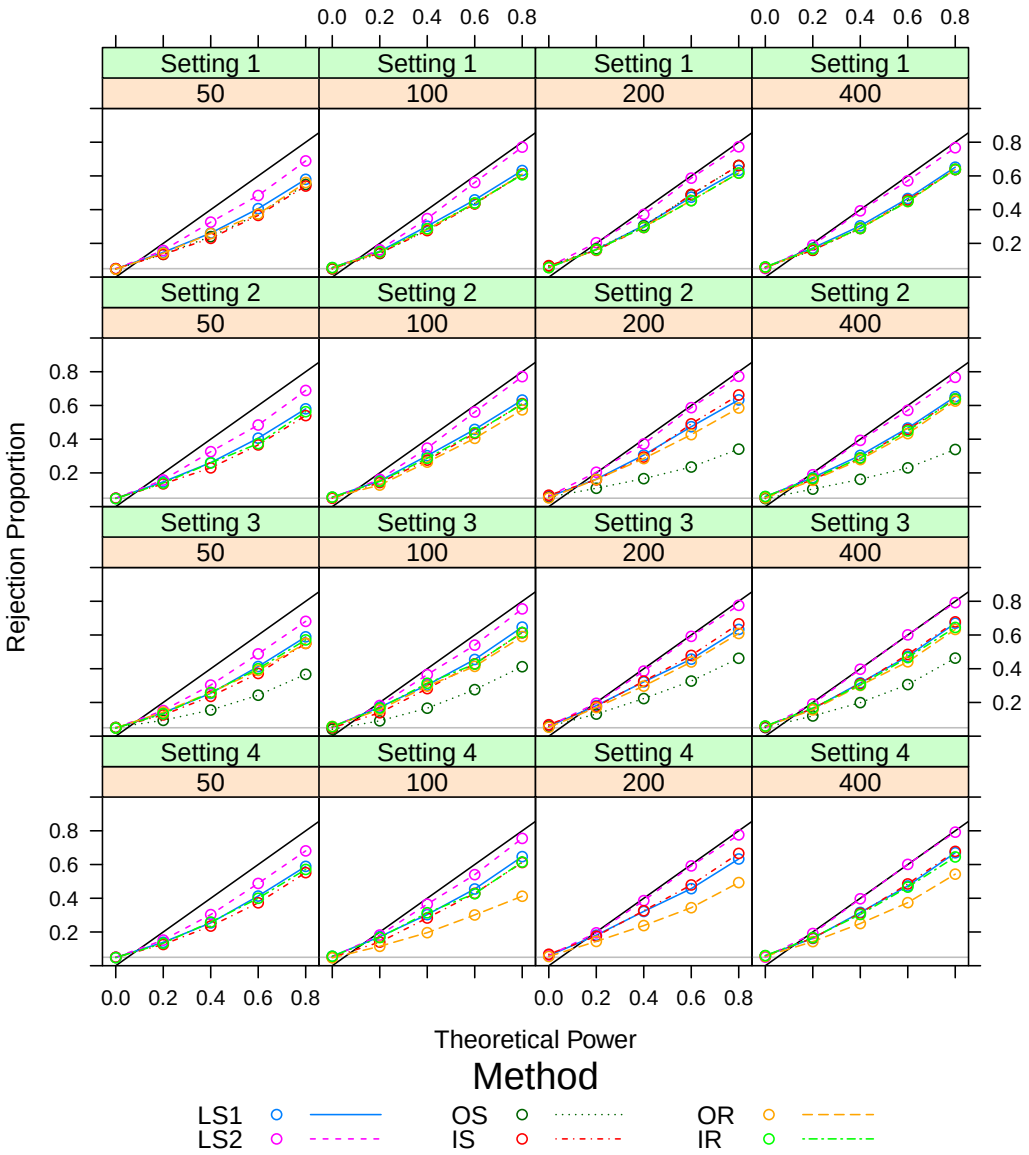

Figure 6.1. Comparison for $\mathbf{Y}_{n\times 2}$ and $\mathbf{X}_{n\times 2}$ using normal error.

In Figure 6.1 the results for one combination of dimensions is shown. The L_2 methods are closest to the theoretical power in all sample sizes and $\mathbf{\Sigma}$ configurations.

The outer standardized methods are especially weak with both unusual $\mathbf{\Sigma}$s and low sample sizes. A missing curve in a panel means that the method could not estimate the co-efficients at all in that testing environment.

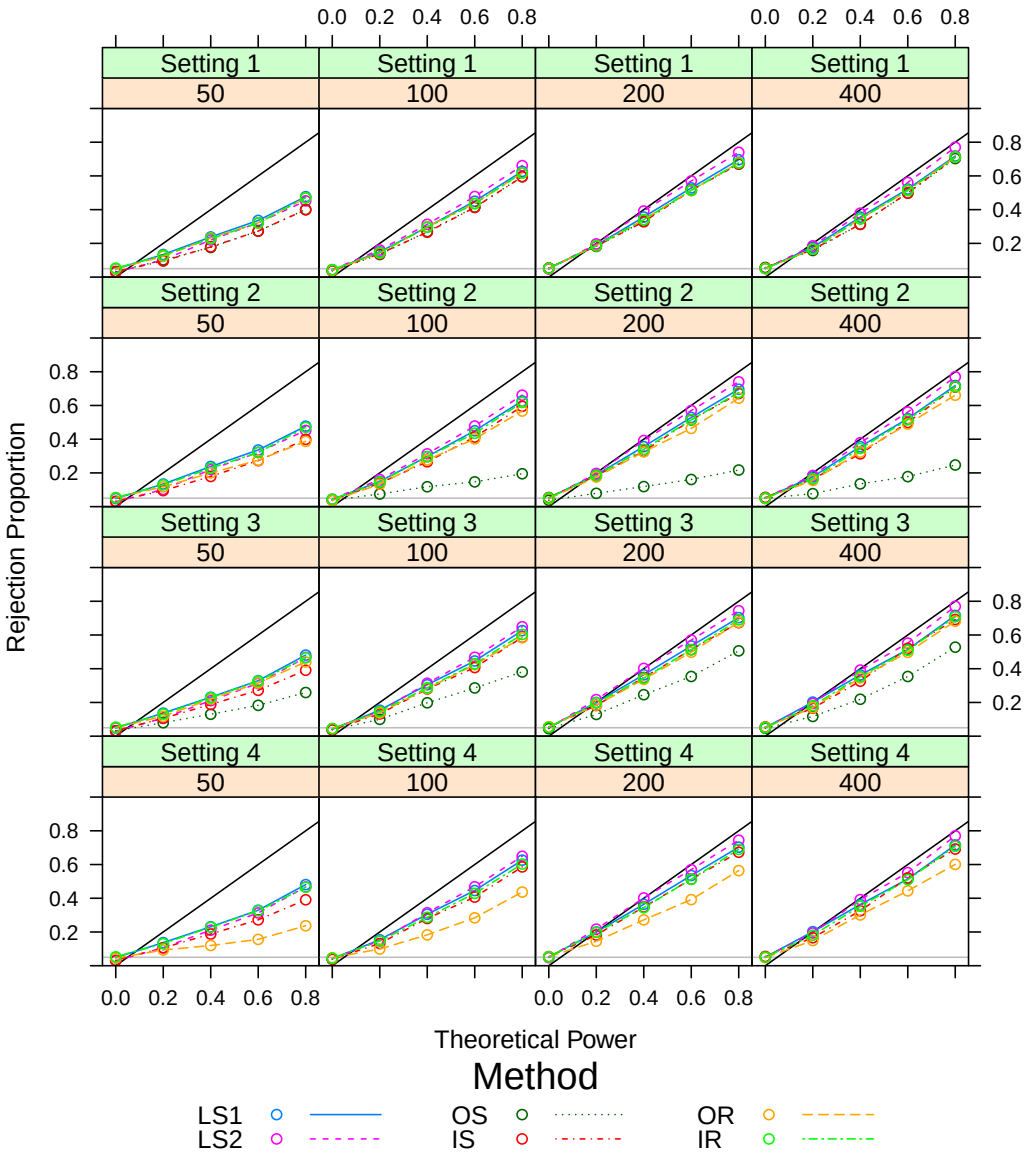

Figure 6.2. Comparison for $\mathbf{Y}_{n\times 5}$ and $\mathbf{X}_{n\times 5}$ using normal error.

These results hold for different sized samples. In Figure 6.2 the results using $\mathbf{Y}$ and $\mathbf{X}$ with more dimensions are shown. Here it should be noted that for small sample sizes all of the methods fail to produce adequate power.

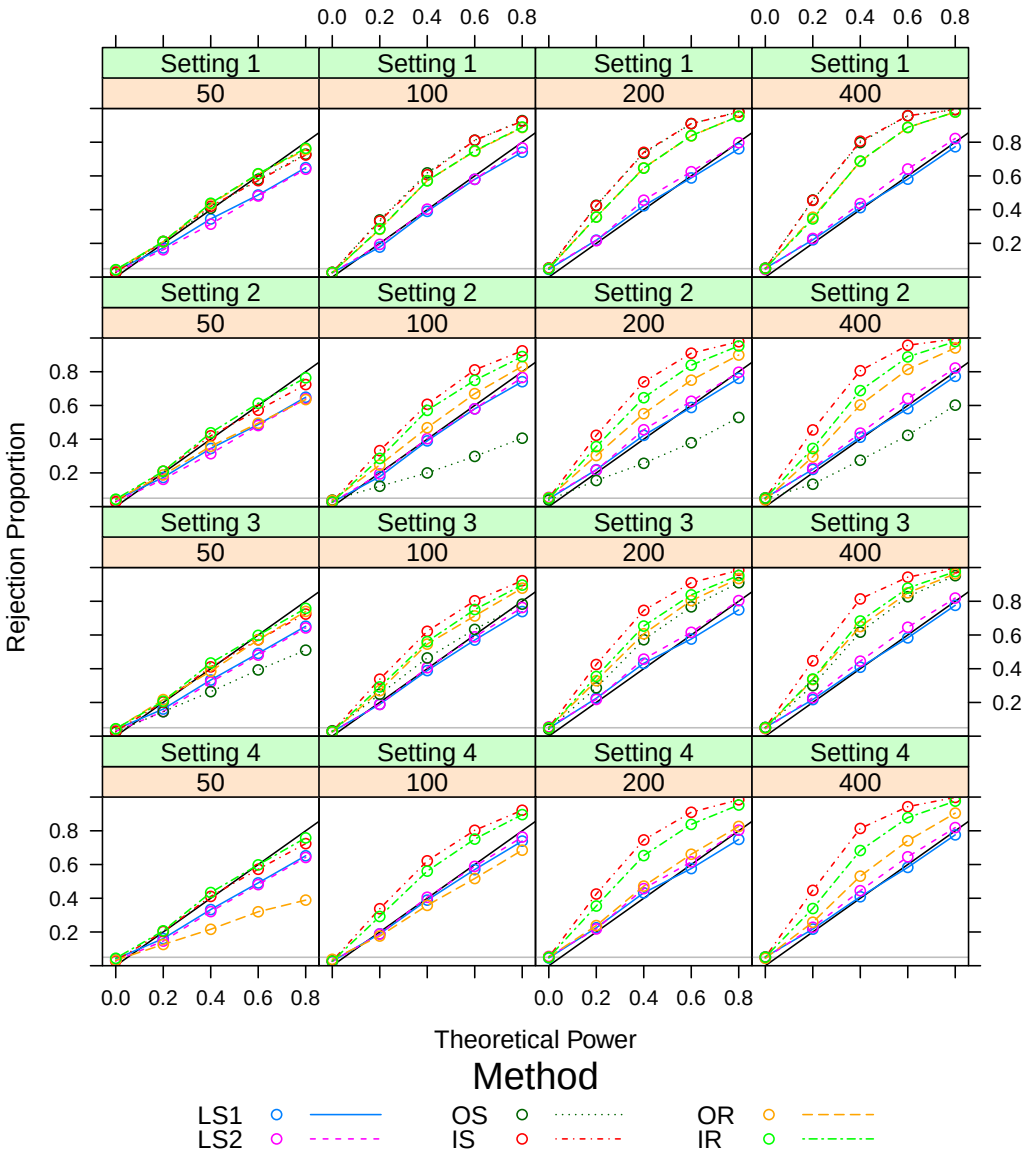

Figure 6.3. Comparison for $\mathbf{Y}_{n\times 5}$ and $\mathbf{X}_{n\times 5}$ using $t_{p,3}$-distributed error.

Figure 6.3 presents the previous test setup using an error term from the t-distribution. It can be seen that the L_2 methods still follow the theoretical power under the normal mode. However, the inner sign and rank methods reject the null hypothesis faster in all cases. This means that they are in fact now more accurately matching the hypotheses.

It should be also noted that outer methods still perform badly with many settings and especially with low sample sizes.

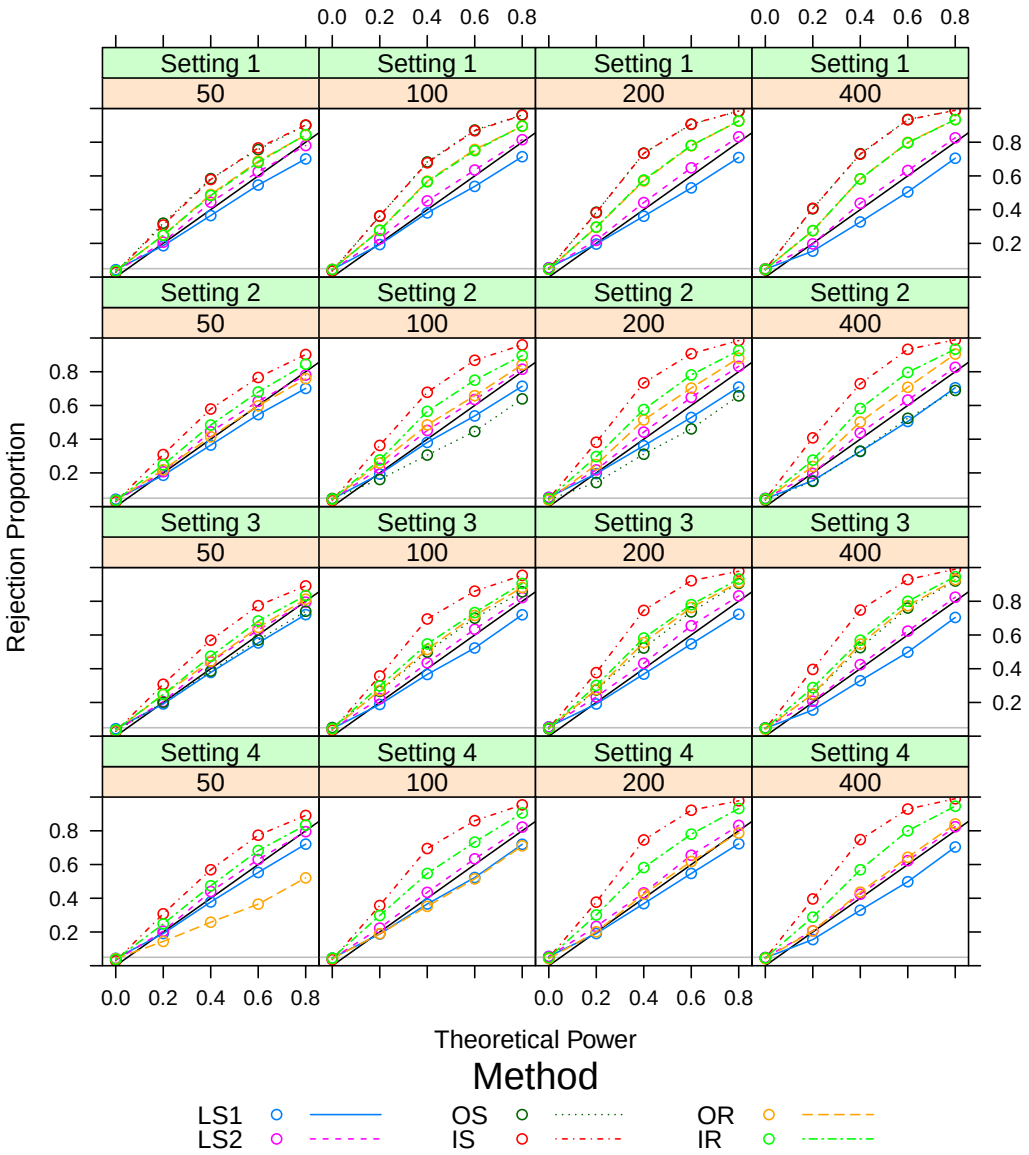

Figure 6.4. Comparison for $\mathbf{Y}_{n\times 3}$ and $\mathbf{X}_{n\times 2}$ using $t_{p,3}$-distributed error.

Another example of t-distributed errors is shown in Figure 6.4. Here the same pattern can be seen, with inner spatial methods rejecting the null hypothesis faster compared to the L_2 methods.

6.2 Testing problem two

In the second testing problem, $\mathbf{X}_1$ will consist of an intercept term and a $q_1 - 1$ -variate normal distribution with expected value $\mathbf{0}_{q-1}$ and variance $\mathbf{I}_{q-1}$. $\mathbf{X}_2$ will be a q_2 -variate normal distribution with expected value $\mathbf{0}_{q_2}$ and variance $\mathbf{I}_{q_2}$. $\boldsymbol{\beta}_1$ will always be a constant $p \times q_1$ matrix of ones.

We will again consider the case $\boldsymbol{\delta} = d\mathbf{1}_p'\mathbf{1}_q$, where d is a non-negative constant. Since calculating the non-centrality parameter for specific powers is more difficult in this testing problem, we will simply use $tr((\boldsymbol{\delta}'\boldsymbol{\delta})(\boldsymbol{\Sigma}^{-1}))$ as the parameter, with pq_1 degrees of freedom.

Then we will form

$$\begin{aligned}
\mathbf{Y}_1 &= \mathbf{X}_1\boldsymbol{\beta}_1 + \boldsymbol{\varepsilon} \\
\mathbf{Y}_1^* &= \mathbf{W}\mathbf{Y}_1 \\
\mathbf{Y}_2 &= \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \boldsymbol{\varepsilon} \\
\mathbf{Y}_2^* &= \mathbf{W}\mathbf{Y}_2
\end{aligned}$$

where $\mathbf{W}$ is a weight matrix to set $\mathbf{cov}(\mathbf{Y})$.

In this testing problem, the same values for n will be used as previously, as well as the different covariance settings. The number of response variables will be either three or five.

We will select q_1 and q_2 for two special cases of tests. We will consider two cases where $q_2 = 1$, corresponding to the hypotheses that just one explaining variable should be rejected. Secondly we will test cases where q_2 will contain approximately half of the variables. The specific testing parameters are

$$\begin{aligned}
q_1 &= \{2\}, q_2 = \{1, 3\} \\
q_1 &= \{5\}, q_2 = \{5\} \\
q_1 &= \{8\}, q_2 = \{1\}
\end{aligned}$$

The figures will use the same notation for covariance structures and test statistics except as follows. For the LS1 estimate the p-value will be calculated using the *anova* function Pillai-Bartlett test statistic. The L_1 tests are compared using the Score test statistic presented in Chapter 4.

The Wald test statistic was also used for comparison. However, in that case the null hypothesis was nearly always rejected regardless of the values of d. Additional tests with higher sample sizes showed that the power curves started to look similar to the Score test curves with a sample size of 10,000. Thus no results are shown for the Wald test, but the case could be studied further.

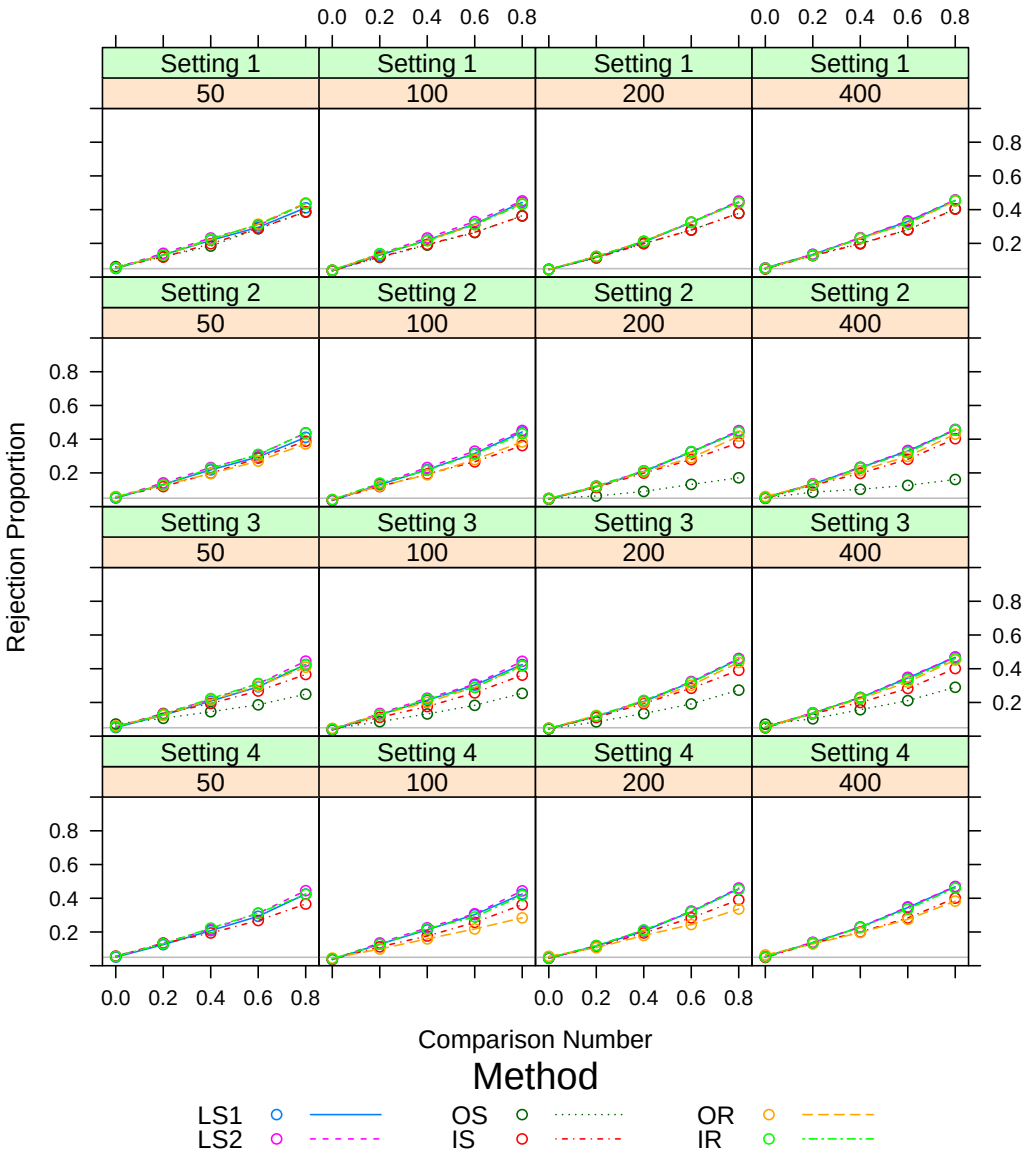

Figure 6.5. Comparison for $\mathbf{Y}_{n\times 3}$, $\mathbf{X}_{1_{n\times 2}}$ and $\mathbf{X}_{2_{n\times 1}}$ using normal error.

One test setting is presented in figure 6.5. Almost all of the methods have similar power, but outer signs perform worse with unusual covariance configurations.

It is important to note that the results can not be directly compared with the first testing problem due to differences in how the co-efficients were calculated.

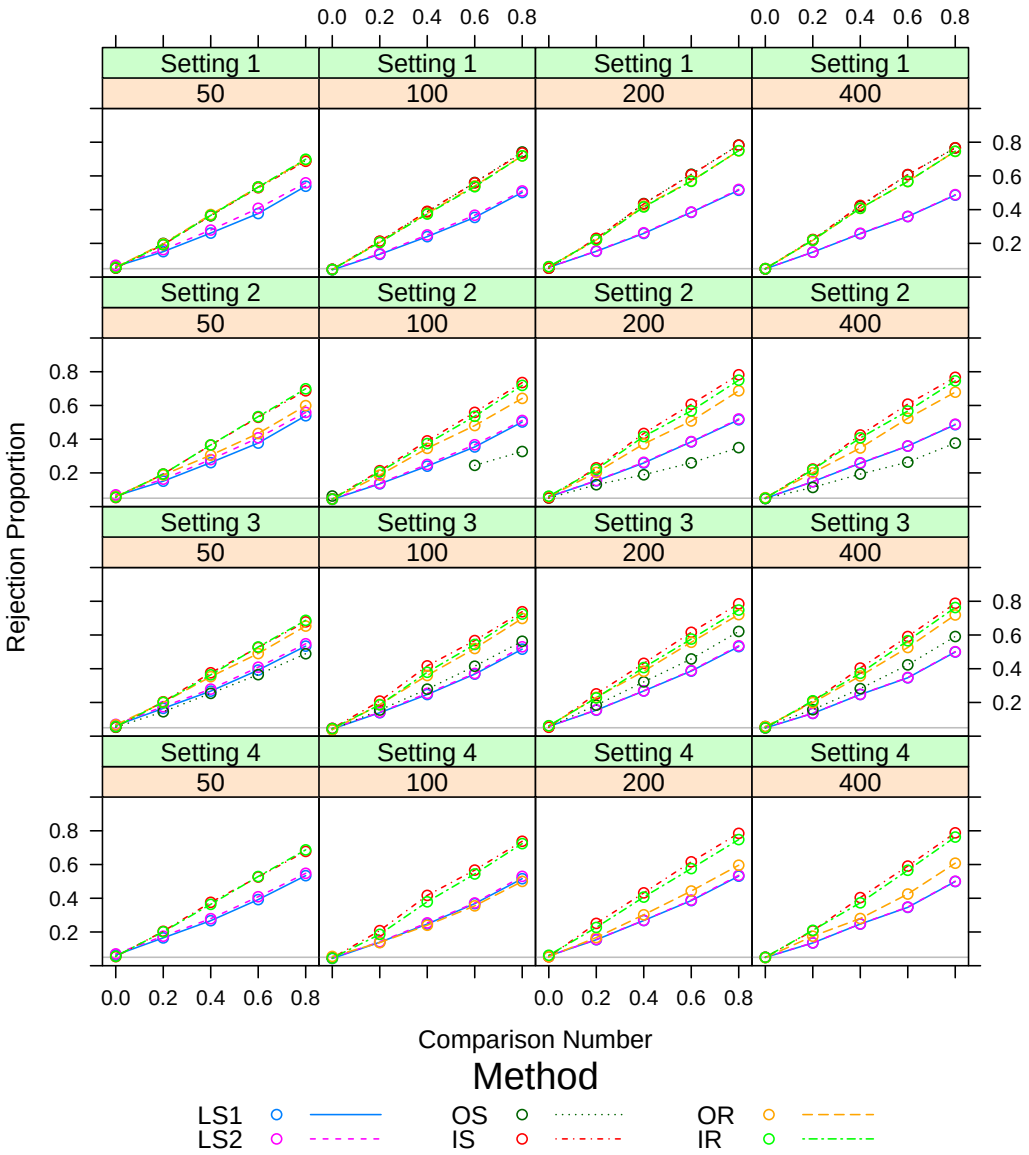

Figure 6.6. Comparison for $\mathbf{Y}_{n\times 3}$, $\mathbf{X}_{1_{n\times 2}}$ and $\mathbf{X}_{2_{n\times 1}}$ using $t_{p,3}$ error.

The same testing configuration using t-distributed error is shown in Figure 6.6. Here we can again see that the inner spatial methods will reject the null hypothesis much faster compared to the normal case. The least squares results do not change significantly.

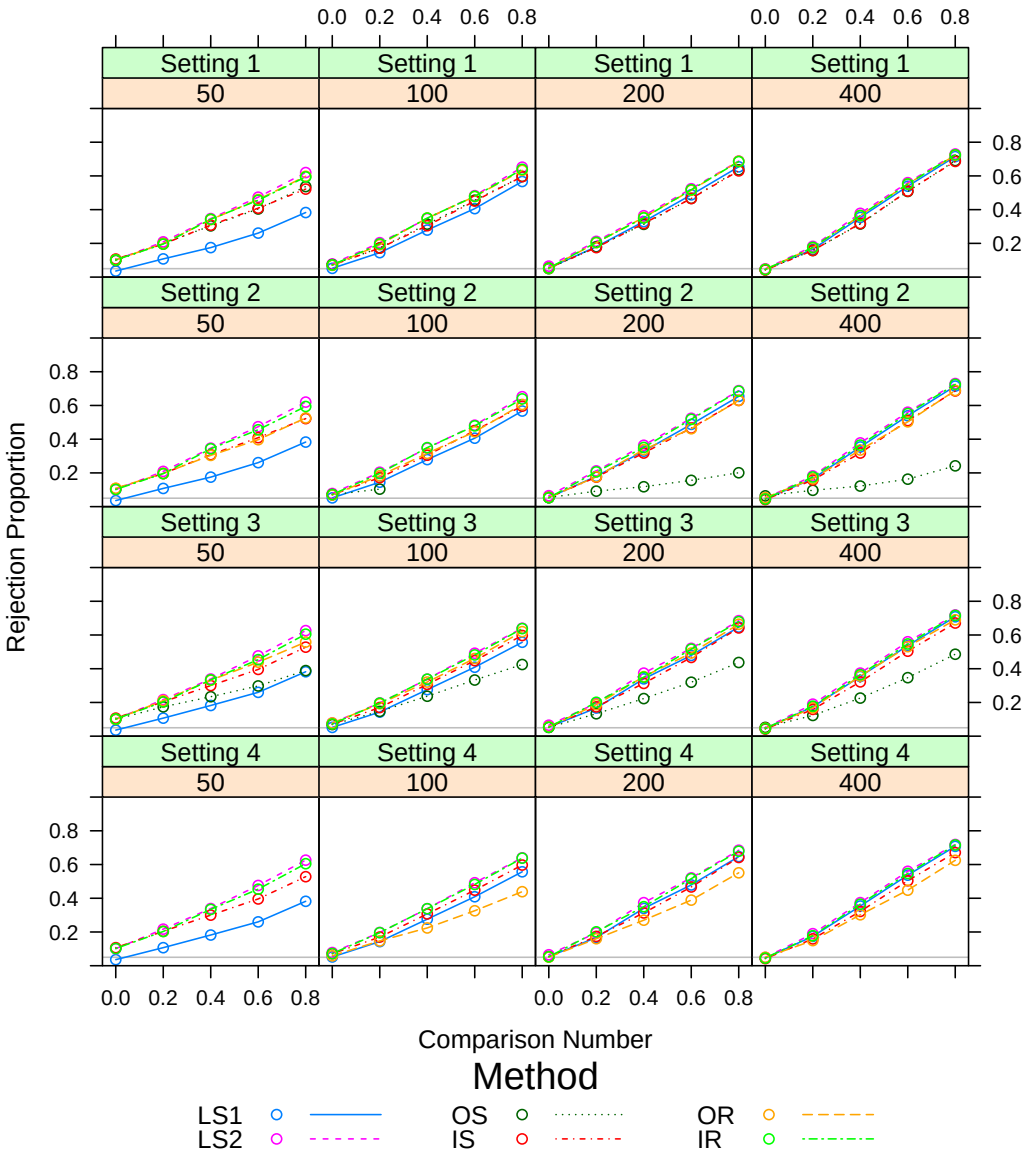

Figure 6.7. Comparison for $\mathbf{Y}_{n\times 5}$, $\mathbf{X}_{1_{n\times 5}}$ and $\mathbf{X}_{2_{n\times 5}}$ using normal error.

In Figure 6.7 another comparison for a different set-up is shown. Here it can be seen that the inner methods are actually comparable to least squares with many sample sizes.

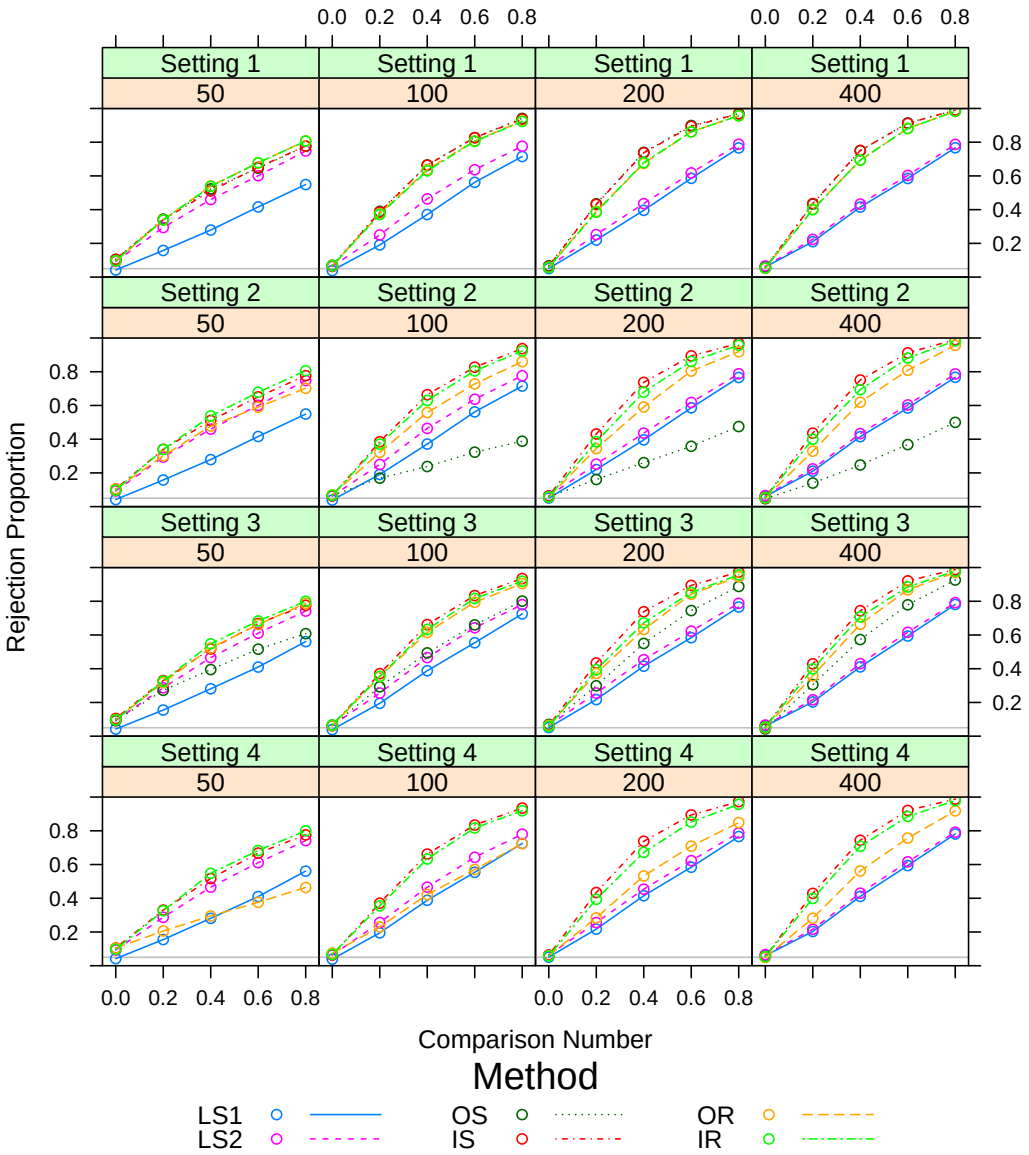

Figure 6.8. Comparison for $\mathbf{Y}_{n\times 5}$, $\mathbf{X}_{1_{n\times 5}}$ and $\mathbf{X}_{2_{n\times 5}}$ using $t_{p,3}$ error.

The same configuration for t-distributed error is shown in Figure 6.8. The same pattern holds as the inner methods reject the hypothesis faster than least squares.

7 Conclusions

This study finds that sign and rank based methods produce regression estimates that can be used in place of traditional L_2 estimates. Although they are computationally more demanding, this can be offset by efficient implementation of used methods.

These estimates can be calculated using functions in R, and can be made faster by integrating code written in C++. This alternate code provides all-around faster computation times, which are made more evident when dealing with larger dimensions of matrices.

The existence of small residuals must be taken into account when calculating the estimates. This can be potentially avoided using a general optimizer, but it was found that this method is much more computationally intensive and thus unpractical in general applications. Thus this method can not be recommended for general use.

When the residual term can be assumed to be Gaussian, the spatial methods perform at similar levels or worse than L_2 estimates. However, using residuals with heavier tailed distributions, the spatial sign and rank methods would provide better results. In the second testing problem, the inner spatial methods can also reject the null hypothesis faster even with a Gaussian error.

In applications where the residuals may not be assumed to be normally distributed, these results show that spatial methods should be considered. This study compares only Gaussian and t-distributed residuals, but other more varied residuals could also be tested using the same framework.

The Wald test statistic should be asymptotically equivalent to the Score statistic, but with the sample sizes used in this thesis there are clear differences. The properties of the Wald statistic could be worth additional investigation. Using the Score statistic should provide more accurate results even with smaller samples.

This thesis was a very preliminary comparison into the differences between the presented methods. Very simple covariance structures and co-efficient matrices were used. It is likely that more differences can be found using other testing set-ups, and this problem could be studied further.

Acknowledgments

I would like to thank my supervisor Klaus Nordhausen for introducing me to this topic and for all of his advice during the process. This thesis has proved to be interesting and has provided many challenges along the way.

I also wish to thank my parents for the support they have given me through my studies.

Bibliography

Chambers, J. M. & Hastie, T. J. (1992), *Statistical Models in S*, Wadsworth & Brooks/Cole.

Choi, K. & Marden, J. (1997), "An approach to multivariate rank tests in multivariate analysis of variance", *Journal of the American Statistical Society*, 92, 1581–1590.

Eddelbuettel, D. (2013), *Seamless R and C++ Integration with Rcpp*, Springer.

Eddelbuettel, D. & Francois, R. (2011), "Rcpp: Seamless R and C++ integration", *Journal of Statistical Software*, 40(8), 1–18.

Eddelbuettel, D. & Sanderson, C. (2014), "RcppArmadillo: Accelerating R with high-performance C++ linear algebra", *Computational Statistics and Data Analysis*, 71, 1054–1063.

Fritz, H., Filzmoser, P. & Croux, C. (2012), "A comparison of algorithms for the multivariate L_1-median", *Computational Statistics*, 27, 393–410.

Genz, A. & Bretz, F. (2009), *Computation of Multivariate Normal and t Probabilities* (Vol. 195 of *Lecture Notes in Statistics*), Springer.

Hettmansperger, T. P. & Randles, R. H. (2002), "A practical affine equivariant multivariate median", *Biometrika*, 89, 851–860.

Hodges, J. L. & Lehmann, E. L. (1963), "Estimation of location based on ranks", *Annals of Mathematical Statistics*, 34, 598–611.

Lehmann, E. L. (2006), *Nonparametrics: Statistical Methods Based on Ranks*, Springer.

Milasevic, P. & Ducharme, G. R. (1987), "Uniqueness of the spatial median", *Annals of Mathematical Statistics*, 25, 1332–1333.

Möttönen, J. & Oja, H. (1995), "Multivariate spatial sign and rank methods", *Journal of Nonparametric Statistics*, 5, 201–213.

Nordhausen, K. & Oja, H. (2011), "Multivariate L_1 methods: The package MNM", *Journal of Statistical Software*, 43, 1–28.

Oja, H. (2010), *Multivariate Nonparametric Methods with R*, Springer.

R Core Team (2012), *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria.

Sanderson, C. (2010), "Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments", Tech. rep., NICTA.

Vardi, Y. & Zhang, C.-H. (1999), "A modified Weiszfeld algorithm for the Fermat-Weber location problem", *Mathematical Programming Series A*, 90, 559–566.

Wilkinson, G. N. & Rogers, C. E. (1973), "Symbolic descriptions of factorial models for analysis of variance", *Applied Statistics*, 22, 392–399.

Appendix A: C++ code

```
#include "armadillo"

using namespace arma;
/* A Class to calculate spatial signs
* using outer and inner standardization.
* Requires as parameters the original
* matrices of response and explaining variables,
* initial beta, maximum number of allowed iterations,
* convergence limit and residual norm tolerance.
*/
class spatialSigns{
private:
 //Initialize class members
 mat Y;
 mat X;
 mat B_init;
 mat S_init;
 int iter;
 int maxiter;
 double eps;
 double eps_S;

public:
 //Initialize public methods and constructor
 mat getB(){
  return B_init;
 }

 mat getS(){
  return S_init;
 }

 int getIter(){
  return iter;
 }
```

```
spatialSigns(mat y, mat x, mat b, int max, double e, double e_S):
Y(y), X(x), B_init(b), iter(0), maxiter(max), eps(e), eps_S(e_S)
{

}

/*Function to calculate outer signs
* As a result the class B_init member is modified
* to contain the estimated beta.
*/
void outer_sign(){
 //Initialize sample size and difference between iterations
 int n = X.n_rows;
 double differ = std::numeric_limits<double>::infinity();
 //Function runs until convergence or maximum iterations reached
 while (differ > eps && iter < maxiter) {
  //Safety so function wont break in case of errors
  try{
   //Calculate residuals and their norm
   mat E = Y - X * B_init;
   mat norm_E = sqrt(sum(square(E),1));
   //Replace too small residuals with limit if found
   if (norm_E.min() < eps_S){
    for(int i = 0; i < n; i++){
     norm_E(i) = norm_E(i) < eps_S ? eps_S : norm_E(i);
    }
   }
   //Calculate new estimate based on residuals
   mat E_sign = E;
   E_sign.each_col() /= norm_E;
   mat X_E = X;
   X_E.each_col() /= sqrt(norm_E);
   mat XEs = (X.t() * E_sign)/n;
   mat XEXE = (X_E.t() * X_E)/n;
   mat ch_XEXE = chol(XEXE);
   mat B_new = B_init + solve(trimatu(ch_XEXE),
       solve(trimatl(ch_XEXE.t()), XEs));
   iter = iter + 1;
   //Calculate difference between estimates
   //and replace old value with new one
   differ = sqrt(accu(square(B_new - B_init)));
   B_init = B_new;
  }
  catch (...){
   iter = maxiter + 1;
```

```
  }
 }
}
/*Function to calculate inner signs
* As a result the class B_init member is modified
* to contain the estimated beta and the member S_init
* is modified to contain the estimated covariance
*/
void inner_sign(){
 //Initialize sample size and difference between iterations
 int n = X.n_rows;
 int p = Y.n_cols;
 double differ = std::numeric_limits<double>::infinity();
 //Initial covariance matrix
 S_init = ((Y - X * B_init).t() * (Y - X * B_init))/n;
 //Function runs until convergence or maximum iterations reached
 while (differ > eps && iter < maxiter) {
  //Safety so function wont break in case of errors
  try{
   //Calculate inverse cov through eigen decomposition
   vec eigval;
   mat eigvec;
   eig_sym(eigval, eigvec, S_init);
   mat S_sqrt = eigvec * sqrt(diagmat(eigval)) * eigvec.t();
   mat S_sqrt_inv = inv_sympd(S_sqrt);
   //Calculate residuals and their norm
   mat E = (Y - X * B_init) * S_sqrt_inv;
   mat norm_E = sqrt(sum(square(E),1));
   //Replace too small residuals with limit if found
   if (norm_E.min() < eps_S){
    for(int i = 0; i < n; i++){
     norm_E(i) = norm_E(i) < eps_S ? eps_S : norm_E(i);
    }
   }
   //Calculate new estimate for beta based on residuals
   mat E_sign = E;
   E_sign.each_col() /= norm_E;
   mat X_E = X;
   X_E.each_col() /= sqrt(norm_E);
   mat XEsSs = ((X.t() * E_sign)/n) * S_sqrt;
   mat XEXE = (X_E.t() * X_E)/n;
   mat ch_XEXE = chol(XEXE);
   mat B_new = B_init + solve(trimatu(ch_XEXE),
   solve(trimatl(ch_XEXE.t()), XEsSs));
   //Calculate new estimate for covariance
```

```
      mat S_new = ((double) p)/n * S_sqrt *
      (E_sign.t() * E_sign) * S_sqrt;

      iter = iter + 1;
      //Calculate difference between beta estimates
      //and replace old values with new ones
      differ = sqrt(accu(square(B_new - B_init)));
      B_init = B_new;
      S_init = S_new;
     }
     catch(...){
      iter = maxiter + 1;
     }
    }
   }
};

/* A Class to calculate spatial ranks
* using outer and inner standardization.
* Requires as parameters the original
* matrices of response and explaining variables
* maximum number of allowed iterations,
* convergence limit and residual norm tolerance.
*/
class spatialRanks{
private:
 //Initialize class members
 mat Y;
 mat X;
 mat Y2;
 mat X2;
 mat B_init;
 mat S_init;
 int iter;
 int maxiter;
 double eps;
 double eps_S;

public:
 //Initialize public methods and constructor
 mat getX2(){
  return X2;
 }

 mat getY2(){
```

```
 return Y2;
}
mat getB(){
 return B_init;
}

mat getS(){
 return S_init;
}

int getIter(){
 return iter;
}

spatialRanks(mat y, mat x, int max, double e, double e_S):
Y(y), X(x), iter(0), maxiter(max), eps(e), eps_S(e_S)
{
 //Calculate matrices of pairwise differences and initial beta
 try{
  X2 = pairwiseDiff(X);
  Y2 = pairwiseDiff(Y);
  mat D_mat = X2.t() * X2;
  mat ch_D = chol(D_mat);
  B_init = solve(trimatu(ch_D), solve(trimatl(ch_D.t()),
  (X2.t() * Y2)));
 }
 catch (...){
  iter = maxiter + 1;
 }
}

/*Function to calculate outer ranks
* As a result the class B_init member is modified
* to contain the estimated beta.
*/
void outer_rank(){
 //Initialize sample size and difference between iterations
 int n = X2.n_rows;
 double differ = std::numeric_limits<double>::infinity();
 //Function runs until convergence or maximum iterations reached
 while (differ > eps && iter < maxiter) {
  //Safety so function wont break in case of errors
  try{
   //Calculate residuals and their norm
   mat E = Y2 - X2 * B_init;
```

```
    mat norm_E = sqrt(sum(square(E),1));
    //Replace too small residuals with limit if found
    if (norm_E.min() < eps_S){
     for(int i = 0; i < n; i++){
      norm_E(i) = norm_E(i) < eps_S ? eps_S : norm_E(i);
     }
    }
    //Calculate new estimate based on residuals
    mat E_sign = E;
    E_sign.each_col() /= norm_E;
    mat X2_E = X2;
    X2_E.each_col() /= sqrt(norm_E);
    mat X2Es = (X2.t() * E_sign)/n;
    mat X2EX2E = (X2_E.t() * X2_E)/n;
    mat ch_X2EX2E = chol(X2EX2E);
    mat B_new = B_init + solve(trimatu(ch_X2EX2E),
    solve(trimatl(ch_X2EX2E.t()), X2Es));
    iter = iter + 1;
    //Calculate difference between estimates
    //and replace old value with new one
    differ = sqrt(accu(square(B_new - B_init)));
    B_init = B_new;
   }
   catch (...){
    iter = maxiter + 1;
   }
  }
}
/*Function to calculate inner ranks
* As a result the class B_init member is modified
* to contain the estimated beta and the member S_init
* is modified to contain the estimated covariance
*/
void inner_rank(){
 //Initialize sample size
 int n1 = Y.n_rows;
 int n = X2.n_rows;
 int p = Y.n_cols;
 //Initialize difference between iterations and covariance
 double differ = std::numeric_limits<double>::infinity();
 S_init = ((Y - X * B_init).t() * (Y - X * B_init))/n;
 //Function runs until convergence or maximum iterations reached
 while (differ > eps && iter < maxiter) {
  //Safety so function wont break in case of errors
  try{
```

```
    //Calculate inverse cov through eigen decomposition
    vec eigval;
    mat eigvec;
    eig_sym(eigval, eigvec, S_init);
    mat S_sqrt = eigvec * sqrt(diagmat(eigval)) * eigvec.t();
    mat S_sqrt_inv = inv_sympd(S_sqrt);
    //Calculate residuals and their norm
    mat E = (Y2 - X2 * B_init) * S_sqrt_inv;
    mat norm_E = sqrt(sum(square(E),1));
    //Replace too small residuals with limit if found
    if (norm_E.min() < eps_S){
     for(int i = 0; i < n; i++){
      norm_E(i) = norm_E(i) < eps_S ? eps_S : norm_E(i);
     }
    }
    //Calculate new estimates for beta and covariance
    mat E_sign = E;
    E_sign.each_col() /= norm_E;
    mat X2_E = X2;
    X2_E.each_col() /= sqrt(norm_E);
    mat X2Es = ((X2.t() * E_sign)/n)  * S_sqrt;
    mat X2EX2E = (X2_E.t() * X2_E)/n;
    mat ch_X2EX2E = chol(X2EX2E);
    mat B_new = B_init + solve(trimatu(ch_X2EX2E),
    solve(trimatl(ch_X2EX2E.t()), X2Es));

    mat S_rank = spatialRank((Y - X * B_new) * S_sqrt_inv);
    mat S_new = ((double) p)/n1 * S_sqrt *
    S_rank.t() * S_rank * S_sqrt;

    iter = iter + 1;
    //Calculate difference between beta estimates
    //and replace old values with new ones
    differ = sqrt(accu(square(B_new - B_init)));
    B_init = B_new;
    S_init = S_new;
   }
   catch (...){
    iter = maxiter + 1;
   }
  }
}

/*Function to calculate the matrix of pairwise differences
* of a matrix which returns the acquired matrix
```

```
*/
mat pairwiseDiff(mat A){
 //Calculate the required number of rows
 int n = A.n_rows;
 int e = 0;
 for(int i = 0; i < n-1; i++){
  for(int j = i+1; j < n; j++){
   e++;
  }
 }
 //Initialize a matrix of required size
 //and calculate the differences
 mat B(e, A.n_cols);
 e = 0;
 for(int i = 0; i < n-1; i++){
  for(int j = i+1; j < n; j++){
   B.row(e) = (A.row(i) - A.row(j));
   e++;
  }
 }
 return B;
}
/*Function to calculate the spatial rank matrix of a matrix.
* Returns the acquired matrix.
*/
mat spatialRank(mat A){
 //Initialize the values and an empty matrix
 int n=A.n_rows;
 int k=A.n_cols;
 double d;

 mat ranks(n,k);
 ranks.fill(0.0);

 vec temp(k);

 //For each pair of rows
 for(int i=0; i<n; i++) {
  for(int j=0; j<n; j++) {
   if(i!=j) {
    //Compute individual differences and the norm
    for(int m=0; m<k; m++){
     temp(m)=A(i,m)-A(j,m);
    }
    d=0.0;
```

```
      for(int m=0; m<k; m++){
       d+=(temp(m)*temp(m));
      }
      d=sqrt(d);
      //Add to the sum of signs
      for(int m=0; m<k; m++){
       ranks(i,m)+=(temp(m)/d);
      }
     }
    }
   }
   mat result = ranks/n;

   return result;
  }
};
```

Appendix B: Optim-implementation

```
#Function to calculate outer signs regression
#using the optim function
optim.outer.signs<-function(Y,X)
{
 require(MNM)
 p <- ncol(Y)       #The number of traits
 q <- ncol(X)       #The number of explaining variables
 n <- nrow(Y)       #The number of cases

 #The average of absolute deviations
 fn<-function(beta,Y,X){
  B <- matrix(beta, q, p)
  E <- Y - X %*% B
  mean(sqrt(diag(E %*% t(E))))
 }

 #The gradient of fn
 grr<-function(beta,Y,X){
  B <- matrix(beta, q, p)
  E<- Y - X %*% B
  norm.E <- SpatialNP:::norm(E)
  E.sign <- sweep(E, 1, norm.E, "/")
  -(1/n) * c(t(X) %*% E.sign)
 }

D.mat <- crossprod(X)
 ch.D <- chol(D.mat)
 B.init <- backsolve(ch.D, forwardsolve(ch.D, crossprod(X,
  Y), upper.tri = TRUE, transpose = TRUE))
 beta0 <-c (B.init)

 res <- optim(beta0, fn, grr, method="BFGS",
  control=list(maxit=10000, reltol=1e-10), Y=Y, X=X)
 beta <- matrix(res$par, q, p)
 value <- res$value
 convergence <- res$convergence
```

```
 norms <- sqrt(diag(beta %*% t(beta)))
 if(is.null(colnames(X))){
  colnames(X) <- c("", paste("x", 1:(q-1), sep=""))
 }
 colnames(X)[1] <- "int"
 colname <- colnames(X)

 list(beta=beta, norms=norms, value=value,
  convergence=convergence, colname=colname)
}
```

Appendix C: Delta values

Values for delta for all definitions of freedom used in Testing Problem One are given in Tables 1 and 2, rounded to three decimals.

Table 1. Values of delta for identity covariance.

	Theoretical power			
dfs (pq)	20	40	60	80
4	0.775	1.124	1.408	1.727
6	0.696	0.996	1.237	1.507
8	0.645	0.916	1.131	1.370
9	0.626	0.885	1.091	1.319
10	0.609	0.859	1.056	1.27
12	0.581	0.815	0.999	1.202
15	0.548	0.765	0.934	1.120
18	0.523	0.727	0.885	1.058
20	0.509	0.706	0.858	1.024
22	0.496	0.687	0.834	0.994
30	0.458	0.631	0.762	0.905
33	0.447	0.614	0.742	0.879
55	0.393	0.535	0.642	0.756

Table 2. Values of delta for extreme covariance.

	Theoretical power			
dfs (pq)	20	40	60	80
4	0.109	0.158	0.198	0.243
6	0.098	0.140	0.174	0.212
8	0.091	0.129	0.159	0.193
9	0.076	0.108	0.133	0.161
10	0.068	0.096	0.118	0.142
12	0.082	0.115	0.141	0.169
15	0.061	0.085	0.104	0.125
18	0.064	0.089	0.108	0.129
20	0.057	0.079	0.096	0.114
22	0.070	0.097	0.117	0.140
30	0.051	0.070	0.085	0.101
33	0.055	0.075	0.091	0.107
55	0.044	0.060	0.072	0.084