PRESELECTION BIAS IN HIGH DIMENSIONAL REGRESSION

by

Kristina Haarr Vinvand

THESIS
for the degree of
MASTER OF SCIENCE

(Master in Modelling and Data Analysis)

Faculty of Mathematics and Natural Sciences
University of Oslo

November 2014

Abstract

In this thesis, we have studied the preselection bias that can occur when the number of covariates in a high dimensional regression problem is reduced prior to a high dimensional regression analysis like the lasso. Datasets in genomics often include ten- or hundred thousands, or even millions, of covariates and a few hundred or less patients. To reduce computations or to make the problem tractable, practitioners often rank the covariates according to univariate importance for the response, and preselect some thousand covariates from the top of the list for multivariate analysis via penalized regression. If the preselection of covariates is not done in a controlled way, this leads to preselection bias.

We have studied the effect of preselection on estimation and prediction and the bias this might induce. With a small preselected dataset, the lasso in combination with cross validation tends to select many covariates, which together are able to explain the data at hand very well. However, for a new independent dataset, these covariates predict rather poorly. This is preselection bias. We have visualized the preselection bias through boxplots in several different datasets from genomics and in simulated data. We have also demonstrated that the problem of preselection bias is most evident in datasets where there is a lot of noise, and where there are heavy dependencies between covariates, as the univariate ranking will not be able to capture the structure of the complex relations in this case.

To be able to trust predictions made from penalized regression on preselected covariates, the preselection should be coupled with some algorithm that controls how many covariates that should be included in order to avoid the bias. We have studied methods like "SAFE", "strong" and "freezing" that all make preselection more safe, the word safe meaning that the lasso analysis for the preselected set of covariates should conclude with the same result as if all covariates were included.

Preface

This thesis was written as a part of my master degree during the period January 2013 to November 2014, alongside various master courses at the University of Oslo. I have really appreciated working on this thesis and have learned even more than I had imagined when I started.

First and foremost, I would like to thank my incredible supervisors Ingrid Kristine Glad and Linn Cecilie Bergersen. Thank you both so much for many good conversations, helpful advice and clever insight. A special thanks to Ingrid for always keeping your door open and for making it easy to ask "dumb" questions, and to Linn Cecilie for taking the time to meet with me even after your work here at the University of Oslo was finished.

I also want to thank Sjur Reppe at Oslo University Hospital (Rikshospitalet) for the Bone data in Chapter 5 and Heidi Lyng at Oslo University Hospital (Radiumhospitalet) for the methylation data in Chapter 8. Without people like you, analysis in genomics, like in this thesis, would be impossible.

I could write a whole chapter just thanking my family. I am so grateful for all the support from my parents and sisters through the years, even though you don't understand a word of what I'm doing. Thank you for always being there! And a special thank you to my husband, Jens Daniel, who has been with me through happiness, frustrations and tough times. You make my life complete, and I could never have done this without you.

Kristina Haarr Vinvand
November 2014

Contents

Chapter 1

Introduction

In many situations calling for statistical regression analysis, especially in genomics and epigenetics, the number of covariates p_{all} is much bigger than the number of observations N ($p_{all} >> N$), as will be illustrated in Chapter 2. In order to handle challenges caused by a high number of covariates, Tibshirani (1996) introduced the concept of lasso in his paper "Regression Shrinkage and Selection via the Lasso" in 1996, and the theory around this problem has grown over the years.

The lasso method introduces a L_1 penalty on the covariates in a regression model and performs variable selection by shrinking some of the coefficients exactly to zero. A penalty parameter λ, also often called a tuning parameter, decides the amount of shrinking and thereby how many of the covariates that are selected by the lasso. There are several ways to find the optimal λ, but as the main interest often is to fit a model that predicts the response as accurately as possible, one often turn to some sort of cross validation.

As the routines for collecting data get more and more advanced, scientists are able to register an enormously large number of covariates. A dataset can be so large that the computation time for analyzing the data becomes very high, and it can even be impossible to read the data into programs like *R* (*The R Foundation for Statistical Computing, version 2.15.2*). The upper size limit for a matrix to be read into *R* is that the whole matrix, $N \times p_{all}$, cannot exceed $2^{31} - 1$. Due to this constraint and the computation time, reducing the number of covariates before doing lasso analyses is useful, even though the lasso works for high dimensions ($p_{all} > N$).

A common way of doing dimension reduction is first to study one covariate at a time and examine how this specific covariate influences the response, and in the subsequent analyses only include the top ranked covariates sorted according to their univariate correlation with the response. The number of covariates that is being included for the subsequent analyses seems, in many cases, to be quite arbitrary. When ranking the data according to their univariate relation with the response, the interaction and correlation between different covariates in the design matrix will not be taken into account.

Both datasets from medical studies and simulated datasets will be used in this thesis to visualize what happens when the dimension of a high dimensional dataset is reduced without supervision. Cross validation will be applied to find the optimal value of the penalty parameter λ. A cross validation curve visualizes the cross validated prediction error for a grid of various λ-values, and the optimal penalty parameter, called λ_{min} in the full dataset, is chosen to be the λ that minimizes this cross validation curve. For different sizes of the preselected set of covariates, the cross validation curve will change, and thereby also the value of λ. We seek the same model as the whole dataset would give, i.e the number of covariates in the preselected set that will give the same penalty parameter λ_{min} as the dataset including all the covariates.

When reducing the number of covariates in high dimensional data by preselection, the lasso solution model may fit very well for the data from which the model was fitted. But large prediction errors may occur when the fitted model is applied on a new dataset. Preselection bias is a concept that is mentioned by several authors writing about high dimensional data, but it is seldom clear what the problem of preselection bias really is. In this thesis, we will demonstrate the problem of preselection bias in high dimensional data, and illustrate how preselection bias will reduce the prediction ability of a model even if it seems that the preselection helps the lasso to arrive at a better model than without. We will focus on preselection in a lasso setting even though preselection bias can appear also when reducing the number of covariates prior to any regression analysis, for instance when reducing the dataset to be able to use ordinary least squares method (OLS) to fit a model. In this case, it is not clear how one could examine the phenomena of preselection bias because it would be impossible to fit a reference model including all the covariates with OLS.

We will also study methods that will make the preselection of covariates more safe, and thereby make the final solution of the lasso more trustworthy and avoiding preseletion bias. Literature proposing various preselection methods to control the bias caused by preselection has grown in recent years. Sure independence screening (SIS) was introduced by Fan and Lv (2008) and is based on correlation learning. SIS reduces the size of the high dimensional dataset from high to a moderate scale that is below sample size, but this method assumes fairly uncorrelated variables. As datasets in genomics often have block-like patterns of correlated variables, there are other methods that are better suited when working with genomic datasets (Richardson and Bottolo in discussions in Fan and Lv (2008)). El Ghaoui et al. (2011) and Tibshirani et al. (2012) developed rules to eliminate covariates for the lasso. They are called the SAFE method and the strong method, respectively, and for a given penalty λ they discard covariates that are clearly not important to describe the response. Furthermore, Bergersen et al. (2013) introduced freezing that can be applied when preselecting covariates so that preselection bias is avoided. The freezing algorithm studies the cross validation curve for different values of λ and include only the top most important covariates of a sorted list of covariates in the subsequent analyses with lasso. In this thesis we focus on genomic datasets, and will study how to avoid preselection bias through the methods SAFE, strong and freezing.

The thesis is organized as follows: In Chapter 2, some high dimensional datasets in genomics are introduced. Chapter 3 contains theory for linear regression and Cox regression before regularized regression and cross validation are introduced. Furthermore, what is meant by preselection and preselection bias will be explained in Chapter 4. Chapter 5 and 6 use lasso in a practical example with linear regression and Cox regression, respectively, and illustrate the problem of preselection bias. SAFE and strong, which are methods that make it possible to discard covariates, but avoid preselection bias, are introduced in Chapter 7. Chapter 8 introduces freezing for more safely including only the top ranked covariates for the lasso analysis, and in this chapter, we also study an example of survival analysis with truly high dimensional data. In Chapter 9 we have done three different simulation studies to examine when preselection bias is most pronounced, before we summarize and make some concluding remarks in Chapter 10.

Chapter 2

High dimensional datasets in genomics

To better understand the background for the analyses in the practical examples in this thesis, we will in this chapter introduce some of the theory about how the human DNA and genes are built. We will also describe several different genomic and epigenetic datasets, in which some are used in Chapters 5, 6 and 8.

DNA and genes

The human body is made up of cells which all contain a nucleus with a genome. All the genetic information of a human body is included in the genome, and it consists of 23 chromosome pairs. The chromosomes are encoded by long strands of DNA (deoxyribonucleic acid) with codes for genes. The human genome contains around 30 000 genes, and the genes can be called the recipe for the proteins that build and control the body (www.yourgenome.org, visited 04.2014). Before the protein is made, mRNA is created from the DNA. When mRNA and protein is made, we say that the gene is expressed. A cell's individual characteristics are defined by the proteins, so which genes that are expressed can tell a lot about a cell, for example if a cancer cell has genes that are expressed differently than the same cell from a healty person (Bioteknologinemnda 2009).

The biological information contained in the DNA is encoded in DNA chains of four different nucleotides; Adenine (A), thymine (T), guanine (G) and cytosine (C). These chains are connected in pairs to form a DNA double helix. A human genome contains over 3 billion DNA base pairs, and how these nucleotides are combined in a sequence defines the cell's characteristics.

By studying the human body on a molecular level, it is possible to detect structures and developments in the DNA that can influence the body's ability to recover from a disease, and the probability of relapse. In this chapter, we will introduce gene expression data, copy numbers, methylation data and SNPs which are four different types of genomic and epigenetic datasets that study the DNA structure of various cells in different ways. In the practical examples in this thesis, gene expression data and methylation data will be used. Gene expression data and copy number data contain typically around 30000 genes, methylation data often have around half a million covariates, and there can be several millions single nucleotide polymorphisms (SNPs) measured in datasets in genome-wide association studies (GWAS).

Microarrays and gene expression data

All the cells in the human body contain identical genetic material, but which genes that are active differentiate from cell to cell. Studying which genes that are active and inactive in different cell types can be helpful to understand both how the cells normally function, and if there are differences in the genes when a cell does not perform as expected. Microarrays make it possible to study thousands of genes at the same time, and determine how active they are in the given cell (National Human Genome Research Institute 2011).

A microarray, or a DNA chip, is a plate of glass or similar that is divided into a grid, where in each spot of the grid, a specified DNA sequence from a cell sample will be attached. This DNA sequence is called a probe and each probe is associated with a particular gene. A microarray can include hundreds of thousand different probes, and is often studied to determine if cells have recognizable patterns during the course of a disease. To detect the differences in the genes of healthy cells and sick cells, DNA from the two distinct sources is often compiled in one microarray, but with different colors. The two sources can for instance be biopsies from the same organ from a healthy person and a person with cancer.

The most common usage of a microarray is in gene expression data. For a gene expression microarray, a much used method to distinguish the expression levels from the different cells is to color, or label, the samples with fluorescent dye Cy3 (green) and Cy5 (red). Through a process called hybridization, the labeled mRNA will get attached to the perfect complementary DNA (cDNA) on the microarray. The amount of color that attaches each spot is proportional to the amount of mRNA in the sample, and if a gene is strongly expressed in one of the samples and not in the other, the color of this probe will be strong. On the other hand, if the gene is expressed the same way in both the sample from the healthy and the tumor cell, the color of the probe will be a combination of the two labeling colors. With this type of comparison of two different samples, a colormap will illustrate clearly which genes that are expressed differently in the tumor cell compared to the healthy cell (Oregon State University 2013). This process of how a microarray is made is shown in Figure 2.1.

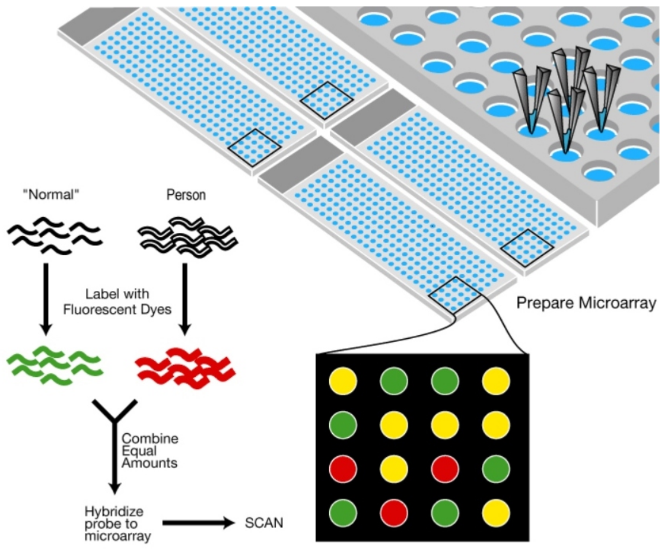


Figure 2.1: A microarray is made by combining samples from a healthy ("normal") person and, for instance, a person with cancer into one grid where mRNA from the different samples are connected to its corresponding spot. Here, a red color on the microarray means that this gene is expressed in the cancer cell, and not in the healthy cell. A yellow spot means that both the cells contain the same amount of mRNA for that specific gene. Usually, microarrays will also include different shades of the colors, representing how much mRNA that is activated. This illustration is from El Camino Hospitals webpage (04.2014).

Gene expression data is usually a matrix made from DNA microarrays from a biopsy, where the colors in a microarray is translated to numbers that describe the intensities, and each element of the matrix reflects the activity of one particular gene. One row of the gene expression matrix corresponds to one sample, and several samples are combined into one matrix. In this thesis, gene expression data are used in both the lasso with linear regression example in Chapter 5 and the lasso with Cox regression example in Chapter 6.

aCGH - array Comparative Genomic Hybridization

Datasets with so called copy numbers are also frequently used to study the DNA. Alterations in the DNA may lead to changes in the number of copies of a gene (Røine, 2013). These mutations may be deletions, insertions, or duplications of the chromosome, or part of the chromosome. The number of copies of a gene influence how the genes are regulated in the cell. Thereby, the number of copies can be related to diseases such as cancer. Remark that mutations in the chromosome is not necessarily due to diseases, there are also normal variations among people.

Array Comparative Genomic Hybridization (aCGH) is used to measure the copy number alterations of a gene. This uses the same colors and methods as microarrays; greater intensity in one spot of the array indicates that there are many copies of this particular gene. It is useful to measure if there are less or more copies than normal, therefore datasets in aCGH is often on log_2-scale (National Cancer Institute 10.2014). This means that the normal value of 2 copies will be represented as 0. A positive value means that there is a gain of copy numbers, while a negative number shows that there is less than 2 copies of the gene represented in this particular spot of the array. We have not studied aCGH data in this thesis, but included the description for completeness.

Methylation data

DNA methylation refers to the addition of a methyl (CH3) group to the cytosine (C) and guanine (G) nucleotides (Mandal 2014). More specific, the methyl can bind to C, but only if C is followed by G (CH3 - CG). The exact role of methylation in the gene expression is currently unknown, but proper DNA methylation appears to be essential for cell differentiation and embryonic development. It is a recurring phenomenon that if a gene is methylated, it is not expressed. It has also been discovered that although the overall methylation levels are similar in different humans, there are significant differences in the methylation levels between different tissue types and between normal cells and cancer cells from the same tissue (Phillips 2008). The methylation level also changes with age.

In Section 8.3, we study the so called beta values for a methylation dataset. On one DNA strand there are half a million probes, and in one cell sample there are several DNA strands. A probes' beta value is a ratio measure describing how many DNA strands in a sample that has the particular probe methylated. There is one beta value for each probe and it takes values between 0 and 1, where 0 indicates that none of the probes in this cell sample are methylated, and 1 corresponds to that all probes are methylated. The numbers in between can be thought of as "how much on" the probe is, and describes the proportion of probes that are methylated in the cell sample.

SNPs

Over the last years there has been an increase in usage of datasets from Genome-wide association studies (GWAS) which aim to find strong associations between what is called SNPs and observable characteristics or traits (phenotypes) in a set of individuals. The goal is to identify genetic risk factors for diseases that are common in the population.

A single-nucleotide polymorphism (SNP) is a generic variation in some location of the genome and involves a nucleotide that differs across different individuals. An example

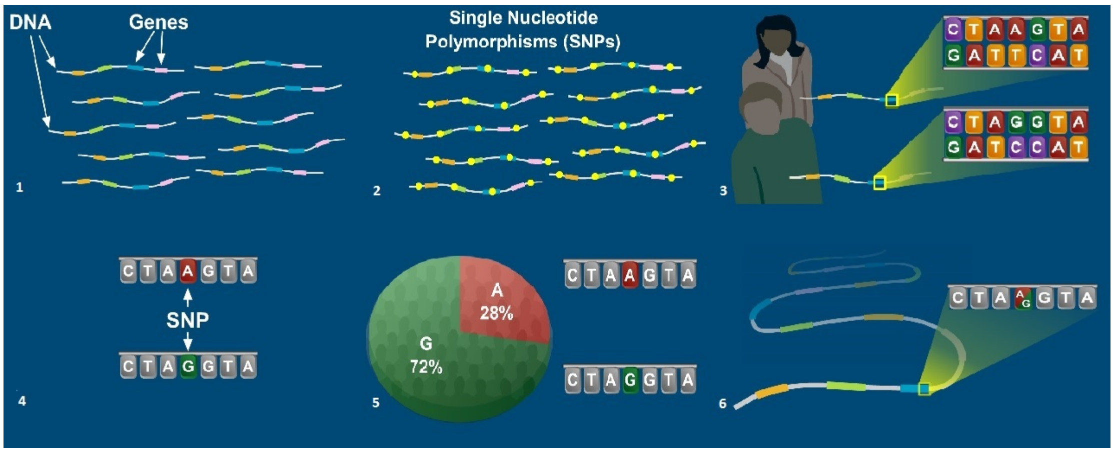


Figure 2.2: A SNP is a variation in a DNA sequence that is present in a large part of the population. ***1:*** *DNA sequences and genes.* ***2:*** *A difference in nucleotides of a large part of the population can be found.* ***3:*** *Two short DNA sequences taken from the same region of the genome of two different persons.* ***4:*** *In the first line of the DNA sequence in picture 3, one nucleotide position had different values for the two persons.* ***5:*** *Only single-nucleotide substitutions in the DNA sequence that are present in a large part of the population are called SNPs.* ***6:*** *At this position of the DNA, a SNP is located. Scientists estimate that there are at least 10 million SNPs in the human DNA. The pictures contains a collection of illustrations taken from the National Human Genome Research Institutes webpage (04.2014).*

is GAA$\underline{\text{T}}$CGT and GAA$\underline{\text{C}}$CGT where the DNA frequence is collected from different individuals. In most cases, there are two different nucleotides for a variant, and the most frequent is often called "0", and the less frequent "1". SNPs are commonly genetic variations, and many SNPs are present in a large proportion of the human population. The less frequent nucleotide variant can for example have a frequency of 40%, i.e that 40% of the population have this variation and 60% have the most frequent variation (Zhang et. al 2012, and Bush and Moore 2012). It is often the less frequent nucleotide that is of interest, because it can be tested if this variation has any association with a common disease. If less than 1% of the population has the less frequent variant, this variant is called a mutant and this is not defined as a SNP (Sjøberg 2006). See Figure 2.2 for an illustration on how SNPs are found.

A SNP is a modern unit of genetic variation and is used more and more in genetic studies. Datasets from GWAS often have one or two million genetic factors. Scientists estimate that our DNA contains approximately 10 million SNPs, but not all of these are detected yet. We have not studied SNPs in this thesis, but the description is, together with the description of aCGH, included for completeness.

Chapter 3

Regression models and penalization

The practical aspects of lasso, cross validation and preselection bias will be studied for both linear regression and Cox regression throughout this thesis. In this chapter we will introduce the underlying theory concerning high dimensional regression problems calling for regularized regression methods.

In many different situations in real life, an outcome of an event depends on one or more variables or covariates. The covariates can influence the response to different degree. Some examples are how age, gender and car type influence the risk of car accidents, how different genes influence how exposed a person is for cancer, or how the blood pressure and age or weight are connected. This can be modeled statistically by introducing covariates x_j, $j = 1, ..., p$, where p is the number of covariates. The coefficients, which indicate how much weight each covariate should have, can be denoted as β_j. Let y_i be the response with $i = 1, ..., N$, where N is the number of observations. When fitting a model to a dataset, there will always be an error since a model can never fully cover all the uncertainty that exists in real life. This error is denoted ϵ_i with $E(\epsilon_i) = 0$.

3.1 Linear regression

In linear regression, the expected response is assumed to be a linear combination of the covariates. By using matrix representation, the linear regression model can be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{3.1}$$

where $\boldsymbol{X}$ is a $N \times (p+1)$ matrix of covariates, $\boldsymbol{y}$ is a vector of responses, $\boldsymbol{\epsilon}$ is a vector of errors, and $\boldsymbol{\beta}$ is a vector of parameters to be estimated, including the intercept β_0. More detailed, formula (3.1) can be written

$$\begin{bmatrix} y_1 \\ . \\ . \\ . \\ y_N \end{bmatrix} = \boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon} = \begin{bmatrix} 1 & x_{11} & ... & x_{1p} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ 1 & x_{N1} & ... & x_{Np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ . \\ . \\ . \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}.$$

The estimated response will be $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$, where the estimated coefficients $\hat{\boldsymbol{\beta}}$ are found through some estimation procedure. These coefficients can furthermore be used to predict the response for a new set of data with new values for the covariates.

An ordinary regression problem, with a larger number of observations than covariates ($p \leq N$), can be solved with the ordinary least squares method (OLS). This is done by minimizing the sum of the residual squared error, or in other words, minimizing the squared difference between the true value $\boldsymbol{y}$ of the response, and the estimated response $\hat{\boldsymbol{y}}$. In the linear regression setting, the least squares estimate is given by

$$\hat{\boldsymbol{\beta}}^{OLS} = \arg\min\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2\}$$

or, with matrix notation,

$$\hat{\boldsymbol{\beta}}^{OLS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

When the noise terms ϵ_i are normally distributed, minimizing the squared error is equivalent to maximizing the likelihood, as will be used when introducing the lasso, giving $\hat{\boldsymbol{\beta}} = \arg\min\{-l(\beta_0, \boldsymbol{\beta})\}$ where l is the log likelihood function.

3.2 Cox regression

In many cases, linear regression is not the proper way to model how the covariates and the response are connected, and there exists a large number of alternative regression methods. In medical research, binary responses are often of interest. Whether a patient is sick or not is a binary response, and it can be tested if the response is affected by some covariates of interest, as the type of medicine or the age of the patient. This can be modeled through a logistic regression model.

Another response common within medical research, is survival time. This will be the focus in this thesis along with the linear response that has already been introduced. Examples of survival data can be time between births, time until divorce, or time from an individual gets cancer treatment until relapse or death. To keep it relatively simple, we will consider time until death from a disease (as for example cancer) in the subsequent description.

Right censored survival data

Right censored survival data is data where the starting point for entering the study is the same for all patients, but the time of death is not always known. A patient's starting point can be time of the diagnosis of an illness, or time of a cancer treatment. The stopping point of an observation, on the other hand, is not always well defined. If observation of the patients is terminated before anything specific is known for all the patients, the survival times are incomplete. Either a death due to the diagnosis that is studied (the event of interest) is observed, or the observation of the patient is ended before anything about the time or cause of death is known. When the observation is terminated before the actual event (death) has happened, the observation is said to be censored. An observed death of other causes than what is studied is also defined as a censored observation.

Survival data like this require special methods to handle the censored observations properly. Several advanced methods exist, but we will focus on the Cox proportional hazard model which is the most common model for right censored survival data (see Aalen et al. (2008)).

Definitions and assumptions

In the following, we will introduce some standard terminology in survival analysis. The survival function $S(t)$ is defined as the probability that the survival time T is larger than t,

$$S(t) = Pr(T > t).$$

For a regression model to fit this framework, a collection of N individuals is registered, and for individual $i \in [1, N]$, $\mathbb{N}_i(t)$ is the counting process that counts the observed occurrences of an event of interest in $[0, t]$. For survival data, it is assumed that there is only one event for each individual, thereby $\mathbb{N}_i(t)$ will only take the values 0 or 1. The intensity process of the counting process $\mathbb{N}_i(t)$ can be written as

$$\eta_i(t) = Y_i(t)\alpha(t|\boldsymbol{x}_i), \tag{3.2}$$

where $Y_i(t)$ is an indicator that gives the value 1 if individual i is under observation just before time t, and zero otherwise. $\alpha(t|\boldsymbol{x}_i)$ is the hazard rate, and $\boldsymbol{x}_i(t) = (x_{i1}(t), x_{i2}(t), ..., x_{ip}(t))^T$ is a vector of covariates that, in general, may be time-dependent. The intensity process $\eta_i(t)$ is the conditional probability that an event occurs for individual i in a small time interval $[t, t + dt)$, and can also be written as $\eta_i(t) = Pr(d\mathbb{N}_i(t) = 1|\mathscr{F}_{t-})$. Here, $\mathscr{F}_{t-}$ is defined as the past and $d\mathbb{N}_i(t)$ denotes the number of jumps of the process in the small time interval $[t, t + dt)$.

The aggregated counting process is defined as $\mathbb{N}_*(t) = \sum_{i=1}^{N} \mathbb{N}_i(t)$ with intensity process

$$\eta_*(t) = \sum_{i=1}^{N} \eta_i(t) = \sum_{i=1}^{N} Y_i(t)\alpha(t|\boldsymbol{x}_i),$$

assuming no joint events. This assumption is reasonable because the time steps can be defined to be infinitesimally small, i.e. dt is a very small number. When the jumps of the counting process occur randomly and independent of each other, the homogeneous Poisson process can be used to model the counting process.

The hazard rate $\alpha(t|\boldsymbol{x}_i)$ in (3.2) is the probability that an individual who has not yet experienced an event, has an event in the next small time interval. More formally, the hazard rate is defined as

$$\alpha(t) = \lim_{dt \to 0} \frac{1}{dt} Pr(t \leq T < t + dt | T \geq t).$$

The hazard rate can be modeled by a regression model. In the Cox regression model, the hazard rate of an individual is on the form

$$\alpha(t|\boldsymbol{x}_i) = \alpha_0(t) exp(\boldsymbol{\beta}^T \boldsymbol{x}_i(t)), \tag{3.3}$$

where $\boldsymbol{x}_i(t) = (x_{i1}(t), ..., x_{ip}(t))^T$ is the vector of covariates for individual i at time t and $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)^T$ is the vector of regression coefficients.[1] $\alpha_0(t)$ is the baseline hazard and describes the hazard for an individual with all covariates equal to zero. The remaining part of the right hand side is the hazard ratio, or the relative risk. The hazard ratio describes the size of the hazard rate that depends on the covariates.

There are some additional assumptions on the hazard rate in Cox regression. It is assumed

1. **Log-linearity:**
$$log\{\alpha(t|\boldsymbol{x})\} = log\{\alpha_0(t)\} + \boldsymbol{\beta}^T \boldsymbol{x}$$
This means that the effect of increasing x by 1 is the same for all values of x.

2. **Proportional hazards:**
$$\frac{\alpha(t|\boldsymbol{x}_2)}{\alpha(t|\boldsymbol{x}_1)} = exp\{\boldsymbol{\beta}^T(\boldsymbol{x}_2 - \boldsymbol{x}_1)\},$$
which says that the ratio is constant over time.

[1] Remark that in Cox regression it is not possible to estimate the value of the intercept β_0. The intercept is a part of the baseline function $\alpha_0(t)$ which is non parametric.

Partial likelihood

In the Cox proportional hazard model, the parameters $\boldsymbol{\beta}$ are estimated through the partial likelihood, instead of the ordinary likelihood. This is necessary because the hazard is semi parametric since it is composed by $exp(\boldsymbol{\beta}^T\boldsymbol{x}_i(t))$, which is parametric, and the non parametric baseline hazard $\alpha_0(t)$. The partial likelihood is found taking the conditional probabilities of observing an event for individual i at time t, given the past and that an event is observed at time t, and multiplying the conditional probabilities for the uncensored observations. With $\mathscr{F}_{t-}$ defined as the past, the conditional probability here is

$$\begin{aligned}
\pi(i|t) &= Pr(d\mathbb{N}_i(t) = 1|d\mathbb{N}_*(t) = 1, \mathscr{F}_{t-}) \\
&= \frac{Pr(d\mathbb{N}_i(t) = 1|\mathscr{F}_{t-})}{Pr(d\mathbb{N}_*(t) = 1|\mathscr{F}_{t-})} \\
&= \frac{\eta_i(t)}{\eta_*(t)} \\
&= \frac{Y_i(t)exp(\boldsymbol{\beta}^T\boldsymbol{x}_i(t))}{\sum_{l=1}^N Y_l(t)exp(\boldsymbol{\beta}^T\boldsymbol{x}_l(t))}.
\end{aligned} \tag{3.4}$$

Thereby, if i_j is the index of an individual who experiences an event at time T_j (i.e. the index of an uncensored individual), the partial likelihood becomes

$$\begin{aligned}
L(\boldsymbol{\beta}) &= \prod_{T_j} \pi(i_j|T_j) \\
&= \prod_{T_j} \frac{Y_{i_j}(T_j)exp(\boldsymbol{\beta}^T\boldsymbol{x}_{i_j}(T_j))}{\sum_{l=1}^n Y_l(T_j)exp(\boldsymbol{\beta}^T\boldsymbol{x}_l(T_j))} \\
&= \prod_{T_j} \frac{exp(\boldsymbol{\beta}^T\boldsymbol{x}_{i_j}(T_j))}{\sum_{l\in\mathscr{R}_j} exp(\boldsymbol{\beta}^T\boldsymbol{x}_l(T_j))}
\end{aligned} \tag{3.5}$$

where $\mathscr{R}_j = \{l|Y_l(T_j) = 1\}$ is the set of persons at risk at time T_j. Using the partial likelihood (3.5), the partial log likelihood for the Cox proportional hazard model is defined as

$$\begin{aligned}
l(\boldsymbol{\beta}) &= logL(\boldsymbol{\beta}) \\
&= \sum_{T_j} \{\boldsymbol{\beta}^T\boldsymbol{x}_{i_j}(T_j) - log(\sum_{l\in\mathscr{R}_j} exp(\boldsymbol{\beta}^T\boldsymbol{x}_l(T_j))\}.
\end{aligned} \tag{3.6}$$

Remark that this will only sum over uncensored events since i_j is the index for an individual that experience an event of interest at time T_j.

For more on the theory around survival analysis and Cox regression, see Aalen et al. (2008).

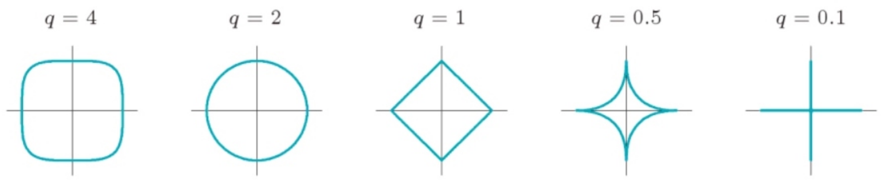


Figure 3.1: The contours of $\sum_j |\beta_j|^q$ for given values of q (in two dimensions, i.e $p = 2$, and we have only two different coefficients, β_1 and β_2). This figure is from the book The Elements of Statistical Learning by Hastie et. al (2009).

3.3 Regularization for regression models

When the number of covariates is larger than the number of observations ($p > N$), the ordinary methods to find the estimated coefficients can not be used to find a unique solution to the regression problem. In this situation, there are p unknown parameters, but only N equations, which leads to an infinite number of solutions. To be able to find a unique solution, regularization of the optimization problem has to be taken into account. This means that we have to introduce restrictions on the coefficients $\boldsymbol{\beta}$ in the regression model (Vidaurre et. al 2013). In this thesis, we will study datasets where the number of covariates p is much larger than the number of observations N ($p >> N$).

Penalized regression

The optimal coefficients in an ordinary regression problem are found by maximizing the likelihood. When restrictions are introduced on the coefficients in a penalized general regression problem, the estimation is performed by minimizing the sum of the negative log-likelihood and a penalty;

$$\hat{\boldsymbol{\beta}} = \arg\min\{-l(\beta_0, \boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_q\}, \tag{3.7}$$

where λ is a penalty parameter and $||\boldsymbol{\beta}||_q = \sum_{j=1}^{p} |\beta_j|^q$ where $q \geq 0$. Different values for q give different shapes on the constraint areas, and Figure 3.1 shows how the constraints will look in the case of two covariates.

Two of the most commonly used penalties are the L_2 and L_1 penalties, which correspond to $q = 2$ and $q = 1$. This is the ridge regression (Hoerl and Kennard 1970) and the lasso (Tibshirani 1996), respectively. The ridge reduces the size of the coefficients (shrinkage), while the lasso also does variable selection by setting some coefficients equal to zero, as will be explained later in this section.

The elastic net (Zou and Hastie 2005), another penalization method, is obtained by combining the ridge and the lasso. The elastic net penalty,

$$\lambda \sum_{j=1}^{p} (\alpha\beta_j^2 + (1-\alpha)|\beta_j|),$$

gives a penalty shape that corresponds to a combination of the two shapes for $q = 2$ and $q = 1$ in Figure 3.1, depending on a parameter α that defines the relationship between the L_1 part and the L_2 part of the constraint.

The "naive" elastic net is defined as

$$\hat{\boldsymbol{\beta}}^{elastic} = \arg\min\{-l(\beta_0, \boldsymbol{\beta}) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} |\beta_j|^2\},$$

where the L_1 part of the penalty generates a model with some coefficients set equal to zero (a sparse model). The quadratic part of the penalty, corresponding to the ridge part, stabilizes the L_1 regularization path, removes the limitation on the number of selected variables and encourages grouping effect.

The lasso, i.e the L_1 regularization for the regression problem, will be the constraint used in this thesis.

Lasso

The lasso was introduced by Robert Tibshirani (1996) and is short for 'least absolute shrinkage and selection operator'. The lasso does variable selection by setting some coefficients equal to zero. Coupled with cross validation to find the penalty parameter λ optimal for prediction, lasso is able to make a sparse model that also has good prediction abilities. The lasso solution of high dimensional regression problems is found by putting $q = 1$ in (3.7), hence

$$\hat{\boldsymbol{\beta}}^{lasso} = \arg\min\{-l(\beta_0, \boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1\}, \tag{3.8}$$

where $||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$. In linear regression an equivalent formulation for the penalized regression problem is

$$\hat{\boldsymbol{\beta}} = \arg\min \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2,$$
$$\text{subject to } \sum_{j=1}^{p} |\beta_j|^q \leq t,$$

where $q = 1$ for the lasso. Here t has the same role as λ in (3.8). The same idea as the ordinary least squares method is used when fitting a model in the penalized regression

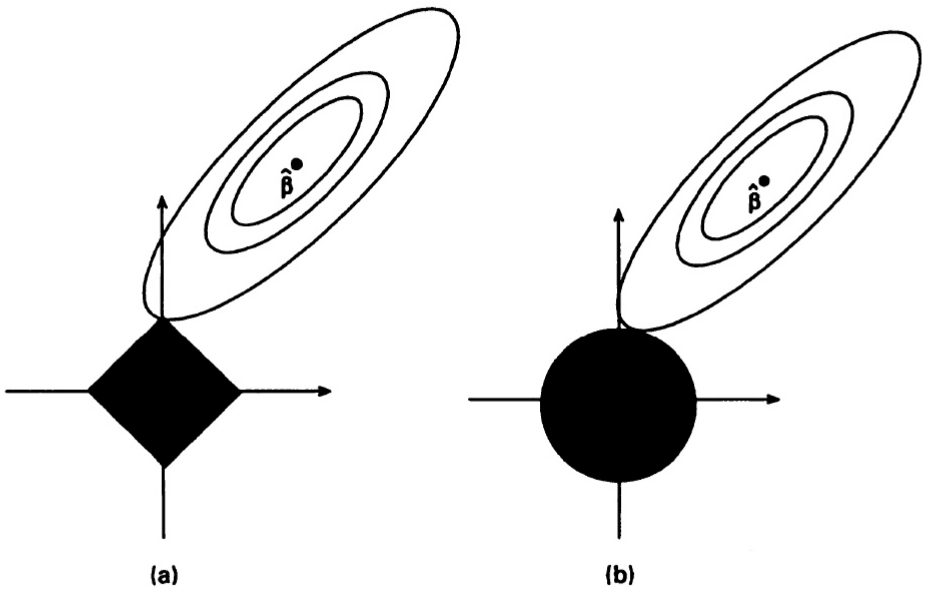


Figure 3.2: Illustration of how lasso (a) and ridge (b) estimation work in the case of linear regression. The figure is from the original paper on lasso of Tibshirani (1996) and illustrates the two dimensional case. $\hat{\boldsymbol{\beta}}$ *is the least squares estimates and the black square and circle is the constraints for the lasso and the ridge, respectively.*

method lasso for linear models, but in addition to minimizing the squared residuals, the sum of the absolute value of the estimates is constrained to be less than a given number.

Figure 3.2 illustrates the difference between lasso and ridge in the case of linear regression. The black square and circle are the penalty (constraint) regions, which are $|\beta_1| + |\beta_2| \leq t$ for lasso, and $\beta_1^2 + \beta_2^2 \leq t$ for ridge. The ellipses are the contours of the least squares error function (Hastie et al. 2009). The point in the middle of the ellipses, $\hat{\beta}$, is the least squares estimate. The solution of the penalized regression problem is defined to be the point where the ellipses and the constraint regions intersect. For the lasso, the figure illustrate that $\beta_1 = 0$ when the black square and the largest ellipse coincide, while for ridge, the ellipse will rarely coincide with the black penalty circle at a point where $\beta_1 = 0$. Therefore, this illustration shows that the ridge only shrinks the model coefficients, while the lasso, in addition to the shrinking, also reduces the number of covariates in a model by setting some coefficients equal to zero.

Lasso for Cox regression is found by using the partial log likelihood (3.6) in the formula for lasso (3.8). More details can be found in Tibshirani (1997).

Adaptive lasso and the group lasso

The original version of the lasso (Tibshirani 1996) is the focus in this thesis, but it is useful to be aware of that there exist several more advanced versions of the lasso. Examples of these are adaptive lasso and group lasso.

The adaptive lasso (Zou 2006) replaces the L_1 penalty in the formula for the lasso (3.8) by a weighted version. The lasso does estimation and variable selection simultaneously, but for certain scenarios Zou (2006) shows that the lasso is inconsistent for variable selection. By introducing adaptive weights when penalizing the different coefficients, the adaptive lasso is consistent and a solution can be found by the same efficient algorithm for solving the lasso. The formula for the adaptive lasso is given by

$$\hat{\boldsymbol{\beta}}^{adapt}(\lambda) = \arg\min\{-l(\beta_0, \boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} w_j |\beta_j|\},$$

where $\boldsymbol{w} = \frac{1}{|\hat{\boldsymbol{\beta}}_{init}|^\gamma}$ is the weight. $\hat{\boldsymbol{\beta}}_{init}$ is an initial estimator and $\gamma > 0$ is a fixed parameter. The lasso can be used as a first step to find the initial estimator, i.e $\hat{\boldsymbol{\beta}}_{init}(\hat{\lambda}_{init})$ can be found through cross validation and (3.8).

The grouped lasso (Yuan and Lin 2006) can be used to solve high-dimensional regression problems where the parameter vector $\boldsymbol{\beta}$ is structured in groups. More information on adaptive lasso, group lasso, and other versions of the lasso can be found in Bühlmann and van de Geer (2011).

The penalty parameter λ for the lasso

The penalty parameter λ in a penalized regression model like the lasso is a tuning parameter that in the end determines how many of the covariates that are selected in the fitted model. The size of λ decides the degree of shrinkage of the coefficients, and when λ gets larger, the number of non-zero coefficients gets smaller because a number of the coefficients are drawn towards zero.

In this thesis, we are interested in the penalty parameter that gives the best prediction, and will therefore apply cross validation to find the optimal λ for this purpose. The result of a large λ will be low variance, but large bias on the βs, because they will be heavily shrunk. On the other hand, when λ is small, the variance increases, but the bias decreases. With cross validation the λ that gives the optimal balance between variance and bias on the βs can be found, as will be explained in the following section.

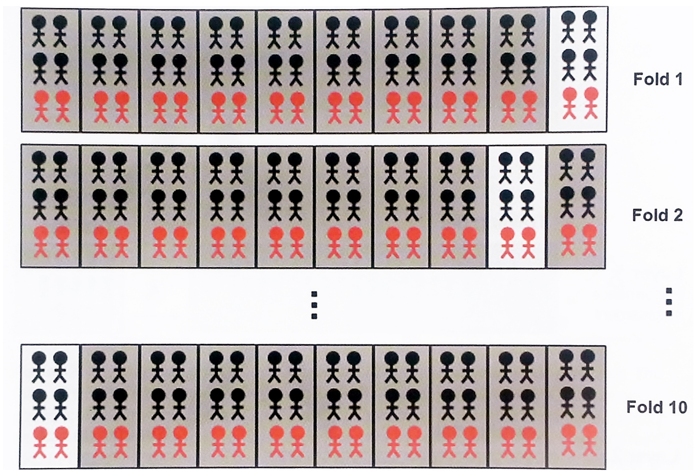


Figure 3.3: 10-folds cross validation. For each iteration, the white box illustrates the validation group of patients, while the grey boxes illustrate the training set where the model is fitted. In studies where both sick patients and healthy references are included, both must be included in the validation group as well as the training set. This illustration is found in Lee (2010).

3.4 Cross validation

K-folds cross validation is a method to find the optimal penalty parameter λ in penalized regression, where λ determines the amount of shrinkage in the model. The cross validation method is illustrated in Figure 3.3 and can be explained as following:

- Divide the observations in K groups
- For different values of the tuning parameter λ:
  - For $k \in [1, K]$, do the following:
    - Put aside the kth group of subjects (patients), called the validation group.
    - Fit a model with the K-1 groups that are left, i.e find the estimated coefficients $\hat{\boldsymbol{\beta}}^{-k}(\lambda)$.
    - Use the model to predict the response $\hat{\boldsymbol{y}}^*$ for the validation group.
    - Compare the predicted response $\hat{\boldsymbol{y}}^*$ with the observed value of the response $\boldsymbol{y}^*$ in the validation group.
  - Find the average prediction error over the K groups.
- Choose the λ that minimizes the average prediction error, e.i. the λ that gives coefficients $\hat{\boldsymbol{\beta}}^{-k}(\lambda)$ that predict the observed value of the response the best over all K groups.

Cross validation for linear regression problems

More precisely, for linear regression, the cross validation function can be written

$$\begin{aligned} CV(\lambda) &= \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in k\text{th part}}(y_i^* - \hat{y}_i^*(\lambda))^2 \\ &= \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in k\text{th part}}(y_i^* - \boldsymbol{x_i}^*\hat{\boldsymbol{\beta}}^{-k}(\lambda))^2, \end{aligned} \tag{3.9}$$

where y_i^* is the observed response in the current validation group and $\hat{y}_i^*(\lambda)$ is the estimated response found from the covariates $\boldsymbol{x_i}^*$ from the current validation group and a model $\hat{\boldsymbol{\beta}}^{-k}(\lambda)$ made by data without the validation group k. After evaluating several different values for λ, the tuning parameter that minimizes $CV(\lambda)$ is chosen. More theory on this subject is found in Hastie et al. (2009) and the slides of Hastie and Tibshirani from 2009 (http://statweb.stanford.edu/ tibs/sta306b/cvwrong.pdf, visited 02.2014).

The main focus when fitting a regression model can be either variable selection or prediction, and this will determine which model is decided to be the best one. The cross validation uses the prediction ability to find the optimal model. Therefore, by applying cross validation to determine the penalty parameter λ in lasso, it is automatically assumed that the aim is prediction.

Cross validation for Cox's proportional hazard model

The idea of tuning a model by leaving out different folds of observations is the same for Cox regression as explained previously in this section. For a linear regression model, the residual sum of squares $\sum(y_i - \boldsymbol{x}_i\hat{\boldsymbol{\beta}})^2$ is, apart from a constant, equal to $-2loglikelihood$. Similarly, the predicted sum of squares $\sum_{k=1}^{K}\sum_{i\in k\text{th part}}(y_i^* - \boldsymbol{x}_i^*\hat{\boldsymbol{\beta}}^{-k})$, used in (3.9), is connected to the cross validated log likelihood.

From Verweij and van Houwelingen (1993), we get that by denoting $l(\boldsymbol{\beta})$ to be the log likelihood, and defining $l^{-k}(\boldsymbol{\beta})$ as the log likelihood when the kth fold is left out, we can define

$$l^k(\boldsymbol{\beta}) = l(\boldsymbol{\beta}) - l^{-k}(\boldsymbol{\beta})$$

to be the contribution of fold k to the log likelihood. This is a general result that holds for all likelihood functions. If the terms in the likelihood were independent, as in a linear model, $l^k(\boldsymbol{\beta})$ would simply sum up to be the likelihood, $\sum l^k(\boldsymbol{\beta}) = l(\boldsymbol{\beta})$. It is defined that $\hat{\boldsymbol{\beta}}^{-k}$ is the value of $\boldsymbol{\beta}$ that maximizes $l^{-k}(\boldsymbol{\beta})$. The general cross validated log likelihood CVL is then given by

$$CVL = \sum_{k=1}^{K} l^k(\hat{\boldsymbol{\beta}}^{-k}).$$

In order to get the K-folds cross validation formula for a Cox model, the partial log likelihood (3.6) must be applied. Since the terms in the Cox partial log likelihood are not independent, cross validation for estimating the penalty parameter λ in a regularized Cox model is a bit more complicated than for the penalized linear regression. The cross validation formula for Cox's proportional hazard model is given by

$$\begin{aligned} CVL(\lambda) &= \sum_{k=1}^{K} l^k(\hat{\boldsymbol{\beta}}^{-k}(\lambda)) \\ &= \sum_{k=1}^{K} \{l(\hat{\boldsymbol{\beta}}^{-k}(\lambda)) - l^{-k}(\hat{\boldsymbol{\beta}}^{-k}(\lambda))\}, \end{aligned} \tag{3.10}$$

and the optimal penalty parameter λ is obtained by maximizing $CVL(\lambda)$, as explained in Bøvelstad et al. (2007). This means that the contribution of all the K folds in the cross validation is studied. The λ that makes all the folds contribute to the partial log likelihood as much as possible, and thereby maximizes the cross validated partial log likelihood, is chosen.

3.5 Programming in R: glmnet

For the practical analyses with the lasso in this thesis, the package *glmnet* (Friedman et. al 2013) in *R* is used. In this package there is a parameter $alpha$ which is called the elastic net mixing parameter. This parameter takes values between 0 and 1 and determines if the penalty in formula 3.7 should be L_1, L_2, or a combination of the two. The penalty is defined as

$$\frac{1-\alpha}{2}||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1,$$

and $alpha = 0$ in *glmnet* gives the ridge penalty, while $alpha = 1$ will give the lasso penalty (default).

To get the lasso solution with coefficients from *glmnet*, the penalty parameter λ must be supplied. The optimal value of the penalty parameter λ is found from cross validation and the function *cv.glmnet*. This does cross validation with $nfolds$ number of folds, where the default is 10 folds.

By specifying the parameter $family$ in *glmnet*, the lasso solution for several different regression models can be found. Particularly, the algorithm for the lasso for Cox regression is used by specifying $family ='' cox''$, and applying the response as a *Surv* object. A more detailed explanation for how lasso for Cox regression is done in *glmnet* can be found in Appendix B2.

Glmnet uses the sequential Strong algorithm to speed up the computations. This discards covariates prior to the lasso analysis, but does not introduce preselection bias, as will be explained in Chapter 7.

There exist other packages that implements the lasso, and the least-angle regression algorithm (LARS) is another famous package made by Efron et al. (2004). Because of differences in the implementations, the results from *glmnet* and LARS may differ.

Chapter 4

Preselection and preselection bias

The number of covariates available in many high dimensional regression situations, is so huge that it is useful to reduce the dimension of the dataset before performing a lasso analysis. Sometimes this data reduction is performed because the data file is so large that it might be hard to read it into programs like R, but most often the reduction of covariates is done in order to reduce the computation time. In the following chapters, we will study how this dimension reduction affects the results in analyses like the lasso. We define p_{all} to be the size of the full dataset, while $p < p_{all}$ is the size of the preselected set of covariates.

4.1 Preselection criterion

Often, a preselection of covariates (also called prescreening or pre-filtering) seems to be done quite arbitrarily, using some preselection criterion, but not thinking to what consequences the reduction of the number of covariates might lead to in the subsequent analyses. An illustration on how covariates can be preselected can be seen in Figure 4.1. Covariates as for example gene expression data is collected from N samples. A sample or an observation often corresponds to one patient, but observations can also include samples from a tumor tissue and a healthy tissue from the same person. The covariates are combined into a Nxp_{all} matrix and sorted according to some preselection criterion before only a reduced number of covariates are used in the subsequent analyses.

Very large sets of data is common in genomics, as described in Chapter 2, and preselection of covariates is therefore often done in this type of datasets. For instance, Cho et al. (2010) performs dimension reduction by studying a single SNP at a time and test if it is correlated to the response. Then they discard SNPs with a weak correlation. They claim that the number of SNPs remaining after the preselection can be determined to avoid computational concerns, and have in their paper reduced the number of SNPs in

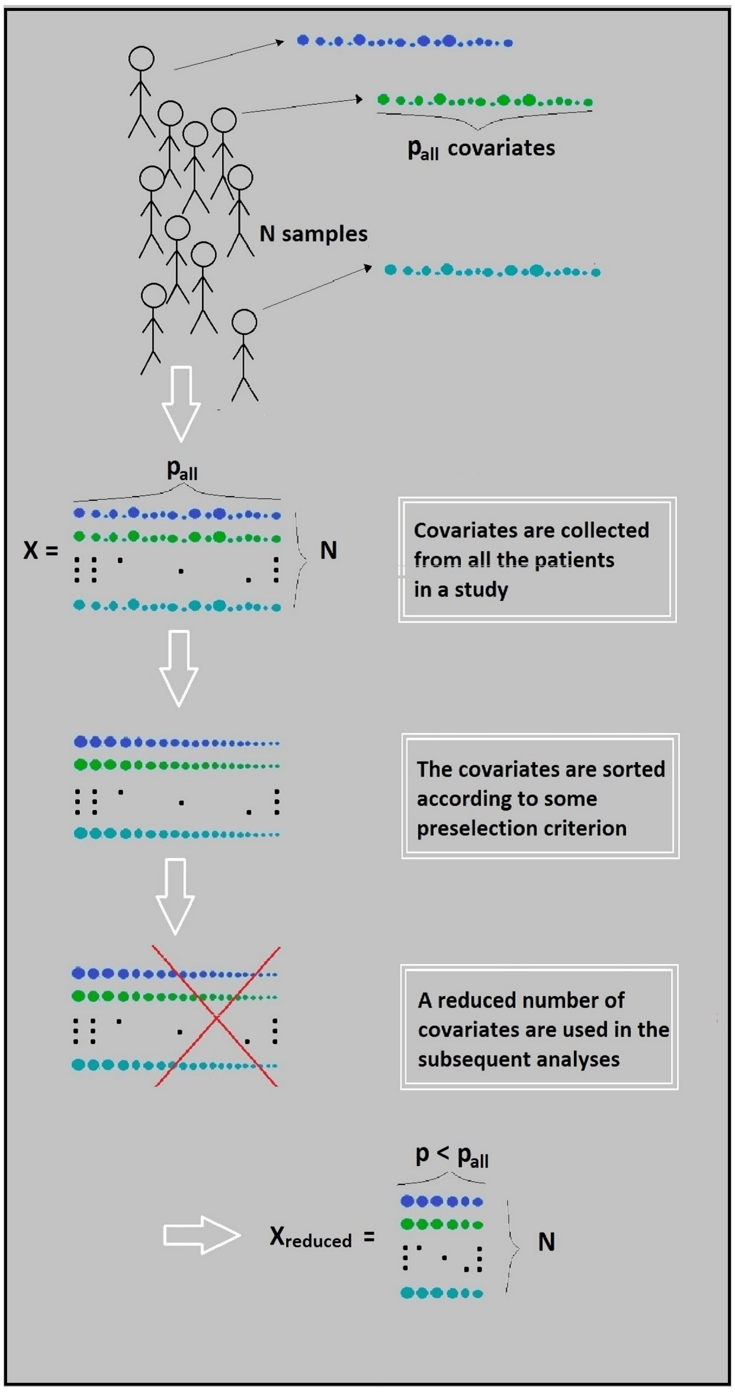

p_{all} covariates
N samples
p_{all}
X =
N
Covariates are collected from all the patients in a study
The covariates are sorted according to some preselection criterion
A reduced number of covariates are used in the subsequent analyses
$p < p_{all}$
$X_{reduced}$ =
N

Figure 4.1: Illustration of now how preselection of covariates can be done. For practical reasons, one dot in this illustration can for instance represent 1000 covariates. The covariates are visualized as dots of different sizes, such that it is possible to see if the covariates are sorted or not.

their dataset from $p_{all} = 327872$ to $p = 1000$ with no further comments.

The correlation between the different covariates and the response is a common preselection criterion when working with linear regression. Studying the correlation between the response and one covariate at a time is, in fact, equivalent with sorting covariates after their P-value from univariate linear regression. A linear regression model can look like

$$y = a + bx,$$

where y is the response, a is the intercept (a constant), b is a coefficient, and x is a covariate. This univariate linear regression is done for each of the covariates in the dataset, and the P-value of b indicates if the value of b is significantly different from zero, i.e if the covariate is correlated with the response or not. The covariates are sorted according to the correlation, and the set of covariates having the highest correlation (in absolute value), or the lowest P-value, are thereafter used in the multiple regression analysis. The challenging question is how many of the sorted covariates should be included in the subsequent analysis in order to obtain trustworthy results?

Another example, in a Cox regression setting, is as in Waldron et. al (2011) where they apply a univariate pre-filter to reduce the dimension of the dataset. They compute a P-value for each covariate by the logrank test, and filter such that only the covariates with a small P-value are used in further computations (they used $P < 0.1$, $P < 0.3$ and $P < 0.5$).

The theory concerning Cox regression is more complicated than linear regression because of the partial likelihood, but the univariate P-values can still be used to preselect covariates before the lasso analysis. By including only one covariate at the time in the survival analysis with the Cox proportional hazard model, a P-value for each of the covariate's coefficient is calculated. A prioritized list of which covariates that should be included in the lasso analysis is found by sorting the coefficients according to their P-value. A low P-value indicates that this covariate is significant when modelling the response. In Waldron et al. (2011), they advise the reader to use caution when applying this so called univariate pre-filter, especially when working with the L_1 penalty (here, combined with L_2 in the elastic net), as the preselection might worsen the prediction. This general effect of preselection is what we will study in the following sections.

4.2 The problem of preselection bias

As we will demonstrate in the following chapters, preselecting a small number of covariates to be included in the lasso analysis will usually result in a low value of the penalty parameter λ. With a weak penalty, many covariates will be included in the final model from the lasso. In general regression, when a model at most has as many covariates as observations, the response can be replicated almost perfectly by adjust-

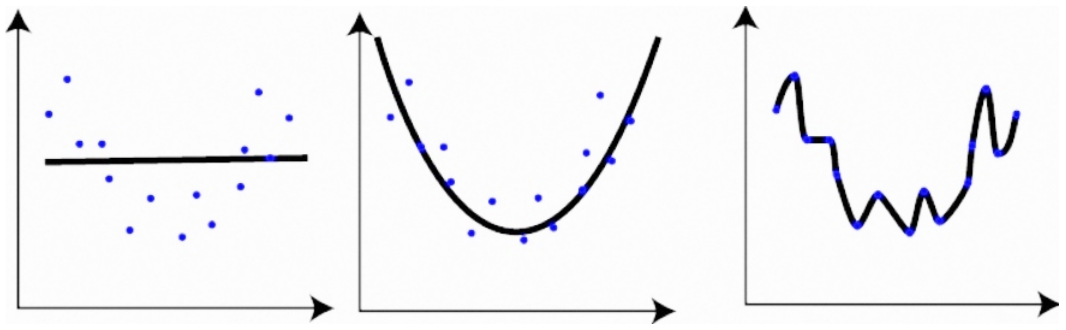


Figure 4.2: Visualization of underfitting (first plot) and overfitting (last plot) in a two dimensional regression problem. The plot in the middle shows the linear regression model that balance bias and variation the best because it captures the trend of the dataset, but not the random noise. This figure is from the webpage The Shape of Data (04.2014).

ing the corresponding coefficients. This concept is called overfitting, and an overfitted model is not likely to predict well for a new dataset. The reason for this is that the overfitted model will not only capture the trend (signal) of the response, but also the random noise in the dataset from which the model was made, as illustrated in the last plot in Figure 4.2. A good model will have a balance between the bias and the variance such that the model will capture as little as possible of the random noise, and as much as possible of the trend of the response.

Reducing the number of covariates in high dimensional data may result in a model that fits very well for the data from which the model was fitted. However, due to overfitting, this might cause large prediction errors on new datasets (Ambroise and McLachlan, 2002). This is an important aspect to be aware of when preselecting covariates. A model should fit as well as possible to the data from which the model was fitted, but often it is even more important how the model can be used for prediction in new datasets. In genomics, the aim can for example be to make a model that is able to predict whether a new patient's probability for survival will increase or decrease if he or she gets a particular cancer treatment.

4.3 Filtering

In some cases, the dimension of a dataset is reduced by discarding covariates that have little variation over the different samples (often patients in genomics). This is done because little variation across the observations indicates that this covariate is unlikely to be able to say anything significant about the response. A possible filter is to discard all covariates which have standard deviation less than for example 0.1. This is not to be confused with what we refer to as a preselection criteria when discussing preselection bias in this thesis. The reason for this is that we wish to shed light on the problems that may occur when we use preselection criteria connected to the relationship between the covariates and the response. Dimension reduction by discarding covariates with low variance only depends on the covariates, and not the response. This will not give the same systematic errors as the preselection bias illustrate (Hastie et al. 2009). We will refer to this type of dimension reduction as filtering.

Chapter 5

Lasso and preselection bias in linear regression

To present an example of the use of lasso and cross validation in high dimensional linear regression problems, we will in this chapter study a dataset from a genomic setting. We will examine the bias caused by preselecting covariates prior to the lasso analysis and thereof also illustrate how the prediction ability of a model changes as the size p of the preselected set of covariates is varied.

5.1 Introduction to the Bone data for linear regression

The dataset studied is a gene expression dataset from Affymetrix microarrays, and we will refer to it as the "Bone data" in the following. All the $N = 84$ patients in the dataset are non-related postmenopausal ethnic Norwegian women between 50 and 86 years of age, and the dataset consists of $p_{all} = 22815$ probes for each patient (after the probes with more than 43% absent calls were removed, as described in Reppe et al. (2010)). As mentioned in Chapter 2, a probe is a DNA sequence that is associated with a particular gene, so in the following, a probe will be denoted a gene. We refer to Reppe et al. (2010) for more details about the dataset and the Affymetrix microarray expression analysis. There are several different responses available in this dataset, and from a medical point of view, the response that is most interesting is the Total Hip T-score, which gives a value for the bone mineral density of a patient. This response can be used to detect which genes stand out for patients with osteoporosis and is what is being analyzed in the paper by Reppe et al. (2010).

Another response in this dataset is the body mass index, BMI, which describes the balance between the height and weight of a person. This type of measure will not distinguish between the weight of fat and the weight of muscles, but will give an indication of whether the patient is overweight or not. The formula for BMI is given by $BMI = w/h^2$, where w is weight in kg and h is the height in meter. A BMI that exceeds 25 will indicate that the patient might be overweight.

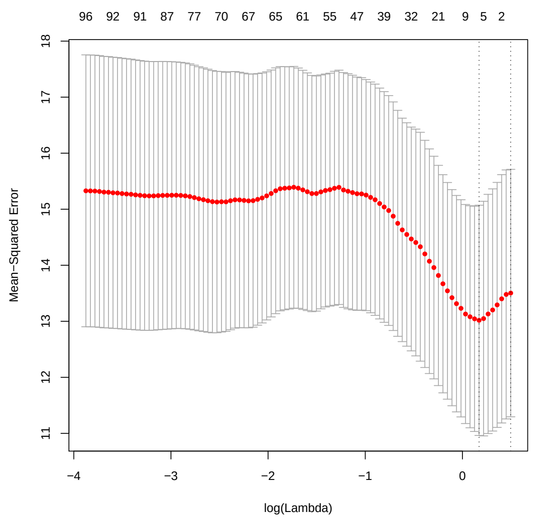
In this example, we have focused on BMI as the response and will use lasso and cross validation, as implemented in the R-package *glmnet*, to evaluate which of the genes influence BMI the most. In Section 5.2, the whole dataset is applied when fitting the lasso, while in Section 5.3 and 5.4, the patients are divided into a training set and a test set to study the preselection bias that may occur when reducing the number of covariates before doing lasso analysis.

5.2 Lasso and cross validation for the Bone data without preselection

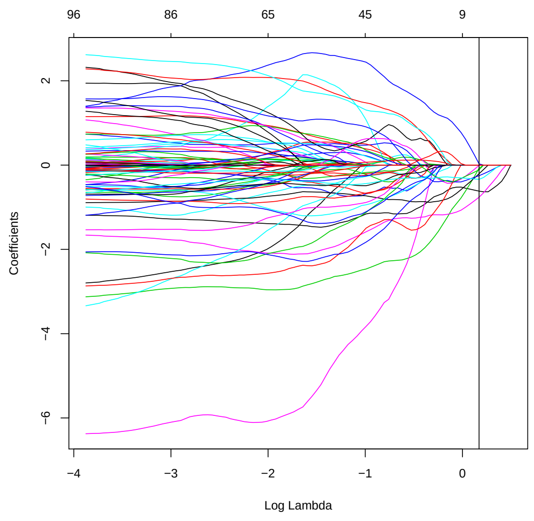
As for all examples in this thesis, we have used 10-folds cross validation to find the optimal value of λ. When doing cross validation in R, we can study a cross validation plot to visualize how the mean squared error changes for different values of the penalty parameter λ. As mentioned, the mean squared error (MSE) measures the difference between the observed response and the predicted response, and should be as small as possible to get the best possible model for prediction. The algorithm goes through a grid of different λ-values and minimizes the difference between the observed $\boldsymbol{y}^*$ in the validation group, and the estimated $\hat{\boldsymbol{y}}^*$ fitted from the model made from the remaining 9 patient groups and the covariates $\boldsymbol{x}^*$ from the validation group. The cross validation process chooses λ_{min} to be the value of the penalty parameter that minimizes the average cross validation mean squared error, which is the average of this MSE for all $K = 10$ different groups. For the Bone data, the cross validation (CV) curve is shown in Figure 5.1(a). Here we clearly see that the λ_{min} (the left, vertical line) is chosen to be the λ-value that minimizes the CV curve.

Figure 5.1(b) shows the value of the different coefficients $\hat{\beta}$ as the size of the penalty parameter λ is changed. The larger λ, the more coefficients are set exactly equal to zero. The numbers on top of the plot show how many coefficients that are not set to zero, i.e how many genes are selected for fitting a model for BMI. Here, 9 genes have coefficients that are not equal to zero. Remark that λ_{min} is the largest λ that gives the minimum of the cross validation curve. By studying this plot, it is clear that a λ-value closer to 1, i.e $log(\lambda)$ closer to zero, would also give the same 9 genes because there is a gap here where no coefficients are put to zero.

The 10 folds in the cross-validation are selected randomly when using the default in *cv.glmnet* in R. This also introduces some randomness in the result of the analysis. To study how much the division of folds influences the result, we ran lasso and cross validation 100 times, where each time the 10 folds were selected at random. As shown



(a) Cross validation plot.



(b) Lasso plot with covariates going towards zero.

*Figure 5.1: The cross validation plot in **(a)** shows how the mean squared error in the cross validation is minimized in λ_{min}, which is marked by the left dotted line. The dotted line to the right is the the largest value of λ such that it is within one standard error of λ_{min}. The red points are the average of the estimated error, while the belt around these are the estimate of the standard error of the cross validation mean squared error. The coefficient plot in **(b)** visualizes how the coefficients go towards zero as the value of the penalty parameter λ is increased. The vertical line shows the optimal value of λ_{min} found from cross validation. The numbers on top of the plot show how many of the covariates that have coefficients that are not equal to zero, and in this case with $N = 84$ and $p_{all} = 22815$, the analysis choose 9 genes to be in the final model.*

Gene index	# times chosen	Name
1275	99	1563182_at
5954	99	206726_at
17489	99	228128_x_at
19965	99	235631_at
7089	96	209309_at
1492	86	1569190_at
3058	86	202085_at
11165	86	217856_at
18143	86	229425_at
22759	15	
13924	7	
6336	1	
18092	1	
20249	1	

Table 5.1: The result of running lasso 100 times for $N = 84$ and $p_{all} = 22815$ with different folds in the cross-validation.

in Table 5.1, most often, the same few genes are selected (here recognized by their gene index), but some randomness is detected in the sense that there are some genes which are selected only for a few divisions of the folds. In fact, there are 9 genes which are selected at least 86% of the times, and 4 genes which are selected 99% of the times. Only the names of the 9 most selected genes are included in the table.

5.3 Cross validation plot for preselected sets of covariates

To better understand the data and the problems arising when preselecting covariates, we wish to study how a fitted model is able to predict in both a training set from which the model was fitted, and a new test set. The allocation of patients to training and test sets is done arbitrarily, and we will in this section study the cross validation plot when the patients are locked into two groups of $N_{training} = 60$ and $N_{test} = 24$. This division was chosen at random. Later we will divide into test and training sets 500 times. The patients are divided into these two groups in order to make it easier to combine the results with the study of the preselection bias later, but for now only the training set of 60 patients will be studied. Also, only one division of the $K = 10$ folds in the cross validation is used. The fold set applied here was one of the fold divisions that gave the same 5 selected genes as the most common result after running lasso and cross validation 100 times for the 60 patients in the training set (see Table A1.1 in Appendix A1). Because the folds are fixed, irregularities due to different folds are avoided, and

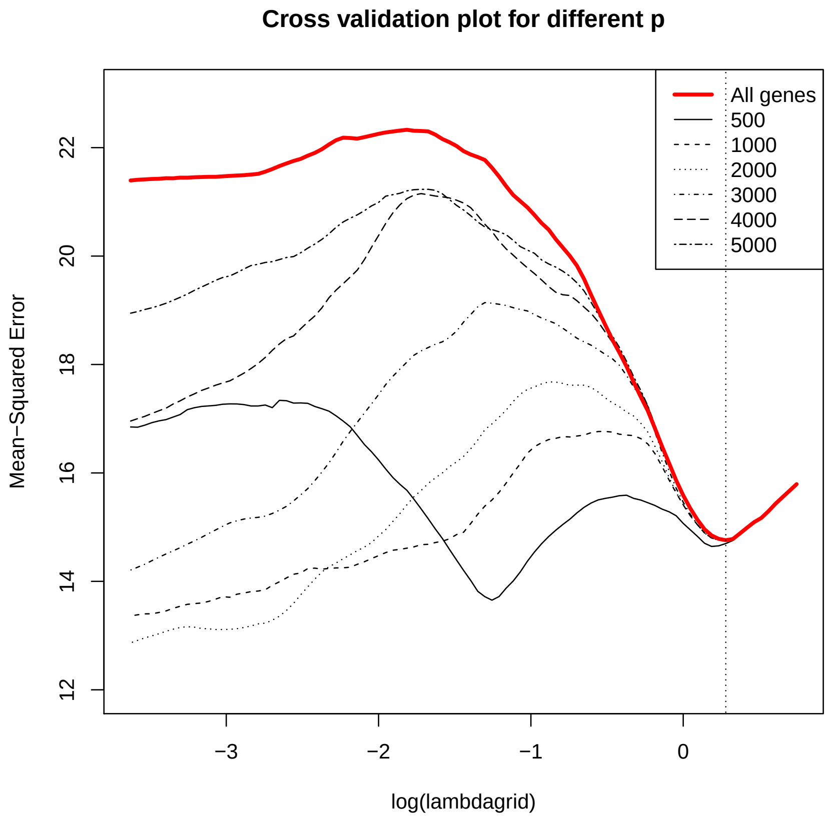


Figure 5.2: Cross validation plot for 60 patients while using the p genes that are most correlated with the response BMI. This plot is made by using the cross validation curves (similar to the one in Figure 5.1(a)) from analyses with different sizes of the preselected set of covariates and combining the CV curve into this one plot. The vertical dotted line shows λ_{min} when all the p_{all} covariates are included and this gives a lasso solution with 5 covariates.

the focus can be directed towards how different values of the size of the preselected set of covariates, p, influence the choice of the penalty λ.

The genes are first sorted, or ranked, according to their correlation with the response, BMI, in a univariate test. Then we perform preselection, by selecting the p top correlated genes for further multivariate analysis with lasso. Hence the p genes that are most correlated are used in the analysis with lasso and cross validation. We consider the result of lasso with the full dataset ($p = p_{all} = 22815$) to be the correct solution, in the sense that there is no preselection bias.

In Figure 5.2, we have plotted the cross validation curves for different values of p for this specific division of patients and folds. Remember that λ_{min} is selected as the value that minimizes the cross validation curve. In this plot, we see that for a preselected set of size $p = 4000$ or larger, the cross validation curve has the same minimum as for the full dataset. With smaller p than this, the penalty will be smaller than for the full set, and too many covariates (and maybe not the correct ones) will be chosen from lasso.

5.4 Preselection bias

To show how preselection bias causes problems that can not be ignored, we study the prediction ability of a model estimated from a preselected set of covariates, both on the dataset from which the model was made, and on a new independent dataset. This is done by dividing the patients in the Bone dataset into two groups and determine how well the model, fitted from the training set, is able to predict the response of both the training set itself and the separate test set.

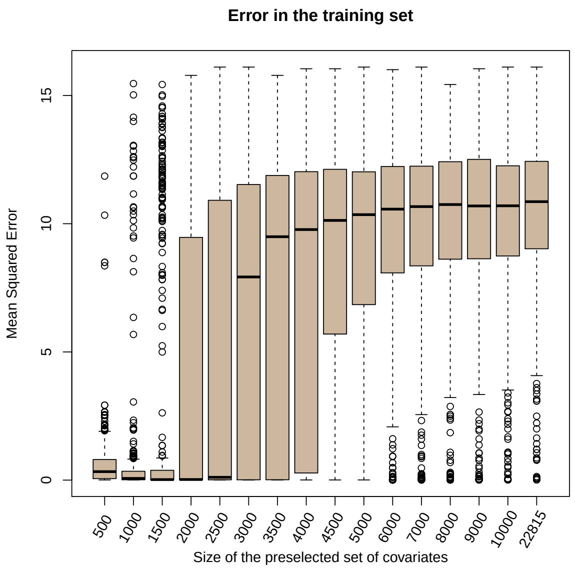
Mean squared error in different patient groups

Since the fitted model will be somewhat influenced by how the patients are divided into a training set and a test set, we will do this division 500 times and fit a model for every new division. For each fitted model, we will first apply the model on the training set and compare our estimated value of the response to the observed response using the mean squared error (MSE). If the model fits the data well, the MSE will be small. The mean squared error is defined as

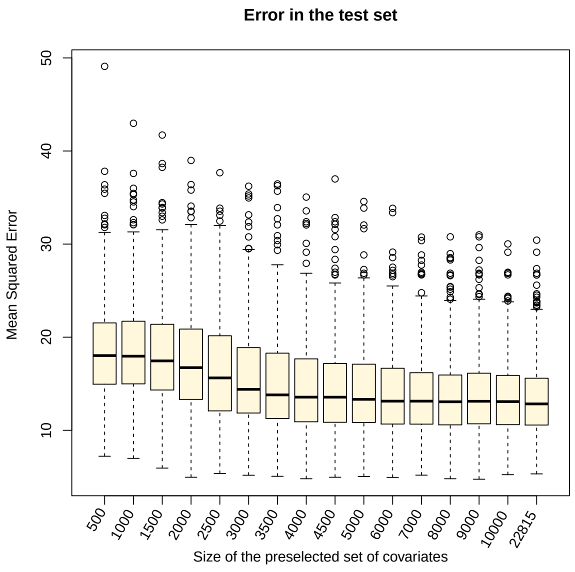
$$MSE_s = \sum_{i}^{N_s} \frac{(y_i - \hat{y}_i)^2}{N_s} \tag{5.1}$$

where y_i is the observed response for patient i, $\hat{y}_i$ is the estimated response and N_s is the size of the set of patients and $s = \{training, test\}$. In this example, $N_{training} = 60$ and $N_{test} = 24$. To evaluate if the model is able to predict the response in a new set of data, we will go through the same procedure for the test set. In this case we apply the model, which we have already estimated from the training set, on the covariates from the test set and compare the estimated response to the observed response in the test set.

For each of the 500 different divisions of the patient groups, we will get a value for the MSE in the training set, and one in the test set. From these, we make one boxplot for 500 MSE values from the training sets, and one for the 500 MSE values from the test sets. A boxplot shows the median of the observations as a line in the middle of a box. The box expands to the 25% percentile and the 75% percentile, i.e it contains the middle 50% of the observations. The lines out from the box will contain most of the remaining observations, except for possible outliers that are visualized by dots.



(a) Boxplots of MSE for the training set with 60 patients for different sizes of preselected datasets.



(b) Boxplots of MSE for the test set with 24 patients for different sizes of preselected datasets.

Figure 5.3: Visualization of how the mean squared error changes when more and more covariates are included in the preselected set of covariates. Each box in the boxplot represents the MSE for 500 different divisions of the patients into training and test sets, and the vertical line in the middle of the plot represents the median of the MSE of these 500 divisions. The last box in each plot is the analysis with the full data set ($p_{all} = 22815$ covariates).

Boxplots for different sizes of the preselected set

As mentioned, the procedure above gives two boxplots of the mean squared error for 500 divisions of the patients, one for the training set of patients and one for the test set. We make such boxplots for different sizes p of the preselected set of covariates. In this way, it is visualized how the error in both the training set and the test set evolve as more and more covariates are included in the lasso.

Figure 5.3(a) shows the result for the training set. Here, it seems that we get an almost perfect model when preselecting only a small number of the original covariates. When letting more of the covariates correlated to the response into the lasso analysis, the difference between the estimated response and the actual response increases drastically. On the other hand, we observe an opposite effect for the test sets in Figure 5.3(b). The prediction mean squared error is quite large when few covariates are preselected and decreases when more and more covariates are included, until it stabilizes from around $p = 4000$ preselected covariates.

Figure 5.4 illustrates the results for both the training set and the test set combined into one figure. We call this a preselection bias plot that illustrates what is meant by preselection bias. The model can predict the response for the patients in the training set very well when some preselection criterion is used to reduce the dimension of the covariates to a relatively small number prior to the lasso analysis. But in practice, what we usually want to achieve is a model that can predict the response for new patients. Therefore, we want the error for the test set of patients to be as small as possible.

In the preselection bias plot in Figure 5.4, we have not included the variation that the different fold divisions in the cross validation provides. In Appendix A1, we see that the effect of folds is negligible in this setting because the variation due to different splittings of the patients dominates the variation due to folds.

The number of covariates in the final model

Boxplots of the number of covariates in the lasso solutions are pictured in Figure 5.5. It is obvious that the number of genes selected from lasso when p is small, is much higher than the number that is selected when we use the whole dataset with $p = p_{all} =$ 22815 genes. We keep this in mind in the following sections and when studying the preselection bias plot in Figure 5.4. When the lasso is run only on the top univariately correlated covariates (the preselected set), the cross validation tends to choose little penalization and many selected genes, so that the model fits well to the current dataset, but will not be able to predict the response of a new dataset in a good way. A sparse model seems to be the best choice, and the preselected set has to be large enough to avoid the phenomenon of overfitting.

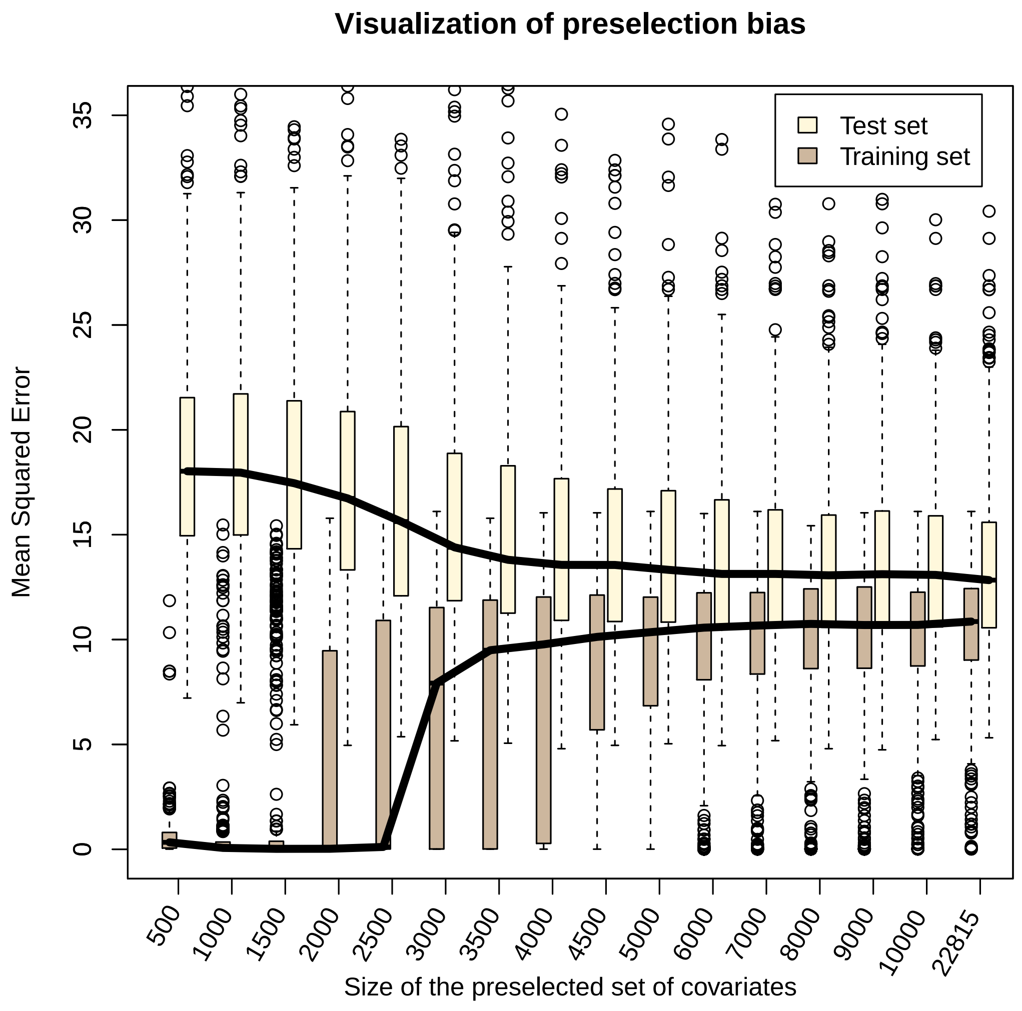


Figure 5.4: The median of the different MSE for 500 divisions of the patients. Here, the errors for both the training set (60 patients) and the test set (24 patients) are plotted together. (Some of the outliers in the test set are out of range in this plot, see Figure 5.3).

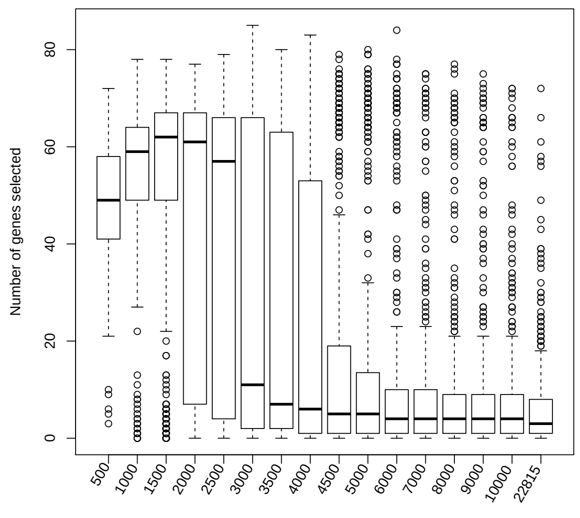


Figure 5.5: Boxplots of the number of genes that is selected in the minimum value of the cross validation curve when doing lasso and cross validation for different p, i.e different size on the preselected set of genes. Each box represents the number of covariates in the lasso model fitted in the training set for 500 different groupings of the patients in the Bone data. Note that in Section 5.3 where the patients were locked in one allocation of the training and test sets, 5 genes were included in the lasso solution from the analysis with p_{all} covariates.

Chapter 6

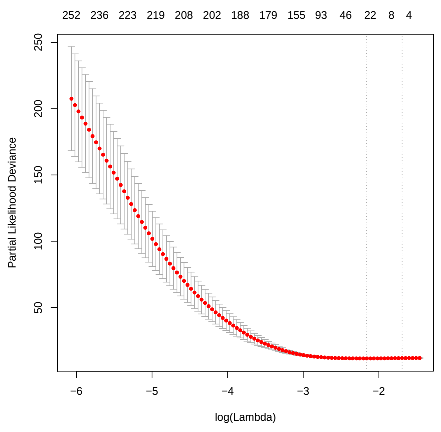
Lasso and preselection bias in Cox regression

The previous chapter illustrated how the preselection bias, described in Chapter 4, can affect the prediction ability of a model in high dimensional linear regression. We now illustrate how the problem of preselection bias applies for survival analysis as well. What happens to the subsequent analyses when only a few of the covariates in survival data are preselected before a model is fitted by the lasso?

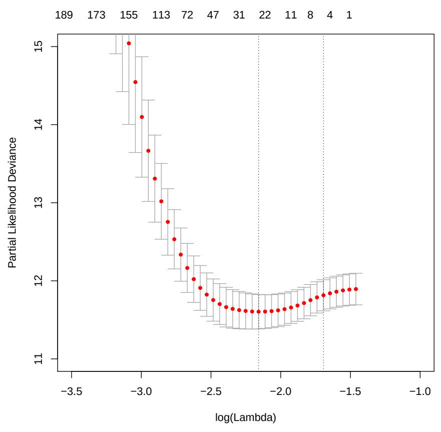
6.1 Introduction to the Lymphoma dataset for survival analysis

To illustrate the lasso and preselection bias on survival data, we have used a dataset studied in Rosenwald et al. (2002). This dataset is also used in Alizadeh et al. (2000) and Simon et al. (2011) and can be downloaded from http://llmpp.nih.gov/DLBCL. The dataset was collected to study the cancer called diffuse large B-cell lymphoma (DL-BCL), which is a cancer of the white blood cells in the body which are responsible for producing antibodies. Patients with DLBCL may respond well to chemotherapy and other treatments, but many will eventually die of the disease. Treating this type of cancer affect the body extensively, and can in some cases cause the patient to die if the patient does not die of the disease first. This means that treating a patient will not always be the best choice because it can lead to a sooner death. Hence this dataset was collected to find out whether it is possible to predict the survival of DLBCL patients by analyzing the disease on a molecular level.

As explained in Alizadeh et al. (2000), only genes that are mainly expressed in lymphoid cells and genes with known or suspected roles in processes important in immunology or cancer are included in this DNA microarray study. We have used this dataset even though this is a type of pre-filtering of covariates. This filtering is based



(a) The original cross validation plot.



(b) Zoomed in on the interesting part of the cross validation plot.

Figure 6.1: Cross validation curve for the survival data with all $N = 240$ patients and all $p_{all} = 7214$ gene expressions. The left vertical dotted line shows the choice of the penalty parameter $\lambda_{min} = 0.1155$, and this gives a model with 24 covariates.

on biological knowledge, and is not the preselection based on statistical analysis that is being discussed in this thesis. But either way, it is worth keeping this filtering of genes in mind when analyzing the results for the following statistical analyses.

Information on imputation of missing values and how we prepared the data for analysis is found in Appendix A2 and A3.

6.2 Lasso and cross validation for the Lymphoma data without preselection

After removing the covariates with more than 50% missing values and imputing the remaining missing values, the dataset consists of $p_{all} = 7214$ gene expressions for $N = 240$ patients. For each patient, follow up time (in years) since diagnosis is used as the response, and status at follow-up defines if the observation is censored or not. If the patient is alive at follow up, the observation is censored. The patients in this dataset had no previous history of lymphoma, and the follow-up time is the time since the untreated diffuse large-B-cell lymphoma was detected. The $p > N$ situation calls for a lasso penalty as before, but this time in combination with a Cox regression. We define the response as a two-column matrix with name "time" and "status", where status is 1 if the patient is dead, and 0 if the observation is censored. This matrix can be created with the *Surv()* function from the *survival* package in *R*.

Gene ID	# times chosen	Gene ID	# times chosen	Gene ID	# times chosen
27774	100	30170	100	27573	84
31242	100	34344	100	28497	68
17236	100	33358	100	25977	68
31981	100	32238	100	34376	68
31669	100	28377	100	24980	37
27585	100	32679	100	16891	37
27731	100	32424	90	29657	37
24376	100	30406	90	34805	32
28328	100	30634	90	17854	16
27267	100	24432	84		

Table 6.1: The 29 most selected genes after running lasso 100 times with different folds in the cross validation.

The cross validation plot for this situation is shown in Figure 6.1, where 6.1(a) is the full plot, while 6.1(b) is the same plot, but zoomed in so that the minimum of the cross validation curve is shown more clearly. This optimal choice of $\lambda_{min} = 0.1155$ gives a lasso solution with 24 selected genes. How the ten folds in the cross validation is divided will give some randomness in which λ is selected, but this λ is the λ that is selected most often when running the code 100 times where the folds change every time. That is, $\lambda_{min} = 0.1155$ is selected 31 times. The penalty parameter that selects 22 genes is chosen 22 times of the divisions, and the λ that selects 27 genes is chosen 21 times. For all the 100 random fold divisions, between 19 and 27 genes are selected. See overview in Table 6.1.

6.3 Cross validation plot for preselected sets of covariates

For survival analysis, univariate Cox regression is often used to preselect covariates. The covariates which have the lowest P-value from the univariate analyses are included in the subsequent analysis with the lasso.

As for the linear regression, we wish to study how the cross validation plot changes as more and more preselected covariates are included in the dataset to be analyzed. We have now divided the patients in two groups: 120 patients in a training set and 120 patients in a test set. For now, only the training set will be used, but the patients are divided in order to compare the results with results later in the chapter. For this reduced dataset, lasso and cross validation is applied. The λ that chooses 22 genes, was the λ_{min} that was selected most times when running cross validation 100 times with different divisions of the folds. Therefore, the folds and the λ-grid that resulted in a model with these 22 genes will be used in the following when checking how the cross

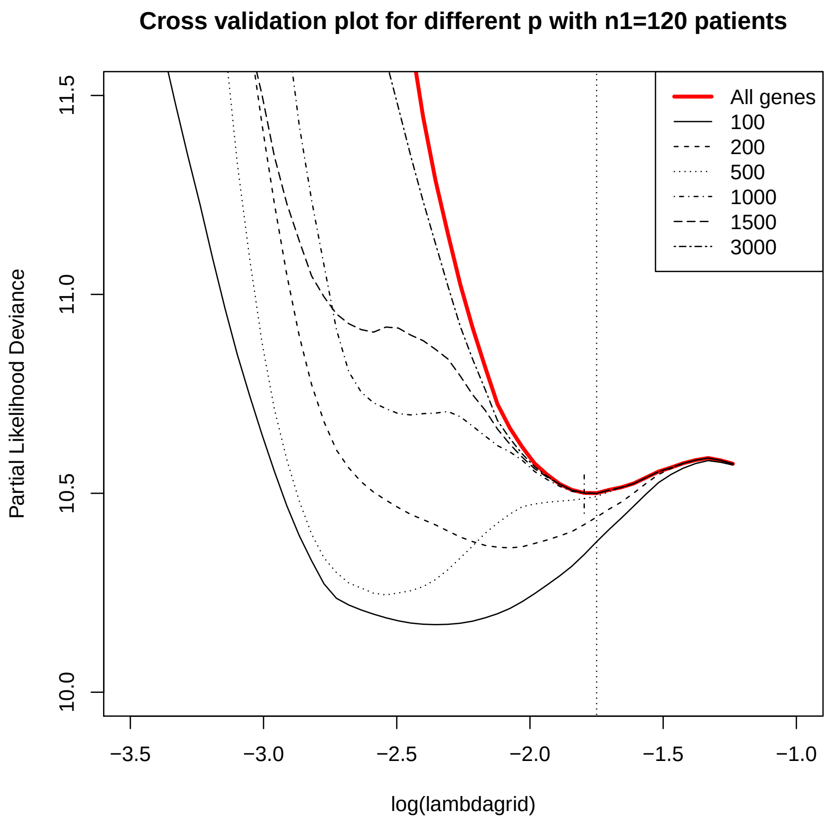


Figure 6.2: Cross validation curves when the size of the preselected covariates is changed. Here, 22 genes are selected in the lasso when all genes are included, and the long dotted line shows the λ_{min} in this point. The short vertical line is the minimum of the cross validation curve when $p = 1000$ of the preselected covariates are included, and with this penalty parameter lasso will choose 23 genes, instead of 22 as the full set of covariates will result in.

validation plot evolves when the size of the preselected set of covariates, p, is changed.

The cross validation plot for different values of p is plotted in Figure 6.2. We seek the p that gives the same minimum of the cross validation curve as the minimum when using all the $p_{all} = 7214$ covariates. In the plot, the correct minimum is not found until $p = 1500$ of the covariates are included. If, for instance, only $p = 500$ covariates are preselected, the cross validation chooses a smaller λ than what would be chosen when including all the covariates in the analysis. Here, 22 genes are selected from the lasso when all genes are included, while as many as 45 are included in the final model when the preselected set contains as few as $p = 500$ covariates. When λ is smaller, the lasso will choose more genes to be significant, and this will result in an overfitted model. This indicates that too many covariates tend to be included in the final model when too few covariates are preselected in the lasso Cox regression as well as in the linear regression in the previous chapter.

6.4 Preselection bias

Check on how well a model fits

The Cox proportional hazard model is, as mentioned in Chapter 3, a semi-parametric model. The baseline hazard α_0 is non-parametric and not estimated by the Cox regression. Therefore, the only element that is estimated in the Cox regression is the relative risk, i.e the survival compared to other patients. The deviance is often used to determine the fit of a model in Cox regression. In general, the deviance of a model is defined as

$$\begin{aligned} D(\hat{\theta}) &= -2[logL(\hat{\theta}) - logL_{\text{saturated model}}] \\ &= -2[logL(\hat{\theta}, X) - logL(\hat{\theta}_{saturated}, X)], \end{aligned}$$

where X is the data, the saturated model is the model that "perfectly" fits the data, and $\hat{\theta}$ is the coefficients in the maximum of the likelihood function. In our setting, the Cox partial likelihood, defined in (3.5), must be used.

It is not straight forward to study how a model fit for a new dataset with the deviance, but we have constructed a check. In our deviance check, we first we studied the deviance calculated from the partial likelihood with covariates and response from the training set together with the model coefficients that are fitted also from the training set. Secondly, we used the deviance calculated from the partial likelihood with the same model coefficients that was fitted from the training set, but now we applied covariates and responses from the test set. For this to work, the size of the training set and the test set must be equal.

More detailed, we find the maximum of the partial likelihood function for the training set, and save the $\hat{\theta} = \hat{\beta}_1, \hat{\beta}_2, ...$ in this point. When we then apply the same $\hat{\theta}$ in the partial likelihood for the test set, we get an impression of whether the likelihood function in the test set is much different from the likelihood function in the training set, since we find the value of the partial likelihood function for the test set in the point where the partial likelihood for the training set had its maximum. If the value of these two are close, it is probable that the likelihood functions are similar, i.e that the model from the training set has a good fit also in the test set.

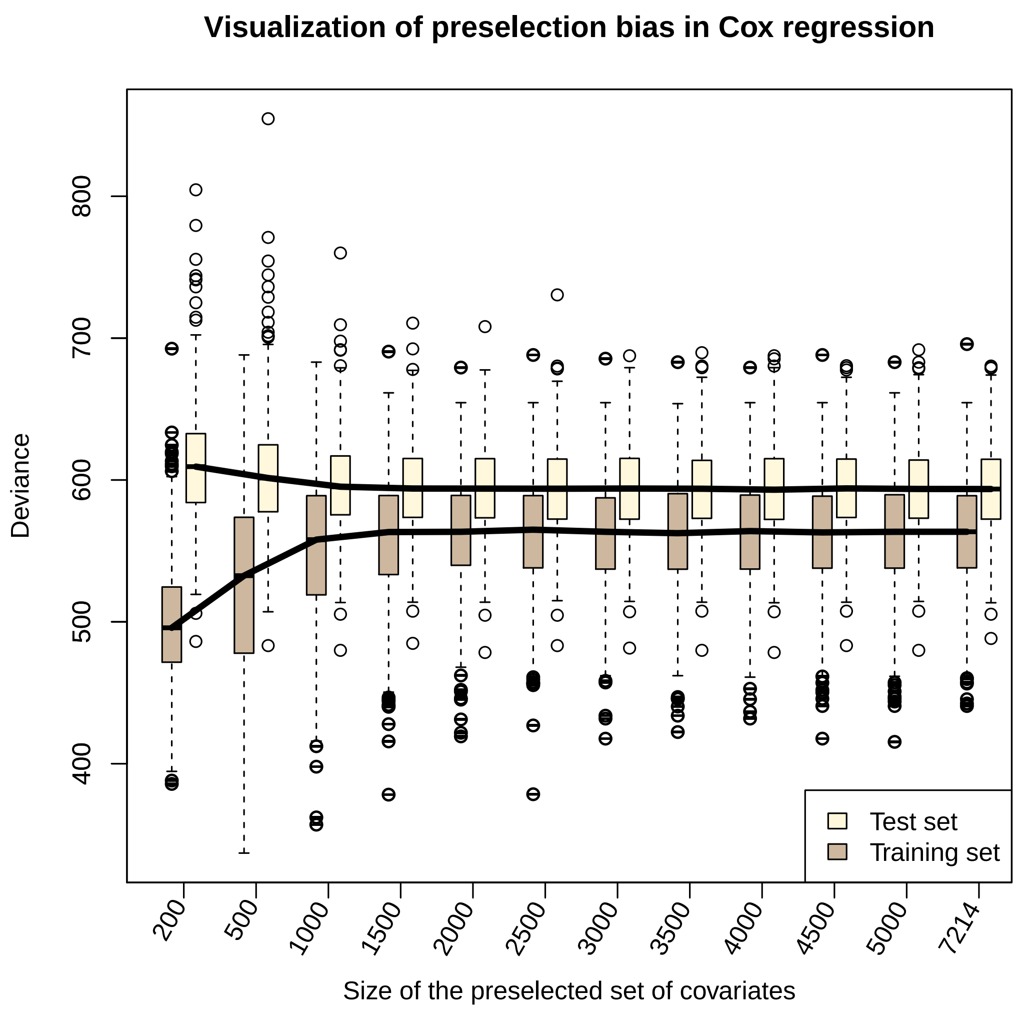


Figure 6.3: Preselection bias plot for Cox regression. The plot shows how the median of the deviance of a model in a training set and a test set evolve when more and more preselected covariates are included in the lasso analysis. The lasso analysis is done on the training set before a check on how the model fits the new data in the test set is performed. Here, the size of the training set and the test set are both 120 patients.

Visualization of preselection bias

The result of the check on how a model works in the training set and in the test set is shown in the preselection bias plot in Figure 6.3. This shows approximately the same pattern as for the preselection bias plot in the linear regression, although in a less conspicuous way. We see that when few covariates are included in the set of preselected covariates, the model from the lasso seems to fit badly in the test set even though it fitted well in the training set. When the size of the preselected set is increased, the model will fit the test set better and better until it stabilizes. On the other hand, for the training set, the fit is well when p is small and gets worse until it stabilizes. This can be explained by the overfitting phenomena since we get a smaller value of the penalty parameter λ when too few preselected covariates are included in the lasso. The difference of the error in the preselection bias plot with deviance is less pronounced than for the example in the previous chapter. This can for instance be a consequence of using a different test to check the fit, the structure of the design matrix in this dataset is different, or that different regression models are applied.

In this example we have fewer outliers in the boxplots than in the linear regression example. This can be explained by the number of patients we have; in the Bone data, we only had 84 patients that we divided into 60 and 24 for the training and test set. In this dataset on survival, we have a total on 240 patients, and this gives less differences for each new spitting of the patients. Hence, the results will be more stable.

Chapter 7

Discarding covariates with SAFE and strong

Is it possible to do preselection and still be able to trust the results from the lasso analysis? In Chapter 8, we will study a method to determine how large a preselected set of covariates should be to avoid preselection bias. This method studies how the the cross validation plot changes when adding more and more covariates to the preselected set. The method is then able to conclude if the preselected set of covariates is large enough to avoid preselection bias, or if more covariates should be included in the analysis.

In this chapter, the methods "SAFE" (El Ghaoui et. al 2011) and "strong" (Tibshirani et al. 2012) will be introduced. These methods are constructed to discard covariates in lasso-type problems, and were originally made to speed up the existing algorithms for lasso. In fact, the sequential strong algorithm is implemented in the *glmnet* package in *R* (Friedman et al. 2013), as mentioned in Section 3.5. This means that when doing lasso analysis in *R*, as we did in the previous two chapters, a selection of covariates is automatically done within the *glmnet* algorithm discarding some covariates to improve computational efficiency. But, as will be explained in this chapter, this selection does not lead to preselection bias because it is checked that the discarding of covariates is done safely through the sequential strong method in combination with with KKT. This discarding comes in addition to the type of preselection described in Chapter 4.

Both the SAFE method and the strong method discard covariates “from the bottom”, i.e. they discard the covariates that are the least correlated to the response, and thereby probably not included in the final model fitted from the lasso. The aim is to discard as many covariates as possible without making any mistakes. For both the SAFE and the strong method, the univariate inner product between the response and each covariate is used to decide if a covariate should be discarded or not. If the inner product is small, the covariate is discarded. It is common practice to standardize the covariates before a lasso analysis is done, but the SAFE and strong methods apply also when covariates are not standardized.

7.1 SAFE

For a given penalty parameter λ, the SAFE rule discards the jth variable if

$$|\boldsymbol{x}_j^T\boldsymbol{y}| < \lambda - ||\boldsymbol{x}||_2||\boldsymbol{y}||_2\frac{\lambda_{max} - \lambda}{\lambda_{max}}, \tag{7.1}$$

where $\lambda_{max} = max_j|\boldsymbol{x}_j^T\boldsymbol{y}|$ is the smallest λ which sets all coefficients equal to zero. Recall that the larger λ is, the more coefficients are put to zero.

This algorithm is safe, in the meaning that if a covariate is discarded with the SAFE method, it is guaranteed that that covariate would not have been a part of the final lasso solution with penalty λ, even if it had remained in the dataset for the lasso analysis.

7.2 Strong

The basic strong rule for the lasso discards the jth covariate if

$$|\boldsymbol{x}_j^T\boldsymbol{y}| < 2\lambda - \lambda_{max}, \tag{7.2}$$

for a given penalty parameter λ. The sequential strong rule uses the univariate inner product between the covariates and the residuals to decide which of the covariates that should be discarded, and discards variables $\boldsymbol{x}_j$ at λ_k if

$$|\boldsymbol{x}_j^T\{\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}(\lambda_{k-1}\}| < 2\lambda_k - \lambda_{k-1},\ k = 1, ..., \tag{7.3}$$

where $\lambda_0 = \lambda_{max}$. This means that for each new value of λ_k, all p_{all} covariates will be ranked based on the fitted model for λ_{k-1}. The lasso is then fitted for the top ranked covariates. This sequential strong rule is what is implemented in *glmnet* in R. Since $\hat{\boldsymbol{\beta}}(\lambda_{max}) = \boldsymbol{0}$, the basic strong rule is a special case of the sequential rule.

The strong rule is often able to discard more covariates than the SAFE rule. When the covariates are standardized, i.e $||\boldsymbol{x}_j||_2 = 1$ for all j, the strong rule will discard more covariates than SAFE because the strong bound will be larger than (or equal to) the SAFE bound. This can be seen by noting that $\lambda_{max} \leq ||\boldsymbol{y}||_2$ when the covariates are standardized, and then the strong bound is the largest:

$$\lambda - 1||\boldsymbol{y}||_2\frac{\lambda_{max} - \lambda}{\lambda_{max}} \leq \lambda - (\lambda_{max} - \lambda) = 2\lambda - \lambda_{max}.$$

Both SAFE and strong have global as well as sequential algorithms (called recursive for SAFE). Sequential algorithms do checks for every new value of λ in contrast to the global methods that discard covariates by doing an overall analysis of the regression problem.

In contrast to the SAFE method, the strong method is not safe. The strong algorithm discards covariates that most likely will not be a part of the final model, but there are no guaranties. In Tibshirani et al. (2012) there where no violations in the 100 simulated sets where $p >> N$ (which is when the rule is mostly used), but some violations may occur when $p \approx N$.

By combining the strong rule with simple checks of the Karush-Kuhn-Tucker (KKT) condition (see next section), it can be made safe. Any approximate rule for discarding covariates can be combined with KKT to ensure that the method finds the exact solution, but the sequential strong rule will in practice discard a very large proportion of the covariates that do not influence the response, and rarely make mistakes. This means that when applying the strong algorithm, the KKT check will rarely be used to reinclude covariates in the dataset.

7.3 KKT condition

The Karush-Kuhn-Tucker condition is an extension to the method of Lagrange multipliers. While the Lagrange multipliers method solve problems that include constraints with one or more equalities, the KKT conditions allow the constraints to be inequalities as well. A problem may look like

$$\begin{gathered} x^* = argminf(x) \\ \text{subject to } h_i(x) = 0 \text{ for } i \in 1, ..., m \\ \text{subject to } g_i(x) \leq 0 \text{ for } i \in 1, ..., n \end{gathered}$$

The optimization problem can then be written as

$$x^* = argmin\{f(x) + \sum_{i=1}^{m} a_i h_i(x) + \sum_{i=1}^{n} b_i g_i(x)\}$$

where a_i and b_i are the KKT multipliers (Gordon and Tibshirani, 09.2014).

As explained in Tibshirani et al. (2012), the KKT condition for the lasso problem is

$$\boldsymbol{x}_j^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) = \lambda s_j$$

for $j = 1, ..., p$ and $s_j \in sign(\beta_j)$ if $\beta_j \neq 0$ ($s_j \in [-1, 1]$ if $\beta_j = 0$). This check will be done for each λ, and if the KKT condition is violated, the corresponding covariates that were excluded in the previous step of the strong algorithm, will be reattached to the dataset. That the KKT condition is violated means that $|\boldsymbol{x}_j^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})| > \lambda s_j$, i.e. there is a high level of correlation between the excluded covariates and the response.

7.4 Strong and SAFE methods applied on Bone data

To compare the SAFE and the strong algorithms, we have used the Bone data to make a plot that is similar to Figure 2 in Tibshirani et al. (2012). This plot shows how many covariates are left after applying the SAFE and strong rules and compares this with how many covariates there are in the lasso solution. We have reused and modified code from the strong paper to study the SAFE and strong methods on the Bone data [1] and the result is displayed in Figure 7.1. The sequential strong algorithm seems to be the best one, since this is the algorithm that discards a large percentage of the covariates. This figure looks similar to the plots in the paper on the Strong method of Tibshirani et al. (2012) based on other datasets.

Furthermore, we see that the recursive and the sequential algorithms seem to be the best ones. Either if we use the sequential algorithms or the global algorithms, strong seems better than SAFE in that strong discards more covariates that will not be included in the lasso solution.

[1] Code from http://statweb.stanford.edu/~tibs/strong/, modified for this context.

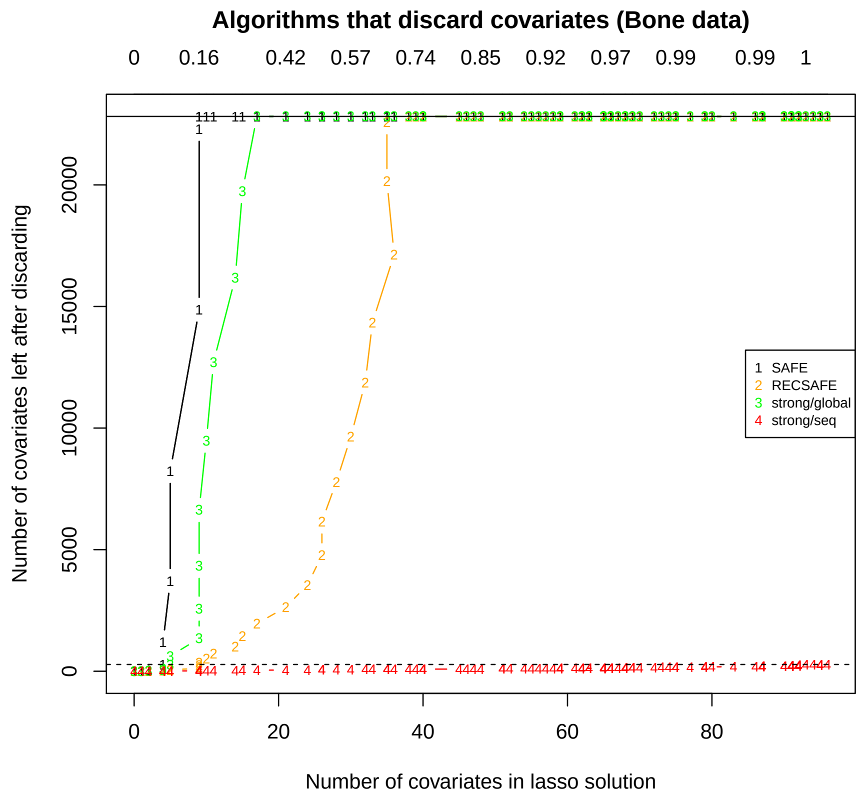


Figure 7.1: SAFE and strong algorithms to discard covariates. Here, the remaining number of covariates, after some have been discarded, is compared to the number of covariates in the lasso solution for a given value of λ (decreasing from left to right). The horizontal full line is the total number of covariates before any covariates are discarded, $p_{all} = 22815$. The maximal number of covariates used in the lasso after ranking all p_{all} covariates for each λ with the sequential strong algorithm is 277 and is visualized by the dotted line. The proportion of the variance explained by the model is shown along the top of the plot.

Chapter 8

The concept of cross validation freezing

Both the "SAFE" and the "strong" method for selecting covariates explore the data and discard the covariates that are the least likely to be important in the final model, as described in the previous chapter. It is also possible to study the problem the other way around, by adding covariates to the preselected set "from the top and down" and include more and more covariates until it is likely that the results from lasso and cross validation will be the same for the preselected dataset as for the full dataset. This preselection method was introduced by Bergersen et al. (2013) as "freezing" and is able to preselect covariates without introducing preselection bias.

Freezing will in practice not be an alternative to the sequential strong algorithm that is implemented in the *glmnet* package in *R*, but will be a complementary method for reducing the dimension of large datasets "outside" the lasso optimization.

8.1 Freezing in previous examples

Bergersen et al. (2013) defines freezing as the point where the minimum of the cross validation curve stays the same even though more covariates are included in the analysis, i.e the point where the penalty parameter λ found in the reduced dataset is the same as the optimal λ_{min} found using all the covariates in the dataset. Figure 5.2 on page 35 visualizes freezing for the Bone data. When the size of the preselected set of covariates, p, exceeds 4000, the cross validation curve is "frozen" and the minimum of the curve is the same minimum as for the whole set of $p_{all} = 22815$ covariates. This means that with only 18% of the dataset, the optimal solution for the lasso regression problem can be found. Similarly, the freezing for the survival data in Chapter 6 is shown in

Figure 6.2(on page 44. Here, the cross validation curve freezes when $p = 1500$. [1] This corresponds to 21% of the full dataset.

When the size p of the preselected set of covariates is larger than the set of covariates in the freezing point, it is expected that the mean squared error in the linear regression, or the partial likelihood deviance in the Cox regression, will not change much even though a higher p is used. In this thesis we have shown that this is indeed true for the datasets we have evaluated. That the prediction error for new data is enlarged when the preselected set of covariates p is smaller than the set where the cross validation curve freezes is visualized in the preselection bias plot in Figure 5.4 on page 39 for the linear regression situation, and in Figure 6.3 on page 46 for the Cox regression situation.

For the examples in this thesis, it is possible to calculate how the cross validation curve for the full dataset would look like, but this is not always the case if the number of covariates becomes too large. Even if the cross validation curve for the full dataset can be obtained, the aim of the freezing algorithm is to track the relevant part of the cross validation curve without using all the covariates, but rather just a subset of these. The concept of freezing evaluates how large the preselected set of covariates, p, must be to find the correct lasso solution, hence without preselection bias.

8.2 Guiding the preselection through freezing

The motivation behind the freezing algorithm is that when increasing p, the cross validation curves will start to coincide from the maximum value, λ_{max} of the λ-grid and down, i.e from the right of the cross validation plot. Bergersen et. al (2013) defines this concept and say that the cross validation curve is freezing in (λ, p).

Some notation must be introduced to be able to explore the concept of freezing in more detail. The set of p ordered covariates are defined as $\boldsymbol{x}_1, ..., \boldsymbol{x}_p$, and p takes values $0 < p_1 < ... < p_{all}$. Then $C_p = \{1, ..., p\}$ can be used to indicate the covariates that are included in an analysis, and $C_F = \{1, ..., p_{all}\}$ is the full dataset. The cross validation curve is named CV, and thereby will the cross validation curve for the dataset containing p of the highest ordered covariates be defined as CV_{C_p}. Λ is a predefined grid of λ-values used in the cross validation.

A curve is defined to be frozen in (λ, p) if

$$\begin{aligned} &CV_{C_{p'}}(\lambda') = CV_{C_F}(\lambda'), \\ &\forall \lambda' \geq \lambda \text{ and } \forall p' \geq p. \end{aligned} \tag{8.1}$$

In words, this means that if the cross validation curve is frozen in (λ, p), then the curve will be identical to the cross validation curve for the full dataset for all $\lambda' > \lambda$, i.e to the

[1]Recall that the cross validation curve for $p = 1000$ has a minimum that gives a smaller value of λ, even though it is difficult to read from the figure.

right of the minimum of the curve. This part of the curve will not change even though more covariates are included in the preselected set ($p' > p$).

To detect the freezing point, the following algorithm can be used. Here, $m = \{1, ..., M\}$ and $p_M = p_{all}$.

Freezing algorithm:

- Compute the cross validation curve CV_{p_m} for all $\lambda \in \Lambda$, for $m = 1, 2, 3, ...,$ until
$$CV_{C_{p_{m+1}}}(\lambda) = CV_{C_{p_m}}(\lambda) \text{ for all } \lambda \in \tilde{\Lambda}.$$
Here, $\tilde{\Lambda} = [\tilde{\lambda}, \lambda_{max}]$ contains a minimum of the cross validation curve $CV_{C_{p_m}}(\lambda)$ for λ-values larger than $\tilde{\lambda}$. This means that the algorithm will stop if the cross validation curves for following datasets, where the last one is larger than the previous, is the same from λ_{max} down to a given value of the penalty parameter $\lambda = \tilde{\lambda}$. The minimum value of λ between $\tilde{\lambda}$ and λ_{max} is defined as $\lambda^*_{p_m}$.
- Use the lasso with penalty parameter $\lambda^*_{p_m}$, and by including only the C_{p_m} covariates. Thereof, the solution $\hat{\boldsymbol{\beta}}_{C_{p_m}}$ of the penalized regression problem can be returned.

Whether the algorithm stops before $p_m = P_{all}$ depends on the ordering of covariates, but even simple ordering will most frequently lead to $p_m << p_{all}$. Even though this algorithm will most often find the correct solution, it is neither guarantied that $\lambda^*_{p_m} = \lambda_{min}$ nor $\hat{\boldsymbol{\beta}}_{C_{p_m}}(\lambda^*_{p_m}) = \hat{\boldsymbol{\beta}}_{C_F}(\lambda_{min})$, so additional sequential checks can be included in the algorithm to detect false stopping. More details on this additional check can be found in Bergersen et al. (2013).

8.3 Survival analysis and freezing for truly high dimensional data

Even without preselecting any covariates, a model for the datasets used in both Chapter 5 and 6 can easily be fitted through the lasso. The datasets contained around 22000 and 7000 covariates, respectively, and the *glmnet* in *R* has no problem with these amounts of data. Even so, preselection is often done for simplicity and to save memory and computation time. Then freezing can be used to control the preselection of covariates and prevent preselection bias. For larger datasets, a reduction of the size of the dataset through preselection of covariates may be crucial to be able to perform analyses with reasonable limited computational resources.

Bergersen et. al (2013) examined lasso in a linear regression setting when introducing the concept of freezing. However, freezing in lasso analysis for survival data has not been examined previously. In chapter 6 in this thesis, the concept of freezing was visualized for survival data with $p_{all} = 7214$, and we will now study a dataset with methylation data to examine the concept of freezing in the Cox regression setting for truly high dimensional data.

In our methylation dataset provided by Heidi Lyng at Oslo University Hospital (Radiumhospitalet), there is a total of 147 patients with cervix cancer. Each patient has corresponding methylation measurements for 484919 probes. Right censored survival data is also available for each patient; the patient has entered the study at time of chemotherapy, and the time of first relapse is considered as an event.

We wish to examine the cross validation plot for methylation data and study how it changes as more and more covariates are included into the preselected set. By comparing the cross validation plot for p_{all} and other $p < p_{all}$, we are able to detect when the curves freeze and thereby how many covariates the freezing algorithm would include in the preselected set of covariates. Because *R* had difficulties running the lasso when all the 484919 probes were included in the dataset, we reduce the size of the dataset by filtering the covariates before we do the preselection that is described in this thesis. The covariates are filtered such that only the covariates with standard deviation larger than 0.1 are included in the dataset, and the data is then reduced from a total of 484919 covariates to $p_{all} = 168045$ covariates. By doing this, we are able to study the cross validation plot for all the covariates that are in our new dataset. This filtering is reasonable because if there is almost no variation between different patients for a given covariate, the probability that this covariate will describe the response is vanishingly small.

Differences in the results from lasso for different fold divisions

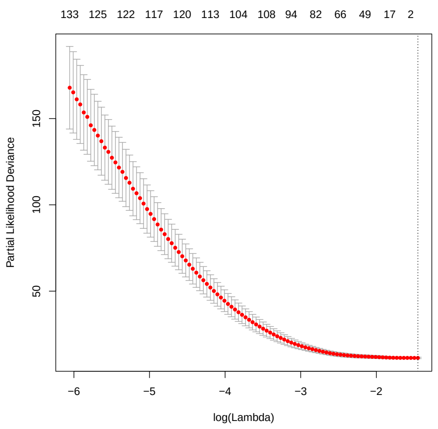
To test the variability of the results for different folds in the cross validation in this dataset, lasso and cross validation is done 100 times where *R* choose the folds at random. Table 8.1 shows how the value of the penalty variable λ varies, and how many of the probes that are selected in the lasso solution for the different values of λ. Gui and Li (2005) claims that a penalized Cox regression model on gene expressions tend to select only a few covariates in the final model, and this is the case in this example. Here, 75% of the fold divisions in the cross validation will conclude to select 5 or less covariates to be included in the final model from the lasso. Which of the probes that are selected are shown by their index in Table 8.2.

Pentalty λ	# times chosen	# genes selected
0.2345	29	1
0.1947	20	5
0.1858	18	9
0.2136	10	2
0.2039	10	3
0.2238	6	1
0.1774	5	12
0.1693	2	16

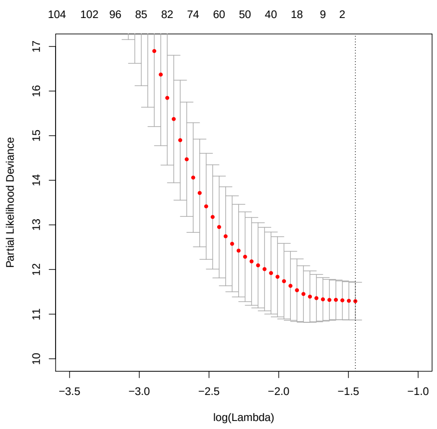
Table 8.1: The result of running lasso 100 times with different folds in the cross-validation for the methylation data. This table shows how many times each of the λ-values are selected, and how many genes will be included in the lasso solution, given λ. Note that the same value of λ gives the same number of selected probes, but this is not necessarily true the other way around.

Probe index	# times chosen	Probe index	# times chosen
128889	100	24931	7
101102	65	32905	7
45758	55	101453	7
129063	45	11243	2
44711	43	27571	2
51339	25	43488	2
69144	25	71840	2
86213	25	110815	2
90636	25		

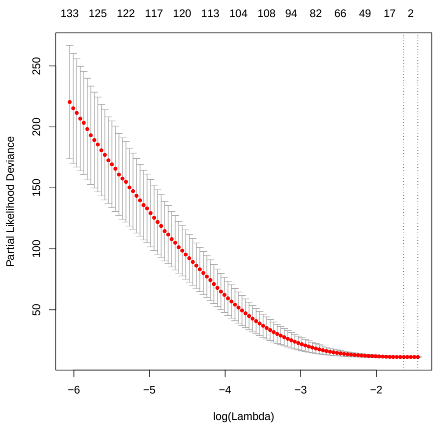
Table 8.2: The probes that get selected when doing the lasso 100 times with different folds in the cross validation for the methylation data.



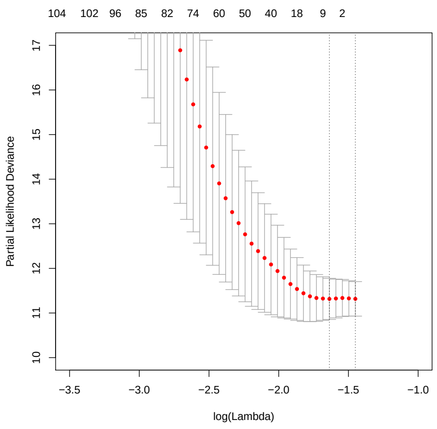
(a) Cross validation plot: $\lambda = 0.2345$ and 1 probe is selected.



(b) Zoomed: $\lambda = 0.2345$.



(c) Cross validation plot: $\lambda = 0.1947$ and 5 probes are selected.



(d) Zoomed: $\lambda = 0.1947$.

Figure 8.1: Cross validation plot for the methylation data. In (c) and (d) it is possible to detect a curve that include the minimum of the cross validation curve, while in (a) and (b), the minimum is set to be in the very end of the cross validation curve.

Cross validation plot to detect freezing

Even though Table 8.1 shows that only one gene is selected from the lasso analysis for the λ-value chosen 29 % of the times, we choose to use $\lambda = 0.1946$ which selects 5 probes in the subsequent analysis. When $\lambda = 0.2345$, the cross validation curve has it's minimum in the end of the curve, as seen in the cross validation plots in Figure 8.1. It is not reasonable to analyze this any further because we are interested in finding which p that will detect the right shape and location around the minimum of the cross validation curve for the full dataset. To be able to detect the differences that various sizes of the preselected sets of covariates give in lasso and cross validation, the cross validation curve in the case where all covariates are included must have a curve such that the minimum can be detected.

With this methylation data, we wish to study the cross validation plot for different sizes of the preselected set of variables, and not the preselection bias boxplot because it is too time consuming to run the lasso for serveral different divisions of the patients. We assume that preselection bias would be visualized in a preselection bias plot if less than the number of covariates in the freezing point were included in the lasso analysis. Therefore, all the patients are included in the following analysis instead of only the training set, as was done in Chapter 5 and 6.

Figure 8.2 clearly shows that also for this dataset, preselection bias can occur if too few of the covariates are chosen to be included in the preselected set. In this dataset, however, the number of covariates can be reduced to as little as 0.6% of the original dataset by applying freezing; the number of covariates to be included in lasso and cross validation can be reduced from $p = 168045$ to $p = 1000$, and the lasso solution will still be the same with 5 probes selected. We say that the cross validation curves are frozen in $(\lambda = 0.1946, p = 1000)$.

When working with datasets as large as this, the advantage of reducing the dataset before any penalized regression model is used is amplified. Even the straight forward operation to read the $p_{all} = 168045$ covariates into R takes over a minute. [2] The computation time when using the whole dataset with p_{all} covariates in the lasso and cross validation is over 10 minutes. In contrast, the same analysis when using the 1000 covariates with the highest influence on the survival in a univariate regression model took less than 10 seconds.

[2]The computer used here is an Asus with a Inter Core i7 processor with 8 GB memory.

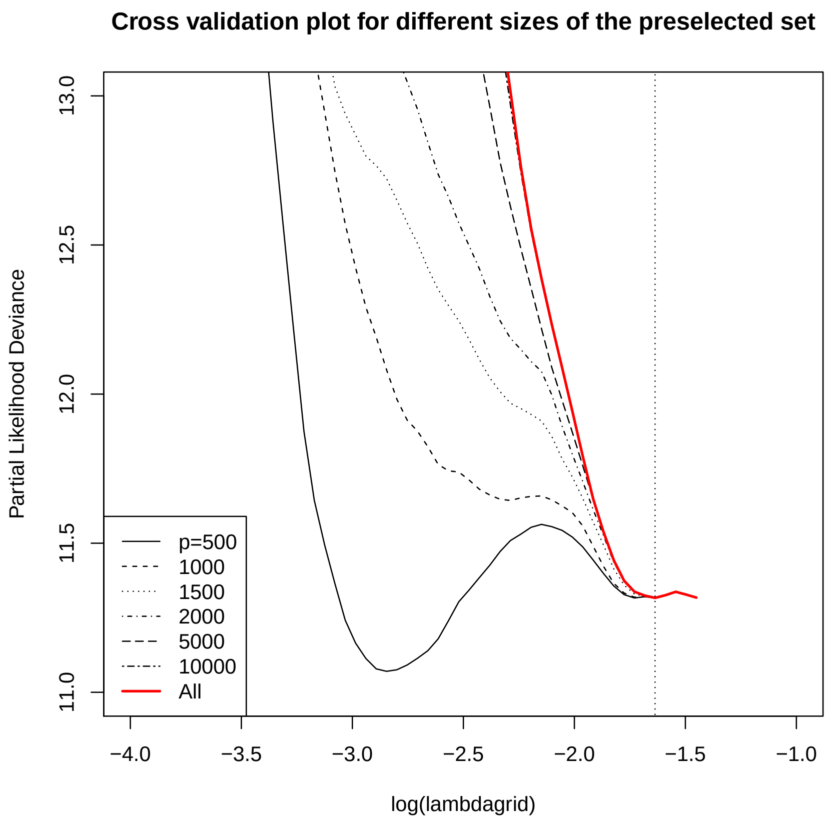


Figure 8.2: Cross validation plot for different size p *of the preselected set of covariates in the methylation data. Here,* $\lambda = 0.1946$ *and 5 probes are selected in the final model. It is not easy to detect by studying the plot, but the line for* $p = 1000$ *has actually the same minimum as the one for the whole dataset. Thereby, this cross validation curve is frozen in* $\lambda = 0.1946$*,* $p = 1000$*.*

Chapter 9

Simulation studies

In Chapter 5 and 6, we studied how preselection bias can be a problem when preselection of covariates is done for real datasets, while Chapter 7 described how some of the covariates are discarded through the sequential strong algorithm when doing lasso with *glmnet* in *R*. To be able to preselect even more covariates prior to the lasso, and still be able to trust the results, Chapter 8 showed how the cross validation curve can be used to check whether the preselected covariates are able to make a trustworthy model. We now simulate data to study preselection bias further. We are interested in studying how the level of noise, and the dependence structure in the design matrix influence the problem of preselection.

9.1 Setup for the simulation studies

For the following simulation studies, we will generate the response $\boldsymbol{y}_{sim}$ from a linear regression model with normally distributed noise $\boldsymbol{\epsilon} \sim N(0, \sigma^2 \boldsymbol{I})$;

$$\begin{aligned} \boldsymbol{y}_{true} &= \boldsymbol{X\beta}, \\ \boldsymbol{y}_{sim} &= \boldsymbol{y}_{true} + \boldsymbol{\epsilon}. \end{aligned} \tag{9.1}$$

To study the effect of the noise level on preselection bias, the signal to noise ratio (SNR) is used to determine the amount of noise in a simulation. The signal to noise ratio describes, as the name indicates, how much noise the dataset has, compared to how strong the signal (trend) is in the observations. A low value of SNR corresponds to a large amount of noise, and the variance is then defined as

$$\sigma^2 = \frac{Var(\boldsymbol{y}_{true})}{SNR}.$$

We choose $SNR = 0.5$ and $SNR = 2$.

The covariate matrix $\boldsymbol{X}$ is defined in various ways in the following different simulation studies, partly following Bergersen et al. (2013).

- **A - Independent:** The covariates are simulated independently from a standard normal distribution.
- **B - Microarray:** The genomic Bone dataset with microarray covariates from Chapter 5 is used.
- **C - Dependency within blocks:** The covariates are simulated in blocks of 100. In each block the covariates have a pairwise correlation between the ith and the jth covariate given by $\rho^{|i-j|}$, but there is independence between blocks.

When simulating data, it is common to repeat independent simulations several times, often thousands, to make sure that the results are not only due to some random factor. When studying the preselection bias, we have, in Chapter 5 and 6, divided the patients into a training and a test set several times to get rid of the randomness due to patient groups. This must be done in the simulation studies as well. We define $s_1 = 100$ to be the number of simulations and $s_2 = 100$ the number of allocations of the patients.[1] For each of the simulation studies (A, B and C), we have simulated the covariates $\boldsymbol{X}$ as explained above, chosen $SNR \in \{0.5, 2\}$ and done the following:

How to do simulations combined with splittings into training and test sets

- Simulate s_1 vectors of responses $\boldsymbol{y}_{sim}$ following (9.1).
  - For every $\boldsymbol{y}_{sim}$, allocate the patients into training and test sets s_2 times.
    - For every allocation of patients and for different sizes p of the preselected set of covariates, fit a model through lasso in the training set and find the mean squared error for both the training set and the test set.
  - Save the median of the the s_2 values of MSE in both patient sets for all p.
- Make boxplots with the MSE medians from s_1 simulations, one boxplot for each p for both the training and the test set.

This means that to visualize the results for the simulation studies, instead of making $s_1 = 100$ preselection bias plots like the one in Figure 5.4, we save the median of the MSE for the s_2 different divisions of the patients. These medians are used to make new boxplots for the results of the s_1 simulations. The width of the boxes are therefore not to be directly compared to those in Chapter 5 and 6, but should still give an impression of the variability.

[1]To limit the computation time, we simulate new responses 100 times instead of thousands. The 100 times 100 lasso analyses for all different values of p took approximately one week to run for each simulation study.

9.2 Three simulation studies

A: Covariates independently simulated from a standard normal distribution

To study how the lasso is effected by preselection bias when covariates are independent of each other, we first simulate a covariate matrix $\boldsymbol{X}$ independently from the standard normal distribution. The number of observations is chosen to be $N = 200$. For each observation, we simulate $p_{all} = 10000$ covariates independently from the standard normal distribution. We define the coefficient vector $\boldsymbol{\beta}$ to be equal to zero, except for the coefficients corresponding to 10 randomly drawn covariates where five are put equal to -2 and five are put equal to 2.

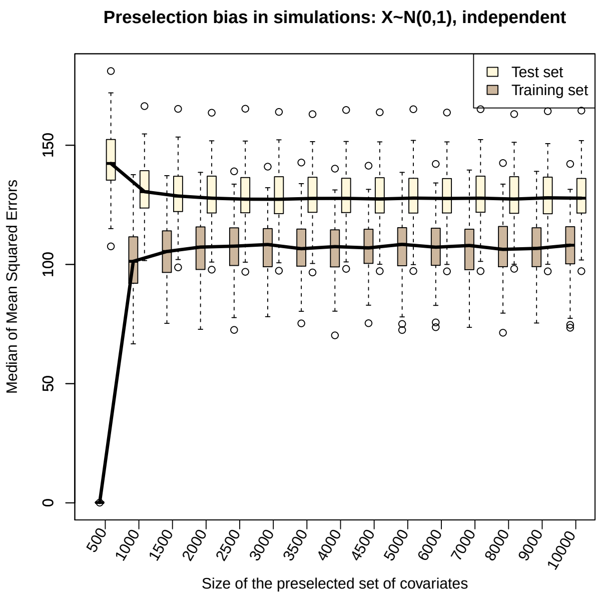
As described above, boxplots of the median of 100 MSE from $s_1 = 100$ simulations are depicted to study the preselection bias in the simulation study. The results for simulation A are visualized in Figure 9.1, both for $SNR = 0.5$ and $SNR = 2$. When the covariates are independent, the preselection bias is not that clear, but we can still see that the MSE in the test set decreases from $p = 500$ to $p = 1500$ (steepest to $p = 1000$) for both $SNR = 0.5$ and $SNR = 2$. This indicates that the univariate sorting of covariates is not necessarily that bad when the covariates are not very correlated. But even in this simple example it is essential that "enough" of the sorted covariates are included in the preselected set. The size of the set of covariates that should be analyzed in lasso could potentially be reduced to as little as 10-15% of the full dataset, as we see in this figure.

The differences in the MSE for various noise levels is noticeable; when $SNR = 2$ and there is little noise, the MSE is low for both the training set and the test set, compared to what is the case when $SNR = 0.5$. Also, the effect of including too few of the covariates in the lasso is less evident when $SNR = 2$

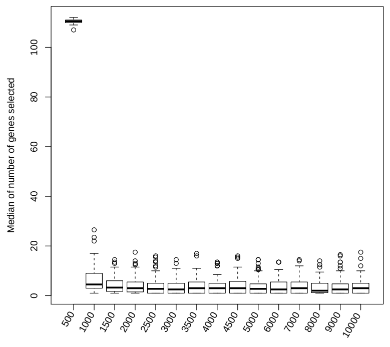
When studying the number of covariates selected in the final model (Figure 9.1 (b) and (d)), we see that when the number of covariates p in the preselected set of covariates reaches a high enough level, the number of non zero covariates in the lasso solution jumps to a low number. Since the response is simulated from a defined set of βs, we know that the correct number covariates that effect the response is 10. Therefore, a model with over 100 covariates different to zero, as the model for $p = 500$, will obviously be an overfitted model.

B: Covariates from the Bone data

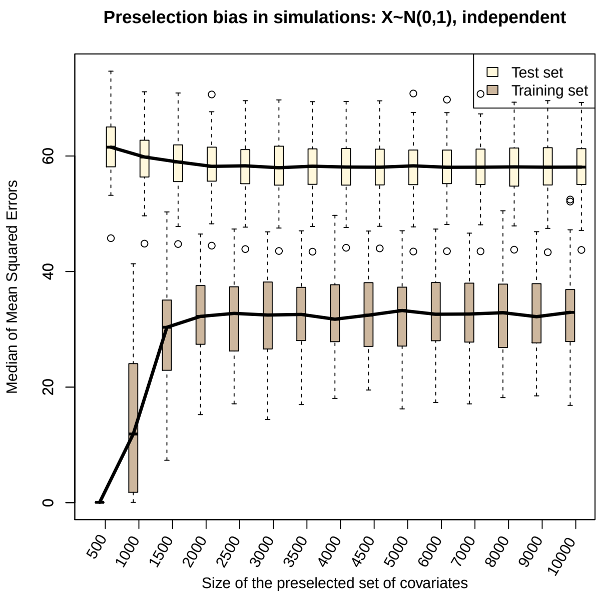
A large datasets with independent covariates, as in simulation A, is not very realistic when it comes to real life datasets. Therefore, we wish to study how the preselection bias changes for different levels of noise when simulating the response using covariates from real data. For this simulation study microarray covariates from the Bone data in



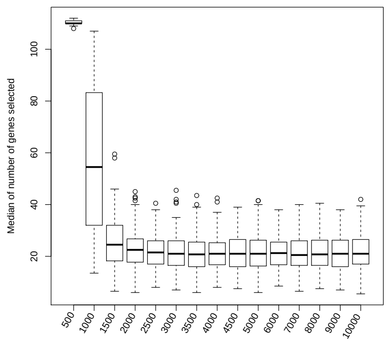
(a) SNR = 0.5



(b) Number of covariates selected

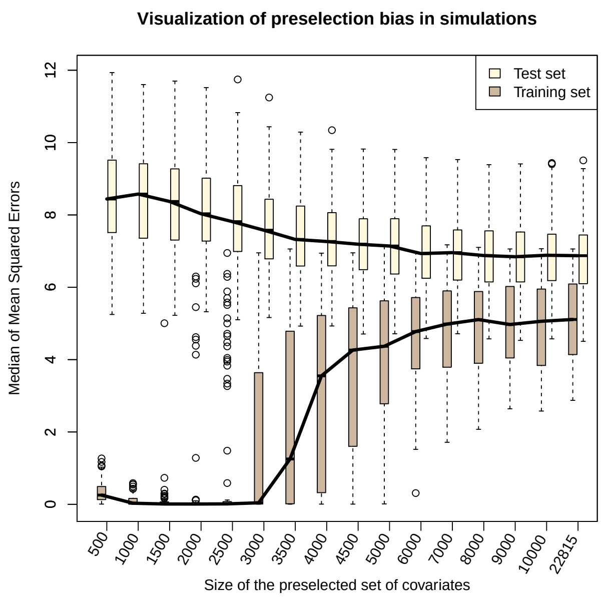


(c) SNR = 2

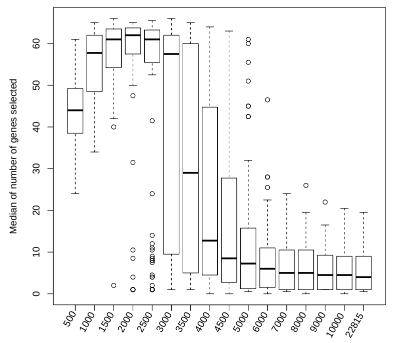


(d) Number of covariates selected

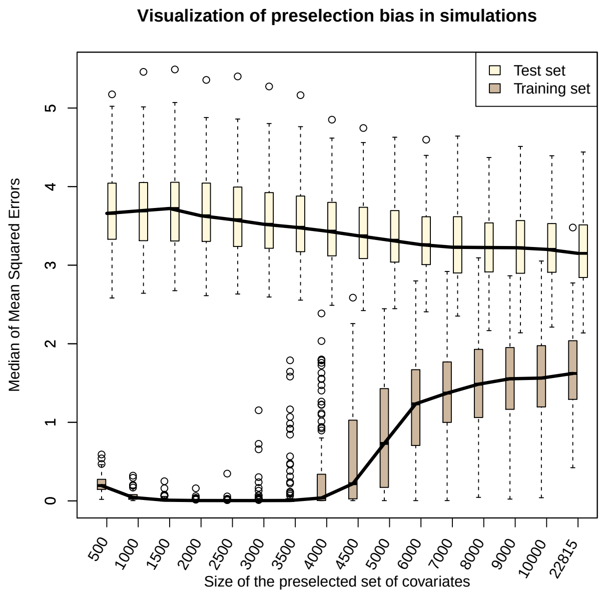
Figure 9.1: ***A.*** *Boxplot of* $s_1 = 100$ *simulations where, for each simulation, the noise* $\boldsymbol{\epsilon}$ *is drawn randomly. For each new simulation, the patients are divided into training and test set* $s_2 = 100$ *times, and the median of the MSE for both the sets are saved. These medians are used to make the boxplots (a) and (c) here. Figure (b) and (d) are made similarly by saving the the median of the number of covariates in the lasso solution in every patient division. Here, the* $p_{all} = 10000$ *covariates are independently simulated from the standard normal distribution.*



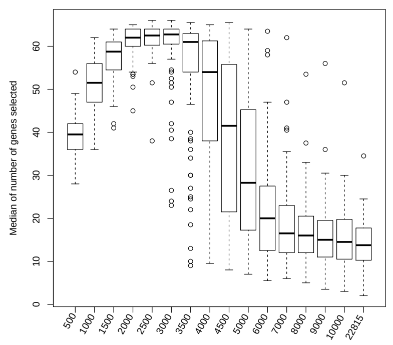
(a) SNR = 0.5



(b) Number of genes selected



(c) SNR = 2



(d) Number of genes selected

*Figure 9.2: **B:** Boxplot of 100 simulations from setting B where the covariates are from the Bone data. See Figure 9.1 for more information on how these plots are made.*

Chapter 5 is used. Here, there are $p_{all} = 22815$ covariates for $N = 84$ patients. When simulating a response for the covariates, we define all the βs to be equal to zero, except for the coefficients corresponding to the 9 genes that were selected from lasso with BMI as the response (see Table 5.1 on page 34). We defined the coefficients corresponding to the first 5 of these genes to be equal to -2 and the remaining 4 coefficients to be 2.

The results for simulations with both $SNR = 0.5$ and $SNR = 2$ are visualized in Figure 9.2. When $SNR = 0.5$ (Figure (a)), i.e there is a lot of noise, the preselection bias that is studied in this thesis is clearly visualized. The median error from 100 simulations will be reduced in the test set as more covariates are included in the preselected set. We see that the median of the MSE stabilizes at around $p = 6000$, and if we had studied the cross validation plot for the $s_1 = 100$ simulations and $s_2 = 100$ divisions of patients, most of the curves would probably freeze at this point where only 26% of the covariates are used. Around this p, the number of covariates selected in the lasso solution also stabilizes at a low level. Even though there seems to be a preselection bias effect when $SNR = 2$ as well, the effect here is less evident. Remark that the y-axes are different in Figure 9.2 (a) and (c), and the error in (c) is quite low for all p. When there is little noise in the dataset, the test set and the training set will be (logically enough) quite similar, and the fitted model will thereby fit fairly well to the test set as well.

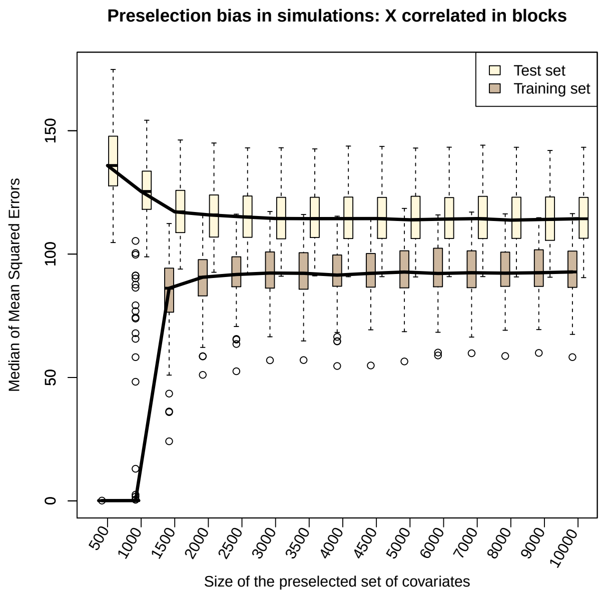
C: Simulation study with covariates correlated in blocks

Simulation B shows that the preselection bias is more evident when a real genomic dataset is used, than when covariates are simulated independently like in Simulation A. In this section, we simulate another covariate matrix $\boldsymbol{X}$ that is more similar to genomic datasets than the covariate matrix with no correlations in A. We introduce a correlation parameter $\rho = 0.9$ and simulate $\boldsymbol{X}$ with blocks of 100 covariates. Within each block, the pairwise correlation between covariate i and j is given by

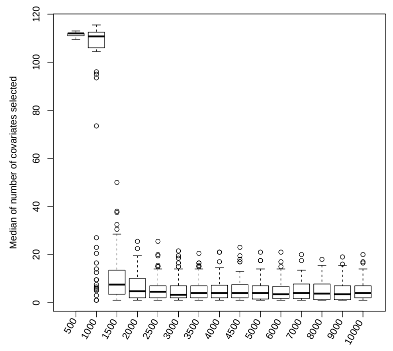
$$cor(i,j) = \rho^{|i-j|},$$

which means that covariates close to each other are highly correlated. The blocks are simulated independently and the size of the full covariate matrix is 200×10000. As in A, five coefficients are chosen to be -2 and five are put equal to 2. The covariates that correspond to these coefficients are all in different blocks.

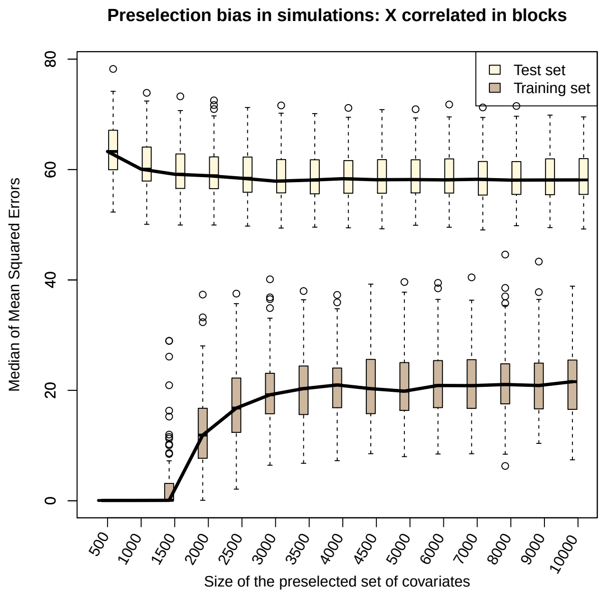
Again, by studying Figure 9.3, we see that the preselection bias effect is most evident when there is a lot of noise in the dataset ($SNR = 0.5$) . In Simulation A, the median of the MSE when $SNR = 0.5$ froze approximately around $p = 1000$ (Figure 9.1(a)), while now, when the covariates are no longer independent, more covariates must be included in the preselected set, but it seems that p between 1500 and 2000 should be satisfactory.



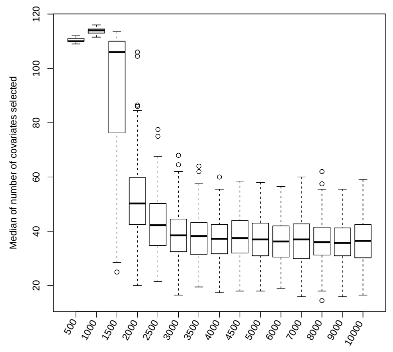
(a) SNR = 0.5



(b) Number of genes selected



(c) SNR = 2



(d) Number of genes selected

*Figure 9.3: **C:** Boxplot of 100 simulations from setting C where the $p_{all} = 10000$ covariates are simulated in blocks of 500 with pairwise correlation from a multivariate normal distribution. See Figure 9.1 for more information on how these plots are made.*

Concluding comments on all the simulations

For all the simulation studies in this thesis it is clear that the preselection bias is most expressed in datasets where there is a large amount of noise. When the training and test set are indeed differentiated because of the high noise level, the fitted model's ability to predict is essential. An overfitted model with too many non-zero coefficients, as the fitted models when the size of the preselected set of covariates is below the size when the cross validation curve freezes, will have poor prediction ability. On the other hand, when there is a less noise and thereby the training and test set are more similar, the ability to predict does not seem to be that effected by the size of the preselected set of covariates because the model fits quite well overall.

Even though the error in the test set is of most interest, the training set can be of help to study tendencies. The changes in the MSE is more pronounced in the boxplots for the training set and can be used to detect how large p must be for the cross validation curves to be frozen. For the test set, the model tends to fit better and stabilizes for the same p as when the training set demonstrates a poorer fit. Also, by studying the boxplots where the number of covariates in the lasso solution are visualized, we see that when the error for the training set jumps to a higher lever, the number of non-zero covariates in the solution is reduced to a noticeably smaller number.

Chapter 10

Summary and conclusion

Summary of the most important topics in this thesis

This thesis is a study of how high dimensional regression problems are affected by the preselection of covariates and when the problem of preselection bias is most evident. High dimensional datasets with thousands or up to millions of covariates, and a significantly smaller number of samples are common in genomics. We have shown examples of different types of genomic datasets in order to give an overview of some settings where high dimensional regression is required and to understand more of the microbiology in this type of data. Furthermore, we have presented theory and methods to deal with high dimensional regression problems, focusing on linear regression and Cox regression for survival analysis. The main focus has been on penalized regression models and the lasso, which is defined with a L_1 penalty so that it does variable selection coupled with prediction. The number of non-zero covariates in a lasso solution tends to be small. That there are few covariates in the lasso model indicates that the penalty parameter λ in the penalized regression model is large. The penalty parameter is found through cross validation for all the examples in this thesis since we have been focusing on optimizing prediction performance.

To avoid working with very large datasets, researchers tend to reduce the number of covariates prior to a penalized regression analysis like the lasso. We have studied how this can lead to preselection bias if the preselection of the covariates is not done properly and shown that the problem of preselection bias is not something that only occurs occasionally in some special datasets. Preselection bias is in fact something one should be aware of when studying all high dimensional datasets, but if the preselection of covariates is done controlled by algorithms as strong (Tibshirani et al. 2012) or freezing (Bergersen et al. 2013), it can be done more safely. But even though a fitted model from almost all datasets may lead to preselection bias when preselection of covariates is done in an ad hoc way, some datasets seem to be more exposed to preselection bias than others. We have seen that for datasets more exposed to preselection bias, the number of covariates needed in the preselected set will often be relatively large.

Challenges and further work

The challenges dealing with huge amounts of data are many, and several choices, that can affect the final result, have to be made along the way in a high dimensional regression analysis. In this thesis, we chose to reduce the dimension of the design matrix for the methylation data in Section 8.3 as a first step in our analysis of the data. The covariates that had less than 0.1 in standard deviation across the patients were excluded from the subsequent analyses. This is a reasonable filtering, in that the probability that these covariates are able to detect anything significant about our response is very small. Our study is on preselection based on the relationship between covariates and the response, but that this extra filtering was necessary shows that unexpected problems often occur when dealing with real data, requiring ad hoc solutions. An alternative solution to filtering the covariates could be to apply the freezing algorithm, but then we would not be able to study the cross validation curve for the "full" dataset to illustrate where the curves froze. Another example of challenges concerning real data is the necessity to manipulate the Lymphoma data in Chapter 6 by using K Nearest Neighbor (KNN) imputation to avoid missing values.

It is also worth mentioning that the lasso is somewhat depending on the ordering of the covariates in $\boldsymbol{X}$. In the Cox example in Chapter 6, 22 covariates were selected by the lasso when the covariates where sorted according to their P-value from a univariate regression model. In contrast, the lasso solution included 23 covariates when the lasso was done before the covariates were sorted. Both these models where fitted with the full dataset with p_{all} covariates. We have used the result from when the covariates were sorted so that the cross validation curve with p_{all} was compatible with the curves when $p < p_{all}$ ordered covariates were used in the lasso.

That freezing occurs when applying the lasso for survival data was shown in both Chapter 6 and 8. In chapter 6, we also studied the preselection bias plot when studying survival data. To be able to analyze how well a Cox regression model fitted a new test set, we had to improvise by studying the partial likelihood for the test set at the point where the partial likelihood for the training set had its maximum. We assumed that the fitted model would give a likelihood function that had its maximum approximately at the same spot if the model fitted the test set as well as the training set. This is an approach open for discussion, and the results in the preselection bias plot in Figure 6.3 can be influenced by that this procedure is less intuitive and straightforward than studying the mean squared error in linear regression. Would another approach show the preselection bias even clearer, in the meaning that the differences in the error when p is increased would be more evident in the test set? Our study only indicates how far away from each other the likelihood functions are, and a check that is able to detect the actual prediction ability of a model in a test set would be desirable. But even though our approach is somewhat weak, we are able to detect a preselection bias effect which coincides with the freezing point in the cross validation plot in Figure 6.2. We have therefore shown that preselection bias is also a problem when dealing with survival data.

The preselected set of covariates will, at some point when p is large enough, include all the covariates that is essential to fit a model without preselection bias. Determining how large p must be to get this same model as if the full dataset was used in the lasso analysis can be done in various ways. As mentioned in the comments in Chapter 9, the point where the preselection bias plot stabilizes (and the cross validation plot freezes) is more evident when studying the error (MSE or deviance) for the training set. This seems to be a recurring phenomenon. Even though it is the error in the test set that is of most interest, we may be able to exploit that the error for the training set increases drastically at the point where the error for the test set stabilizes on a low level. This observation indicates that we can detect how large the preselected set of covariates should be to avoid preselection bias without even studying a test set. Datasets in genomics often have few samples, and by dividing the samples into a training and a test set, the number of samples in each of the two analyses are reduced, $N_{test} \leq N_{training} < N$. To keep the number of samples in the lasso analysis as large as possible, it could be interesting to examine the errors in this thesis without studying a test set. We could rather examine how the error for the model, fitted from the dataset with all the samples N, changes as the size of the preselected set of covariates p is increased. When the error increases drastically and the number of covariates in the lasso solution is reduced, the optimal p is probably found. This approach would also result in that the irregularities for each run of lasso and cross validation are only due to different folds in the cross validation, and not different divisions of the patients. The size of the preselected set of covariates from this approach should correspond to p when the cross validation curves are frozen.

Prediction rules to avoid preselection bias rely on how the covariates are initially ordered. All the covariates included in the lasso solution for the full dataset must be contained in the preselected set if the same solution should be found with a reduced dataset. To be able to find this result, a list of organized covariates must be evaluated and p increased until all essential covariates are included in the preselected set and the correct lasso solution has been found. The ordering of the covariates must be such that the most important covariates are ranked highest. An optimal ordering would reflect the lasso behavior, but in practice an approximation of such ranking is used. By ranking the covariates based on simple univariate computations, Bergersen et al. (2013) showed that the size of the set of covariates to be included in the lasso can be reduced to a small percentage of the original dataset. This conclusion has been reinforced in this thesis.

The prediction challenges that preselection of covariates lead to are most emphasized in high dimensional regression problems with dependencies between covariates, and where there is a large amount of noise, as shown in the simulation studies in Chapter 9. To fit a model that has a good ability to predict is challenging in this setting, and it is important to avoid all effects that may lead to a weaker model. To control the preselection of covariates and thereby avoid preselection bias is therefore an important aspect when it comes to dealing with high dimensional data with dependencies and noise. Independent covariates are rarely the case in real life situations, so the simulation study with correlated covariates are more interesting. Further research should include simulations with interactions between covariates, as is often the case in genomic datasets. We expect that the preselection bias will get even more evident when interactions are included because the univariate sorting of the covariates will not be able to capture the interaction structure of the design matrix. It is then likely that the preselected set of covariates must be larger than for the data with independent covariates and the data with correlated covariates to be able to include all the essential covariates in the lasso analysis.[1] Furthermore, it is likely that by studying even larger datasets than what have been the main focus in this thesis, the problems appearing when there are correlations in the design matrix will be even more evident.

Conclusion

What is new in this thesis is a clear understanding and visualization of how preselection bias affects a model's ability to predict. The connection between the point where the prediction error for a new dataset stabilizes at a low level in a preselection bias plot and the freezing point of a cross validation curve has also been visualized. The concept of freezing has not previously been studied in survival analysis, and we have shown that freezing occurs in survival analysis, as well as in the linear regression setting. Also, the idea that the preselection bias will be more evident in datasets where the univariate sorting is unable to detect the dependence structure in the design matrix is emphasized through the simulation studies. We have stressed the problems that may occur when preselection of covariates is done without applying algorithms designed to control the preselection. This will hopefully guide students and researches to be more alert when manipulating high dimensional datasets by reducing the number of covariates before applying analyses like penalized regression.

[1]Note that the Bone data used in Simulation B is gene expression data from medical studies, and probably contains interactions between some covariates. In this situation, the size of the preselection set included more covariates when the mean squared error for the test set stabilized at a low level, than for the two other simulation studies.

Bibliography

Aalen, O.O, Borgan, Ø., and Gjessing,H.K (2008)
“Survival and Event History Analysis”
Oslo: Springer

Alizadeh, A.A., Eisen, M.B., Davis, R.E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J.C., Sabet, H., Tran, T., Yu, X., Powell, J.I., Yang, L., Marti, G.E., Moore, T., Hudson, J.Jr, Lu, L., Lewis, D.B.,Tibshirani, R., Sherlock, G., Chan, W.C., Greiner, T.C., Weisenburger, D.D., Armitage, J.O., Warnke, R., Levy, R., Wilson, W., Grever, M.R., Byrd, J.C., Botstein, D., P. O. Brown, P.O., and Staudt, L.M. (2000)
"Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling"
Nature, 403:503-511
Dataset: http://llmpp.nih.gov/lymphoma/

Ambroise, C. and McLachlan, G.J. (2002)
“ Selection bias in gene extraction on the basis of microarray gene-expression data”
Proceedings of the National Academy of Sciences (PNAS), 99(10):6562-6566.

Bergersen, L.C., Ahmed, I., Frigressi, A., Glad, I.K., and Richardson, S. (2013)
“Preselection in Lasso-type Problems by Cross-validation Freezing”
Manuscript

Bioteknologinemnda (2009)
"Arv og genetikk"
http://www.bion.no/filarkiv/2010/07/temaark_arv_og_genetikk_v2_2009.pdf, visited 22.04.14.

Bush, W.S., and Moore, J.H. (2012)
"Chapter 11: Genome-Wide Association Studies"
PLOS Computational Biology, 8(12):e1002822.

Bühlmann, P. and van de Geer, S. (2011)
“Statistics for High-Dimensional Data”
Berlin: Springer.

Bøvelstad, H.M., Nygård, S., Størvold, H.L., Aldrin, M., Borgan, Ø., Frigessi, A., and Lingjærde, O.C. (2007)
"Predicting survival from microarray data - a comparative study"
Bioinformatics, 23(16):2080-2087.

Cho, S., Kim, K., Kim, Y. J., Lee, J.-K., Cho, Y.S., Lee, J.-Y., Han, B.-G., Kim, H., Ott, J., and Park, T. (2010)
"Joint identification of multiple genetic variants via elastic-net variable selection in a genome-wide association analysis"
Annals of Human Genetics, 74(5):416-428.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R. (2004)
"Least Angle Regression"
The Annals of Statistics, 32(2):407-499.

El Camino Hospital, Genomic Medicine Institute (04.2014)
Illustration from webpage:
http://elcaminogmi.dnadirect.com/img/content/tests/acgh/hybridization.gif, vistited 28.04.14.

El Ghaoui, L., Viallon, V., and Rabbani, T (2011)
"Safe feature elimination for the lasso and sparse supervised learning problems"
ArXiv e-prints, 1009.4219.

Fan, J. and Lv, J. (2008)
"Sure independent screening for ultra high dimmensional feature space" (SIS)
Journal of the Royal Statistical Sosiety: Series B (Statistical Methodology), 70(5):849-922.

Friedman, J., Hastie, T., and Tibshirani, R. (2013)
"Pachage 'glmnet' "
http://cran.r-project.org/web/packages/glmnet/glmnet.pdf, visited 20.02.14.

Gordon, G. and Tibshirani, Ryan (09.2014)
"Karush-Kuhn-Tucker conditions" (slides)
https://www.cs.cmu.edu/~ggordon/10725-F12/slides/16-kkt.pdf,
visited 22.09.14.

Gui, J. and Li, H. (2005)
" Penalized Cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data"
Bioinformatics, 21(13):3001-3008.

Hastie, T., Tibshirani, R., and Friedman, J. (2009)
"Elements of statistical learning"
Springer, 2nd edition

Hoerl, A. E. and Kennard, R. (1970)
"Ridge regression: Biased estimation for nonorthogonal problems"
Technometrics, 12:55-67.

Lee, J. K. (2010)
"Statistikal Bioinformatics - For Biomedical and Life Science Researchers"
New Jersey: Wiley-Blackwell

Mandal, A. (2014)
"DNA Methylation - What is DNA Methylation?"
http://www.news-medical.net/health/DNA-Methylation-What-is-DNA-Methylation.aspx, visited 24.04.14.

National Cancer Institute (10.2014)
"aCGH copy number data"
https://wiki.nci.nih.gov/display/TCGA/aCGH+copy+number+data, visited 10.10.14.

National Human Genome Research Institute (2011)
"DNA Microarray Technology"
http://www.genome.gov/10000533, visited 28.04.14.

National Human Genome Research Institute (04.2014)
Illustration from webpage:
http://www.genome.gov/Pages/Education/DNADay/TeachingTools/MakingSNPsMakeSense.html , visited 28.04.14.

Oregon State University (2013)
"Microarray - How does It Work?"
http://www.unsolvedmysteries.oregonstate.edu/microarray_07,
visited 28.04.14.

Phillips, T. (2008)
"The Role of Methylation in Gene Expression"
Nature Education, 1(1):116.

Reppe, S., Refvem, H., Gautvik, V.T., Olstad, O.K., Høvring, P.I., Reinholt, F.P., Holden, M., Frigessi, A., Jemtland, R., and Gautvik, K.M. (2010)
"Eight genes are highly associated with BMD variation in postmenopausal Caucasian women"
Bone, 46(3):604-12.

Rosenwald, A., Wright, G., Chan, W.C., Connors, J.M., Campo, E., Fisher, R.I., Gascoyne, R.D., Muller-Hermelink, H.K., Smeland, E.B., and Staudt, L.M. (2002)
"The use of Molecular Profiling to Predict Survival after Chemotherapy for Diffuse Large-B-Cell Lymphoma"
The New England Journal of Medicine, 346(25):1937-1947.

The dataset used in the chapter wtih Cox regression is found on http://llmpp.nih.gov/DLBCL/ , where the patient data is under Supplemental Data, and the gene expression data is found in the file Web Figure 1 Data file.

Røine, M. (2013)
"Weighted Lasso Analysis for Cervix Cancer Data"
Thesis, University of Oslo.

The Shape of Data (04.2014)
Illustration from webpage:
http://shapeofdata.wordpress.com/2013/03/26/general-regression-and-over-fitting/, visited 30.04.14.

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2011)
"Regularization Paths for Cox's Proportional Hazards Model via Coordinat Descent"
Journal of Statistical Software, 39(5) .

Sjøberg, N. O. (2006)
"Molekylær genetikk"
Forlaget Vett & Viten AS, 4.utgave

Tibshirani, R. (1996)
"Regression Shrinkage and Selection via the Lasso"
Journal of the Royal Statistical Sosiety, 58(1):267-288.

Tibshirani, R. (1997)
"The Lasso method for variable selection in the Cox model"
Statistics in Medicine, 16:385-395.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., and Tibshirani, R.J (2012)
"Strong rules for Discarding Predictiors in Lasso-type Problems"
Journal of the Royal Statistical Society: Series B, 74(2):245-266.
We have also used slides on this subject (Hastie and Tibshirani 2009) from Tibshiranis homepage (http://statweb.stanford.edu/ tibs/research.html, visited 09.05.14).

Verweij, P. J. M. and van Houwelingen, H. C. (1993)
"Cross-validation in survival analysis"
Statistics in Medicine, 12:2305-2314.

Vidaurre, D., Bielza, C., and Larrañaga, P. (2013)
"A Survey of L_1 Regression"
International Statistical Review, 81(3):361-387.

Waldron, L., Pintilie, M., Tsao, M.-S., Shephard, F., Huttenhower, C., and Jurisica, I. (2011)
"Optimized application of penalized regression methods to diverse genomic data"
Bioinformatics, 27(24):3399-3406.

Yuan, M. and Lin, Y. (2006)
"Model selection and estimation in regression with grouped variables"
Journal of the Royal Statistical Sosiety, 68(1):49-67.

Zou, H. (2006)
"The Adaptive Lasso and its Oracle Properties"
Journal of the American Statistical Association, 101(476):1418-1429.

Zou, H. and Hastie, T. (2005)
"Regularization and variable selection via the elastic net"
Journal of Royal Statistical Society, 67(2):301-320.
Slides on the topic: www.stanford.edu/~hastie/TALKS/enet_talk.pdf

Zhang, X., Huang, S., Zhang, Z., and Wang, W. (2012)
"Chapter 10: Mining Genome-Wide Genetic Markers"
PLOS Computational Biology, 8(12): e1002828.

Appendix A

A1: The effect of folds in the preselection bias plot

The solution from a lasso analysis is somewhat affected by the folds in the cross validation. In the preselection bias plots throughout this thesis we have made boxplots showing the variation due to the splitting of patients into training and test sets. The effect of the folds is neglected, and in this appendix we will show that the variation due to the folds will not affect how the medians evolve in the preselection bias plots. The variation due to the division of patients into training and test sets dominates the variation due to the folds. The check is constructed using the Bone dataset from Chapter 5, but we assume that the results will hold for all examples in this thesis.

When visualizing the preselection bias through boxplots with different divisions of patients, we let the *cv.glmnet* algorithm in *R* choose folds in the cross validation at random for each of the patient divisions. Table 5.1 showed that this was acceptable when all the patients were included because the lasso and cross validation chose approximately the same genes even though different folds were used. This is checked for a random training set of 60 patients as well. Table A1.1 indicates that this also gives a quite stable result. To validate that these stable result hold for different divisions of the patients, and not only for this random patient group, we have constructed a check using boxplots.

Gene index	Chosen
5954	97
7208	97
7089	94
11194	73
16482	73
1492	13

Table A1.1: The result of running lasso 100 times with different folds in the cross validation for one random training set of 60 patients.

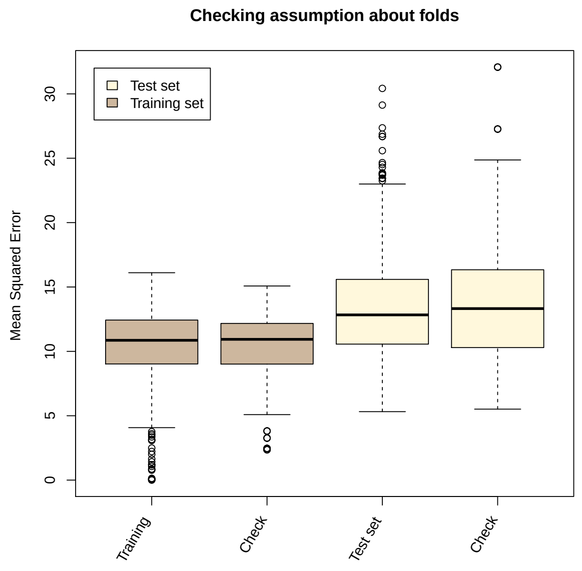


Figure A1.1: The boxes with labels "Training" and "Test set" show the median of the different MSE for 500 divisions of the patients and the full dataset using all covariates, i.e the same as the last box in Figure 5.3 (a) and (b). The "Check"-boxes show 100 different divisions of folds for each of 100 divisions of patients. The median of the two sets are approximately at the same level in the original boxplot and in the check to detect if the random folds for each division of the patients would ruin the credibility of the visualization of the preselection bias.

When testing with 100 different folds for every division of patients, it is assumed that there will be more variation than with only one set of folds for each division of patients. The median should be approximately at the same level as in the boxplot for p_{all} with folds chosen at random for each division of the patients. We do this to be able to trust the results in Figure 5.3 and 5.4, where we have not taken into account the effect the different folds have on the results. Figure A1.1 shows that our assumption seems to hold, and that we do not need to take the effect of the folds into account when we study the median of several patient divisions. The effect of different splittings of the patients will by far exceed the effect of different folds.

Note that because of the time of computation, we check with 100 different folds when dividing the patients into training and test sets 100 times, not 500 times that is the number of patient splittings done in the preselection bias plots.[2] Therefore, the variance might not be that large in the checking of the effect of different folds as we would expect if we tested with 100 folds for all the 500 patients groups.

The conclusion here is that it should be safe to trust the result in Figure 5.3 and the

[2]The 100 times 100 computations of lasso and cross validation took approximately one week to run.

preselection bias plot in Figure 5.4, without controlling for different folds for each new division of the patient groups and for each new p.

A2: Imputation

When analyzing real data, a common problem is that there are missing values in the dataset. This can for example be caused by faults in the measuring equipment when collecting gene expression data. How these missing values are dealt with will influence the results of the analyses that is done. Setting the missing values to zero is maybe the easiest way out of the problem with missing values, but will not always make sense when advanced analyses of the data is of interest.

The most common way of dealing with missing values is to impute the NA (not available) numbers with some kind of mean of the samples that are closest to the sample with a missing value. One example of such an imputation is the K Nearest Neighbor (KNN) method that we applied on the dataset for survival of patients with Large-B-Cell lymphoma from Rosenwald et.al (2002). In this dataset, some genes had a large number of missing values. To impute a missing gene expression for a patient S, the k (we used k=8) patients that are most similar to patient S are found, and the missing gene expression value for patient S is imputed to be the mean of the gene expressions for the specific gene from these k patients.

Because it will be difficult to compare different patients if most of the patients has the same covariate as missing, KNN is not used if more than some percentage of the observations are missing. In our example with survival data, we removed the genes that had more than 50% missing values and applied KNN on the remaining genes.

A3: Details about the dataset for Cox regression

The dataset studied in Chapter 6 is downloaded from http://llmpp.nih.gov/DLBC. We have connected the patient data (under Supplemental Data) and the gene expression data from DNA microarrays ("Web Figure 1 Data file") by the "LYM-number".

Five of the patients had zero as follow up time, and to get the Cox proportional hazard analysis to work, we define that these patients have $time = 0.0001$. Since this is such a small number, the analysis will be marginally effected by this change. Around 10 percent of the total 1.8 million (7399x240) gene expressions was not defined. We have removed the 185 genes that had more than 50% missing values. This is 2.5% of the original 7399 genes. Furthermore, we have imputed the rest of the missing values by using the K Nearest Neighbors method (KNN) with $k = 8$. This is done through the function $impute.knn()$ from the package *impute* from Bioconductor version 2.13.

Appendix B: Scripts in R

B1: Code for Chapter 5

A linear regression model is used when we analyze the Bone data with BMI as the response. The following code shows how to implement many of the main ideas in Chapter 6, 8 and 9 although modifications will be needed for the different regression models and datasets.

```
#****************************
# THE BONE DATA
#****************************
# THE COVARIATES:
data     <-  read.csv("Bone.csv",sep=",",header=TRUE,row.names=1)
data     <- as.matrix(data)
gener    <- t(data) #nxp matrise

#Delete columns with Total Hip T-score and with ,,,
gener    <- gener[,-(1:2)]
gennavn <- colnames(gener)
id       <- rownames(gener)
p        <- length(gennavn) #22815 genes
n        <- length(id)      #84 women

# THE RESPONSE:
respons <- read.csv("MoreResponses.csv")
bmi      <- respons[3,]
bmi      <- bmi[,-1]
idres    <- colnames(bmi)
bmi      <- as.numeric(bmi)

#****************************
# RUN LASSO MANY TIMES
#****************************
# Load package
library(glmnet)

dfny  <- rep(0,100)
maxdf <- 100
keepindex <- matrix(rep(0,maxdf*100),nrow=100,maxdf)
keepcoef  <- matrix(rep(0,maxdf*100),nrow=100,maxdf)
for (i in 1:100){
  # Cross-validation
  cv  <- cv.glmnet(gener,bmi,alpha=1)
  fit <- glmnet(gener,bmi,alpha=1,lambda=cv$lambda.min)
```

```r
  dfny[i] <- fit$df
  co      <- coef(fit)
  index   <- rep(0,fit$df)
  k=1
  for (l in 1:p){
    if (co[l] != 0) {
      index[k] = l
      k=k+1
    }
  }
  keepindex[i,] <- c(index,rep(0,(maxdf-length(index))) )
  keepcoef[i,]  <- c(co[index],rep(0,(maxdf-length(co[index]))))
}

#*****************************************************************
# CROSS VALIDATION PLOT WHEN MORE AND MORE COVARIATES ARE INCLUDED
#*****************************************************************
# Training set of 60 patients(test set will not be used yet)
gener1 <- gener[sett1,] #sett1 contains 60 indexes between 1 and 84
bmi1   <- bmi[sett1]

# Sort covariates by univariate correlation with the response (sett1)
cormat <- matrix(rep(0,p*2),ncol=2)
for (i in 1:p) { # go through all the genes
  # i+1 to correspond with the indexes from glmnet (intercept)
  cormat[i,1] <- i+1
  cormat[i,2] <- cor(gener1[,i],bmi1, method="pearson")
}
corsortabs <- cormat[sort.list(abs(cormat[,2]),decreasing = TRUE),]

# Fix folds
# "minefolder" is a vector of numbers between 1 and 10, length 60

# ---- Run lasso one time to fix lambda grid ----
# Cross validation in training set (n1 patients)
cv1 <- cv.glmnet(gener1,bmi1,alpha=1,foldid=minefolder)
plot(cv1)
# Choose lambda with lowest prediction error
lambda1min<- cv1$lambda.min
fit        <- glmnet(gener1,bmi1,alpha=1,lambda=lambda1min)
df1_all    <- fit$df # number of non-zero coefficients in model
lambdagrid<- cv1$lambda

# Change for pny = c(500,1000,2000,3000,4000,5000,p_all)
#----------------------------------
pny        <- 500
X_500      <- gener1[ ,corsortabs[1:500,1]-1]
cv_500     <- cv.glmnet(X_500,bmi1,alpha=1,
              foldid=minefolder,lambda=lambdagrid)
lambda1min<- cv_500$lambda.min
fit        <- glmnet(X_500,bmi1,alpha=1,lambda=lambda1min)
df1_500    <- fit$df
co         <- coef(fit)
behold     <- rep(0,fit$df)
j=1
for (i in 1:(pny+1)){
  if (co[i]!=0) {
    behold[j] <- i
    j=j+1
}}
koeffisientind[1, ] <- c(behold,rep(0,90-length(behold)))
koeffisienter[1, ]  <- c(co[behold],rep(0,90-length(behold)))
```

```
gennavnny[1, ]        <- c(colnames(X_500)[behold -1],rep(0,90- fit$df))
#-----------------------------------

# Freezing plot
# Put all the cross validation curves in one plot
plot(log(lambdagrid),cv_500$cvm,ylim=c(12,23),
     type="l",lty=1,ylab="Mean-Squared Error",
     main="Cross validation plot for different p")
lines(log(lambdagrid),cv_1000$cvm,type="l",lty=2)
lines(log(lambdagrid),cv_2000$cvm,type="l",lty=3)
lines(log(lambdagrid),cv_3000$cvm,type="l",lty=4)
lines(log(lambdagrid),cv_4000$cvm,type="l",lty=5)
lines(log(lambdagrid),cv_5000$cvm,type="l",lty=6)
lines(log(lambdagrid),cv1$cvm,type="l",lty=1,lwd=3,col=2)
lines(rep(log(lambda1min),20),seq(10,29,by=1),type="l",lty=3)
legend("topright",
      c("All genes","500","1000","2000","3000","4000","5000"),
       lty=c(1,1:6),col=c(2,1,1,1,1,1,1),lwd=c(3,rep(1,6)))

#*******************************************************
# STUDY THE EFFECT OF PRESELECTION IN TRAINING AND TEST SET
#*******************************************************
n1 <- 60; n2 <- n-n1
pvektor <- c(500,1000,1500,2000,2500,3000,3500,4000,
             4500,5000,6000,7000,8000,9000,10000,p)
one1    <- rep(1,n1); one2 <- rep(1,n2)

# Function to make test set, given the training set
settcontains <- function(i) {
  for (k in 1:n1) {
    if (sett1[k]==i) { return(TRUE) }}
  return(FALSE)
}

#******************************
# FUNCTION
# - divide patients S times
# - does lasso on training set
# - MSE for both training and test
#**********************************
MSE_in_training_test <- function(S,oppdel_navn) {
for (s in 1:S) { # divisions of patients
  # Define the training set and the test set
  sett1 <- sort(sample(c(1:n),n1, replace=F))
  X1    <- X[sett1,]
  y1    <- y[sett1]
  sett2 <- rep(0,n2)
  k=1
  for (i in 1:n) {
   if (!settcontains(i)) { sett2[k]=i; k=k+1 }
  }
  X2   <- X[sett2,]
  y2   <- y[sett2]

  # sort according to the correlation with the response
  cormat <- matrix(rep(0,p*2),ncol=2)
  for (i in 1:p) {
    cormat[i,1] <- i+1
    cormat[i,2] <- cor(X1[,i],y1, method="pearson")
  }
  corsortabs <- cormat[sort.list(abs(cormat[,2]),decreasing = TRUE),]

  dfall <- rep(0,length(pvektor))
  e1all <- rep(0,length(pvektor))
  e2all <- rep(0,length(pvektor))
  for (t in 1:length(pvektor)) {
    pny  <- pvektor[t]
    Xny1 <- X1[,corsortabs[1:pny,1]-1]
    cv1  <- cv.glmnet(Xny1,y1,alpha=1)
    lambdamin <- cv1$lambda.min
    fit  <- glmnet(Xny1,y1,alpha=1,lambda=lambdamin)
    df1  <- fit$df
    dfall[t] <- df1

    co   <- coef(fit)
    betaindex <-  rep(0,df1)
    j=1
    for (i in 1:(pny+1)){
     if (co[i]!=0) {
        betaindex[j] = i; j=j+1
      }
    }
    betaene <- co[betaindex]
    # Training set
    Xene1   <- cbind(one1,Xny1[,betaindex -1])
    yhat1   <- Xene1 %*% betaene
    e1      <- y1-yhat1
    e1all[t]<- sum(e1^2)/n1
    # Test set
    Xny2    <- X2[,corsortabs[1:pny,1]-1]
    Xene2   <- cbind(one2,Xny2[,betaindex -1])
    yhat2   <- Xene2 %*% betaene
    e2      <- y2-yhat2
    e2all[t]<- sum(e2^2)/n2
  }
  save(lambdaall,cvverdi,dfall,e1all,e2all,
     file=paste(oppdel_navn,run,".RData",sep=" "))
}
} # end of function ----------------------

S <- 500 # number of different divisions of the patients
MSE_in_training_test(S=S, oppdel_navn="oppdeling")

#*************************************************************
# MAKE PRESELECTION BIAS PLOT FOR 500 DIVISIONS OF THE PATIENTS
#*************************************************************
plength    <- length(pvektor) # number of different p's
lambdamat <- matrix(rep(0,S*plength),nrow=S)
cvverdimat<- matrix(rep(0,S*plength),nrow=S)
dfmat      <- matrix(rep(0,S*plength),nrow=S)
e1mat      <- matrix(rep(0,S*plength),nrow=S)
e2mat      <- matrix(rep(0,S*plength),nrow=S)

for (s in 1:S) {
  load(paste("oppdeling",s,".RData",sep=" "))
  lambdamat[s,]  <- lambdaall
  cvverdimat[s,] <- cvverdi
  dfmat[s,] <- dfall
  e1mat[s,] <- e1all
  e2mat[s,] <- e2all
}
med_e1 <- apply(e1mat,2,median)
med_e2 <- apply(e2mat,2,median)

```

```
# Preselection bias plot
# Training and test set in the same plot
boxplot(e1mat, at = 0:15*3 + 1,ylim=c(0,35) , xaxt = "n",
  col="bisque3",ylab="Mean Squared Error",
  xlab="Size of the preselected set of covariates",
  main="Visualization of preselection bias")
boxplot(e2mat, at = 0:15*3 + 2, xaxt = "n", add = TRUE, col="cornsilk1")
axis(1,at=0:15*3+1.5,labels=FALSE)
text(x =0:15*3+1.5, y = par("usr")[3] - 1, srt = 60, adj = 1.2,
     labels = pvektor, xpd = TRUE)
legend(x=35,y=36,c("Test set","Training set"),fill=c("cornsilk1","bisque3"))
lines(0:15*3 + 1,med_e1,lwd=5))
lines(00:15*3+2,med_e2,lwd=5)
```

B2: Code for Chapter 6

In this chapter, we use Cox regression instead of linear regression that was used in Chapter 5. This makes the programming a bit more advanced, but the idea is the same as for the programs in B1 for Chapter 5. The main differences is

- The response y must be defined as a *Surv* object including time and status. *Surv* is a part of the *survival*-package.
- The covariates are sorted according to their P-value from a univariate Cox regression model before the top p covariates are included in the preselected set of covariates. We have used the default P-value in *coxph* from the *survival*-package, which is calculated from the Wald test. The Wald test statistics is

$$Z = \frac{\hat{\beta}_j}{se(\hat{\beta}_j)},$$

  which is approximately standard normally distributed under the null hypothesis that $\beta_j = 0$.
- In *glmnet*, the variable $family$ must be defined as $"cox"$.
- It is not possible to predict the responses for a new dataset, so the deviance is used to determine how well a model fits for the new dataset.

When doing cross validation for survival data in the *glmnet*-package in *R*, specifying $family = "cox"$, the default is $grouped = TRUE$. This choice gives a more efficient use of the risk sets because here the cross validation partial likelihood is obtained for the Kth fold by taking the log partial likelihood evaluated on the $(K-1)/K$ of the dataset and subtract the log partial likelihood evaluated on the full dataset, which corresponds to formula (3.10) in Section 3.4. This is explained on the page for the *R*-package *glmnet* (Friedman et al. 2013).

```r
#***************************
# DATA FOR SURVIVAL ANALYSIS
#***************************
library(survival)

dat2 <- read.table("http://llmpp.nih.gov/DLBCL/NEJM_Web_Fig1data",
          header=T,sep="\t",fill=T,quote="")
dim(dat2) # 7399  295
# the first two columns of the file include unique ID and name
X     <- as.matrix(dat2[,3:295] )
p     <- dim(X)[1]
rownames(X) <- dat2[,1]

patients <- read.table(
  "http://llmpp.nih.gov/DLBCL/DLBCL_patient_data_NEW.txt",
  header=T,sep="\t")
dim(patients)# 240  12
n_patients <- dim(patients)[1]

# EXTRACT "LYM NUMBER" FROM COLNAMES OF X
n_LYM       <- 274    # Stop at 274
LYM_string <- rep(0,n_LYM)
LYMno       <- rep(0,n_LYM)
for (i in 1:n_LYM){
  LYM_string[i] <- substr(colnames(X)[i],13,15)
}
# some names has a bit different names:
LYM_diff_names <- c(144,233,171,12,90)
for (i in LYM_diff_names){
  LYM_string[i] <- substr(colnames(X)[i],14,16)
}
for (i in 1:n_LYM) {
  LYMno[i] <- as.numeric(LYM_string[i])
}

Xnew    <- X[,1:n_LYM]
dim(Xnew) #7399  274
colnames(Xnew) <- LYMno

# CONNECT GENES AND SURVIVAL OF PATIENTS
LYMpatients <- patients[,1]
# Connect the LYM-numbers from the gene expressions to the patients
Xnew      <- Xnew[, colnames(Xnew) %in%  LYMpatients ]
# sort the genes and patients after LYM number
Xnew     <- Xnew[,sort.list(as.numeric(colnames(Xnew)),decreasing=F)]
patients<-patients[sort.list(as.numeric(patients[,1]),decreasing=F),]
dim(Xnew)# 7399  240

# TIME AND STATUS FOR THE SURVIVAL DATA
time_years      <- patients[,3] # Follow up years
time_years      <- patients[,3] # Follow up years
# change time 0 to a small number to be able to include these
time0_patients <- which(time_years==0)
time_years[which(time_years==0)] <- 0.0001
# Status: 1=dead, 0=censored
deadalive <- patients[,4]
status     <- rep(NA,length(deadalive))
for (i in 1:length(deadalive)) {
  if (deadalive[i] == "Alive") {status[i]=0}
  else if (deadalive[i] == "Dead") {status[i]=1}
}
```

```

# DEAL WITH MISSING VALUES
length(which(is.na(Xnew)))/(7399*240) # approx 0.1 was NA
#source("http://bioconductor.org/biocLite.R")
#     biocLite("impute"))
library(impute)

# Remove genes with more than 50% missing
missing50    <- length(which(countNA>0.5)) #  185 (2,5%)
Xnew_remove <- Xnew[-which(countNA>0.5),]
dim(Xnew_remove) # 7214  240

# Impute missing values with KNN, k=8
Xtest_more_rem <- impute.knn(Xnew_remove,k=8)
Xtest_rem        <- Xtest_more_rem$data
Xnew             <- Xtest_rem

#*******************************
# RUN LASSO MANY TIMES
#*******************************
y      <- Surv(time_years,status)

iseed <- 171:270
#save one row for each run of glmnet
Results <- matrix(NA,ncol=2,nrow=length(iseed))
rownames(Results) <- iseed
colnames(Results) <- c("lambdamin", "df")
listnonzero        <- list()

for(seed in iseed){
  set.seed(seed)
  print(paste("seed =", seed, sep=""))
  cv.fit <- cv.glmnet(x=t(Xnew), y=y, family="cox",
               standardize=T,alpha=1)
  fit     <- glmnet(x=t(Xnew), y=y,family="cox",
               lambda=cv.fit$lambda.min,alpha=1)
  Results[rownames(Results)==seed,] <- c(cv.fit$lambda.min,fit$df)
  coef     <- as.vector(fit$beta)
  coef1    <- cbind(rownames(Xnew),coef)
  nonzero <- coef1[coef!=0][1:fit$df]
  nonzerobeta <- coef1[coef!=0][(fit$df+1):(2*fit$df)]
  listnonzero[[paste("seed", seed , sep="")]] <- list(nonzero,nonzerobeta)
}

#*************************************
# PRESELECTION BIAS IN DATA FOR SUVIVAL
#*************************************
p_all <- dim(Xnew)[1]
n      <- dim(patients)[1]
X      <- t(Xnew)

# divide the patients into two different groups
n1 <- 120; n2 <- n-n1
pvektor <- c(200,500,1000,1500,2000,2500,
              3000,3500,4000,4500,5000,p_all)

S <- 500
for (s in 1:S) {
  # Define training and test set
  sett1 <- sort(sample(c(1:n),n1, replace=F))
  X1     <- X[sett1,]
  y1     <- Surv(time_years[sett1],status[sett1])

  sett2 <- rep(0,n2)
  k=1
  for (i in 1:n) {
    if (!settcontains(i)) { sett2[k]=i; k=k+1 }
  }
  X2   <- X[sett2,]
  y2   <- Surv(time_years[sett2],status[sett2])

  # Find the p-value for every probe from univarite cox regression:
  save_pvalue <- matrix(rep(NA,p_all*2),ncol=2)
  for (i in 1:p_all){
    x   <- (X1[,i])
    fit <- coxph(y1~x)
    save_pvalue[i,1] <- i
    save_pvalue[i,2] <- summary(fit)$coef[5]
  }
  # and sort the probes after their P-value
  probe_pvalue <- cbind(save_pvalue[,1],colnames(X),save_pvalue[,2])
  class(probe_pvalue) <- "numeric"
  probe_sort    <- probe_pvalue[sort.list(probe_pvalue[,3],
                   decreasing=F),]
  probe_sort    <- as.matrix(probe_sort)

  lambdaall <- rep(0,length(pvektor))
  cvverdi    <- rep(0,length(pvektor))
  dfall      <- rep(0,length(pvektor))
  devtrainall <- rep(0,length(pvektor))
  devtestall  <- rep(0,length(pvektor))

  for (t in 1:length(pvektor)) {
    pny  <- pvektor[t]
    X1ny <- X1[,probe_sort[1:pny,1]]
    cv1  <- cv.glmnet(X1ny,y1,alpha=1,family="cox")
    lambdamin    <- cv1$lambda.min
    lambdaall[t]<- lambdamin
    cvverdi[t]  <- min(cv1$cvm)
    fit        <- glmnet(X1ny,y1,alpha=1,lambda=lambdamin,family="cox")
    df1        <- fit$df
    dfall[t] <- df1
    coef.max <- coef(fit)
    # training set
    devtrainall[t] <- coxnet.deviance(y=y1,x=X1ny,beta=coef.max)
    # test set
    X2ny       <- X2[,probe_sort[1:pny,1]]
    dev.test <- coxnet.deviance(y=y2,x=X2ny,beta=coef.max)
    devtestall[t]  <- dev.test
  }
  save(lambdaall,cvverdi,dfall,devtestall,devtrainall,
       file=paste("newdata_KNN_oppdeling",s,".RData",sep=""))
}
```

B3: Code for Chapter 8

As the dataset in Chapter 6, the methylation data is survival data and we specify *family* $=''$ *cox''* in *glmnet*. The freezing plots (cross validation plot for different sizes *p* of the preselected set of covariates) are made similar for the datasets in Chapter 5, 6 and 8, but in the latter two the covariates are sorted according to the P-value from a univariate Cox regression model instead of the Pearson correlation used in the linear regression case in Chapter 5.

```r
#*****************************
# METHYLATION DATA (SURVIVAL)
#*****************************
Metyl_info    <- read.table(
    "dataset/Methylation_beta_Normalized_AnaPasUSNPflt_detPFjern_Stdv0.1.txt",
    header=T, sep="\t", na.strings = "")
Patient_data <- read.table("dataset/Patient_data.txt",
                  header=F, skip=1)
colnames(Patient_data)<- c("147","Illumina_pasienter","Tid","Status")

# Prepare the dataset
PatientsID <- Patient_data$Illumina_pasienter[1:150]
xm          <- Metyl_info[, colnames(Metyl_info) %in%  PatientsID]
index_Patients <- which(PatientsID %in% colnames(xm))
# update PasientID with only the pasients that have Metyl_info
PatientsID <- Patient_data$Illumina_pasienter[index_Patients]
t           <- Patient_data$Tid[Patient_data[,2] %in% PatientsID]
s           <- Patient_data$Status[Patient_data[,2] %in% PatientsID]
y           <- Surv(t,s)

p_all <- 168045
n     <- length(PatientsID) #147 patients

#**************************
# RUN LASSO MANY TIMES
#**************************
# Similar as in Chapter 6, but with different seeds
# (and longer computation time)
iseed <- 123:222

#**************************
# FREEZING PLOT
#**************************
# Do univariate cox regression for each probe
# This takes a long time!
save_pvalue <- rep(NA,p_all)
for (i in 1:p_all){
  x   <- t(xm[i,])
  fit <- coxph(y~x)
  save_pvalue[i] <- summary(fit)$coef[5]
}
# sort the probes after their P-value
probe_pvalue<-cbind(probeID,save_pvalue)
probe_sort   <-probe_pvalue[sort.list(probe_pvalue[,2],decreasing=F),]
# gives the index of the probes

X          <- t(xm)
seed_now <- 172

# Cross validation plot for different sizes p,
# is made similarly as in Chapter 5
set.seed(seed_now)
pny      <- 500
X_pre   <- X[ ,probe_sort[1:pny,1]]
cv_pre <- cv.glmnet(X_pre,y,lambda=lambdagrid, family="cox")
lambda1min <- cv_pre$lambda.min
```

```
fit     <- glmnet(X_pre,y,family="cox",lambda=lambda1min)
df_pre <- fit$df
```

B4: Code for Chapter 9

All the datasets here are simulated from linear models with Gaussian noise. Therefore, the programmed function for Chapter 5 in B1 can be used when splitting the patients into different training and test sets and computing the mean squared error.

```
#**********************************************************
# FUNCTION TO SIMULATE NEW NOISE
# and save data to make preselection bias plot
#**********************************************************
simulate_this <- function(snr,S,oppdel_navn) {
  # ADD NOISE
  sigma1 <- sqrt(apply(ysim,2,sd)^2/snr)
  eps     <- rnorm(n,0,sigma1)
  y       <- ysim + eps

  # Function from code for Chapter 5: Calculate MSE
  # in training and test set for S divisions of patients
  MSE_in_training_test(S, oppdel_navn)
  # (Saved only dfall,e1all,e2all here)
}

#**********************************
# FUNCTION TO DO HUNDRED SIMULATIONS
#**********************************
hundred_simulations <- function(snr=snr,S=S,oppdel_navn,run_navn){
  for (run in 1:num_runs) {
    simulate_this(snr=snr,S=S,oppdel_navn)

    dfmat <- matrix(rep(0,S*plength),nrow=S)
    e1mat <- matrix(rep(0,S*plength),nrow=S)
    e2mat <- matrix(rep(0,S*plength),nrow=S)

    # Collect data from the S different divisions of the patiens
    for (s in 1:S) {
      load(paste(oppdel_navn,s,".RData",sep=""))
      dfmat[s,] <- dfall
      e1mat[s,] <- e1all
      e2mat[s,] <- e2all
    }
    # Save median of each run (median of 100 divisions of patients)
    # One for each p
    med_e1 <- apply(e1mat,2,median) # median of columns
    med_e2 <- apply(e2mat,2,median)
    med_df <- apply(dfmat,2,median)

    # save for each run
    save(med_e1,med_e2,med_df,
         file=paste(run_navn,run,".RData",sep=""))
  }
}


```

```
#************************************
# DEFINE VARIABLES IN ALL SIMULATIONS
#************************************
pvektor <- c(500,1000,1500,2000,2500,3000,3500,
             4000,4500,5000,6000,7000,8000,9000,p)
plength <- length(pvektor)

S        <- 100 # divisions of patients
num_runs <- 100 # simulations

# For the simulations of covariates in sim A and B:
p  <- 10000 # number of covariates
n  <- 200   # number of patients
# Size of the two groups
n1 <- 100; n2 <- n-n1
one1 <- rep(1,n1); one2 <- rep(1,n2) #for the intercept

#**************************************
# A: COVARIATES SIMULATED INDEPENDENTLY
#**************************************
# MAKE THE COVARIATES
X  <- matrix(rep(NA,p*n),nrow=n)
# draw X. Covariates simulated independently
# from the standard normal distribution
for (i in 1:n) {
  X[i,] <- rnorm(p,0,1)
}

# MAKE RESPONSE
# Define beta
set.seed(132)
betaindex <- sample(1:p,10)
#6823 9526 2895 4750 4959 2888 8565 1068 2380 7491
realbeta <- rep(0,p)
realbeta[betaindex[1:5]]  <- -2
realbeta[betaindex[6:10]] <- 2

ysim <- X %*% realbeta

#-------------------------------------
# define variables
# Change between SNR=0.5 and SNR=2
oppdel_navn <- "norm_sim_snr05"
run_navn    <- "norm_run_snr05"
snr         <- 0.5
#-------------------------------------

hundred_simulations(snr=snr,S=S,oppdel_navn,run_navn)

#*************************************
# B: SIMULATE WITH GENES FROM BONE DATA
#*************************************
# MAKE RESPONSE
# Define beta: Choose the genes that have been selected
# in previous analysis to have beta not equal to 0
# gene index chosen earlier:
important = c(1276,1493,3059,5955,7090,11166,17490,18144,19966)
# Remark: Must subtract 1 to get the right gene since the indexes
# is with beta0 at index 1
realbeta <- rep(0,p)
realbeta[important[1:5]-1] <- -2
realbeta[important[6:9]-1] <- 2
```

```

X     <- gener # from Bone data , see Chapter 5
ysim <- X %*% realbeta

#----------------------------------------
# define variables
# Change between SNR=0.5 and SNR=2
oppdel_navn <- "sim_snr05"
run_navn     <- "run_snr05"
snr          <   0.5
#----------------------------------------

hundred_simulations(snr=snr,S=S,oppdel_navn)


#**********************************
# C: CORRELATED COVARIATES
#**********************************
library(MASS)
# MAKE THE COVARIATES
# Covariates close to each other have higher correlation
# 1 on the diagonal
blocksize <- 100
blocks     <- p/blocksize
rho         <- 0.9
Sigma_cor <- matrix(rep(NA,blocksize^2),nrow=blocksize)
for (i in 1:blocksize) { for (j in 1:blocksize)  {
    Sigma_cor[i,j] <- rho^abs(i-j)
}}

Mu <- rep(0,blocksize)

# draw X:  multivariate standard normal distribution
# equal correlation 100 covariates
X <- matrix(rep(NA,n*p),nrow=n)
for (block in 1:blocks) {
  X[,((blocksize*(block-1))+1):(blocksize*(block))] =
    mvrnorm(n, Mu, Sigma_cor) # one samle in each row
}
dim(X)   #200 10000

# MAKE RESPONSE
# Define beta
set.seed(5) #choose this seed s.t no block has more than one beta!=0
betaindex <- sample(1:p,10)
#2003 6852 9167 2844 1047 7008 5277 8074 9558 1104
realbeta   <- rep(0,p)
realbeta[betaindex[1:5]]   <- -2
realbeta[betaindex[6:10]] <- 2

ysim <- X %*% realbeta

#----------------------------------------
# define variables
# Change between SNR=0.5 and SNR=2
oppdel_navn <- "cor_sim_snr05"
run_navn     <- "cor_run_snr05"
snr          <- 0.5
#----------------------------------------

hundred_simulations(snr=snr,S=S,oppdel_navn)
```