Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Simulation Studies of Sparse Canonical Correlation Analysis in Differing Dimensions and Variances, with an Application to Genomic Data

Anton van Wieren (565075)

Supervisor:	P.C. Schoonees
Second assessor:	F. Frasincar
Date final version:	2nd July 2023

Abstract

Canonical correlation analysis (CCA) is a useful tool in analysing two sets of variables measured on the same sample. This statistical technique maximizes the correlation between linear combinations of the two sets of variables. However, in high-dimensional data and data with high variance, CCA faces limitations including inaccuracy, lack of interpretability, and even inapplicability in certain cases. Witten et al. (2009) introduce a sparse variant of this method, which selects only the most important variables, solving most issues. In this paper, we assess the robustness of sparse CCA when dimensionality and variance changes, using accuracy and sparsity recognition measurements. This can help researchers to make informed choices about applying sparse CCA and developing new methods. Simulation studies comparing sparse CCA with regular CCA reveal that sparse CCA performs better in most scenarios, including high-dimensional and noisy data. This paper demonstrates strengths and weaknesses of sparse CCA as opposed to regular CCA. Furthermore, we apply sparse CCA to genomic data from breast cancer patients to identify genes that are correlated with DNA copy number changes, providing insights into potential underlying causes.

1 Introduction

Canonical correlation analysis (CCA) is a statistical technique that has been around for a long time, first introduced by Hotelling (1936). Suppose $\mathbf{X}$ and $\mathbf{Z}$ are two data matrices with measurements of p and q variables on the same sample of n observations, respectively. CCA can identify linear relationships between the two sets of variables by choosing optimal values for canonical vectors $\mathbf{u}$ and $\mathbf{v}$, so that the correlation between the linear combinations $\mathbf{Xu}$ and $\mathbf{Zv}$ is maximized; $\max\limits_{\mathbf{u},\mathbf{v}}\{\text{cor}(\mathbf{Xu},\mathbf{Zv})\}$. However, CCA has some shortcomings in interpreting the results, especially when working with high-dimensional data. This is because CCA mainly focuses on maximizing the correlation between the linear combinations $\mathbf{Xu}$ and $\mathbf{Xv}$, rather than identifying which variables explain the correlation. In addition, CCA is not applicable to all data sets since it only gives unique solutions if $n > p$ and $n > q$.

Witten et al. (2009) introduce a sparse variant of CCA; sparse CCA. This technique does not only maximize the correlation between $\mathbf{Xu}$ and $\mathbf{Xv}$, but also imposes sparsity constraints on the canonical vectors $\mathbf{u}$ and $\mathbf{v}$. Consequently, a part of their components become zero and only a subset consisting of the most important variables is selected. This helps in the interpretability of the resulting canonical vectors, which is especially useful in cases with a high number of variables, since sparse CCA serves as a dimensionality reduction technique. Moreover, in contrast to regular CCA, sparse CCA can be applied to all types of data sets. In this paper, similar algorithms as described in Witten et al. (2009), solving their penalized matrix decomposition (PMD), are used to obtain the sparse canonical vectors.

In this paper, we compare the performance of regular CCA with the sparse CCA method of Witten et al. (2009) in different data settings, using a variety of accuracy and sparsity recognition metrics. Specifically, we investigate how the two methods react to differing dimensionality by altering the number of variables and observations in the data, and how they behave when differing the variance and including outliers in the data. Wilms and Croux (2015b) find that the sparse CCA method of Witten et al. (2009) outperforms the regular CCA in high-dimensional data and noisy data. They use simulated covariance matrices, without simulating the true underlying canonical vectors. In this paper, we compare the accuracy of CCA and sparse CCA in different dimensions and for different variances by simulating the data matrices according to the true (simulated) canonical vectors, enabling direct assessment of variable selection and a more explicit evaluation of the estimation accuracy.

Understanding how sparse CCA behaves in different dimensions and different variances is important for understanding the robustness, generalizability and scalability of this method, and for assessing the strengths and weaknesses of sparse CCA. This is relevant to know when one considers applying sparse CCA. Additionally, recognizing the limitations of sparse CCA can help in developing new methods and techniques specifically to handle data with explicit dimensions or different variances where sparse CCA does not perform well.

To evaluate the properties of sparse CCA and regular CCA, we conduct simulation studies. We increase the dimensionality by expanding the number of observations and variables in the simulated data sets. Additionally, we examine the performance of CCA and sparse CCA by changing the variance in the simulated data sets. Also, we augment the variance of 10% of the simulated data to see how CCA and sparse CCA react to outliers.

In genomic research, there are often multiple sets of different measurements on the same set of samples. These assessments can include, for example, gene expression measurements, DNA copy number changes, single nucleotide polymorphism (SNP) data, and other phenotype measurements. These data sets often have a very large number of variables. Therefore, sparse CCA is generally an adequate method to find correlations between different genomic data sets measured on the same sample. In this paper, a data set of breast cancer patients is considered. Various cancer types can be identified by regions of chromosomal gains or losses. Therefore, we use sparse CCA to identify a set of genes that are correlated with DNA copy number changes (comparative genomic hybridization (CGH) spots). Breast cancer is one of the most prevalent types of cancer amongst woman. Identifying those genes can help conducting cancer research in this field, which is crucial. This application is performed using a data set of gene expression measurements and a data set of CGH spots of 89 breast cancer patients (Chin et al. (2006)).

The main findings of this paper are that both CCA and sparse CCA are affected by changing the number of variables and observations in the data. Increasing the number of variables leads to less accurate performance of CCA, while sparse CCA is more robust to this increase. Altogether, sparse CCA performs more accurately than CCA, except when there are a small number of variables and a high number of observations. In addition, CCA is sensitive to outliers, whereas sparse CCA is more robust to outliers. Finally, when we apply the sparse CCA method of Witten et al. (2009) to a genomic data set about breast cancer, we find that sparse CCA is a good method for identifying a set of genes that are correlated with DNA copy number changes. Sparse CCA also performs well in a holdout out-of-sample validation method using this genomic data set.

This paper replicates parts of Witten et al. (2009). They consider two variants of their PMD that can be applied to solve sparse CCA. We replicate the simulation study comparing those two methods with simulated CGH data and extend it by utilizing some accuracy and sparsity recognition measurements in different simulation settings. We do this to obtain more specific and measurable differences between the methods. Furthermore, we extend the simulation studies of Witten et al. (2009) by altering some characteristics, including the number of variables, the number of observations, and the variance. This contributes to obtain knowledge about the behaviour of sparse CCA in different data settings. Finally, we replicate the application of sparse CCA to a data set about breast cancer patients.

The aim of this paper is to assess the strengths and weaknesses of sparse CCA by identifying its relative advantages opposed to regular CCA, and understanding its behaviour in data with differing number of observations, variables, and different variances. Also, this paper aims to explore the application of sparse CCA in identifying correlations between gene measurements and DNA copy number changes in a breast cancer data set.

In the next section, Section 2, we discuss relevant literature that can be linked to our research. Then in Section 3, we elaborate on the data that we use in our paper. In Section 4, we explain the methods and algorithms that are used in our research and we introduce some metrics for evaluating (sparse) CCA. After that, in Section 5, we discuss the obtained results. Finally, in Section 6 we draw conclusions and suggest further research.

2 Literature

In this section, we discuss relevant literature to this paper. First, we examine some related matrix decomposition methods. Then, we discuss other literature that uses sparse canonical correlation analyses and are relevant to this research.

2.1 Matrix decomposition methods

The penalized matrix decomposition (PMD) of Witten et al. (2009) is a generalization of singular value decomposition (SVD, further explained in Section 4.1.1), introduced in the modern formulation by Lanczos (1950). This PMD of Witten et al. (2009) is not the first generalized improvement over SVD. Lee and Seung (1999) and Lee and Seung (2001) introduced a non-negative matrix factorization (NNMF) method, adding non-negative constraints to the SVD to improve interpretability. Hoyer (2002) developed the non-negative sparse coding (NNSC) method adding sparsity constraints to the NNMF method. The results of NNMF and NNSC ensure non-negativity in the obtained factors, so the results can differ substantially from the factors obtained after performing PMD. Lazzeroni and Owen (2002) developed the cluster-based Plaid model, which can give interpretative results. The Plaid model can cluster genes and conditions simultaneously, based on their expression profiles. As opposed to previous matrix decomposition methods, it is in a non-convex form so it cannot be precisely optimized.

2.2 Sparse canonical correlation analysis

A useful application of PMD is sparse canonical correlation analysis (sparse CCA), a variant of CCA (first introduced by Hotelling (1936)). Sparse CCA is often used in genomic research, since it becomes increasingly common for biological researchers to use more than one evaluation on a single set of observations. Witten et al. (2009) use sparse CCA to obtain correlations between data on DNA copy number changes and gene expression measurements for a breast cancer data set, in order to identify a set of genes that have correlated expressions with a set of chromosomal gains or losses. Other papers which also identify sparse correlations between genomic data sets are Parkhomenko et al. (2007), Lin et al. (2014), Du et al. (2020). For Parkhomenko et al. (2007) specifically, the goal is to identify the most relevant sets of genes that are associated with different phenotypes for a leukaemia and lung cancer data set. Parkhomenko et al. (2007) also perform a genomic study applying a sparse CCA method, closely related to the PMD based sparse CCA of Witten et al. (2009). Sparse CCA can also be applied to non-genomic data sets. For example, Iaci et al. (2010) apply the method to an environmental data set and Hastie et al. (2015) apply sparse CCA to Netflix data for possible marketing purposes.

Furthermore, Parkhomenko et al. (2009) propose a variant of sparse CCA, adaptive sparse CCA, that can automatically adapt the sparsity level in the correlation coefficients. Waaijenborg et al. (2008) propose a penalized CCA method with elastic net type penalty functions and investigate DNA copy number data and gene measurement data for brain tumours. The algorithms they use are quite similar to the algorithms of Witten et al. (2009).

The sparse CCA procedures of Witten et al. (2009), Waaijenborg et al. (2008) and Parkhomenko et al. (2009) are three of the most popular sparse CCA methods.

Wilms and Croux (2015b) consider the limitations of CCA from a predictive point of view. They compare these three popular methods with regular CCA, using several performance measurements (as in Rothman et al. (2010)) in different simulation settings using low- and high-dimensional data. They find that the sparse CCA method of Witten et al. (2009) outperforms regular CCA, especially in high-dimensional data. Wilms and Croux (2015a) compare CCA, robust CCA (Karnel (1991)), sparse CCA and the newly introduced method robust sparse CCA, using similar median-based metrics as Wilms and Croux (2015b). They consider different alterations to simulated data in order to make it more varianced and compare the methods. They find that all methods are affected by the data alterations and show the advantages of sparse CCA opposed to regular CCA. Note that both Wilms and Croux (2015b) and Wilms and Croux (2015a) use simulated covariance matrices, whereas we use simulated underlying canonical vectors. In addition, they both only investigate the cases of high- and low-dimensional data, while we increase the number of observations and variables step wise to gain a better insight in the behaviour of regular and sparse CCA.

In genomic research, there are often multiple assays to represent only one set of observations. Sometimes more than two data sets are available for analysis. First of all, outcome measurements of all samples may be available, for example survival time in case of cancer samples. Witten and Tibshirani (2009) developed a method called supervised sparse CCA to include this kind of data in the analysis. Secondly, next to DNA copy number changes data and gene measurement data, often single nucleotide polymorphism (SNP) data is also available. Witten and Tibshirani (2009) developed the PMD based sparse multiple CCA to handle more than two sets of variables. This can be very useful if one wishes to analyse more than two data sets measured on the same set of observations.

There are also other algorithms to consider when solving sparse CCA. Chu et al. (2013) developed an algorithm to solve sparse CCA that is competitive and sometimes even better than the algorithms used in Witten et al. (2009).

3 Data

Certain cancer types can be identified by regions of chromosomal gains or losses. Therefore we investigate the correlation between DNA copy numbers changes and gene expression measurements of breast cancer patients. In this section, we describe the data about breast cancer patients where the sparse CCA method of Witten et al. (2009) is applied to. The results of this application can be found in Section 5.3.

The data used in this paper is the same as in Chin et al. (2006) and can be found in the 'PMA' package in R (Witten and Gross (2011)). The data set consists of 89 observations of gene expression measurements and DNA copy number changes measurements of breast cancer patients. The data set contains nine different elements. The first element is a matrix of 2149 comparative genomic hybridization (CGH) spots for all 89 samples. A CGH spot measures DNA copy number changes along chromosomes. The second element is a matrix containing 19 672 gene expression measurements in 89 different samples. The third and fourth element describe the CGH spots by the chromosomal location and the nucleotide position of each CGH spot, respectively. Note that the chromosomal location of each CGH spot corresponds with one of 23 different chromosomes pairs, since humans have 22 autosomal chromosome pairs, and one sex chromosome pair. Therefore, these locations range between the values 1 and 23. The remaining elements describe the gene expression data by the accession numbers, names, descriptions, chromosomal locations and nucleotide locations for all gene expressions. Note that in the gene expression data, the range of the chromosomal locations go from 1 to 24. This is because the gene expressions that correspond with the sex chromosome pair are split up in location 23 and 24 for the X and the Y chromosome, respectively. Finally, note that there are some gene expressions that do not have a specific chromosomal location.

We alter the data before analysing it by centering the data matrices around zero. That is, subtracting the mean of the data matrices from each value. To analyse the relationship between DNA copy number and gene expression data, sparse CCA is executed for each individual chromosome. For each chromosome, this process uses the DNA copy number information specific to that chromosome, and all available gene expression data (so for all chromosomes).

4 Methodology

4.1 Penalized matrix decomposition

First of all, we describe the general form of penalized matrix decomposition (PMD) as proposed in Witten et al. (2009).

4.1.1 Singular value decomposition

The PMD is a generalization of a singular value decomposition (SVD) introduced in the modern formulation by Lanczos (1950). SVD can decompose a matrix $\mathbf{X}$ into matrices $\mathbf{U}, \mathbf{D}$ and $\mathbf{V}$:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}', \qquad \text{where} \qquad \mathbf{U}'\mathbf{U} = \mathbf{I}_n, \qquad \mathbf{V}'\mathbf{V} = \mathbf{I}_p, \qquad d_1 \geq d_2 \geq \cdots \geq d_K > 0. \tag{1}$$

Here, $\mathbf{X}$ denotes an $n \times p$ matrix with $\text{rank}(\mathbf{X}) = K \leq \min\{n, p\}$. $\mathbf{U}$ denotes an $n \times K$ matrix where $\mathbf{u}_k$ indicates the k_{th} column of $\mathbf{U}$. $\mathbf{V}$ denotes a $p \times K$ matrix where $\mathbf{v}_k$ indicates the k_{th} column of $\mathbf{V}$ ($k \in \{1, \dots, K\}$). Furthermore, $\mathbf{D}$ denotes a $K \times K$ diagonal matrix with diagonal elements $d_1, \dots, d_K$, and $\mathbf{I}_q$ stands for an identity matrix of size q. Lastly, we assume without loss of generality that the mean of $\mathbf{X}$ is 0. By centering $\mathbf{X}$ around 0, we remove bias and ensure that SVD analyses the underlying variability and correlations and not the trends. Note that throughout this paper, matrices and vectors are indicated in bold for clarity.

Eckart and Young (1936) derived that for any $r \leq K$, the following equation holds:

$$\sum_{k=1}^{r} d_k \mathbf{u}_k \mathbf{v}_k' = \underset{\hat{\mathbf{X}} \in M_r}{\arg\min} \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2. \tag{2}$$

Here, $\hat{\mathbf{X}}$ denotes an rank-r approximation of $\mathbf{X}$, M_r denotes a set of all $n \times p$ matrices with rank r and $\|\cdots\|_F^2$ stands for the squared Frobenius norm which sums all squared elements of a matrix. This means that the first r factors of SVD provide the best approximation of a matrix with rank r, indicated with regard to the Frobenius norm, which is a useful property.

4.1.2 PMD for a single factor

Witten et al. (2009) propose a generalization of the SVD by adding penalties on the magnitudes of $\mathbf{U}$ and $\mathbf{V}$. In this subsection, we look at an approximation of the PMD with rank 1 (so $K = 1$). From Equation (2), we know that the best approximation of a matrix with rank r equals $\sum_{k=1}^{r} d_k \mathbf{u}_k \mathbf{v}_k'$. Therefore, for a single factor ($r = 1$) this best approximation equals $d\mathbf{u}\mathbf{v}'$ (for the case of a single factor, we denote d_1 as d, $\mathbf{u}_1$ as $\mathbf{u}$ and $\mathbf{v}_1$ as $\mathbf{v}$). That is why Witten et al. (2009) propose the following optimization problem:

$$\min_{d,\mathbf{u},\mathbf{v}} \frac{1}{2}\|\mathbf{X} - d\mathbf{u}\mathbf{v}'\|_F^2 \quad s.t. \quad \|\mathbf{u}\|_2^2 = 1, \quad \|\mathbf{v}\|_2^2 = 1, \quad P_1(\mathbf{u}) \leq c_1, \quad P_2(\mathbf{v}) \leq c_2, \quad d \geq 0. \tag{3}$$

P_1 and P_2 denote convex penalty functions and c_1 and c_2 are the scalar upperbounds for those convex penalty function. Note that $\|\mathbf{w}\|_q$ of a vector $\mathbf{w}$ denotes its L_q-norm $\sum_i |w_i^q|^{\frac{1}{q}}$. In Appendix A, we explain how we choose the values of c_1 and c_2. The convex penalty functions P_1 and P_2 can take a lot of different forms, but we only consider two different convex penalty functions:

- LASSO: $P_1(\mathbf{u}) = \sum_{i=1}^{n} |u_i|$
- Fused LASSO: $P_1(\mathbf{u}) = \sum_{i=1}^{n} |u_i| + \lambda \sum_{i=2}^{n} |u_i - u_{i-1}|, \qquad \lambda > 0$

This fused LASSO penalty form is motivated in Tibshirani et al. (2005).

We can rewrite Problem (3) by using the following equality:

$$\frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{D}\mathbf{V}'\|_F^2 = \frac{1}{2}\|\mathbf{X}\|_F^2 - \sum_{k=1}^{K} \mathbf{u}_k'\mathbf{X}\mathbf{v}_k d_k + \frac{1}{2}\sum_{k=1}^{K} d_k^2. \tag{4}$$

The proof of this equality is given in Appendix B. In the situation of $K = 1$, we can rewrite Problem (3) into the following problem:

$$\max_{\mathbf{u},\mathbf{v}} \{\mathbf{u}'\mathbf{X}\mathbf{v}\} \quad s.t. \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1, \quad P_1(\mathbf{u}) \leq c_1, \quad P_2(\mathbf{v}) \leq c_2. \tag{5}$$

Problem (5) solves for the same $\mathbf{u}$ and $\mathbf{v}$ as Problem (3) and the value of d solving Problem (3) is $\mathbf{u}'\mathbf{X}\mathbf{v}$. $\mathbf{u}'\mathbf{X}\mathbf{v}$ is bi-linear in $\mathbf{u}$ and $\mathbf{v}$, meaning that it is linear in $\mathbf{u}$ if $\mathbf{v}$ is fixed, and it is linear in $\mathbf{v}$ if $\mathbf{u}$ is fixed. Note that we replaced the equality constraints with inequality constraints, since $\|\mathbf{u}\|_2^2 \leq 1$ also satisfies $\|\mathbf{u}\|_2^2 = 1$ given that c_1 is chosen so that $\mathbf{u}$ can have a Euclidean norm greater or equal than 1 (Boyd and Vandenberghe (2004)).

Witten et al. (2009) propose an algorithm to solve a single-factor PMD model:

Algorithm 1[1]

```
1. Initialize $\mathbf{v}$ to have a Euclidean norm of 1.
2. Iterate until convergence:[2]
    (a) $\mathbf{u} = \arg\max_{\mathbf{u}}\{\mathbf{u}'\mathbf{X}\mathbf{v}\}$ subject to $P_1(\mathbf{u}) \leq c_1$ and $\|\mathbf{u}\|_2^2 \leq 1$.
```

[1]All algorithms in this paper are cited from Witten et al. (2009).

[2]That is, until 20 iterations are performed, or when the difference between the new $\mathbf{v}$ and the old $\mathbf{v}$ is smaller than $1e^{-7}$.

   (b) $\mathbf{v} = \arg\max_{\mathbf{v}}\{\mathbf{u}'\mathbf{X}\mathbf{v}\}$ subject to $P_2(\mathbf{v}) \leq c_2$ and $\|\mathbf{v}\|_2^2 \leq 1$.

3. $d = \mathbf{u}'\mathbf{X}\mathbf{v}$.

Witten et al. (2009) initialize $\mathbf{v}$ in the following way:

- If $p > n$, $\mathbf{v}_{init} = \mathbf{X}'\mathbf{v}_{SVD(\mathbf{X}\mathbf{X}')}$.
- If $p \leq n$, $\mathbf{v}_{init} = \mathbf{v}_{SVD(\mathbf{X}'\mathbf{X})}$.
- Lastly, standardize $\mathbf{v}_{init}$ so that it has a Euclidean norm of 1.

Where $\mathbf{v}_{SVD(\mathbf{X}\mathbf{X}')}$ denotes the vector $\mathbf{v}$ obtained after performing SVD on $\mathbf{X}\mathbf{X}'$ (similar for $\mathbf{v}_{SVD(\mathbf{X}'\mathbf{X})}$). Note that this algorithm may not always reach a global optimum in general, but empirical studies conducted by Witten et al. (2009) show that the algorithm does converge to interpretable results if the penalty sizes are chosen correctly.

4.1.3 PMD for multiple factors

When we want to obtain a rank-K approximation of a matrix $\mathbf{X}$, we need to attain multiple factors of the PMD, therefore we solve the optimization problem for the single factor PMD (Problem (5)) repeatedly. Witten et al. (2009) propose an algorithm to solve a K-factor PMD model:

Algorithm 2

```
1. Let $\mathbf{X}^1 = \mathbf{X}$.
2. For $k \in \{1, ..., K\}$:
    (a) Find $\mathbf{u}_k$, $\mathbf{v}_k$ and $d_k$ by applying Algorithm 1 to $\mathbf{X}^k$.
    (b) $\mathbf{X}^{k+1} = \mathbf{X}^k - d_k\mathbf{u}_k\mathbf{v}_k'$.
```

Here, for each $k \in \{1, ..., K\}$, the residuals obtained by subtracting the $1, ..., k-1$ factors found from the data matrix is used as the $\mathbf{X}$ matrix. Note that the solutions are not orthogonal due to the P_1 and P_2 penalty functions ($\mathbf{u}_k$ and $\mathbf{v}_k$ are not in the column and row spaces of $\mathbf{X}^k$ for each $k \in \{1, ..., K\}$), whereas the solutions of SVD are orthogonal.

4.1.4 Special forms of PMD

Witten et al. (2009) propose two specific forms of the penalty functions in the PMD. These specific forms are denoted by PMD(A,B), where A denotes the type of the P_1 penalty function on $\mathbf{u}$, and B denotes the type of the P_2 penalty function on $\mathbf{v}$. In this paper, we consider two forms; PMD(L_1, L_1) and PMD(L_1, FL). L_1 stands for a LASSO type L_1 penalty function, and FL represents a fused LASSO type penalty function. In the simulation studies in Section 5.1, we apply PMD(L_1, L_1). In Section 5.2, we compare the performance of PMD(L_1, L_1) with PMD(L_1, FL).

PMD(L_1, L_1) set L_1 penalty norms on both $\mathbf{u}$ and $\mathbf{v}$. This leads to the following optimization problem for a single factor:

$$\max_{\mathbf{u},\mathbf{v}}\{\mathbf{u}'\mathbf{X}\mathbf{v}\} \quad s.t. \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1, \quad \|\mathbf{u}\|_1 \leq c_1, \quad \|\mathbf{v}\|_1 \leq c_2. \tag{6}$$

We regulate the values of c_1 and c_2 by restricting them to be in the ranges $[1; \sqrt{n}]$ and $[1; \sqrt{p}]$, respectively, as explained in Witten et al. (2009). This special form of PMD leads to sparse factors $\mathbf{u}$ and $\mathbf{v}$ for appropriate choices of c_1 and c_2.

When considering an optimization problem of the form $\max_{\mathbf{u}}\{\mathbf{u'a}\}$ s.t $\|\mathbf{u}\|_2^2 \leq 1$, $\|\mathbf{u}\|_1 \leq c$, $\mathbf{u}$ is solved for $\mathbf{u} = \frac{S(\mathbf{a},\Delta)}{\|S(\mathbf{a},\Delta)\|_2}$, where $\Delta = 0$ if the condition $\|\mathbf{u}\|_1 \leq c$ is satisfied, and otherwise Δ is adjusted such that $\|\mathbf{u}\|_1 = c$ is satisfied. This is proven in Appendix C. Here, the function $S(\cdot)$ indicates the soft thresholding operator. This operator takes the form of $S(a, c) = sign(a)(|a| - c)_+$ where c is a positive scalar and $(x)_+$ equals x if $x > 0$ and 0 if $x \leq 0$. Knowing the solution to this optimization problem, we can adjust our PMD algorithm for a single factor for the case of PMD(L_1, L_1) in the following way:

Algorithm 3

```
1. Initialize $\mathbf{v}$ to have a Euclidean norm of 1.
2. Iterate until convergence:
    (a) $\mathbf{u} = \frac{S(\mathbf{Xv},\Delta_1)}{\|S(\mathbf{Xv},\Delta_1)\|_2}$ where $\Delta_1 = 0$ if this results in $\|\mathbf{u}\|_1 \leq c_1$; otherwise, $\Delta_1$ is chosen to
        be a positive constant such that $\|\mathbf{u}\|_1 = c_1$.
    (b) $\mathbf{v} = \frac{S(\mathbf{X'u},\Delta_2)}{\|S(\mathbf{X'u},\Delta_2)\|_2}$ where $\Delta_2 = 0$ if this results in $\|\mathbf{v}\|_1 \leq c_2$; otherwise, $\Delta_2$ is chosen
        to be a positive constant such that $\|\mathbf{v}\|_1 = c_2$.
3. $d = \mathbf{u'Xv}$.
```

Binary search is employed to determine the values of Δ_1 and Δ_2, for each update of $\mathbf{u}$ and $\mathbf{v}$. We compare the results of PMD(L_1, L_1) with the SVD method described in Section 4.1.1. A simulation study shows that the PMD method is better in identifying underlying factors and can be found in Appendix D.

PMD(L_1, FL) is another variant of the PMD method of Witten et al. (2009). This variant sets an L_1 penalty function on $\mathbf{u}$, and a fused LASSO penalty function on $\mathbf{v}$. This leads to the following optimization problem for a single factor:

$$\max_{\mathbf{u},\mathbf{v}}\{\mathbf{u'Xv}\} \quad s.t. \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1, \quad \|\mathbf{u}\|_1 \leq c_1, \quad \sum_j |v_j| + \lambda \sum_j |v_j - v_{j-1}| \leq c_2. \tag{7}$$

To solve this optimization problem, we first rewrite it in a minimization problem using the Lagrange form on the constraints on $\mathbf{v}$:

$$\min_{\mathbf{u},\mathbf{v}}\{-\mathbf{u'Xv} + \frac{1}{2}\mathbf{v'v} + \lambda_1 \sum_j |v_j| + \lambda_2 \sum_j |v_j - v_{j-1}|\} \quad s.t. \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{u}\|_1 \leq c_1. \tag{8}$$

This Lagrange form gives the same solutions for $\mathbf{u}$ and $\mathbf{v}$ as the bound form in Problem (7). This special form of PMD leads to sparse factors $\mathbf{u}$ and sparse and rather smooth factors $\mathbf{v}$. The values of λ_1 and λ_2 are chosen by cross validation ($\lambda_1, \lambda_2 \geq 0$).

We can solve this PMD(L_1, FL) by adjusting PMD Algorithm 1 for a single factor in the following way:

Algorithm 4

```
1. Initialize $\mathbf{v}$ to have a Euclidean norm of 1.
2. Iterate until convergence:
    (a) $\mathbf{u} = \frac{S(\mathbf{X}\mathbf{v},\Delta_1)}{\|S(\mathbf{X}\mathbf{v},\Delta_1)\|_2}$ where $\Delta_1 = 0$ if this results in $\|\mathbf{u}\|_1 \leq c_1$; otherwise, $\Delta_1$ is chosen to
        be a positive constant such that $\|\mathbf{u}\|_1 = c_1$.
    (b) $\mathbf{v} = \arg\min_{\mathbf{v}} \frac{1}{2}\|\mathbf{X}'\mathbf{u} - \mathbf{v}\|_2^2 + \lambda_1 \sum_j |v_j| + \lambda_2 \sum_j |v_j - v_{j-1}|$.
3. $d = \mathbf{u}'\mathbf{X}\mathbf{v}$.
```

In Algorithm 4, the $\mathbf{v}$ in step 2(b) can be updated using software implementations of fused LASSO regressions as used and described in Tibshirani and Wang (2007), Friedman et al. (2007), and Hoefling (2010).

Lastly, we can solve PMD(L_1, L_1) and PMD(L_1, FL) for multiple factors in the same way as Algorithm 2, but instead of repeatedly applying Algorithm 1, we repeatedly apply Algorithm 3 and Algorithm 4, respectively for PMD(L_1, L_1) and PMD(L_1, FL).

4.2 Canonical correlation analysis

The PMD of Witten et al. (2009) can by applied in multiple fields. One application of the PMD method is in canonical correlation analysis (CCA), introduced by Hotelling (1936). CCA is a statistical technique that can identify relationships between two sets of variables, measured on the same sample. Suppose that there are $p + q$ variable measurements on n observations and these variable measurements can be naturally split up into two sets of p and q variables, respectively. Let $\mathbf{X}$ denote the $n \times p$ matrix of the first part of variables and $\mathbf{Z}$ the $n \times q$ matrix of the second part of variables. CCA finds linear combinations between the two sets of variables by choosing optimal canonical vectors $\mathbf{u}$ and $\mathbf{v}$ that maximize the correlation between $\mathbf{Xu}$ and $\mathbf{Zv}$. This is obtained by solving the following problem:

$$\max_{\mathbf{u},\mathbf{v}}\{\mathbf{u}'\mathbf{X}'\mathbf{Z}\mathbf{v}\} \quad s.t. \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1. \tag{9}$$

Note that the constraints in (9) are different than those from Hotelling (1936). We replaced the constraints $\mathbf{u}'\mathbf{X}'\mathbf{X}\mathbf{u} \leq 1$ and $\mathbf{v}'\mathbf{X}'\mathbf{X}\mathbf{v} \leq 1$ by $\|\mathbf{u}\|_2^2 \leq 1$ and $\|\mathbf{v}\|_2^2 \leq 1$, respectively. This is because it has been shown that changing the covariance matrix into an identity matrix performs well in high-dimensional data sets (Dudoit et al. (2002) and Tibshirani et al. (2003)). Regular CCA can be solved analytically, and is further explained in Johnson and Wichern (2019).

4.2.1 Sparse canonical correlation analysis

It can be useful to investigate which variables drive the correlation between the two data sets, especially in high-dimensional settings. By incorporating sparsity constraints to the canonical vectors $\mathbf{u}$ and $\mathbf{v}$, a part of the components in $\mathbf{u}$ and $\mathbf{v}$ become zero, and only the most important variables remain, which can help interpreting the results. When including sparsity constraints in CCA, the method is called sparse CCA and has the following formulation:

$$\max_{\mathbf{u},\mathbf{v}}\{\mathbf{u}'\mathbf{X}'\mathbf{Z}\mathbf{v}\} \quad s.t. \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1 \quad P_1(\mathbf{u}) \leq c_1, \quad P_2(\mathbf{v}) \leq c_2. \tag{10}$$

Again, P_1 and P_2 are convex penalty functions and c_1 and c_2 are the penalty sizes for the penalty function. The convex penalty functions ensure sparsity in the canonical vectors $\mathbf{u}$ and $\mathbf{v}$. This optimization problem is very similar to the PMD Problem (5). The only difference is that $\mathbf{X}$ is replaced by $\mathbf{X}'\mathbf{Z}$. Sparse CCA can be solved using the same algorithms that are used to solve PMD, explained in Section 4.1 .

Different penalty functions can be included in solving sparse CCA. In Section 4.1.4, we consider two specific forms of PMD; PMD(L_1,L_1) and PMD(L_1,FL). These two specific forms can also be applied to sparse CCA. Sparse CCA(L_1,L_1) leads to sparse canonical vectors $\mathbf{u}$ and $\mathbf{v}$ when c_1 and c_2 are small enough. Sparse CCA(L_1,FL) also leads to sparse canonical vectors $\mathbf{u}$ and $\mathbf{v}$ when c_1 and c_2 are sufficiently small, but also to a somewhat smooth vector $\mathbf{v}$. Sparse CCA(L_1,FL) is often used when the variables in $\mathbf{Z}$ are ordered.

Sparse CCA is particularly useful when dealing with high-dimensional data sets, especially when the number of variables is larger than the number of observations ($p, q \geq n$). In this case, regular CCA result in non-unique canonical vectors and is inapplicable, whereas sparse CCA can still be applied. Sparse CCA is also very useful when handling data with high multi-collinearity, since it only selects the most important features and resolves this issue.

4.3 Performance measurements

In this subsection, we introduce and explain some metrics to evaluate the fit of CCA and sparse CCA.

4.3.1 Accuracy metrics

We measure the accuracy of the canonical vectors obtained by CCA and sparse CCA by the angle between subspaces spanned by the canonical vectors, as was done in Wilms and Croux (2015b) and Wilms and Croux (2015a). We measure the angle between the subspace spanned by the estimated canonical vectors and the true canonical vectors. The smaller the angle, the more alike the true and the estimated canonical vectors are, so the more accurate the estimated canonical vectors are. The angle for canonical vector $\mathbf{U}$ is measured in the following way:

- First, calculate the QR-decomposition of the estimated $\hat{\mathbf{U}}$ and the real $\mathbf{U}$. $\hat{\mathbf{U}} = \mathbf{Q}_{\hat{\mathbf{U}}}\mathbf{R}_{\hat{\mathbf{U}}}$ and $\mathbf{U} = \mathbf{Q}_{\mathbf{U}}\mathbf{R}_{\mathbf{U}}$.
- Secondly, calculate the SVD in the following way: $\mathbf{Q}'_{\hat{\mathbf{U}}}\mathbf{Q}_{\mathbf{U}} = \mathbf{U}_{svd}\mathbf{D}_{svd}\mathbf{V}'_{svd}$.
- The $\mathbf{D}_{svd}$ matrix is a diagonal matrix with diagonal elements $d_1 \geq d_2 \geq ... \geq d_K > 0$. The minimum angle between the subspaces spanned by $\hat{\mathbf{U}}$ and $\mathbf{U}$ is given by $arccos(d_1)$.

The angle for canonical vector $\mathbf{V}$ is measured in the same way.

This measurement is determined over M simulations, and the average angle is given in the following way:

$$\theta(\hat{\mathbf{U}}, \mathbf{U}) = \frac{1}{M}\sum_{m=1}^{M} \theta^m(\hat{\mathbf{U}}^m, \mathbf{U}), \qquad \theta(\hat{\mathbf{V}}, \mathbf{V}) = \frac{1}{M}\sum_{m=1}^{M} \theta^m(\hat{\mathbf{V}}^m, \mathbf{V}). \tag{11}$$

Note that the range of $\theta(\hat{\mathbf{U}}, \mathbf{U})$ is $[arccos(1); arccos(0)] \approx [0; 1.5708]$.

Furthermore, for the sparse CCA method, we measure the accuracy using the Frobenius norm. Here, we multiply the matrices $\mathbf{X}$ and $\mathbf{Z}$ to obtain the $p \times q$ matrix $\mathbf{X}'\mathbf{Z}$. From this $\mathbf{X}'\mathbf{Z}$ matrix, we randomly remove 10% and apply the PMD(L_1, L_1) method to obtain $\mathbf{U}_{PMD}$, $\mathbf{V}_{PMD}$ and $d_{1,PMD}, ..., d_{K,PMD}$. Then, we estimate the matrix $\mathbf{X}'\mathbf{Z}$ by $\hat{\mathbf{X}'\mathbf{Z}} = \sum_{k=1}^{K} d_{k,PMD}\mathbf{u}_{k,PMD}\mathbf{v}'_{k,PMD}$. The Frobenius norm accuracy is given by:

$$FN = \frac{1}{M}\sum_{m=1}^{M} \frac{\sum_{i=1}^{p}\sum_{j=1}^{q} \mathbb{I}_{\{(i,j)\in R_m\}}((\mathbf{X}'\mathbf{Z})_{(i,j)} - (\hat{\mathbf{X}'\mathbf{Z}})_{(i,j)})^2}{|R_m|}. \tag{12}$$

This accuracy measurement indicates the average squared difference between the true (removed) values and the estimated values. Here, R_m denotes the set of indices that are removed from $\mathbf{X}'\mathbf{Z}$ in simulation m and $|R_m|$ denotes the cardinality of this set. Furthermore $\mathbb{I}_{\{(i,j)\in R_m\}}$ denotes the indicator function that equals 1 if element (i, j) (row i and column j) is in R_m and 0 otherwise.

4.3.2 Sparsity recognition measurements

We evaluate the recognition of the true underlying sparseness in the canonical vectors. Rothman et al. (2010) consider the sparsity recognition measurements true positive rate (TPR) and true negative rate (TNR). The true positive rate measures the amount of true nonzero variables in the estimated canonical vectors. The true negative rate measures the hit rate of excluding unimportant variables in the canonical vectors. We only consider the sparseness accuracy of the estimated canonical vectors of sparse CCA, since the canonical vectors of regular CCA do not account for sparsity. The TPR and TPN are defined for $\mathbf{U}$ as follows:

$$TPR(\hat{\mathbf{U}}_{sCCA}, \mathbf{U}) = \frac{\sum_{i=1}^{p}\sum_{j=1}^{K} \mathbb{I}_{\{(\hat{\mathbf{U}}_{sCCA})_{(i,j)}\neq 0\}}\mathbb{I}_{\{\mathbf{U}_{(i,j)}\neq 0\}}}{\sum_{i=1}^{p}\sum_{j=1}^{K} \mathbb{I}_{\{\mathbf{U}_{(i,j)}\neq 0\}}}. \tag{13}$$

$$TNR(\hat{\mathbf{U}}_{sCCA}, \mathbf{U}) = \frac{\sum_{i=1}^{p}\sum_{j=1}^{K} \mathbb{I}_{\{(\hat{\mathbf{U}}_{sCCA})_{(i,j)}= 0\}}\mathbb{I}_{\{\mathbf{U}_{(i,j)}= 0\}}}{\sum_{i=1}^{p}\sum_{j=1}^{K} \mathbb{I}_{\{\mathbf{U}_{(i,j)}= 0\}}}. \tag{14}$$

The TPR and TPN are defined in the same way for $\mathbf{V}$.

To assess the sparsity in the obtained (sparse) canonical vectors, some measurements of sparsity are required. Hurley and Rickard (2009) compare a collection of commonly used sparsity measurements using six criteria. They found that in terms of those criteria, the Gini coefficient is the best way to measure sparsity, since it is the only sparsity measurement that satisfies all of those criteria. The following procedure is applied to calculate the Gini coefficient as in Hurley and Rickard (2009). First, we concatenate all values in the estimated matrices $\hat{\mathbf{U}}_{sCCA}$ and $\hat{\mathbf{V}}_{sCCA}$ in $Kp \times 1$ and $Kq \times 1$ vectors $\mathbf{u}$ and $\mathbf{v}$, respectively. Then we take the absolute value of all elements in vectors $\mathbf{u}$ and $\mathbf{v}$ and sort them such that $|u_1| \leq |u_2| \leq ... \leq |u_{Kp}|$ and $|v_1| \leq |v_2| \leq ... \leq |v_{Kq}|$. Next, we calculate the Gini coefficient for vector $\mathbf{u}$ as follows:

$$Gini(\mathbf{u}) = 1 - 2\sum_{i=1}^{Kp} \frac{Kp - i + \frac{1}{2}}{Kp} \frac{|u_i|}{\|\mathbf{u}\|_1}. \tag{15}$$

This is done in a similar way for $\mathbf{v}$, where p is replaced by q. Interestingly, the Gini coefficient has a useful graphical interpretation as further explained in Hurley and Rickard (2009).

5 Results

In this section, we compare different aspects of CCA and sparse CCA by applying simulation studies. Then, we compare two variants of PMD to establish their relative advantages. Finally, the sparse CCA method is applied to a real life breast cancer data set.

5.1 Simulation study

We conduct simulation studies in this section, by applying sparse CCA(L_1, L_1). In all following tables, to determine the penalty sizes, 5-fold cross validation is used. When significant differences are mentioned, a two-sided t-test is performed with a 5% significance level.

5.1.1 Differing dimensionality

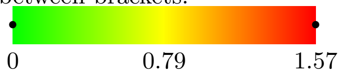
We investigate how regular CCA and sparse CCA perform in different dimensional settings. We simulate the true values of $\mathbf{U}$ and $\mathbf{V}$ from a standard normal distribution and we apply 100 simulations ($M = 100$). Also, we use two factors ($K = 2$). The simulation details are given in Appendix E. Note that the true canonical vectors are not sparse.

The results of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ in different dimensional settings are displayed in Table 1. The values are given a colour, lower values of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ are associated with a greener colour in the table, indicating higher accuracy. Higher values of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ are associated with a more red colour, indicating lower accuracy. The results indicate that when the number of variables (p, q) increases, regular CCA performs significantly less accurate for each increase in p and q. When we increase the number of samples (n), for low values of p and q, the values for $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ first decrease until $n = 101$ or $n = 251$ and then increase when n gets higher. For high values of p and q, the values for $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ are monotonically decreasing for increasing values of n. Note that around the tipping point from decreasing to increasing values, $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ does not change significantly most of the time. This means that for small numbers of variables, increasing the number of observations initially improves accuracy, but after a certain point, accuracy decreases. However, for high numbers of variables, increasing the number of observations improves accuracy. Finally, we notice that when the number of observations (n), and the number of variables (p, q) are close, CCA performs inadequately. The results of $\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$ are similar and given in Appendix F.

The results of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ are shown in Table 2. Sparse CCA also performs most accurate in terms of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ when there are few variables. For relatively small numbers of observations (n), there is a common upgoing trend in $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ when p and q increase, since it never drops significantly. For relatively large numbers of observations, the values of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ first increase for low values of p and q, then decrease again for somewhat higher values of p, q, and they rise again for really high values of p and q. When we increase the number of observations (n), $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ takes a general form for all values of p and q. It starts relatively high for low values of n, then it goes down, stays somewhat stable for intermediate values of n, and goes up again for high values of n. Here, $p, q = 4$ and $p, q = 250$ are exceptions. We obtained similar results for $\theta_{sCCA}(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$. Those are shown in Appendix F.

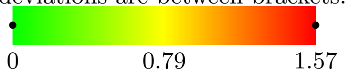
	p,q							
$n\downarrow$	4	7	10	25	50	100	250	1000
5	0.630 (0.371)	-	-	-	-	-	-	-
8	0.447 (0.347)	1.245 (0.168)	-	-	-	-	-	-
11	0.355 (0.263)	1.058 (0.217)	1.356 (0.107)	-	-	-	-	-
26	0.304 (0.225)	0.614 (0.200)	0.849 (0.188)	1.458 (0.048)	-	-	-	-
51	0.244 (0.198)	0.482 (0.170)	0.659 (0.153)	1.272 (0.127)	1.496 (0.030)	-	-	-
101	0.238 (0.215)	0.458 (0.174)	0.559 (0.164)	0.836 (0.107)	1.365 (0.071)	1.518 (0.019)	-	-
251	0.275 (0.235)	0.495 (0.181)	0.558 (0.140)	0.664 (0.094)	0.825 (0.092)	1.258 (0.074)	1.538 (0.012)	-
1001	0.312 (0.245)	0.580 (0.212)	0.633 (0.229)	0.695 (0.135)	0.662 (0.075)	0.727 (0.062)	0.984 (0.041)	1.556 (0.006)

Table 1: $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ values when differing the number of observations (n) and the number of variables (p and q). Lower values of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ correspond to a more green colour (indicating higher accuracy), and higher values of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ correspond to a more red colour (indicating lower accuracy). Standard deviations are between brackets.



	p,q							
$n\downarrow$	4	7	10	25	50	100	250	1000
5	0.221 (0.179)	0.501 (0.236)	0.575 (0.238)	0.644 (0.243)	0.698 (0.294)	0.674 (0.249)	0.638 (0.211)	0.653 (0.227)
8	0.267 (0.210)	0.455 (0.233)	0.507 (0.205)	0.540 (0.192)	0.541 (0.177)	0.550 (0.160)	0.563 (0.158)	0.607 (0.166)
11	0.256 (0.216)	0.424 (0.201)	0.441 (0.161)	0.460 (0.157)	0.469 (0.136)	0.493 (0.103)	0.517 (0.113)	0.573 (0.169)
26	0.258 (0.215)	0.371 (0.172)	0.375 (0.162)	0.379 (0.094)	0.393 (0.066)	0.408 (0.068)	0.450 (0.107)	0.480 (0.163)
51	0.236 (0.197)	0.403 (0.187)	0.411 (0.180)	0.395 (0.132)	0.383 (0.097)	0.407 (0.103)	0.452 (0.168)	0.458 (0.171)
101	0.215 (0.192)	0.365 (0.167)	0.433 (0.205)	0.375 (0.114)	0.367 (0.066)	0.385 (0.097)	0.463 (0.179)	0.486 (0.214)
251	0.275 (0.229)	0.470 (0.204)	0.518 (0.186)	0.409 (0.109)	0.364 (0.066)	0.369 (0.087)	0.476 (0.163)	0.634 (0.319)
1001	0.362 (0.276)	0.624 (0.224)	0.647 (0.230)	0.606 (0.160)	0.472 (0.083)	0.402 (0.046)	0.365 (0.037)	0.775 (0.166)

Table 2: $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ values when differing the number of observations (n) and the number of variables (p and q). Lower values of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ correspond to a more green colour (indicating higher accuracy), and higher values of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ correspond to a more red colour (indicating lower accuracy). Standard deviations are between brackets.



When comparing the performance of CCA in Table 1 with the performance of sparse CCA in Table 2, we notice that in general, sparse CCA performs more accurate than regular CCA. However, there are some exceptions for small number of variables (p and q) and high number of observations (n) where sparse CCA and regular CCA do not differ significantly. Furthermore, we notice that the range of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ is much smaller than the range of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$. The range of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ is $[0.2146; 0.7752]$ and the range of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ is $[0.2378; 1.5560]$. This implies that sparse CCA is more robust to high dimensionality in a data set than regular CCA in terms of the subspace spanned by the canonical vectors. Again, similar results hold for

$\mathbf{V}$ and can be found by comparing the tables in the appendices. It is interesting to note that, even though the underlying factors are not sparse, the sparse canonical vectors of sparse CCA are more accurate than the non-sparse canonical vectors of regular CCA in most cases.

Finally, we compute the Frobenius norm measurement for differing number of variables and observations. The results show that more variables in the data set correspond to more accurate results for sparse CCA in terms of the Frobenius norm measurement, as we would expect. In general the more samples in the data set implies significantly higher values of the Frobenius norm measurement. The details and further descriptions of the results can be found in Appendix G.

5.1.2 Differing variance

We apply the regular CCA method and the sparse CCA method of Witten et al. (2009) to a structured simulated data set with sparse underlying factors (simulation details are given in Appendix H). Again, we use 100 simulations ($M = 100$) and 2 factors ($K = 2$). The results are shown in Table 3. They indicate that regular CCA performs poorly in terms of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ and $\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$. The values of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ and $\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$ slightly improve when increasing the variance from 0.3^2 to 1.2^2, but are still relatively high. Additionally, for sparse CCA we notice that this method performs relatively well in cases of a low variance in terms of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ and $\theta_{sCCA}(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$, but when the variance increases, we see that sparse CCA performs rapidly worse. When we compare regular CCA and sparse CCA, we observe that sparse CCA performs significantly more accurate when the variance is relatively low. For variance $\in \{0.3^2, 0.6^2, 0.9^2, 1.2^2\}$, sparse CCA performs significantly better than regular CCA, but when the variance is 1.5^2 or higher, sparse CCA and regular CCA do not differ significantly. Additionally, in terms of the Frobenius norm, sparse CCA exhibits a significant increase in each step of expanding variance.

Variance	$\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$	$\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$	$\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$	$\theta_{sCCA}(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$	Frobenius norm sparse CCA
0.3^2	1.477 (0.037)	0.478 (0.136)	1.480 (0.032)	0.473 (0.137)	0.036 (0.001)
0.6^2	1.437 (0.054)	1.116 (0.182)	1.438 (0.045)	1.099 (0.171)	0.141 (0.004)
0.9^2	1.419 (0.058)	1.327 (0.098)	1.431 (0.048)	1.323 (0.107)	0.316 (0.008)
1.2^2	1.406 (0.067)	1.360 (0.084)	1.403 (0.063)	1.368 (0.078)	0.560 (0.013)
1.5^2	1.399 (0.063)	1.376 (0.078)	1.405 (0.063)	1.389 (0.082)	0.871 (0.022)
1.8^2	1.396 (0.051)	1.396 (0.076)	1.404 (0.065)	1.390 (0.076)	1.257 (0.029)
2.1^2	1.396 (0.063)	1.386 (0.073)	1.398 (0.067)	1.405 (0.071)	1.711 (0.039)
2.4^2	1.409 (0.059)	1.393 (0.076)	1.399 (0.064)	1.394 (0.083)	2.231 (0.053)
2.7^2	1.396 (0.075)	1.397 (0.070)	1.393 (0.069)	1.395 (0.080)	2.822 (0.071)
3.0^2	1.399 (0.063)	1.383 (0.082)	1.398 (0.069)	1.405 (0.069)	3.482 (0.085)
9.0^2	1.392 (0.065)	1.397 (0.067)	1.391 (0.062)	1.398 (0.073)	31.222 (0.678)

Table 3: Differing variances in sparse CCA and regular CCA, using accuracy measurements. Standard deviations are given between brackets. Lower values correspond to higher accuracy of the method.

5.1.3 Contamination of the data

We compare two different types of simulation studies using alterations to 10% of the data. The type with a known (sparse) structure, as explained in Appendix H, and the type with an unknown (non-sparse) structure as explained in Appendix E (note that we only draw values 2 to p and q from a standard normal distribution and set the first value equal to 0 to make the

TNR feasible). We evaluate the performance of regular and sparse CCA in these two types of structures using accuracy measurements and sparsity recognition measurements. We also vary the variance from 0.3^2 to 3^2. Finally, each time we do not only run our simulation as described in the appendices, we also make 10% of the data more varianced. We do this by randomly removing 10% of the simulated $\mathbf{U}$ and $\mathbf{V}$ for each simulation $m \in \{1, ..., M\}$, and replace these values with values drawn from a normal distribution with mean 0 and a standard deviation equal to 9 times the normally used standard deviation. So if we consider the case where var $= 1^2$, the outliers are simulated from $\mathcal{N}(0, 9 \times 1^2)$. Again, we use $M = 100$ and $K = 2$.

The results are shown in Table 4. Regular CCA performs significantly better without the outliers in terms of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$. Sparse CCA also performs more accurate without including the outliers for a low variance (0.3^2) with regard to $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$. However, with higher variances, sparse CCA performs better when outliers are included compared to when they are not included. This can be explained by the fact that in the resulting estimated canonical vectors, a large part of the simulated outliers are selected. We again notice that sparse CCA performs better than regular CCA in terms of angle between the subspaces spannend. Additionaly, regular CCA performs more accurate in terms of $\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$ in the unstructured simulation setting. However, sparse CCA performs significantly better in terms of $\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$ in the structured simulation settings for var $\in \{0.3^2, 1^2\}$ (except for var $= 1^2$ in the case of outliers). These are perspicuous results, since one could expect CCA to perform better in unstructured settings as their resulting canonical vectors are unstructerd. One could also expect sparse CCA to perform better in structured settings as their resulting canonical vectors incorporates sparsity, so they have a certain structure.

The TPRs are all significantly higher without outliers and the TNRs are all significantly higher including outliers (except for var $= 0.3^2$ with the structured details). The TPR is most often significantly higher in the unstructured simulation setting than in the structured simulation setting, while the TNR is significantly higher in the structured data. The Gini coefficients are always significantly higher when including the outliers, so the canonical vectors become more sparse with outliers included.

The results are similar for $\mathbf{V}$ and are shown in Appendix J.

Simulation details	$\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$		$\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$		$TPR(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$		$TNR(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$		GINI($\mathbf{U}$)	
Structured	w.o 10%	with 10%	w.o 10%	with 10%	w.o 10%	with 10%	w.o 10%	with 10%	w.o 10%	with 10%
Var = 0.3^2	1.471	1.505	0.459	0.556	0.984	0.921	0.939	0.970	0.640	0.909
Var = 1^2	1.402	1.485	1.346	0.460	0.894	0.840	0.982	0.990	0.778	0.959
Var = 3^2	1.395	1.487	1.393	0.362	0.908	0.855	0.981	0.990	0.763	0.961
Unstructured	w.o 10%	with 10%	w.o 10%	with 10%	w.o 10%	with 10%	w.o 10%	with 10%	w.o 10%	with 10%
Var = 0.3^2	1.505	1.519	0.380	0.491	0.992	0.960	0.537	0.608	0.610	0.759
Var = 1^2	1.434	1.493	1.199	0.448	0.959	0.889	0.482	0.658	0.793	0.947
Var = 3^2	1.397	1.486	1.385	0.372	0.961	0.869	0.395	0.640	0.737	0.953

Table 4: Results of $\mathbf{U}$, altering 10% of the data to become outliers, using accuracy and sparsity recognition metrics. On the left of each element, the values without alterations are given and on the right the values with alterations are given. The table including standard deviations is given in Appendix I.

5.2 Comparing PMD(L_1, L_1) with PMD(L_1, FL)

Witten et al. (2009) consider two specific variants of their PMD method. Here, we compare the two variants of the PMD to find the respective merits and limitations of those methods. To compare PMD(L_1, L_1) with PMD(L_1, FL), we simulate a data set containing 12 samples with 1000 copy number measurements on a single chromosome. The simulated data has five samples that contain a region of gain in the copy numbers 100 up to 500. The simulation details are given in Appendix K. Figure 1 indicates that both methods discover the five samples that contain the region of gains in $\mathbf{u}$. Additionally, the results of canonical vector $\mathbf{v}$ in PMD(L_1, FL) are more smooth than the results in PMD(L_1, L_1). Figure 1 clearly shows that $\mathbf{v}$ almost exclusively consists of nonzero weights in the region 100 to 500. This is not the case in PMD(L_1, L_1), since there are a lot of spikes in that area. Also in PMD(L_1, L_1), there are a lot more fluctuations besides the true region of gain than in PMD(L_1, FL). PMD(L_1, FL) is here the better method to find the region of gain in the copy number measurements.

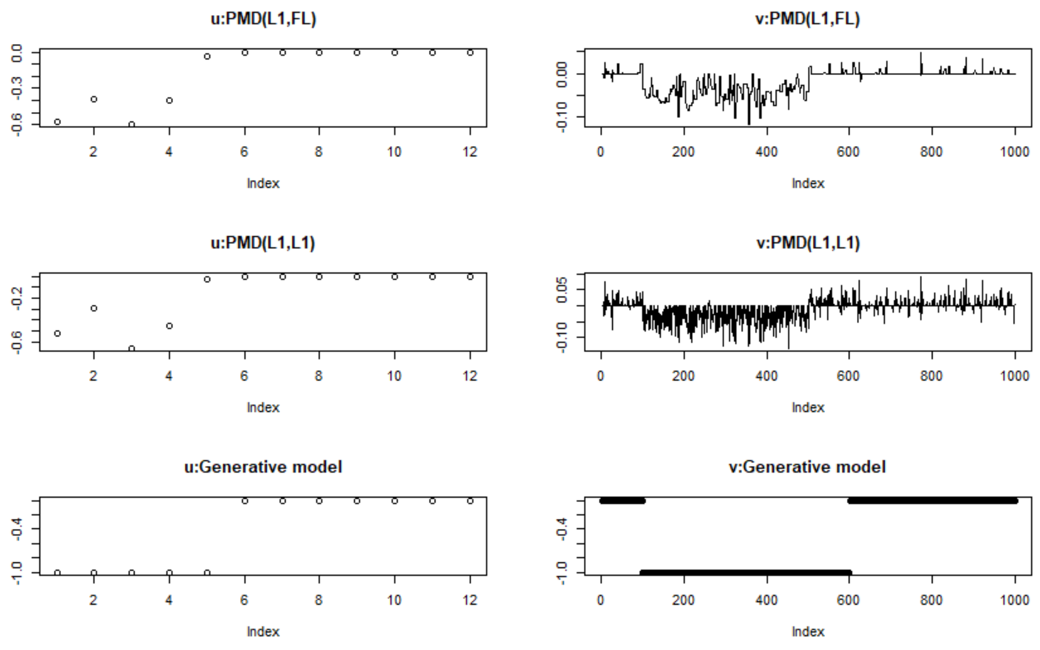


Figure 1: Performance of PMD(L_1, L_1) and PMD(L_1, FL). On top the results of PMD(L_1, FL), in the middle the results PMD(L_1, L_1) and on the bottom the generative model.

Furthermore, we compare sparse CCA(L_1, L_1) with sparse CCA(L_1,FL) using accuracy and sparsity recognition metrics in three different simulation settings. First of all, a structured version is considered where the true values of $\mathbf{U}$ and $\mathbf{V}$ are known and either -1, 1 or 0 (Appendix H using $n = 50$). The second variant is an unstructured variant. Here, the true values of $\mathbf{U}$ and $\mathbf{V}$ include randomness (Appendix E, setting the first values equal to 0 to make the TNR feasible). Lastly, we consider a combination of the above two variants. In this case, we simulate $\mathbf{U}$ according to the unstructured simulation details and $\mathbf{V}$ according to the structured simulation details. Again, we use 100 simulations ($M = 100$) and 2 factors ($K = 2$).

The results are shown in Table 5. For the structured data, sparse CCA(L_1, L_1) performs more accurate for $\mathbf{U}$ with regard to $\theta(\hat{\mathbf{U}}, \mathbf{U})$, gives higher TPR's (amount of true nonzero variables in the canonical vectors), and is more accurate in terms of the Frobenius norm measurement. Opposingly, sparse CCA(L_1, FL) performs more accurate for $\mathbf{V}$ with regard to $\theta(\hat{\mathbf{V}}, \mathbf{V})$, gives higher TNR's (hit rate of excluding unimportant variables), and is more sparse in terms of the Gini coefficients.

In the unstructured data setting, sparse CCA(L_1, L_1) performs more accurate for both $\mathbf{U}$ and $\mathbf{V}$. It also results in higher TPR's and better Frobenius norm measurements for sparse CCA(L_1, L_1). However, sparse CCA(L_1, FL) results are still more sparse in the Gini coefficients and give higher hit rates of excluding unimportant variables (TNR).

In the case of combination type data, the differences in all measurements are similar to the results of the structured data.

Simulation details	$\theta(\hat{\mathbf{U}}, \mathbf{U})$	$\theta(\hat{\mathbf{V}}, \mathbf{V})$	$TPR(\hat{\mathbf{U}}, \mathbf{U})$	$TPR(\hat{\mathbf{V}}, \mathbf{V})$	$TNR(\hat{\mathbf{U}}, \mathbf{U})$	$TNR(\hat{\mathbf{V}}, \mathbf{V})$	GINI($\mathbf{U}$)	GINI($\mathbf{V}$)	FN
Structured									
sparse CCA(L_1,L_1)	0.496 (0.155)	0.494 (0.156)	0.981 (0.009)	0.981 (0.009)	0.955 (0.046)	0.954 (0.062)	0.674 (0.155)	0.674 (0.155)	0.020 (0.001)
sparse CCA(L_1,FL)	0.839 (0.282)	0.121 (0.054)	0.951 (0.032)	0.978 (0.003)	0.964 (0.117)	0.990 (0.002)	0.863 (0.149)	0.794 (0.029)	0.022 (0.001)
Unstructured									
sparse CCA(L_1,L_1)	0.417 (0.139)	0.418 (0.138)	0.989 (0.011)	0.988 (0.011)	0.598 (0.142)	0.572 (0.184)	0.631 (0.116)	0.630 (0.116)	0.022 (0.001)
sparse CCA(L_1,FL)	0.550 (0.263)	0.808 (0.098)	0.973 (0.044)	0.983 (0.006)	0.627 (0.072)	0.612 (0.098)	0.714 (0.161)	0.808 (0.061)	0.042 (0.161)
Combination									
sparse CCA(L_1,L_1)	0.398 (0.077)	0.402 (0.052)	0.992 (0.002)	0.986 (0.003)	0.570 (0.143)	0.942 (0.109)	0.614 (0.080)	0.601 (0.086)	0.021 (0.001)
sparse CCA(L_1,FL)	0.647 (0.282)	0.132 (0.057)	0.964 (0.052)	0.981 (0.004)	0.613 (0.152)	0.990 (0.003)	0.768 (0.178)	0.761 (0.053)	0.025 (0.001)

Table 5: Performance of sparse CCA(L_1, L_1) and sparse CCA(L_1, FL) using different simulation settings. Standard deviations are between brackets.

5.3 Application of sparse CCA to a breast cancer data set

We apply the sparse CCA(L_1, FL) method to a data set about breast cancer patients. The method is executed to identify a set of genes that is correlated with DNA copy number changes (CGH spots) for each chromosome. We define the gene measurement data to be the $\mathbf{X}$ data matrix, and the DNA copy number change data to be the data matrix $\mathbf{Z}$. We apply sparse CCA(L_1, FL) instead of sparse CCA(L_1, L_1) because, as shown in Section 5.2, this method give more sparse results, which is favoured in this type of data. Also, when data matrix $\mathbf{Z}$ is ordered, sparse CCA(L_1, FL) performs more accurate for the corresponding canonical vectors $\mathbf{v}$. Here, $\mathbf{Z}$ corresponds to DNA copy number changes, which is an ordered data matrix, so sparse CCA(L_1, FL) is a suitable method. Finally, sparse CCA(L_1, FL) performs better in excluding unimportant variables in the canonical vectors, which is a useful advantage over sparse CCA(L_1, L_1).

Table 6 shows the most important genes after performing sparse CCA(L_1, FL) using the CGH spots on chromosome 1. Here, the value of penalty size c_1 is chosen such that the $\mathbf{u}$ vector have only 25 nonzero values. Table 6 shows the names of the 25 genes with nonzero weights. i defines the index of the gene and u_i defines the corresponding weight. Note that all nonzero genes are located on chromosome 1. This is expected, since the method finds genes that have a high correlation with DNA copy number changes on chromosome 1, and a copy number change on a chromosome should be correlated with genes on that chromosome.

Results of other chromosomes can be obtained by using the data of CGH spots of different chromosomes. For each chromosome k, the 25 genes with nonzero weights are all located on chromosome k, as we would expect.

i	Name of gene	Chromosome	u_i
68	jumping translocation breakpoint	1	0.0656
1235	translocated promoter region (to activated MET oncogene)	1	0.1401
1435	glyceronephosphate O-acyltransferase	1	0.2645
1463	NADH dehydrogenase (ubiquinone) Fe-S protein 2 (49kD) (NADH-coenzyme Q reductase)	1	0.2596
1679	nucleoporin 133kD	1	0.0380
1811	geranylgeranyl diphosphate synthase 1	1	0.1432
1863	rab3 GTPase-activating protein, non-catalytic subunit (150kD)	1	0.2741
2140	peroxisomal biogenesis factor 11B	1	0.1674
2324	phosphatidylinositol glycan, class C	1	0.0948
3175	tubulin-specific chaperone e	1	0.0864
4237	protoporphyrinogen oxidase	1	0.0690
5241	tuftelin 1	1	0.0511
5696	S-phase response (cyclin-related)	1	0.0021
8240	papillary renal cell carcinoma (translocation-associated)	1	0.0704
8342	splicing factor 3b, subunit 4, 49kD	1	0.4508
9481	UDP-Gal:betaGlcNAc beta 1,4- galactosyltransferase, polypeptide 3	1	0.2687
11634	hypothetical protein FLJ12671	1	0.2189
13650	ESTs, mitochondrial precursor [*H.sapiens*]	1	0.0176
14872	hypothetical protein HSPC155	1	0.1681
15333	mitochondrial ribosomal protein L24	1	0.2165
15354	HSPC003 protein	1	0.3895
15407	hypothetical protein FLJ10876	1	0.0966
15452	CGI-78 protein	1	0.1722
15777	chromosome 1 open reading frame 27	1	0.1344
18440	hypothetical protein My014	1	0.2781

Table 6: Non-zero genes in **u** using CGH spots on chromosome 1.

5.3.1 Correlations using test and training samples

To assess the usefulness of the sparse CCA method on the breast cancer data set, we divided the samples into a test and a training set. We draw 10 times randomly and independently 75% of the sample to serve as the training set. The remaining 25% of the sample is used as the test set. We applied sparse CCA(L_1,FL) to the training set to obtain the canonical vectors $\mathbf{u}_{tr}$ and $\mathbf{v}_{tr}$. The correlation of the training sample is calculated as $cor(\mathbf{X}_{tr}\mathbf{u}_{tr}, \mathbf{Z}_{tr}\mathbf{v}_{tr})$. The correlation of the test sample is calculated using the canonical vectors of the training sample and the data matrices of the test sample: $cor(\mathbf{X}_{te}\mathbf{u}_{tr}, \mathbf{Z}_{te}\mathbf{v}_{tr})$. Here $\mathbf{X}_{te}$ and $\mathbf{Z}_{te}$ denote the data matrices containing the 25% of the test data and $\mathbf{X}_{tr}$ and $\mathbf{Z}_{tr}$ denote the data matrices containing the 75% of the training data. The correlations are calculated as the mean over the 10 iterations. The results are displayed in Figure 2. They show that the correlations of the training sample are around 0.8. The correlations of the test sample are slightly lower than the training sample but still relatively high for most chromosomes. This means that in this holdout out-of-sample validation method, sparse CCA performs reasonably well.

5.3.2 Chromosomal gains and losses

We performed sparse CCA(L_1, FL) using all gene measurements and the DNA copy number changes of only chromosome 1. The values of the resulting canonical vector **v** (most important chromosomal gains and losses) of chromosome 1 are shown in Figure 3. It is shown that the resulting **v** is sparse and somewhat smooth. The higher the magnitude presented in Figure 3, the higher the value of $|v_i|$. The figures for the other chromosomes (**v**) can be found in Appendix L.

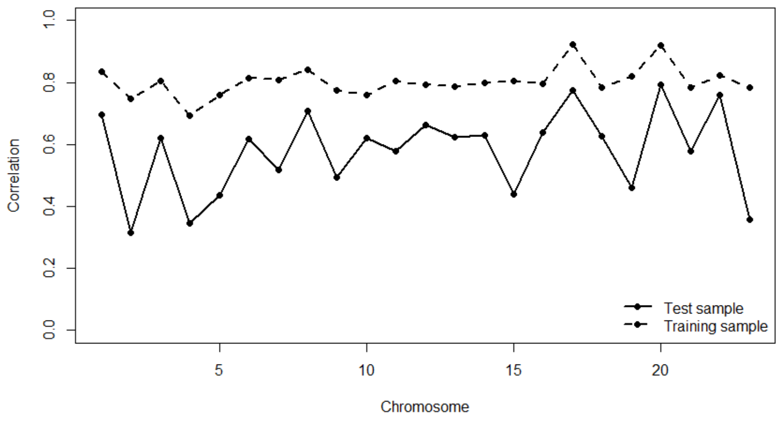


Figure 2: Correlations between gene expressions and DNA copy number changes. The correlations of the training sample in the dotted line and the correlations of the test sample in the black line.

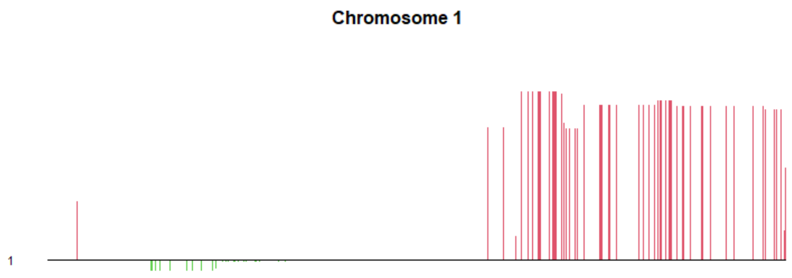


Figure 3: $\mathbf{v}$ for chromosome 1 obtained using PMD(L_1, FL). The horizontal axes denote the nucleotide position of the CGH spots along chromosome 1 and the vertical axes denote the value of each element in the resulting canonical vector $\mathbf{v}$.

6 Conclusion

In this paper, we investigated the performance of sparse canonical correlation analysis (sparse CCA) using the penalized matrix decomposition method of Witten et al. (2009). We performed different simulation studies to assess properties of regular CCA and sparse CCA, and examined the generalizability by differing dimensionalities and variances in data sets. We also applied the sparse CCA method to a real life breast cancer data set to find correlations between DNA copy number changes and gene measurements.

We investigated how regular CCA and sparse CCA perform in varying dimensional settings by changing the number of variables and samples in the data. We evaluated the accuracy using the angle between the subspace spanned by the estimated canonical vectors and the true canonical vectors. In general, we found that sparse CCA outperforms regular CCA in high-dimensional data sets, except when the data consists of a small number of variables and a high number of observations. The accuracy of regular CCA decreases with an increase in variables, while sparse CCA remains more robust to this increase. Both methods showed improved accuracy when increasing the number of observations, with an exception of cases where the number of observations reached a high threshold. Altogether, sparse CCA is more robust to highly dimensional data sets than regular CCA.

Additionally, we studied the performance of sparse CCA (and regular CCA) when increasing the variance in the data set. We found that regular CCA performs overall inaccurately. Sparse CCA performs more accurately than regular CCA in cases of low variance, but when the variance increases, sparse CCA performs quickly increasingly inaccurate. Sparse CCA and regular CCA perform equally poor for data sets with a high variance. Moreover, we investigated how CCA and sparse CCA deal with outliers in the data by making 10% of the data highly varianced. We did this for unstructured and structured data to assess the robustness of the methods. Regular CCA does not manage outliers well, while sparse CCA deals much better with outliers, especially for data with high variance, since it selects a part of the outliers. In addition, we observed that CCA performs better in structured data compared to unstructured data, whereas sparse CCA performs better in unstructured data than in structured data, as we would expect. Furthermore, results of sparse CCA become more sparse when outliers are included.

Finally, we applied the sparse CCA method to a real life genomic data set of breast cancer patients, to find correlations between gene measurements and DNA copy number changes. We found that sparse CCA is an adequate method to analyse this data. All genes that have the highest correlation with DNA copy number changes on a specific chromosome, are also found on that specific chromosome, as expected. We also established that sparse CCA performs well on this data set using a holdout out-of-sample method utilizing training and test data.

In this paper, we only considered two specific types of sparse CCA with LASSO type and fused LASSO type constraints. For further research, it might be interesting to consider other convex penalty functions. For example, one might consider to use the adaptive LASSO (Zou (2006)) as a convex penalty function. Adaptive LASSO gives adaptive weights to the LASSO penalty term based on estimated coefficients regarding the model fit, which can be useful in identifying the most important variables in two data sets.

Additionally, we only examine a few different simulation settings. This can limit the methods to not be generalizable to other data sets with different characteristics. In follow-up research, the CCA and sparse CCA methods can be compared using different simulation settings to determine if they perform similar or completely different. For instance, it can be interesting to simulate the data matrices in a manner such that the data is highly correlated, or not correlated at all. Then one can compare the performance of sparse CCA and regular CCA to assess how they behave in this sort of data.

Finally, we solely applied sparse CCA to a genomic data set. The sparse CCA method can also be applied to other types of data sets and can give interesting insights in different topics. For example, it can be applied to large environmental data sets to analyse potential harmful factors to the environment.

In conclusion, this paper explores the behaviour of sparse CCA in different data settings, which can provide insights for researchers regarding the use of sparse CCA. By understanding the behaviour of sparse CCA, researchers can make informed decisions when considering to apply sparse CCA, enhancing the effectiveness of their analyses.

References

Boyd, S. P. and Vandenberghe, L. (2004). *Convex-optimization.* Cambridge University Press.

Chin, K., DeVries, S., Fridlyand, J., Spellman, P. T., Roydasgupta, R., Kuo, W.-L., Lapuk, A., Neve, R. M., Qian, Z., Ryder, T., and et al. (2006). Genomic and transcriptional aberrations linked to breast cancer pathophysiologies. *Cancer Cell*, 10(6):529–541.

Chu, D., Liao, L.-Z., Ng, M. K., and Zhang, X. (2013). Sparse canonical correlation analysis: New formulation and algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):3050–3065.

Du, L., Liu, K., Yao, X., Risacher, S. L., Han, J., Saykin, A. J., Guo, L., and Shen, L. (2020). Detecting genetic associations with brain imaging phenotypes in alzheimer's disease via a novel structured scca approach. *Medical Image Analysis*, 61:101656.

Dudoit, S., Fridlyand, J., and Speed, T. P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457):77–87.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. J. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2).

Hastie, T., Tibshirani, R. J., and Wainwright, M. (2015). *Statistical learning with sparsity: The lasso and generalizations*, page 215–216. CRC Press, Taylor amp; Francis Group.

Hoefling, H. (2010). A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321.

Hoyer, P. (2002). Non-negative sparse coding. *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing.*

Hurley, N. and Rickard, S. (2009). Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741.

Iaci, R., Sriram, T., and Yin, X. (2010). Multivariate association and dimension reduction: A generalization of canonical correlation analysis. *Biometrics*, 66(4):1107–1118.

Johnson, R. A. and Wichern, D. W. (2019). *Applied Multivariate Statistical Analysis.* Pearson India Education Services.

Karnel, G. (1991). Robust canonical correlation and correspondence analysis. pages 335–354.

Kuhn, H. W. and Tucker, A. W. (2013). Nonlinear programming. *Traces and Emergence of Nonlinear Programming*, page 247–258.

Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255.

Lazzeroni, L. C. and Owen, A. (2002). Plaid models for gene expression data.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. 14th Annual Neural Information Processing Systems Conference, NIPS 2000 ; Conference date: 27-11-2000 Through 02-12-2000.

Lin, D., Calhoun, V. D., and Wang, Y.-P. (2014). Correspondence between fmri and snp data by group sparse canonical correlation analysis. *Medical image analysis*, 18(6):891–902.

Owen, A. B. and Perry, P. O. (2009). Bi-cross-validation of the svd and the nonnegative matrix factorization. *The Annals of Applied Statistics*, 3(2).

Parkhomenko, E., Tritchler, D., and Beyene, J. (2007). Genome-wide sparse canonical correlation of gene expression with genotypes. *BMC Proceedings*, 1(S1).

Parkhomenko, E., Tritchler, D., and Beyene, J. (2009). Sparse canonical correlation analysis with application to genomic data integration. *Statistical Applications in Genetics and Molecular Biology*, 8(1):1–34.

Rothman, A. J., Levina, E., and Zhu, J. (2010). Sparse multivariate regression with covariance estimation. *Journal of Computational and Graphical Statistics*, 19(4):947–962.

Tibshirani, R. J., Hastie, T., Narasimhan, B., and Chu, G. (2003). Class prediction by nearest shrunken centroids, with applications to dna microarrays. *Statistical Science*, 18(1).

Tibshirani, R. J., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(1):91–108.

Tibshirani, R. J. and Wang, P. (2007). Spatial smoothing and hot spot detection for cgh data using the fused lasso. *Biostatistics*, 9(1):18–29.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R. J., Botstein, D., and Altman, R. B. (2001). Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525.

Waaijenborg, S., Verselewel de Witt Hamer, P. C., and Zwinderman, A. H. (2008). Quantifying the association between gene expressions and dna-markers by penalized canonical correlation analysis. *Statistical Applications in Genetics and Molecular Biology*, 7(1).

Wilms, I. and Croux, C. (2015a). Robust sparse canonical correlation analysis. *SSRN Electronic Journal*.

Wilms, I. and Croux, C. (2015b). Sparse canonical correlation analysis from a predictive point of view. *Biometrical Journal*, 57(5):834–851.

Witten, D. M. and Tibshirani, R. J. (2009). Extensions of sparse canonical correlation analysis with applications to genomic data. *Statistical Applications in Genetics and Molecular Biology*, 8(1):1–27.

Witten, D. M., Tibshirani, R. J., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534.

Witten, D, T. R. and Gross, S. (2011). *'PMA'*. R package version 1.0.7.1.

Wold, S. (1978). Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429.

Appendix

A Algorithm to determine penalty sizes c_1 and c_2

Algorithm 5[3]

1. From the original data matrix $\mathbf{X}$, construct 10 data matrices $\mathbf{X}_1, ..., \mathbf{X}_{10}$, each for which is missing a non-overlapping 10% of the elements, sampled at random from the rows and columns.
2. For each candidate value of c_1 and c_2:
   (a) For $i \in 1, ..., 10$:
      i) Fit the PMD to $\mathbf{X}_i$ with tuning parameters c_1 and c_2, and calculate $\hat{\mathbf{X}}_i = d\mathbf{u}\mathbf{v}'$, the resulting estimate of $\mathbf{X}_i$.
      ii) Record the mean squared error (MSE) of the estimate $\hat{\mathbf{X}}_i$. This MSE is obtained by computing the mean of the squared differences between elements of $\mathbf{X}$ and the corresponding elements of $\hat{\mathbf{X}}_i$, where the mean is taken only over elements that are missing from $\mathbf{X}_i$.

   (b) Record the average mean squared error across $\mathbf{X}_1, ..., \mathbf{X}_{10}$ for tuning parameters c_1 and c_2.
3. The optimal values of c_1 and c_2 are those which correspond to the lowest MSE.

In practise this algorithm from Witten et al. (2009) takes a lot of computational space. Therefore, we do not consider different candidate values for c_1 and c_2, but only different candidate values for parameter c. The values of c_1 and c_2 are then computed by $c_1 = c\sqrt{n}$ and $c_2 = c\sqrt{p}$.

Similar cross-validation techniques are used in Wold (1978) and Owen and Perry (2009). In Owen and Perry (2009) they performed a kind of cross validation for SVD and the non-negative matrix factorization as described in Lee and Seung (1999).

We use a slightly different optimization problem when performing PMD while handling missing data values:

$$\max_{\mathbf{u},\mathbf{v}} \sum_{(i,j)\in C} X_{ij} u_i v_j \quad s.t. \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1, \quad P_1(\mathbf{u}) \leq c_1, \quad P_2(\mathbf{v}) \leq c_2. \tag{16}$$

Here, C denotes the set of all non-missing elements in the data matrix $\mathbf{X}$, X_{ij} denote the element in row i and column j of matrix $\mathbf{X}$ and u_i and v_j denote the i_{th} and j_{th} element from vectors $\mathbf{u}$ and $\mathbf{v}$ respectively. A similar approach for handling missing data in a singular value decomposition (SVD) is proposed in Troyanskaya et al. (2001).

Note that we did not program Problem (16) ourselves, since it is already incorporated in the 'PMA' package in R (Witten and Gross (2011)).

[3]This algorithm is also taken directly from Witten et al. (2009). In this case, some sort of 10-fold cross-validation is used, where in Section 5.1 we apply 5-fold cross-validation.

B Proof of Equation (4)

In this appendix, we give the proof of Equation (4) in Section 4.1.2.

$$\frac{1}{2}\|\mathbf{X}-\mathbf{UDV}'\|_F^2 = \frac{1}{2}\|\mathbf{X}\|_F^2 - \sum_{k=1}^{K}\mathbf{u}_k'\mathbf{X}\mathbf{v}_k d_k + \frac{1}{2}\sum_{k=1}^{K} d_k^2.$$

Using the trace trick, we can rewrite $\|\mathbf{X}-\mathbf{UDV}'\|_F^2$ as follows:

$$\begin{aligned}\|\mathbf{X}-\mathbf{UDV}'\|_F^2 &= tr((\mathbf{X}-\mathbf{UDV}')'(\mathbf{X}-\mathbf{UDV}'))\\ &= tr((\mathbf{X}'-\mathbf{VDU}')(\mathbf{X}-\mathbf{UDV}'))\\ &= tr(\mathbf{X}'\mathbf{X}-\mathbf{X}'\mathbf{UDV}'-\mathbf{VDU}'\mathbf{X}+\mathbf{VDU}'\mathbf{UDV}')\\ &= \|\mathbf{X}\|_F^2 - tr((\mathbf{X}'\mathbf{UDV}')') - tr(\mathbf{VDU}'\mathbf{X}) + tr(\mathbf{VDU}'\mathbf{UDV}')\\ &= \|\mathbf{X}\|_F^2 - 2tr(\mathbf{VDU}'\mathbf{X}) + tr(\mathbf{VDU}'\mathbf{UDV}').\end{aligned}$$

Here, we use the trace properties $tr(A+B) = tr(A) + tr(B)$, $tr(AB) = tr(BA)$ and $tr(A) = tr(A')$. We can rewrite $tr(\mathbf{VDU}'\mathbf{UDV}')$ as follows:

$$\begin{aligned}tr(\mathbf{VDU}'\mathbf{UDV}') &= tr(\mathbf{UDV}'\mathbf{VDU}')\\ &= tr(\mathbf{UDI}_p\mathbf{DU}')\\ &= tr(\mathbf{UDDU}')\\ &= tr(\mathbf{DU}'\mathbf{UD})\\ &= tr(\mathbf{DI}_n\mathbf{D})\\ &= tr(\mathbf{DD})\\ &= \sum_{k=1}^{K} d_k^2.\end{aligned}$$

Here, we use that $\mathbf{D}$ is a diagonal matrix, so $\mathbf{D}' = \mathbf{D}$, and we use that $\mathbf{U}'\mathbf{U} = \mathbf{I}_n$ and $\mathbf{V}'\mathbf{V} = \mathbf{I}_p$, where $\mathbf{I}_q$ is an identity matrix of size q.

We can rewrite $tr(\mathbf{VDU}'\mathbf{X})$ as $tr(\mathbf{DU}'\mathbf{XV})$. It can be derived that the diagonal elements of $\mathbf{DU}'\mathbf{XV}$ are equal to $d_k\mathbf{u}_k'\mathbf{X}\mathbf{v}_k$ for $k \in \{1, ..., K\}$. That implies that:

$$tr(\mathbf{VDU}'\mathbf{X}) = \sum_{k=1}^{K} d_k\mathbf{u}_k'\mathbf{X}\mathbf{v}_k.$$

These above derivations together derives that indeed

$$\frac{1}{2}\|\mathbf{X}-\mathbf{UDV}'\|_F^2 = \frac{1}{2}\|\mathbf{X}\|_F^2 - \sum_{k=1}^{K}\mathbf{u}_k'\mathbf{X}\mathbf{v}_k d_k + \frac{1}{2}\sum_{k=1}^{K} d_k^2.$$

C Proof of Equation in 4.1.4

In this appendix, we prove that in the problem

$$\max_{\mathbf{u}}\{\mathbf{u}'\mathbf{a}\} \quad s.t \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{u}\|_1 \leq c$$

$\mathbf{u}$ is solved for $\mathbf{u} = \frac{S(\mathbf{a},\Delta)}{\|S(\mathbf{a},\Delta)\|_2}$, where $\Delta = 0$ if this results in $\|\mathbf{u}\|_1 \leq c$ and otherwise Δ is chosen such that $\|\mathbf{u}\|_1 = c$ is satisfied.

This maximization problem is equivalent to the minimization problem

$$\min_{\mathbf{u}}\{-\mathbf{u}'\mathbf{a}\} \quad s.t \quad \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{u}\|_1 \leq c$$

and is equivalent to the following Lagrange form:

$$\min_{\mathbf{u}}\{z(\mathbf{u};\lambda,\Delta)\} = \min_{\mathbf{u}}\{-\mathbf{u}'\mathbf{a} + \lambda(\|\mathbf{u}\|_2^2 - 1) + \Delta(\|\mathbf{u}\|_1 - c)\}.$$

When we differentiate this objective function with respect to $\mathbf{u}$ and equalize to 0, we get:

$$\frac{\delta z(\mathbf{u};\lambda,\Delta)\}}{\delta \mathbf{u}} = 0 \Leftrightarrow -\mathbf{a} + 2\lambda\mathbf{u} + \Delta\Gamma = 0.$$

Here, $\Gamma_i = \begin{cases} sign(u_i) & \text{if } u_i \neq 0 \\ \in [-1,1] & \text{otherwise} \end{cases}$. The Karush-Kuhn-Tucker (KKT) conditions (as further explained in Kuhn and Tucker (2013)) for optimality are given by:

1. $-\mathbf{a} + 2\lambda\mathbf{u} + \Delta\Gamma = 0$
2. $\lambda(\|\mathbf{u}\|_2^2 - 1) = 0$
3. $\Delta(\|\mathbf{u}\|_1 - c) = 0$

From these KKT conditions, we can see that if $\lambda > 0$ we get

$$\mathbf{u} = \frac{S(\mathbf{a},\Delta)}{2\lambda}.$$

Given the above KKT condition 2, we need either $\lambda = 0$ (if feasible) or $\lambda \neq 0$ and needs to be chosen so that $\|\mathbf{u}\|_2 = 1$. This implies that we can solve for $\mathbf{u}$ in the following way:

$$\mathbf{u} = \frac{S(\mathbf{a},\Delta)}{\|S(\mathbf{a},\Delta)\|_2}.$$

Given the above KKT condition 3, either $\Delta = 0$ (if feasible) or $\Delta \neq 0$ and needs to be chosen so that $\|\mathbf{u}\|_1 = c$. This means that we use $\Delta = 0$ if $\|\mathbf{u}\|_1 \leq c$ and otherwise we choose Δ so that $\|\mathbf{u}\|_1 = c$.

D Penalized matrix decomposition versus SVD

We replicate the simulation study of Witten et al. (2009) that compares their PMD method with the singular value decomposition (SVD) method to demonstrate the advantages of the PMD method. The penalized matrix decomposition (PMD) method of Witten et al. (2009) is a generalization of the singular value decomposition (SVD). In Figure 4, we compare PMD(L_1, L_1) with SVD. The simulation details are given in Appendix H, only we use $n = 50$ in this case. We apply PMD(L_1, L_1) and SVD with $K = 2$. Figure 4 shows that the PMD method performs quite well in identifying the underlying factors. In general, the PMD gives nonzero values in the canonical vectors where there are nonzero values in the simulated canonical vectors. Even though it gives not exactly the same results as the true canonical vectors, it performs way better than SVD in identifying the underlying factors. SVD gives vague forms of how the canonical vectors should look like, but there is a lot of noise. In this case PMD is a better method for canonical correlation analysis than SVD, since it defines the linear combinations of the true data sets in the best way.

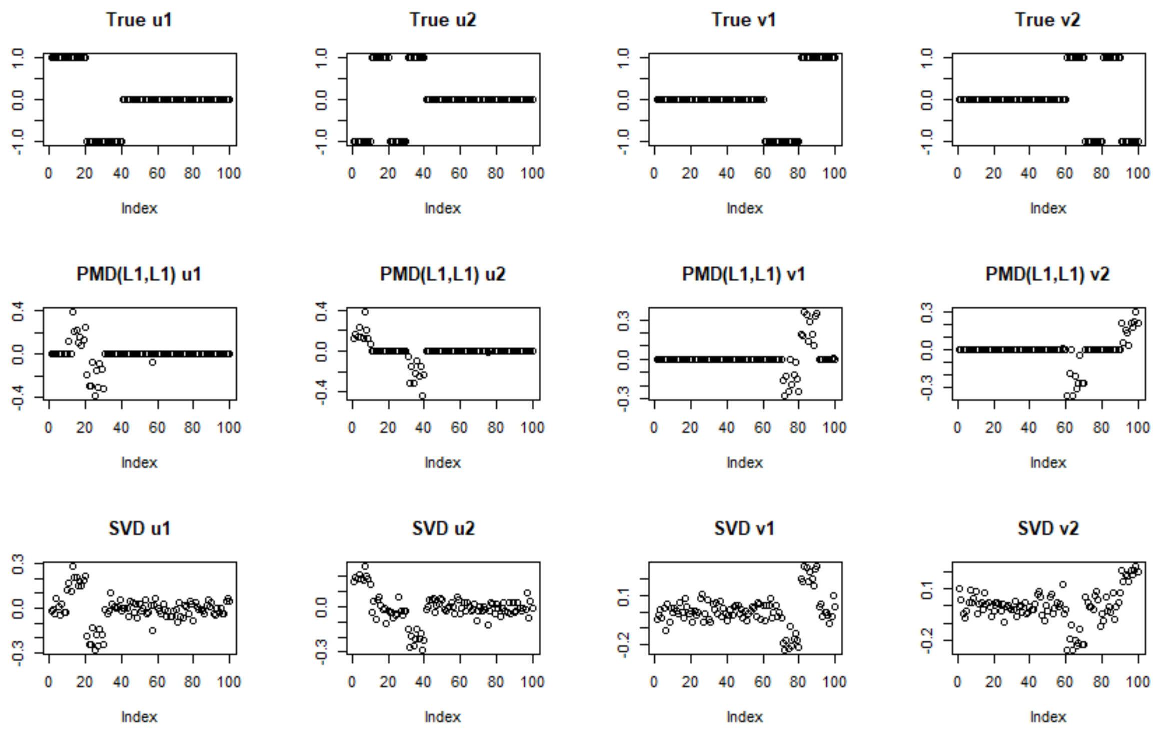


Figure 4: PMD versus SVD. On top the latent canonical vectors, in the middle the results of PMD(L_1, L_1), and on the bottom the results of SVD.

E Simulation details for Table 1 and Table 2

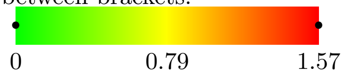
We construct data matrices $\mathbf{X}$ and $\mathbf{Z}$ using $p \times 1$ vectors $\mathbf{u}_1$, and $\mathbf{u}_2$ and $q \times 1$ vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively. All values of the vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$ and $\mathbf{v}_2$ are drawn from a standard normal distribution. We simulate two $n \times 1$ orthonormal vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ by drawing them from a standard normal distribution, then we apply the Gramm-Schmidt process to make them orthonormal. We simulate the elements of data matrices $\mathbf{X}$ and $\mathbf{Z}$ as follows:

- $X_{ij} \sim \mathcal{N}(w_{1,i}u_{1,j} + w_{2,i}u_{2,j}, 0.3^2)$
- $Z_{ij} \sim \mathcal{N}(w_{1,i}v_{1,j} + w_{2,i}v_{2,j}, 0.3^2)$

F Results Section 5.1.1 for canonical vector V

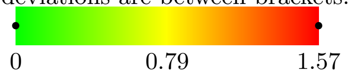
				p,q				
$n\downarrow$	4	7	10	25	50	100	250	1000
5	0.615 (0.373)	-	-	-	-	-	-	-
8	0.476 (0.330)	1.249 (0.189)	-	-	-	-	-	-
11	0.365 (0.279)	1.057 (0.238)	1.344 (0.124)	-	-	-	-	-
26	0.292 (0.209)	0.615 (0.226)	0.870 (0.194)	1.461 (0.054)	-	-	-	-
51	0.211 (0.179)	0.504 (0.184)	0.634 (0.173)	1.257 (0.123)	1.496 (0.024)	-	-	-
101	0.254 (0.195)	0.451 (0.175)	0.526 (0.151)	0.861 (0.109)	1.368 (0.071)	1.521 (0.020)	-	-
251	0.300 (0.249)	0.490 (0.200)	0.536 (0.174)	0.659 (0.094)	0.836 (0.085)	1.257 (0.081)	1.536 (0.011)	-
1001	0.322 (0.249)	0.573 (0.223)	0.660 (0.205)	0.677 (0.145)	0.671 (0.083)	0.726 (0.056)	0.985 (0.042)	1.555 (0.006)

Table 7: $\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$ values when differing the number of observations (n) and the number of variables (p and q). Lower values of $\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$ correspond to a more green colour (indicating higher accuracy), and higher values of $\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$ correspond to a more red colour (indicating lower accuracy). Standard deviations are between brackets.



				p,q				
$n\downarrow$	4	7	10	25	50	100	250	1000
5	0.231 (0.234)	0.517 (0.239)	0.566 (0.264)	0.650 (0.246)	0.714 (0.289)	0.674 (0.254)	0.638 (0.205)	0.652 (0.224)
8	0.282 (0.234)	0.460 (0.226)	0.476 (0.212)	0.542 (0.195)	0.539 (0.186)	0.550 (0.174)	0.559 (0.152)	0.603 (0.163)
11	0.266 (0.211)	0.425 (0.188)	0.454 (0.214)	0.467 (0.159)	0.464 (0.137)	0.487 (0.103)	0.511 (0.110)	0.571 (0.170)
26	0.261 (0.217)	0.370 (0.188)	0.367 (0.152)	0.385 (0.097)	0.390 (0.065)	0.416 (0.065)	0.453 (0.100)	0.481 (0.162)
51	0.241 (0.213)	0.384 (0.186)	0.415 (0.177)	0.381 (0.127)	0.382 (0.093)	0.406 (0.106)	0.453 (0.165)	0.458 (0.169)
101	0.257 (0.196)	0.389 (0.144)	0.423 (0.166)	0.378 (0.119)	0.361 (0.060)	0.390 (0.091)	0.462 (0.175)	0.485 (0.210)
251	0.289 (0.261)	0.452 (0.204)	0.500 (0.180)	0.402 (0.097)	0.377 (0.058)	0.373 (0.074)	0.477 (0.161)	0.633 (0.319)
1001	0.309 (0.272)	0.614 (0.218)	0.667 (0.242)	0.584 (0.149)	0.467 (0.093)	0.402 (0.041)	0.364 (0.039)	0.775 (0.160)

Table 8: $\theta_{sCCA}(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$ values when differing the number of observations (n) and the number of variables (p and q). Lower values of $\theta_{sCCA}(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$ correspond to a more green colour (indicating higher accuracy), and higher values of $\theta_{sCCA}(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$ correspond to a more red colour (indicating lower accuracy). Standard deviations are between brackets.



G Frobenius norm for differing dimensionality

We compare the accuracy of sparse CCA in terms of the Frobenius norm measurement, as described in Section 4.3.1. The results are shown in Table 9. We see that the Frobenius norm measurement decreases significantly in each step for all n when increasing the number of variables (p, q). This means that sparse CCA tends to become more accurate in terms of the Frobenius norm measurement as the number of variables in the data set increases. Additionally, in general, having a larger number of samples (n) in the data set leads to significantly higher values of the Frobenius norm measurement, meaning less accurate performance of sparse CCA. There are some exceptions, for $p, q = 10, n \in \{5, 8, 11\}$ the Frobenius norm decreases, but not significantly. Also for low values of p, q and low values of n, the Frobenius norm measurement does not always increase significantly in each step.

	p,q							
n↓	4	7	10	25	50	100	250	1000
5	0.680 (0.591)	0.421 (0.267)	0.294 (0.168)	0.070 (0.021)	0.025 (0.004)	0.010 (0.001)	0.003 (<0.001)	0.001 (<0.001)
8	0.786 (0.673)	0.421 (0.243)	0.288 (0.111)	0.074 (0.021)	0.028 (0.004)	0.011 (0.001)	0.004 (<0.001)	0.001 (<0.001)
11	0.786 (0.648)	0.433 (0.250)	0.279 (0.152)	0.077 (0.016)	0.030 (0.004)	0.012 (0.001)	0.004 (<0.001)	0.001 (<0.001)
26	0.872 (0.688)	0.518 (0.272)	0.316 (0.129)	0.088 (0.016)	0.038 (0.003)	0.017 (0.001)	0.006 (<0.001)	0.002 (<0.001)
51	0.880 (0.503)	0.544 (0.289)	0.359 (0.132)	0.108 (0.016)	0.047 (0.003)	0.022 (0.001)	0.008 (<0.001)	0.002 (<0.001)
101	0.985 (0.675)	0.666 (0.262)	0.412 (0.135)	0.139 (0.016)	0.064 (0.004)	0.030 (0.001)	0.012 (<0.001)	0.003 (<0.001)
251	1.233 (0.683)	0.805 (0.319)	0.581 (0.155)	0.199 (0.017)	0.095 (0.005)	0.047 (0.001)	0.018 (<0.001)	0.004 (<0.001)
1001	2.159 (1.232)	1.403 (0.409)	1.016 (0.224)	0.394 (0.039)	0.189 (0.009)	0.092 (0.002)	0.036 (<0.001)	0.009 (<0.001)

Table 9: Values of the Frobenius norm measurement when differing the number of observations (n) and the number of variables (p and q). Standard deviations are between brackets.

H Simulation details for Table 3 and Figure 4

We use $n = 150$, $p = 100$ and $q = 100$. In this simulation study, we simulate data matrices $\mathbf{X}$ and $\mathbf{Z}$ using $p \times 1$ vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ and $q \times 1$ vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively. The details of vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$ and $\mathbf{v}_2$ are:

- $\mathbf{u}_1$ consists of 20 times 1, 20 times -1 and 60 times 0.
- $\mathbf{u}_2$ consists of 10 times -1, 10 times 1, 10 times -1, 10 times 1 and 60 times 0.
- $\mathbf{v}_1$ consists of 60 times 0, 20 times -1 and 20 times 1.
- $\mathbf{v}_2$ consists of 60 times 0, 10 times 1, 10 times -1, 10 times 1 and 10 times -1.

Furthermore, we simulate two $n \times 1$ orthonormal vectors $\mathbf{w}_1$ and $\mathbf{w}_2$. Here, we simulate both $\mathbf{w}_1$ and $\mathbf{w}_2$ from a standard normal distribution and than apply the Gramm-Schmidt process to make them orthonormal. Then we simulate data matrices $\mathbf{X}$ and $\mathbf{Z}$ as follows:

- $X_{ij} \sim \mathcal{N}(w_{1,i}u_{1,j} + w_{2,i}u_{2,j}, 0.3^2)$
- $Z_{ij} \sim \mathcal{N}(w_{1,i}v_{1,j} + w_{2,i}v_{2,j}, 0.3^2)$

In Table 3 the variance of generating X_{ij} and Z_{ij} (0.3^2) differs.

I Results Section 5.1.3 with standard deviation

Simulation details	$\theta_{CCA}(\hat{\mathbf{U}}_{CCA}, \mathbf{U})$		$\theta_{sCCA}(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$		$TPR(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$		$TNR(\hat{\mathbf{U}}_{sCCA}, \mathbf{U})$		GINI($\mathbf{U}$)	
Structured										
Var = 0.3^2	1.471 (0.040)	1.505 (0.026)	0.459 (0.114)	0.556 (0.180)	0.984 (0.008)	0.921 (0.105)	0.939 (0.142)	0.970 (0.121)	0.640 (0.123)	0.909 (0.096)
Var = 1^2	1.402 (0.051)	1.485 (0.040)	1.346 (0.093)	0.460 (0.221)	0.894 (0.190)	0.840 (0.129)	0.982 (0.011)	0.990 (0.001)	0.779 (0.164)	0.959 (0.041)
Var = 3^2	1.395 (0.068)	1.487 (0.031)	1.393 (0.070)	0.362 (0.222)	0.908 (0.182)	0.855 (0.123)	0.981 (0.013)	0.990 (0.001)	0.763 (0.164)	0.961 (0.028)
Unstructured										
Var = 0.3^2	1.505 (0.021)	1.519 (0.021)	0.380 (0.088)	0.491 (0.208)	0.992 (0.003)	0.960 (0.077)	0.537 (0.205)	0.608 (0.152)	0.610 (0.093)	0.759 (0.176)
Var = 1^2	1.434 (0.053)	1.493 (0.031)	1.199 (0.152)	0.448 (0.163)	0.959 (0.072)	0.889 (0.122)	0.482 (0.246)	0.658 (0.036)	0.793 (0.155)	0.947 (0.038)
Var = 3^2	1.397 (0.066)	1.486 (0.036)	1.385 (0.066)	0.372 (0.197)	0.961 (0.083)	0.869 (0.128)	0.395 (0.287)	0.640 (0.103)	0.737 (0.168)	0.953 (0.036)

Table 10: Metrics using 10% outliers. On the left of each element, the normal values are given and on the right the values using 10% outliers. This is for the outcomes of $\mathbf{U}$.

J Results contamination of V

Simulation details	$\theta_{CCA}(\hat{\mathbf{V}}_{CCA}, \mathbf{V})$		$\theta_{sCCA}(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$		$TPR(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$		$TNR(\hat{\mathbf{V}}_{sCCA}, \mathbf{V})$		GINI($\mathbf{V}$)	
Structured										
Var = 0.3^2	1.482 (0.034)	1.504 (0.024)	0.455 (0.116)	0.545 (0.176)	0.984 (0.007)	0.921 (0.105)	0.949 (0.071)	0.976 (0.102)	0.639 (0.123)	0.910 (0.096)
Var = 1^2	1.413 (0.062)	1.483 (0.034)	1.349 (0.086)	0.466 (0.221)	0.895 (0.189)	0.840 (0.129)	0.981 (0.012)	0.991 (0.001)	0.779 (0.164)	0.959 (0.039)
Var = 3^2	1.403 (0.067)	1.483 (0.040)	1.397 (0.078)	0.368 (0.209)	0.905 (0.172)	0.854 (0.122)	0.980 (0.013)	0.991 (0.001)	0.763 (0.164)	0.962 (0.026)
Unstructured										
Var = 0.3^2	1.509 (0.024)	1.519 (0.021)	0.376 (0.092)	0.488 (0.202)	0.992 (0.004)	0.960 (0.077)	0.563 (0.174)	0.612 (0.140)	0.609 (0.094)	0.759 (0.175)
Var = 1^2	1.431 (0.055)	1.494 (0.035)	1.193 (0.150)	0.438 (0.190)	0.958 (0.073)	0.887 (0.121)	0.477 (0.244)	0.663 (0.023)	0.792 (0.155)	0.948 (0.037)
Var = 3^2	1.391 (0.060)	1.486 (0.034)	1.384 (0.063)	0.382 (0.212)	0.959 (0.083)	0.868 (0.127)	0.400 (0.290)	0.650 (0.077)	0.739 (0.167)	0.954 (0.035)

Table 11: Metrics using 10% outliers. On the left of each element, the normal values are given and on the right the values using 10% outliers. This is for the outcomes of $\mathbf{V}$.

K Simulation details for Figure 1

We use $n = 12$ and $p = 1000$ by simulating the $n \times p$ matrix $\mathbf{X}$. We simulate matrix such that the first 5 samples have regions of gain or loss between positions 100 and 500. This is done in the following way:

- All values X_{ij} are generated from a standard normal distribution $\mathcal{N}(0, 1)$
- except for $i \in \{1, ..., 5\}$ and $j \in \{100, ..., 500\}$, they are generated from $\mathcal{N}(1, 1)$.

L Weights of v for the other Chromosomes

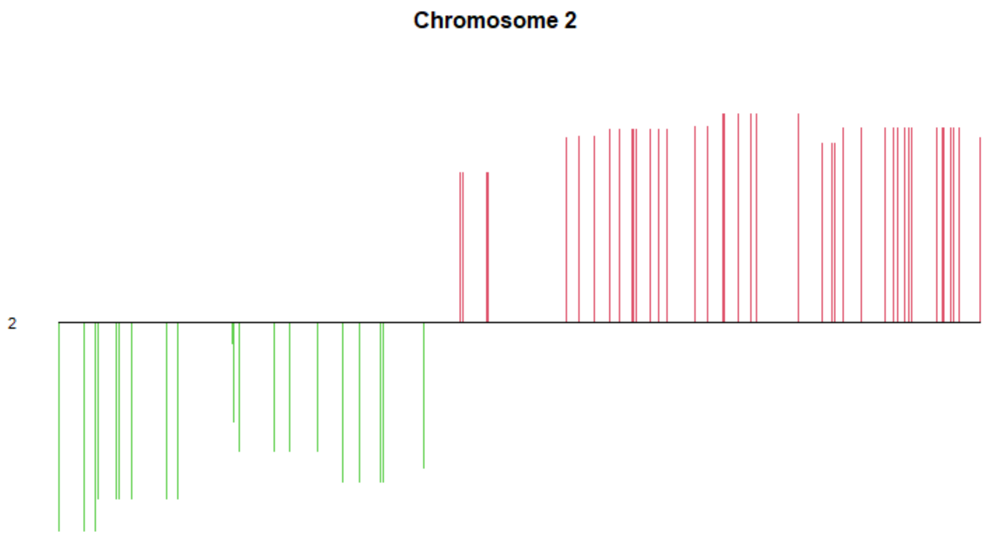


Figure 5: **v** for Chromosome 2

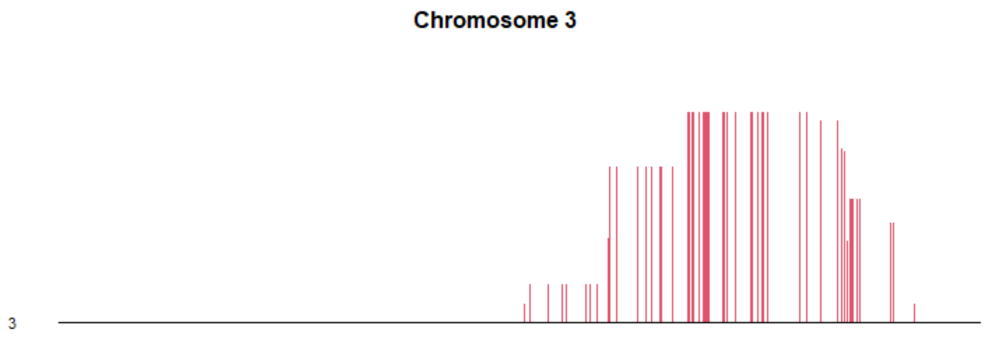


Figure 6: **v** for Chromosome 3

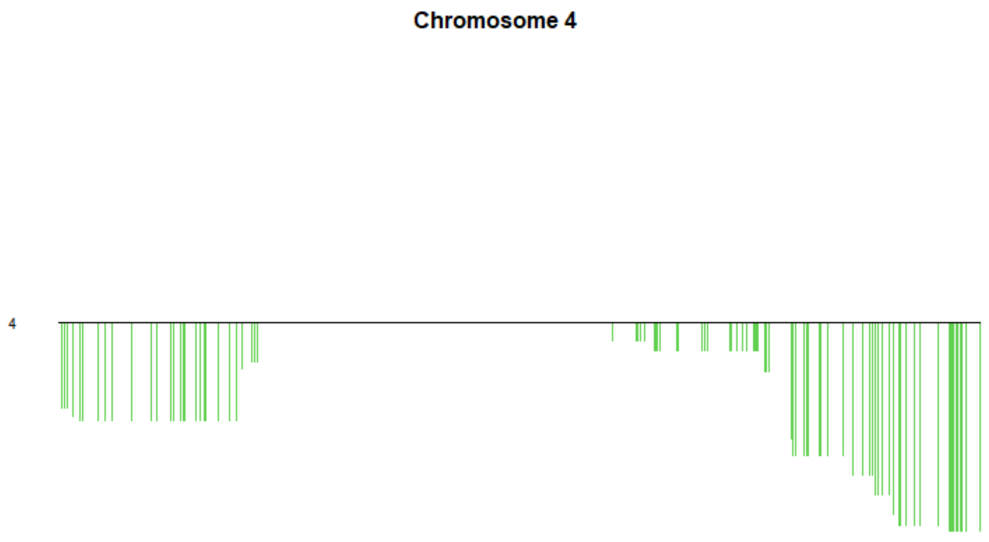


Figure 7: **v** for Chromosome 4

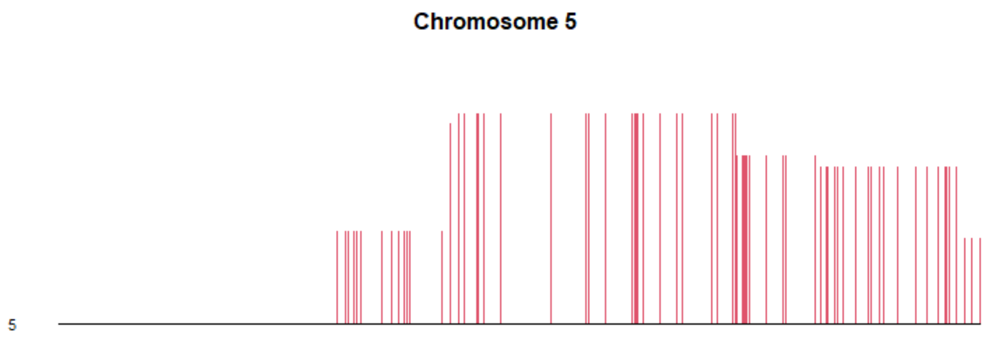


Figure 8: **v** for Chromosome 5

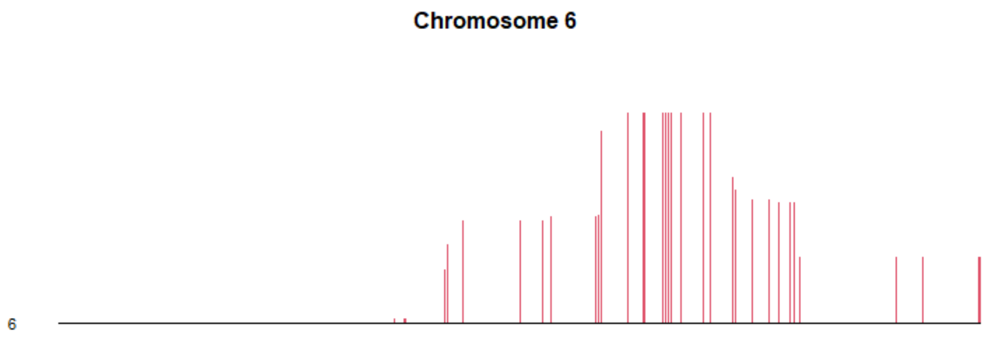


Figure 9: **v** for Chromosome 6

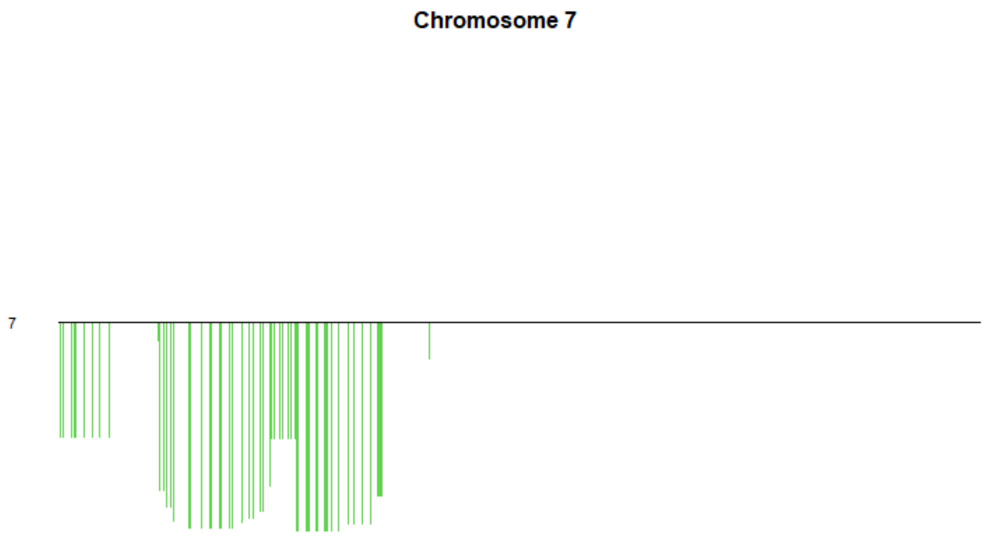


Figure 10: **v** for Chromosome 7

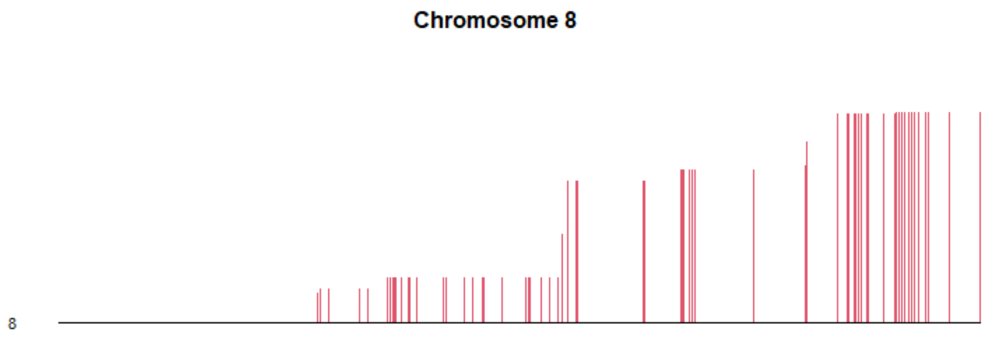


Figure 11: **v** for Chromosome 8

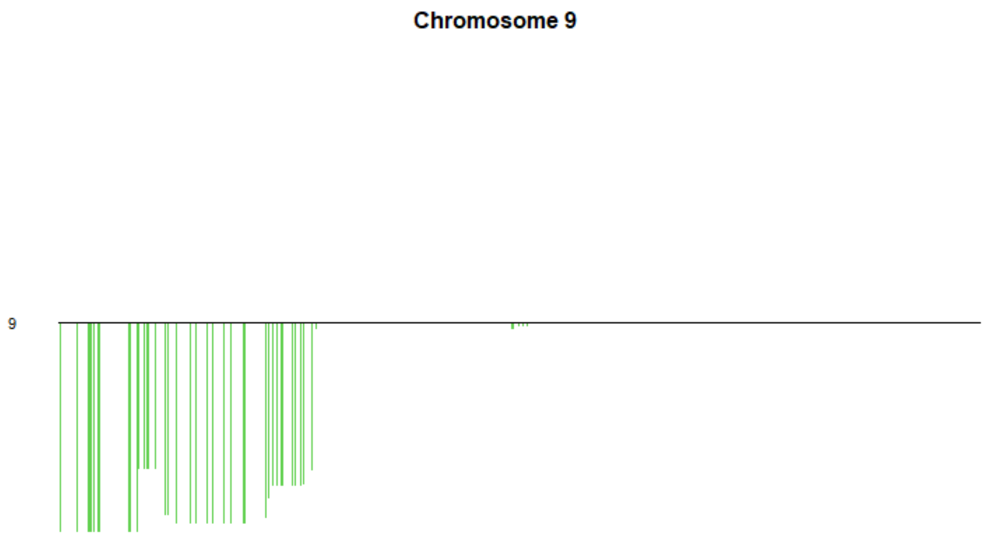


Figure 12: **v** for Chromosome 9

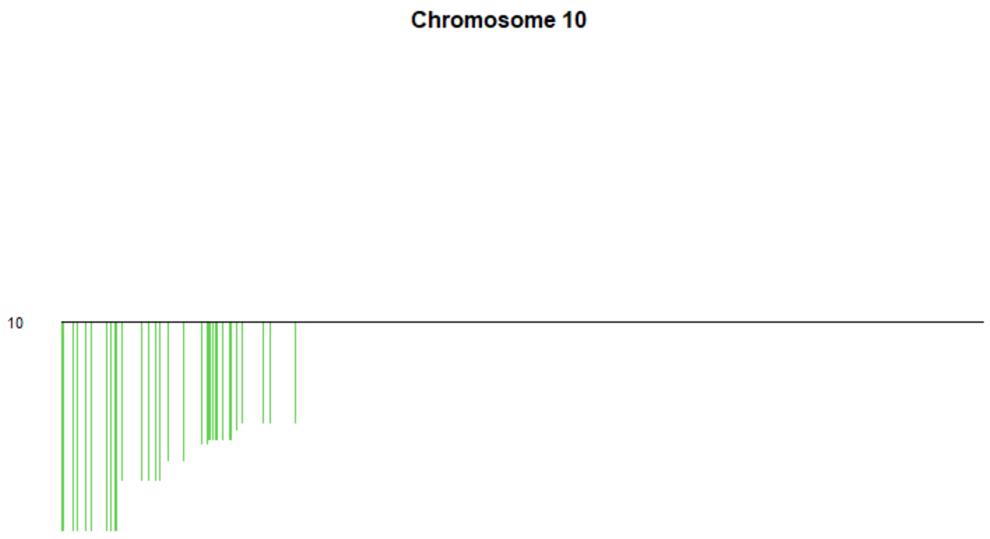


Figure 13: **v** for Chromosome 10

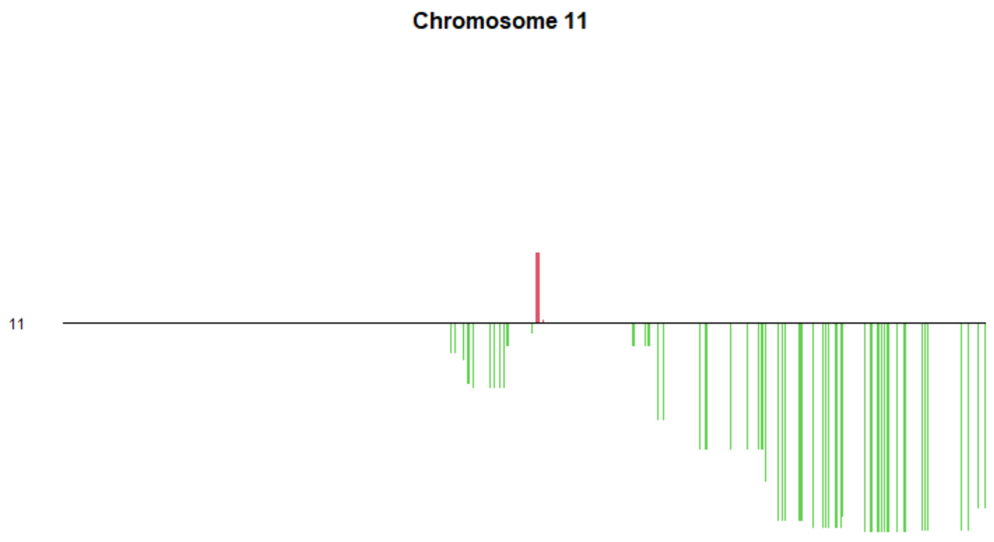


Figure 14: **v** for Chromosome 11

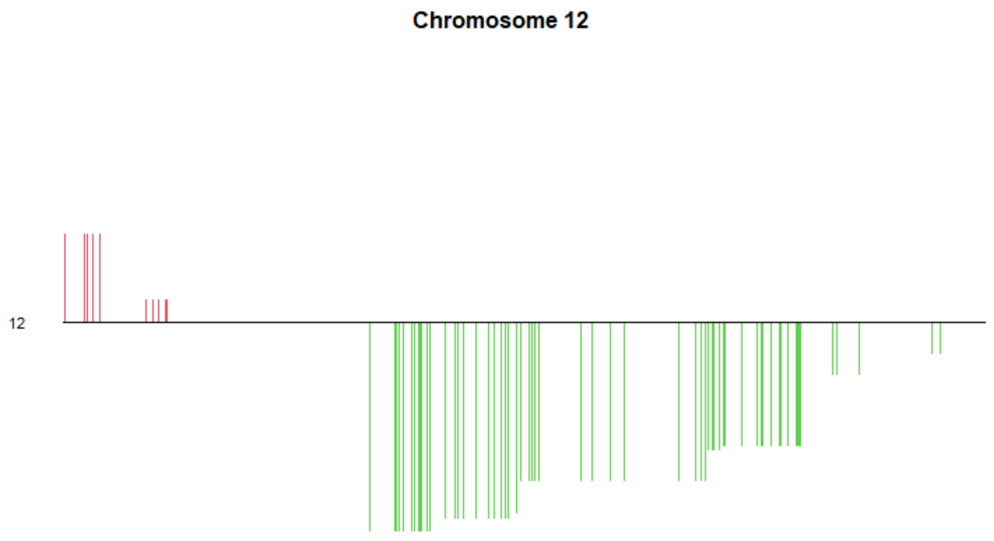


Figure 15: **v** for Chromosome 12

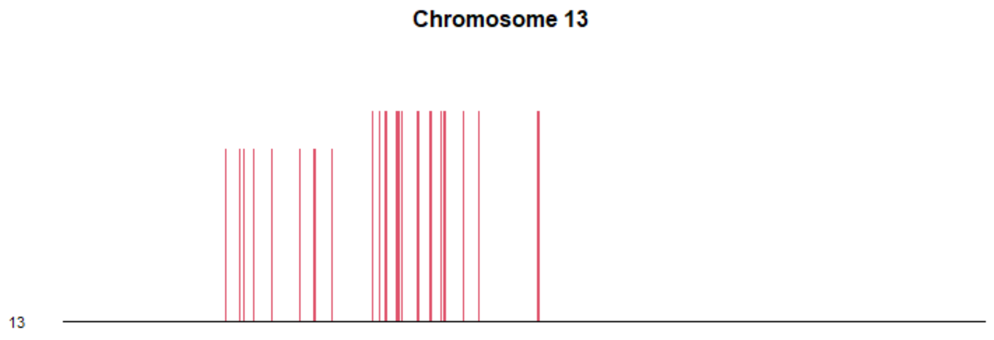


Figure 16: **v** for Chromosome 13

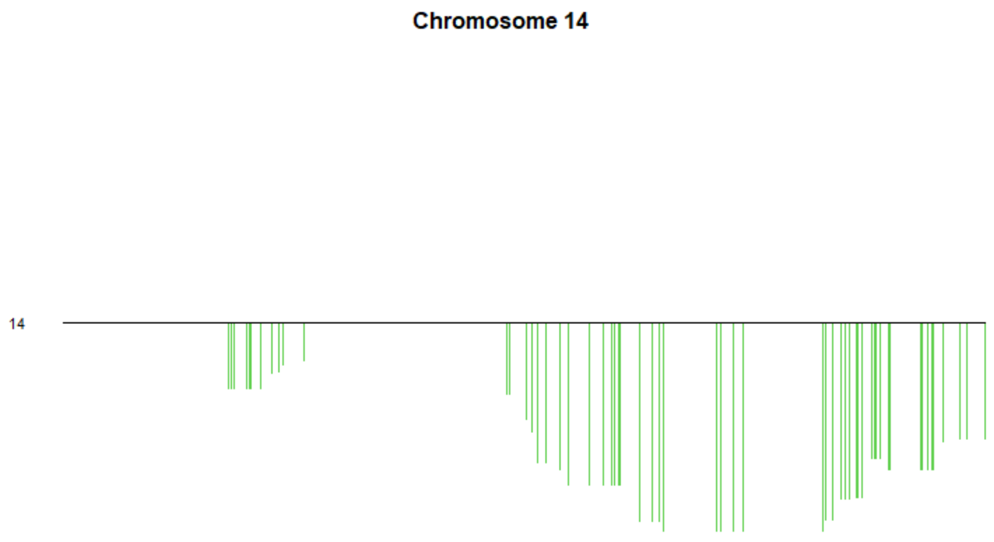


Figure 17: **v** for Chromosome 14

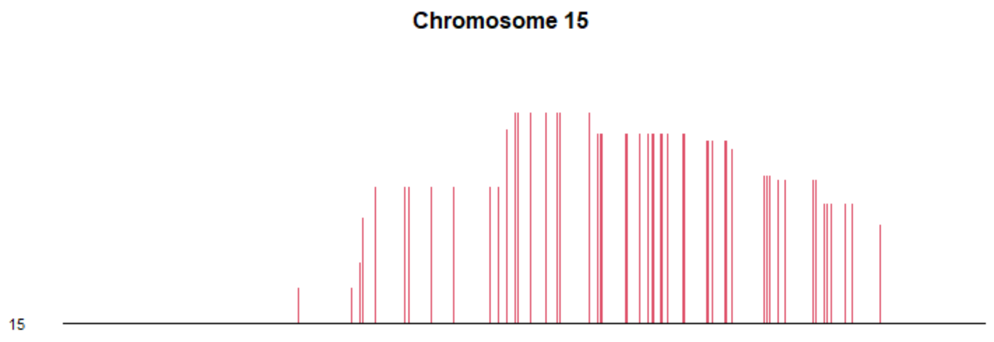


Figure 18: **v** for Chromosome 15

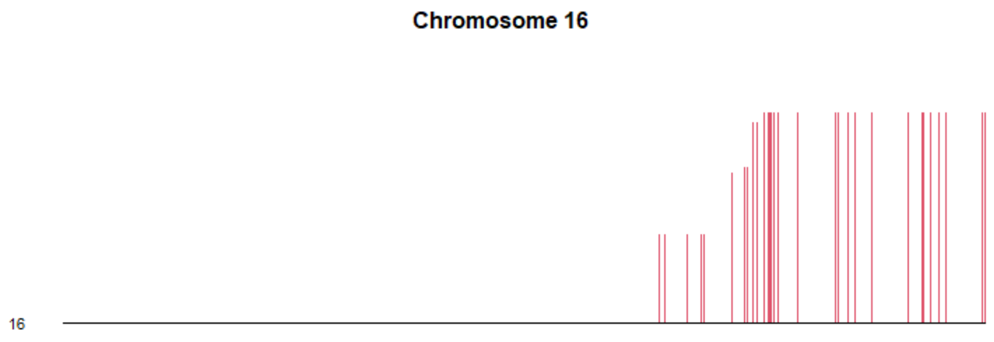


Figure 19: **v** for Chromosome 16

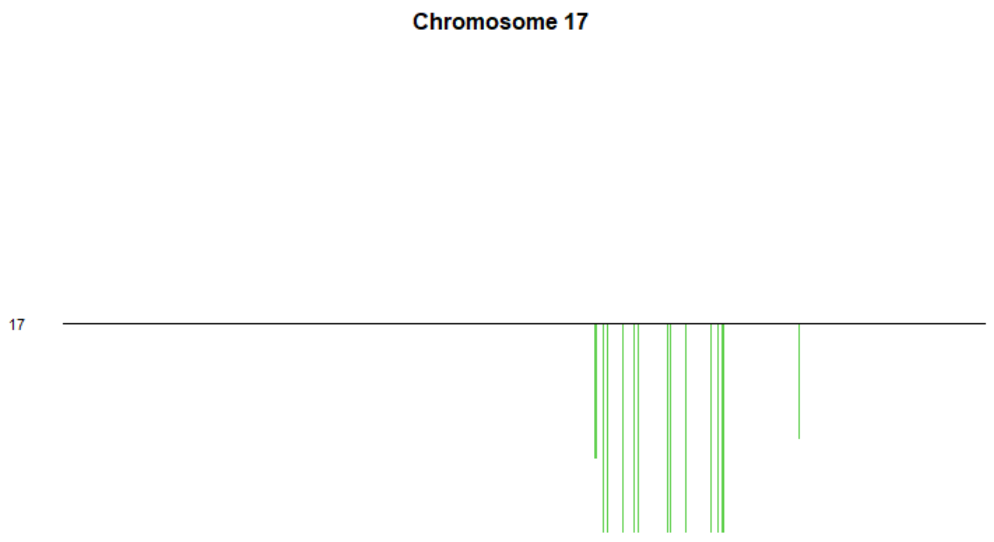


Figure 20: **v** for Chromosome 17

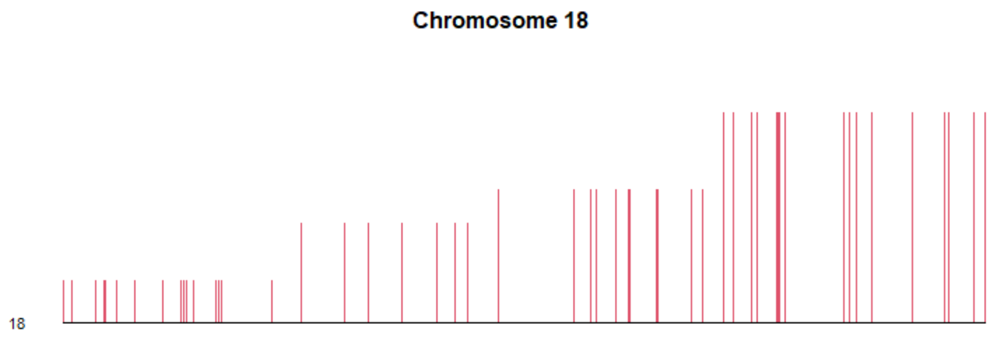


Figure 21: **v** for Chromosome 18

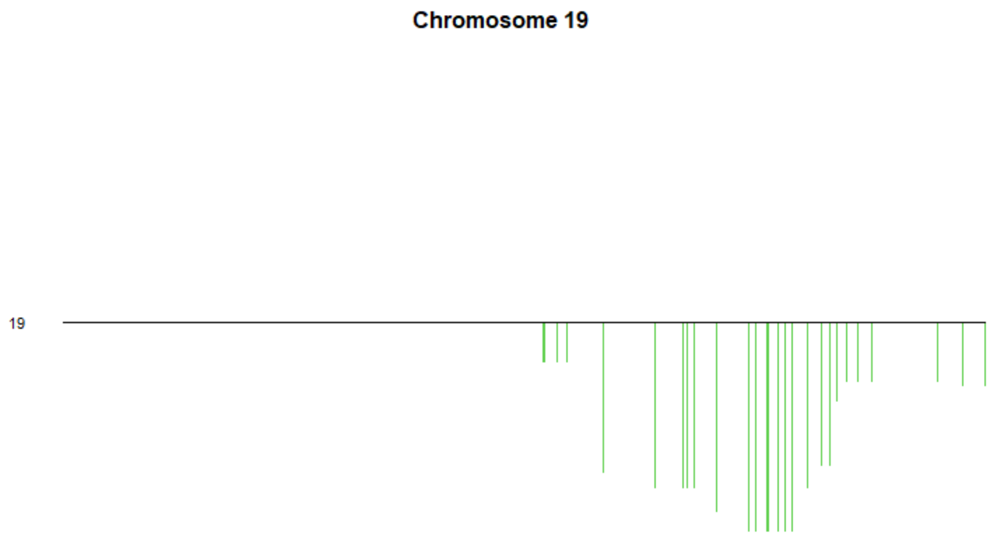


Figure 22: **v** for Chromosome 19

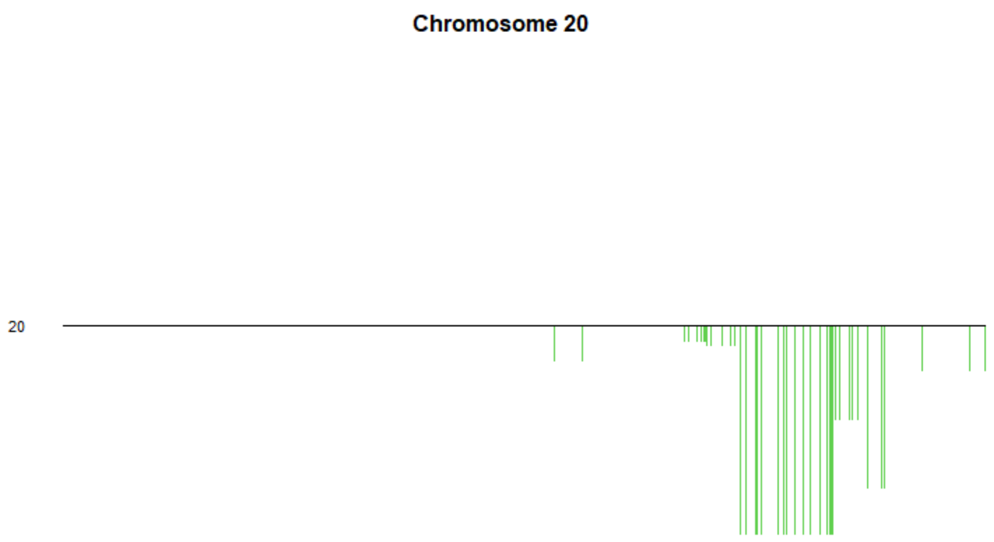


Figure 23: **v** for Chromosome 20

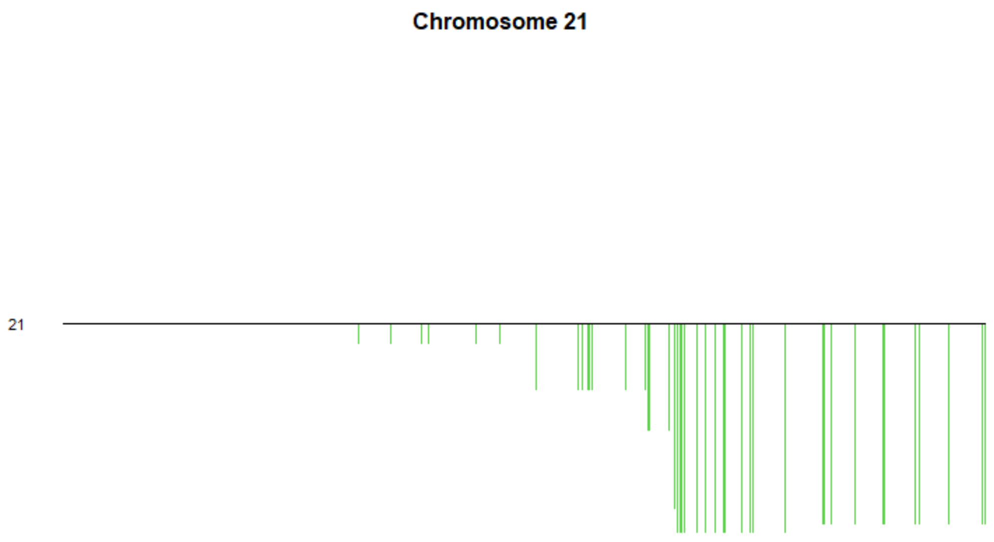


Figure 24: **v** for Chromosome 21

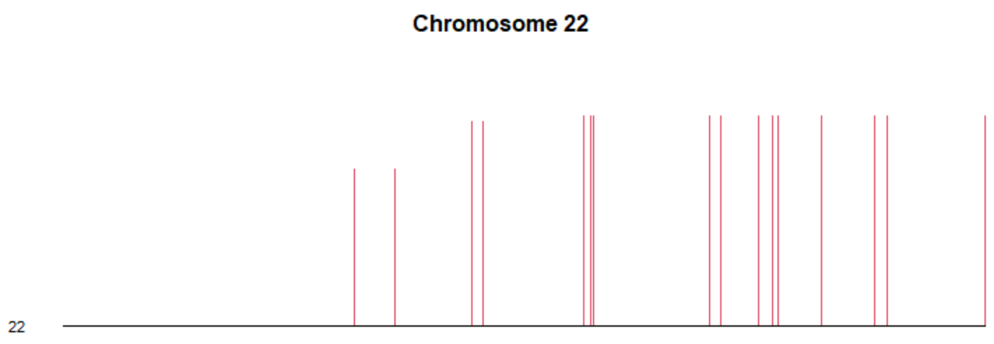


Figure 25: **v** for Chromosome 22

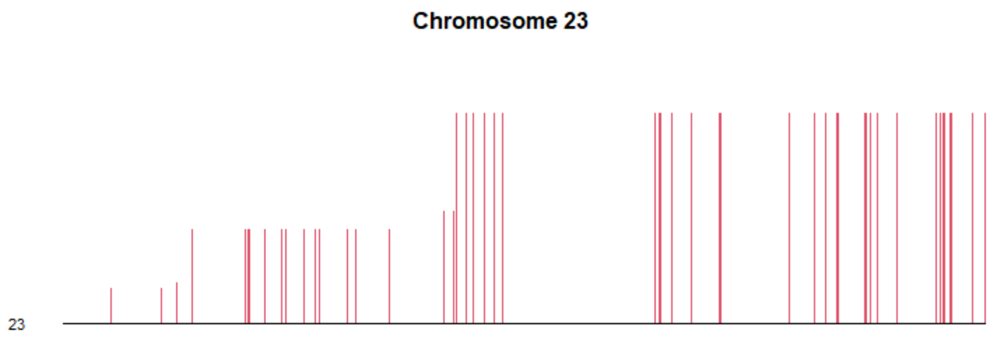


Figure 26: **v** for Chromosome 23