Linear Discriminant Analysis (LDA)

Lieven Clement

statOmics, Ghent University (https://statomics.github.io)

Contents


```
library(tidyverse)
library(gridExtra)
```

1 Breast cancer example

- Schmidt *et al.*, 2008, Cancer Research, **68**, 5405-5413
- Gene expression patterns in n=200 breast tumors were investigated (p=22283 genes)
- After surgery the tumors were graded by a pathologist (stage 1,2,3)

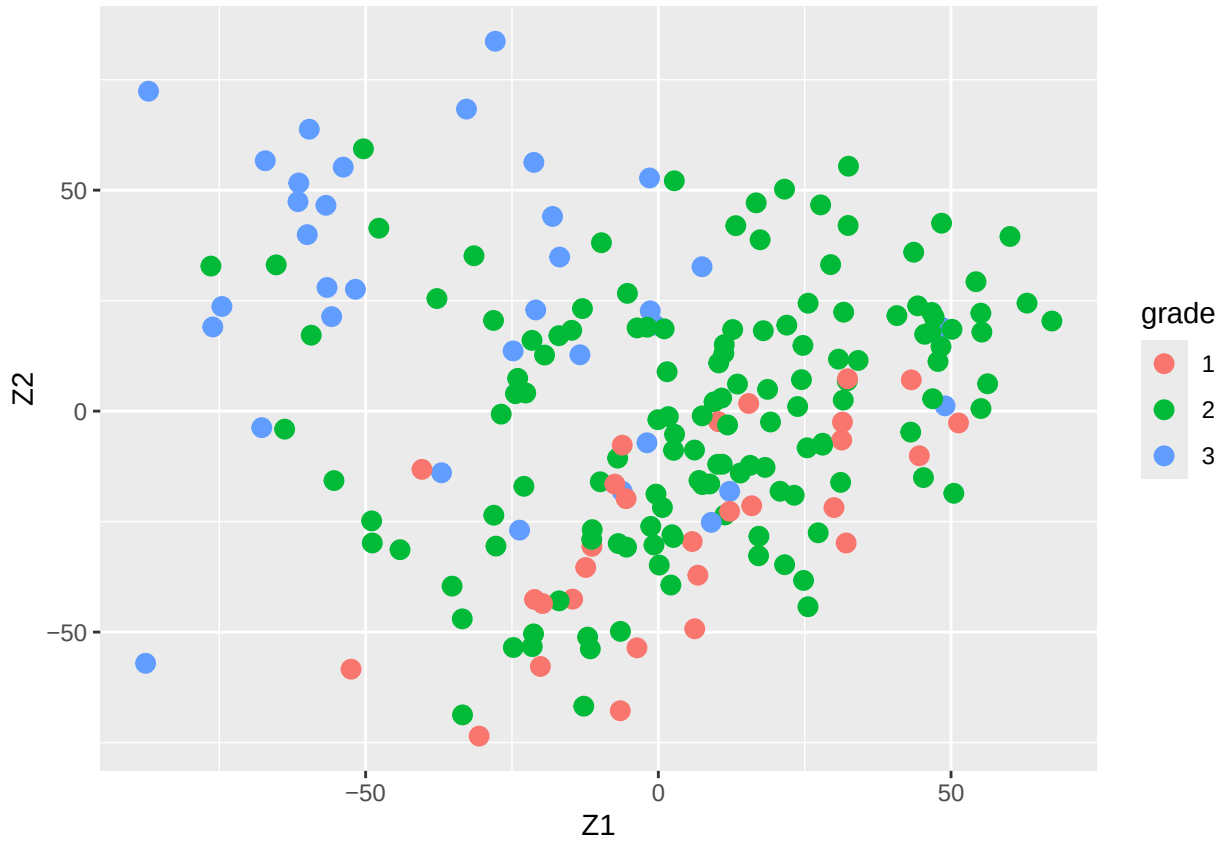
1.1 Data

```
#BiocManager::install("genefu")
#BiocManager::install("breastCancerMAINZ")

library(genefu)
library(breastCancerMAINZ)
data(mainz)
```

```r
X <- t(exprs(mainz)) # gene expressions
n <- nrow(X)
H <- diag(n)-1/n*matrix(1,ncol=n,nrow=n)
X <- H%*%X
Y <- pData(mainz)$grade
table(Y)
```

```
#> Y
#>   1   2   3
#>  29 136  35
```

```r
svdX <- svd(X)
k <- 2
Zk <- svdX$u[,1:k] %*% diag(svdX$d[1:k])
colnames(Zk) <- paste0("Z",1:k)

Zk %>%
  as.data.frame %>%
  mutate(grade = Y %>% as.factor) %>%
  ggplot(aes(x= Z1, y = Z2, color = grade)) +
  geom_point(size = 3)
```



2 Linear discriminant analysis

Fisher's construction of LDA is simple: it allows for classification in a dimension-reduced subspace of $\mathbb{R}^p$.

First we assume that Y can only take two values (0/1).

Fisher aimed for a direction, say $\mathbf{a}$, in the p-dimensional predictor space such that the orthogonal projections of the predictors, $\mathbf{x}^t\mathbf{a}$, show maximal ratio between the between and within sums of squares:

$$\mathbf{v} = \text{ArgMax}_a \frac{\mathbf{a}^t\mathbf{B}\mathbf{a}}{\mathbf{a}^t\mathbf{W}\mathbf{a}} \text{ subject to } \mathbf{a}^t\mathbf{W}\mathbf{a} = 1,$$

where $\mathbf{W}$ and $\mathbf{B}$ are the within and between covariance matrices of $\mathbf{x}$. The restriction is introduced to obtain a (convenient) unique solution.

2.1 Between and within sums of squares

- In the training dataset, let $\mathbf{x}_{ik}$ denote the ith p-dimensional observation in the kth group ($k = 0, 1$ referring to $Y = 0$ and $Y = 1$, resp.), $i = 1, ..., n_k$.
- Let $z_{ik} = \mathbf{a}^t\mathbf{x}_{ik}$ denote the orthogonal projection of $\mathbf{x}_{ik}$ onto $\mathbf{a}$
- For the one-dimensional z-observations, consider the following sum of squares:

$$\text{SSE} = \text{within sum of squares} = \sum_{k=0,1}\sum_{i=1}^{n_k}(z_{ik} - \bar{z}_k)^2$$

$$\text{SSB} = \text{between sum of squares} = \sum_{k=0,1}\sum_{i=1}^{n_k}(\bar{z}_k - \bar{z})^2 = \sum_{k=0,1} n_k(\bar{z}_k - \bar{z})^2$$

  with $\bar{z}_k$ the sample mean of z_{ik} within group k, and $\bar{z}$ the sample mean of all z_{ik}.
- To reformulate SSE and SSB in terms of the p-dimensional $\mathbf{x}_{ik}$, we need the sample means

$$\bar{z}_k = \frac{1}{n_k}\sum_{i=1}^{n_k} z_{ik} = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{a}^t\mathbf{x}_{ik} = \mathbf{a}^t\frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{x}_{ik} = \mathbf{a}^t\bar{\mathbf{x}}_k$$

$$\bar{z} = \frac{1}{n}\sum_{k=0,1}\sum_{i=1}^{n_k} z_{ik} = \cdots = \mathbf{a}^t\bar{\mathbf{x}}.$$

- SSE becomes

$$\text{SSE} = \sum_{k=0,1}\sum_{i=1}^{n_k}(z_{ik} - \bar{z}_k)^2 = \mathbf{a}^t\left(\sum_{k=0,1}\sum_{i=1}^{n_k}(\mathbf{x}_{ik} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ik} - \bar{\mathbf{x}}_k)^t\right)\mathbf{a}$$

- SSB becomes

$$\text{SSB} = \sum_{k=0,1} n_k(\bar{z}_k - \bar{z})^2 = \mathbf{a}^t\left(\sum_{k=0,1} n_k(\bar{\mathbf{x}_k} - \bar{\mathbf{x}})(\bar{\mathbf{x}_k} - \bar{\mathbf{x}})^t\right)\mathbf{a}$$

- The $p \times p$ matrix

$$\mathbf{W} = \sum_{k=0,1}\sum_{i=1}^{n_k}(\mathbf{x}_{ik} - \bar{\mathbf{x}}_k)(\mathbf{x}_{ik} - \bar{\mathbf{x}}_k)^t$$

  is referred to as the matrix of **within** sum of squares and cross products.
- The $p \times p$ matrix

$$\mathbf{B} = \sum_{k=0,1} n_k(\bar{\mathbf{x}_k} - \bar{\mathbf{x}})(\bar{\mathbf{x}_k} - \bar{\mathbf{x}})^t$$

  is referred to as the matrix of **between** sum of squares and cross products.
- Note that on the diagonal of $\mathbf{W}$ and $\mathbf{B}$ you find the ordinary univariate within and between sums of squares of the individual components of $\mathbf{x}$.

2.2 Obtain projections

An equivalent formulation:

$$\mathbf{v} = \text{ArgMax}_a \mathbf{a}^t\mathbf{B}\mathbf{a} \text{ subject to } \mathbf{a}^t\mathbf{W}\mathbf{a} = 1.$$

This can be solved by introducing a Langrange multiplier:

$$\mathbf{v} = \text{ArgMax}_a \mathbf{a}^t\mathbf{B}\mathbf{a} - \lambda(\mathbf{a}^t\mathbf{W}\mathbf{a} - 1).$$

Calculating the partial derivative w.r.t. $\mathbf{a}$ and setting it to zero gives

$$\begin{aligned} 2\mathbf{B}\mathbf{a} - 2\lambda\mathbf{W}\mathbf{a} &= 0 \\ \mathbf{B}\mathbf{a} &= \lambda\mathbf{W}\mathbf{a} \\ \mathbf{W}^{-1}\mathbf{B}\mathbf{a} &= \lambda\mathbf{a}. \end{aligned}$$

From the final equation we recognise that $\mathbf{v} = \mathbf{a}$ is an **eigenvector** of $\mathbf{W}^{-1}\mathbf{B}$, and λ is the corresponding **eigenvalue**.

The equation has in general rank($\mathbf{W}^{-1}\mathbf{B}$) solutions. In the case of two classes, the rank equals 1 and thus only one solution exists.

- A training data set is used for the calculation of $\mathbf{W}$ and $\mathbf{B}$. $\longrightarrow$ This gives the eigenvector $\mathbf{v}$
- The training data is also used for the calculation of the centroids of the classes (e.g. the sample means, say $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$). $\longrightarrow$ The projected centroids are given by $\bar{\mathbf{x}}_1^t\mathbf{v}$ and $\bar{\mathbf{x}}_2^t\mathbf{v}$.
- A new observation with predictor $\mathbf{x}$ is classified in the class for which the projected centroid is closest to the projected predictor $z = \mathbf{x}^t\mathbf{v}$.

An advantage of this approach is that $\mathbf{v}$ can be interpreted (similar as the loadings in a PCA) in terms of which predictors x_j are important to discriminate between classes 0 and 1.

2.3 More than two classes

When the outcome Y refers to more than two classes, say m classes, then Fisher's method is constructed in exactly the same way. Now

$$\mathbf{W}^{-1}\mathbf{B}\mathbf{a} = \lambda\mathbf{a}$$

will have $r = \text{rank}(\mathbf{W}^{-1}\mathbf{B}) = \min(m-1, p, n)$ solutions (eigenvectors and eigenvalues). (n: sample size of training data)

Let $\mathbf{v}_j$ and λ_j denote the r solutions, and define

- $\mathbf{V}$: $p \times r$ matrix with collums $\mathbf{v}$
- $\mathbf{L}$: $r \times r$ diagonal matrix with elements $\lambda_1 > \lambda_2 > \cdots > \lambda_r$

The p-dimensional predictor data in $\mathbf{X}$ may then be transformed to the r-dimensional scores

$$\mathbf{Z} = \mathbf{X}\mathbf{V}.$$

For eigenvectors $\mathbf{v}_i$ and $\mathbf{v}_j$, it holds that

$$\text{cov}\left[Z_i, Z_j\right] = \text{cov}\left[\mathbf{X}\mathbf{v}_i, \mathbf{X}\mathbf{v}_j\right] = \mathbf{v}_i^t\mathbf{W}\mathbf{v}_j = \delta_{ij},$$

in which the covariances are defined within groups. Hence, within the groups (classes) the scores are uncorrelated.

3 High dimensional predictors

With high-dimensional predictors

- Replace the $p \times p$ matrices $\mathbf{W}$ and $\mathbf{B}$ by their diagonal matrices (i.e. put zeroes on the off-diagonal positions)
- **Sparse LDA** by imposing an L_1-penalty on $\mathbf{v}$.

Two approaches: Zhou *et al.* (2006), Journal of Computational and Graphical Statistics , **15**, 265-286, and Clemmensen *et al.* (2011), Technometrics, **53**.

4 Breast cancer example

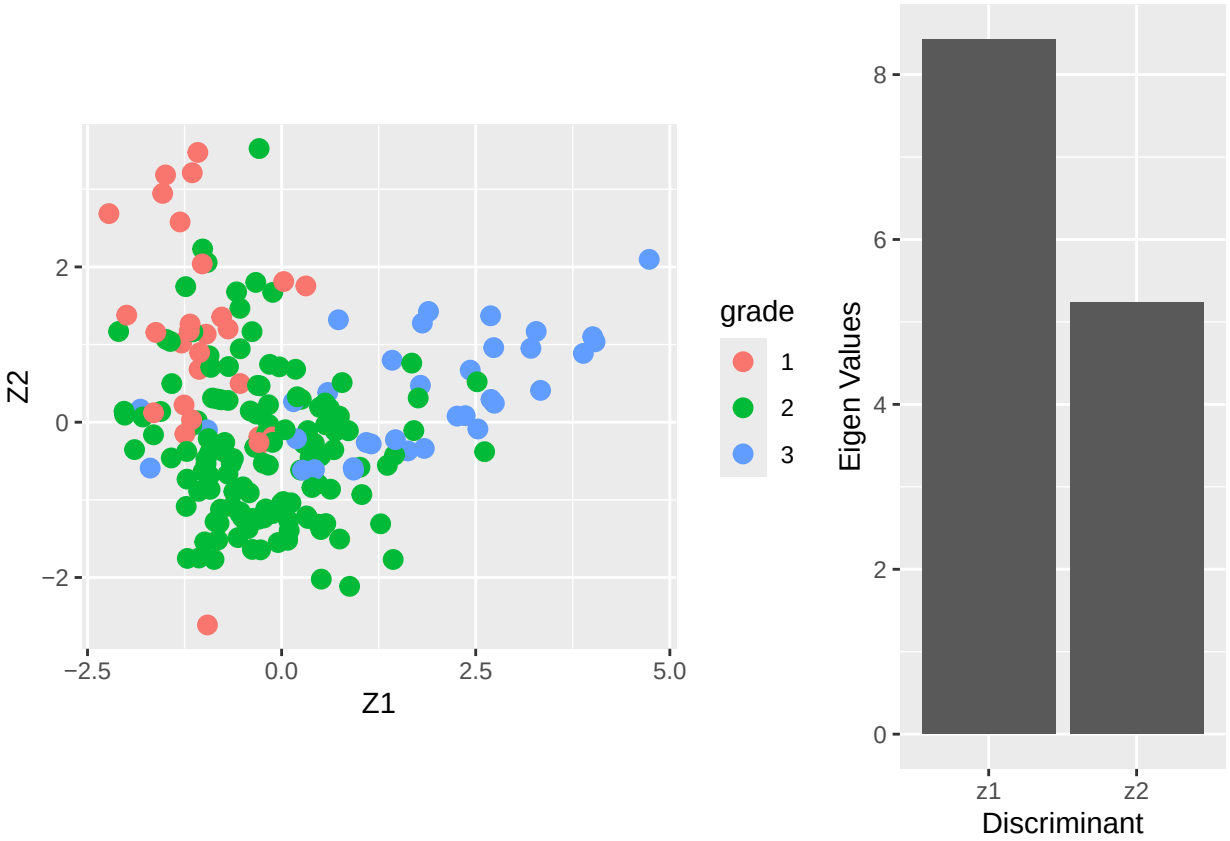
4.1 All genes

4.1.1 LDA

- Fisher's LDA is illustrated on the breast cancer data with all three tumor stages as outcome.
- We try to discriminate between the different stages according to the gene expression data of all genes.
- We cache the result because the calculation takes 10 minutes.

```
breast.lda <- MASS::lda(x = X, grouping = Y)
```

```
Vlda <- breast.lda$scaling
colnames(Vlda) <- paste0("V",1:ncol(Vlda))

Zlda <- X%*%Vlda
colnames(Zlda) <- paste0("Z",1:ncol(Zlda))

grid.arrange(
  Zlda %>%
    as.data.frame %>%
    mutate(grade = Y %>% as.factor) %>%
    ggplot(aes(x= Z1, y = Z2, color = grade)) +
    geom_point(size = 3) +
    coord_fixed(),
  ggplot() +
    geom_bar(aes(x = c("z1","z2"), y = breast.lda$svd), stat = "identity") +
    xlab("Discriminant") +
    ylab("Eigen Values"),
  layout_matrix = matrix(
    c(1,1,2),
    nrow=1)
  )
```



- The columns of the matrix $\mathbf{V}$ contain the eigenvectors. There are $\min(3-2, 22283, 200) = 2$ eigenvectors. The 200×2 matrix $\mathbf{Z}$ contains the scores on the two Fisher discriminants.
- The eigenvalue λ_j can be interpreted as the ratio

$$\frac{\mathbf{v}_j^t \mathbf{B} \mathbf{v}_j}{\mathbf{v}_j^t \mathbf{W} \mathbf{v}_j},$$

  or (upon using $\mathbf{v}_j^t \mathbf{W} \mathbf{v}_j = 1$) the between-centroid sum of squares (in the reduced dimension space of the Fisher discriminants)

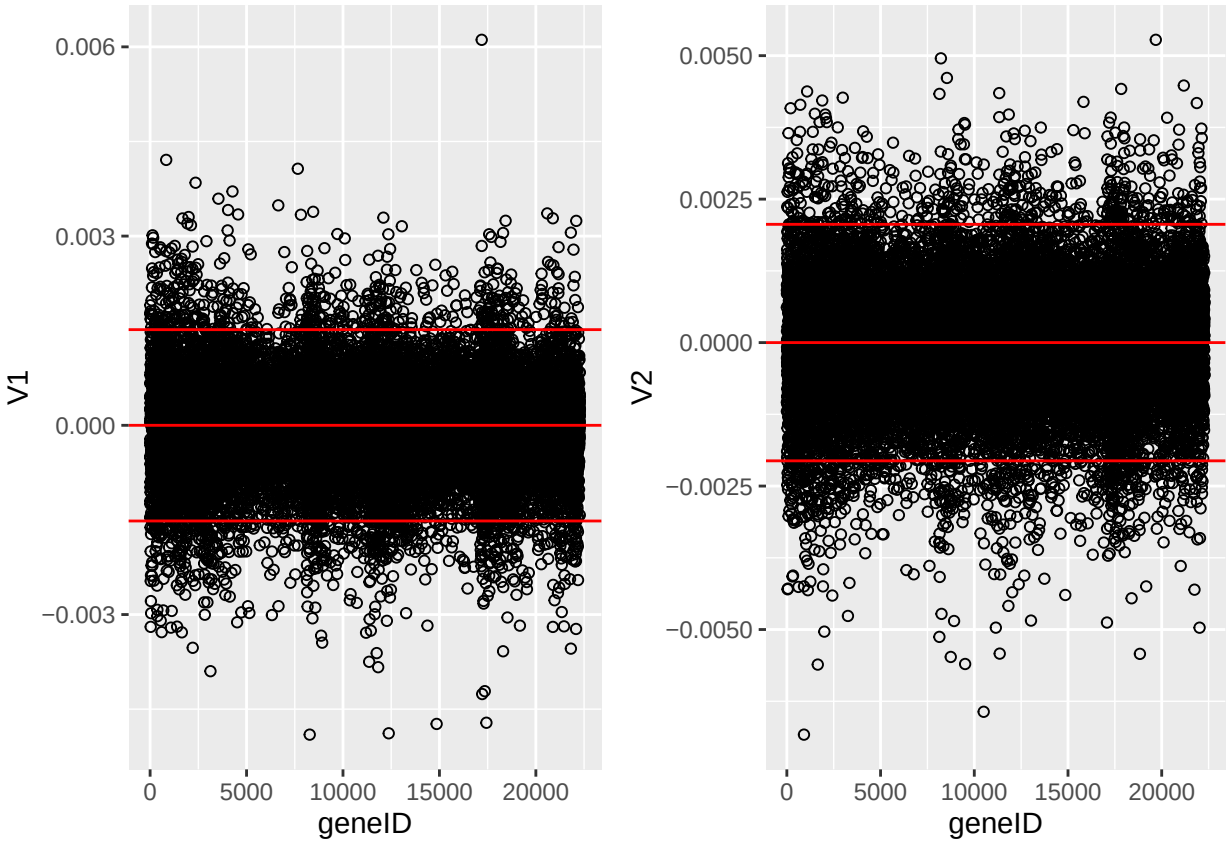
$$\mathbf{v}_j^t \mathbf{B} \mathbf{v}_j.$$

- From the screeplot of the eigenvalues we see that the first dimension is more important than the second (not hugely) in terms of discriminating between the groups.
- From the scatterplot we can see that there is no perfect separation (discrimination) between the three tumor stages (quite some overlap).
- To some extent the first Fisher discriminant dimension discriminates stage 3 (green dots) from the other two stages, and the second dimension separates stage 1 (black dots) from the two others.

4.1.2 Interpretation of loadings

```
grid.arrange(
  Vlda %>%
    as.data.frame %>%
```

```
  mutate(geneID = 1:nrow(Vlda)) %>%
  ggplot(aes(x = geneID, y = V1)) +
  geom_point(pch=21) +
  geom_hline(yintercept = c(-2,0,2)*sd(Vlda[,1]), col = "red"),

Vlda %>%
  as.data.frame %>%
  mutate(geneID = 1:nrow(Vlda)) %>%
  ggplot(aes(x = geneID, y = V2)) +
  geom_point(pch=21) +
  geom_hline(yintercept = c(-2,0,2)*sd(Vlda[,2]), col = "red"),
ncol = 2)
```



The loadings of the Fisher discriminants are within the columns of the **V** matrix.

- Since we have 22283 genes, each discriminant is a linear combination of 22283 gene expression. Instead of looking at the listing of 22283 loadings, we made an index plot (no particular ordering of genes on horizontal axis).
- The red horizontal reference lines correspond to the average of the loading (close to zero) and the average plus and minus twice the standard deviation of the loadings.
- If no genes had any "significant" discriminating power, then we would expect approximately 95% of all loadings within the band. Thus loadings outside of the band are of potential interest and may perhaps be discriminating between the three tumor stages.
- In the graphs presented here we see many loadings within the bands, but also many outside of the band.

We repeat the analysis, but now with the sparse LDA method of Clemmensen et al. (2011).

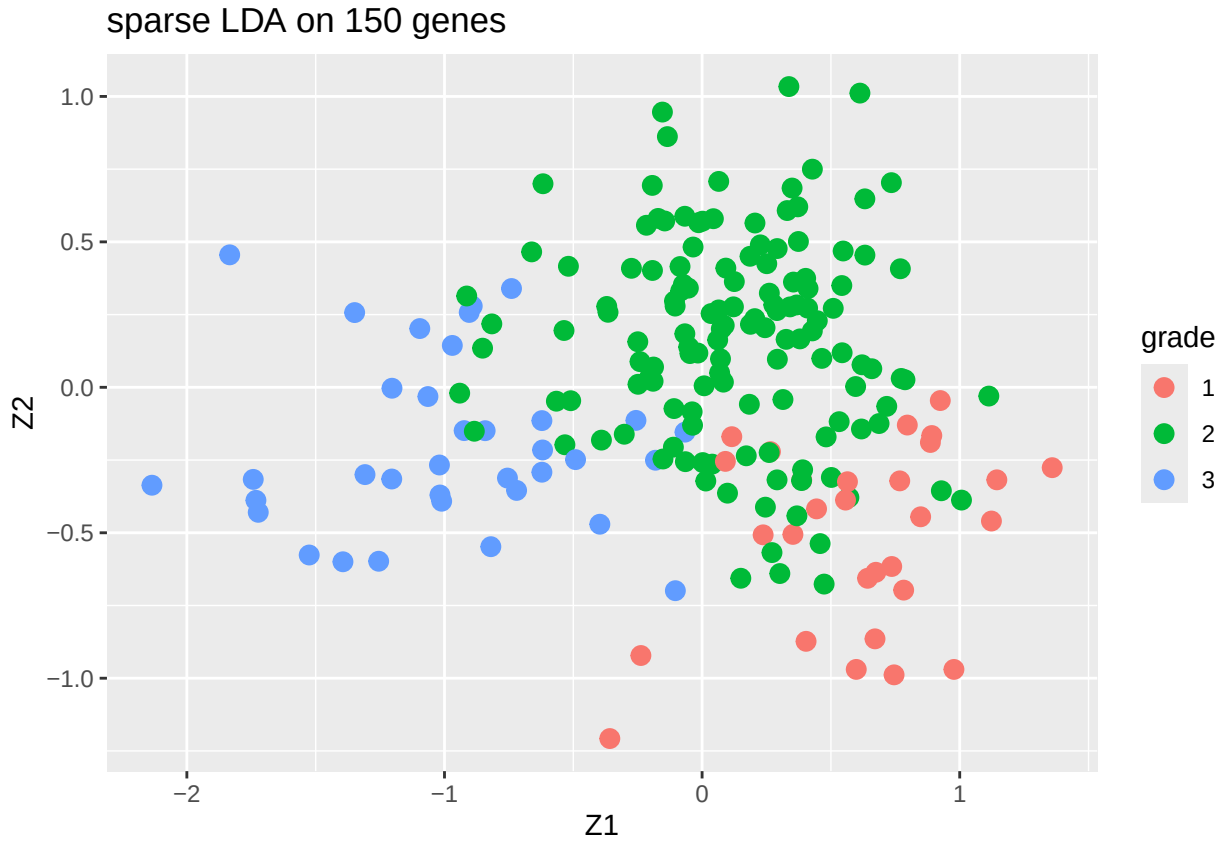
4.2 Sparse LDA based on 150 random genes

- We only present the results of the sparse LDA based on a random subset of 150 genes (ordering of genes in datamatrix is random).
- The discrimination seems better than with classical LDA based on all genes. This is very likely caused by too much noise in the full data matrix with over 20000 predictors.

```
# BiocManager::install("sparseLDA")

library(sparseLDA)

YDummy <- data.frame(
  Y1 = ifelse(Y == 1, 1, 0),
  Y2 = ifelse(Y == 2, 1, 0),
  Y3 = ifelse(Y == 3, 1, 0)
  )

X2 <- X[,1:150]

breast.slda <- sda(x = X2,
  y = as.matrix(YDummy),
  lambda = 1e-6,
  stop = -50,
  maxIte = 25,
  trace = TRUE)
```

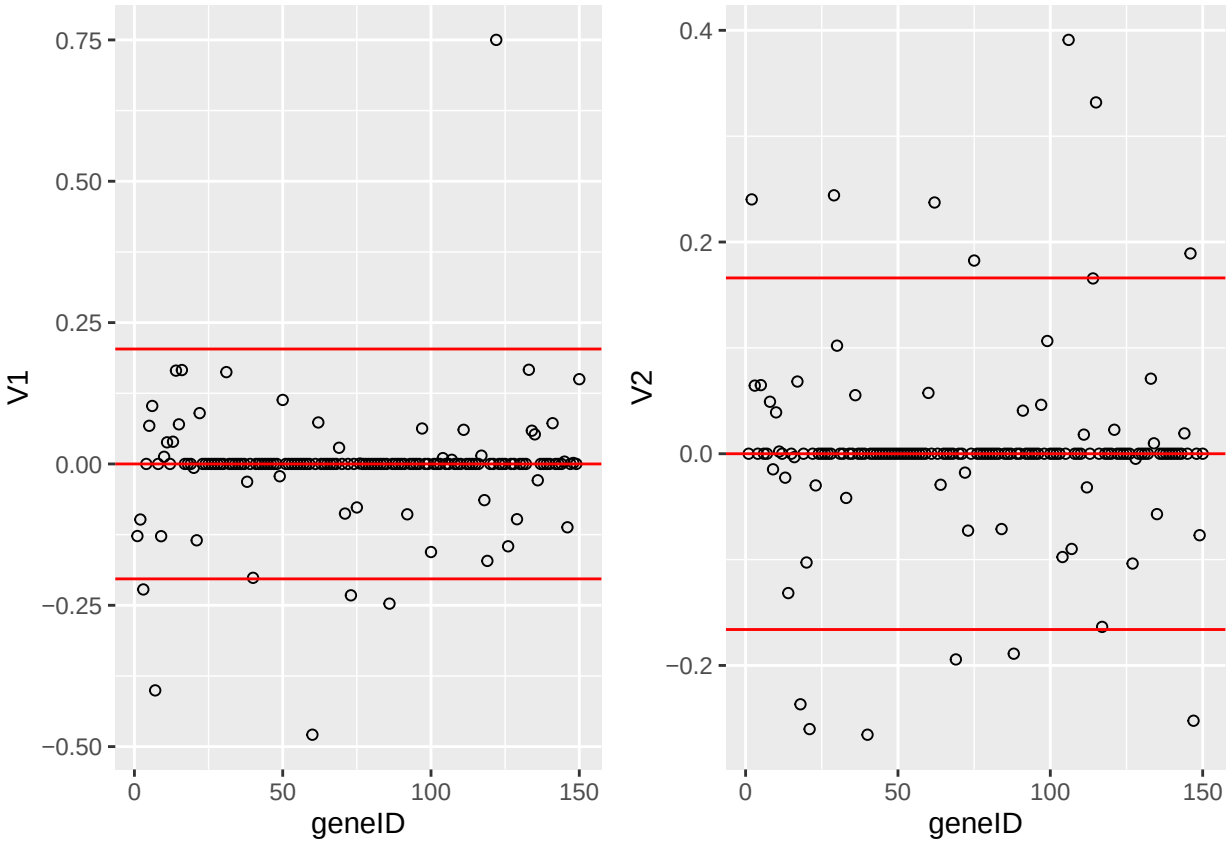
```
#> ite:  1  ridge cost:  122.577  |b|_1:  5.787398
#> ite:  2  ridge cost:  106.3113  |b|_1:  6.571866
#> ite:  3  ridge cost:  100.2607  |b|_1:  5.653801
#> ite:  4  ridge cost:  90.02758  |b|_1:  6.229192
#> ite:  5  ridge cost:  91.62978  |b|_1:  5.511954
#> ite:  6  ridge cost:  90.99971  |b|_1:  5.444292
#> ite:  7  ridge cost:  89.10579  |b|_1:  5.699823
#> ite:  8  ridge cost:  88.16107  |b|_1:  5.843783
#> ite:  9  ridge cost:  88.14556  |b|_1:  5.836932
#> ite:  10  ridge cost:  88.07781  |b|_1:  5.846502
#> ite:  11  ridge cost:  87.9259  |b|_1:  5.874251
#> ite:  12  ridge cost:  87.85233  |b|_1:  5.887925
#> ite:  13  ridge cost:  87.81639  |b|_1:  5.894665
#> ite:  14  ridge cost:  87.79876  |b|_1:  5.897986
#> ite:  15  ridge cost:  87.79008  |b|_1:  5.899624
#> ite:  16  ridge cost:  87.78581  |b|_1:  5.900431
#> ite:  17  ridge cost:  87.78371  |b|_1:  5.900828
#> ite:  18  ridge cost:  87.78267  |b|_1:  5.901025
#> ite:  19  ridge cost:  87.78216  |b|_1:  5.901121
#> ite:  20  ridge cost:  87.78191  |b|_1:  5.901169
#> ite:  21  ridge cost:  87.78178  |b|_1:  5.901192
#> ite:  22  ridge cost:  87.78172  |b|_1:  5.901204
#> ite:  1  ridge cost:  134.7777  |b|_1:  5.379259
#> ite:  2  ridge cost:  134.7777  |b|_1:  5.379259
#> final update, total ridge cost:  222.5595  |b|_1:  11.28046
```

```
Vsda <- matrix(0, nrow=ncol(X2), ncol=2)
Vsda[breast.slda$varIndex,] <- breast.slda$beta
colnames(Vsda) <- paste0("V",1:ncol(Vsda))

Zsda <- X2%*%Vsda
colnames(Zsda) <- paste0("Z",1:ncol(Zsda))

Zsda %>%
 as.data.frame %>%
 mutate(grade = Y %>% as.factor) %>%
 ggplot(aes(x= Z1, y = Z2, color = grade)) +
 geom_point(size = 3) +
 ggtitle("sparse LDA on 150 genes")
```




```
grid.arrange(
   Vsda %>%
     as.data.frame %>%
     mutate(geneID = 1:nrow(Vsda)) %>%
     ggplot(aes(x = geneID, y = V1)) +
     geom_point(pch=21) +
     geom_hline(yintercept = c(-2,0,2)*sd(Vsda[,1]), col = "red") ,

   Vsda %>%
     as.data.frame %>%
     mutate(geneID = 1:nrow(Vsda)) %>%
     ggplot(aes(x = geneID, y = V2)) +
```

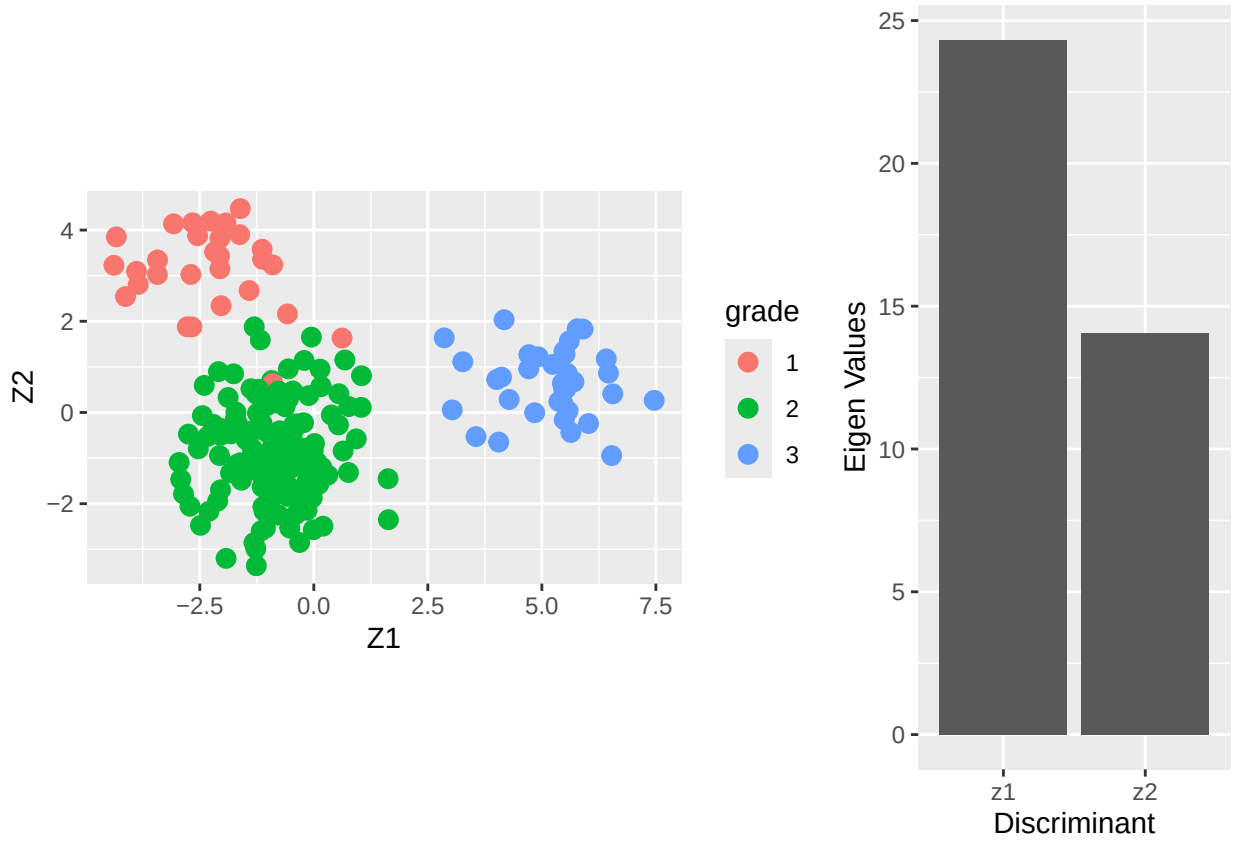
```
    geom_point(pch=21) +
    geom_hline(yintercept = c(-2,0,2)*sd(Vsda[,2]), col = "red"),
  ncol = 2)
```



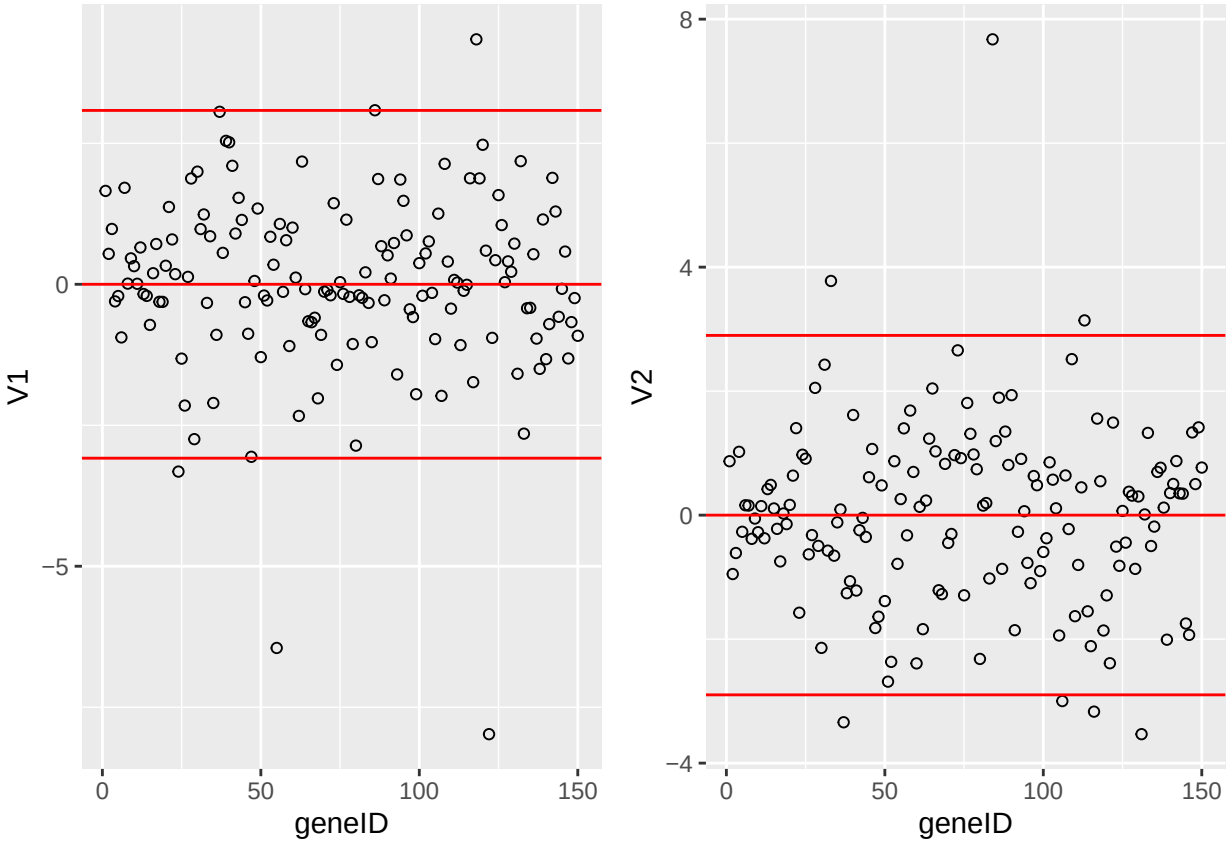
4.3 LDA based on 150 random genes

```
breast.lda150 <- MASS::lda(x = X2, grouping = Y)

Vlda <- breast.lda150$scaling
colnames(Vlda) <- paste0("V",1:ncol(Vlda))

Zlda <- X2%*%Vlda
colnames(Zlda) <- paste0("Z",1:ncol(Zlda))

grid.arrange(
  Zlda %>%
    as.data.frame %>%
    mutate(grade = Y %>% as.factor) %>%
    ggplot(aes(x= Z1, y = Z2, color = grade)) +
    geom_point(size = 3) +
    coord_fixed(),
  ggplot() +
    geom_bar(aes(x = c("z1","z2"), y = breast.lda150$svd), stat = "identity") +
    xlab("Discriminant") +
    ylab("Eigen Values"),
```

```
layout_matrix = matrix(
  c(1,1,2),
  nrow=1)
)
```




```
grid.arrange(
  Vlda %>%
    as.data.frame %>%
    mutate(geneID = 1:nrow(Vlda)) %>%
    ggplot(aes(x = geneID, y = V1)) +
    geom_point(pch=21) +
    geom_hline(yintercept = c(-2,0,2)*sd(Vlda[,1]), col = "red"),

  Vlda %>%
    as.data.frame %>%
    mutate(geneID = 1:nrow(Vlda)) %>%
    ggplot(aes(x = geneID, y = V2)) +
    geom_point(pch=21) +
    geom_hline(yintercept = c(-2,0,2)*sd(Vlda[,2]), col = "red"),
  ncol = 2)
```



4.4 Wrapup

LDA on all 22283 genes gave poorer result than on 150 genes. This is probably caused by numerical instability when working with large **W** and **B** matrices

- Sparse LDA gave slightly poorer result than LDA on the subset of 150 genes. This may be caused by overfitting of the LDA.
- When (sparse) LDA is used to build a prediction model/classifier, then CV methods, or splitting of dataset into training and test datasets should be used to allow for an honest evaluation of the final prediction model.
- The graphs in the first two Fisher discriminant dimensions shown on the previous slides should only be used for data exploration.
- When the objective is to try to understand differences between groups in a high dimensional space, Fisher LDA is preferred over PCA.

Acknowledgement

- Olivier Thas for sharing his materials of Analysis of High Dimensional Data 2019-2020, which I used as the starting point for this chapter.

Session info

Session info

```
#> [1] "2024-10-02 16:21:45 CEST"
#> - Session info ---------------------------------------------------------------
#>  setting  value
#>  version  R version 4.4.0 RC (2024-04-16 r86468)
#>  os       macOS Big Sur 11.6
#>  system   aarch64, darwin20
#>  ui       X11
#>  language (EN)
#>  collate  en_US.UTF-8
#>  ctype    en_US.UTF-8
#>  tz       Europe/Brussels
#>  date     2024-10-02
#>  pandoc   3.1.1 @ /Applications/RStudio.app/Contents/Resources/app/quarto/bin/tools/ (via rmarkdown)
#>
#> - Packages -------------------------------------------------------------------
#>  package           * version    date (UTC) lib source
#>  AIMS              * 1.36.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  AnnotationDbi       1.66.0     2024-05-01 [1] Bioconductor 3.19 (R 4.4.0)
#>  Biobase           * 2.64.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  BiocFileCache       2.12.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  BiocGenerics      * 0.50.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  biomaRt           * 2.60.1     2024-06-26 [1] Bioconductor 3.19 (R 4.4.0)
#>  Biostrings          2.72.1     2024-06-02 [1] Bioconductor 3.19 (R 4.4.0)
#>  bit                 4.5.0      2024-09-20 [1] CRAN (R 4.4.1)
#>  bit64               4.5.2      2024-09-22 [1] CRAN (R 4.4.1)
#>  blob                1.2.4      2023-03-17 [1] CRAN (R 4.4.0)
#>  bookdown            0.40       2024-07-02 [1] CRAN (R 4.4.0)
#>  bootstrap           2019.6     2019-06-17 [1] CRAN (R 4.4.0)
#>  breastCancerMAINZ * 1.42.0     2024-05-02 [1] Bioconductor 3.19 (R 4.4.0)
#>  bslib               0.8.0      2024-07-29 [1] CRAN (R 4.4.0)
#>  cachem              1.1.0      2024-05-16 [1] CRAN (R 4.4.0)
#>  class               7.3-22     2023-05-03 [1] CRAN (R 4.4.0)
#>  cli                 3.6.3      2024-06-21 [1] CRAN (R 4.4.0)
#>  cluster             2.1.6      2023-12-01 [1] CRAN (R 4.4.0)
#>  codetools           0.2-20     2024-03-31 [1] CRAN (R 4.4.0)
#>  colorspace          2.1-1      2024-07-26 [1] CRAN (R 4.4.0)
#>  crayon              1.5.3      2024-06-20 [1] CRAN (R 4.4.0)
#>  curl                5.2.3      2024-09-20 [1] CRAN (R 4.4.1)
#>  data.table          1.16.0     2024-08-27 [1] CRAN (R 4.4.1)
#>  DBI                 1.2.3      2024-06-02 [1] CRAN (R 4.4.0)
#>  dbplyr              2.5.0      2024-03-19 [1] CRAN (R 4.4.0)
#>  digest              0.6.37     2024-08-19 [1] CRAN (R 4.4.1)
#>  dplyr             * 1.1.4      2023-11-17 [1] CRAN (R 4.4.0)
#>  e1071             * 1.7-16     2024-09-16 [1] CRAN (R 4.4.1)
#>  elasticnet          1.3        2020-05-15 [1] CRAN (R 4.4.0)
#>  evaluate            1.0.0      2024-09-17 [1] CRAN (R 4.4.1)
#>  fansi               1.0.6      2023-12-08 [1] CRAN (R 4.4.0)
#>  farver              2.1.2      2024-05-13 [1] CRAN (R 4.4.0)
#>  fastmap             1.2.0      2024-05-15 [1] CRAN (R 4.4.0)
#>  filelock            1.0.3      2023-12-11 [1] CRAN (R 4.4.0)
#>  fontawesome         0.5.2      2023-08-19 [1] CRAN (R 4.4.0)
#>  forcats           * 1.0.0      2023-01-29 [1] CRAN (R 4.4.0)
#>  future              1.34.0     2024-07-29 [1] CRAN (R 4.4.0)
```

```
#>  future.apply        1.11.2     2024-03-28 [1] CRAN (R 4.4.0)
#>  genefu            * 2.36.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  generics            0.1.3      2022-07-05 [1] CRAN (R 4.4.0)
#>  GenomeInfoDb        1.40.1     2024-06-16 [1] Bioconductor 3.19 (R 4.4.0)
#>  GenomeInfoDbData    1.2.12     2024-04-24 [1] Bioconductor
#>  ggplot2           * 3.5.1      2024-04-23 [1] CRAN (R 4.4.0)
#>  globals             0.16.3     2024-03-08 [1] CRAN (R 4.4.0)
#>  glue                1.8.0      2024-09-30 [1] CRAN (R 4.4.1)
#>  gridExtra         * 2.3        2017-09-09 [1] CRAN (R 4.4.0)
#>  gtable              0.3.5      2024-04-22 [1] CRAN (R 4.4.0)
#>  highr               0.11       2024-05-26 [1] CRAN (R 4.4.0)
#>  hms                 1.1.3      2023-03-21 [1] CRAN (R 4.4.0)
#>  htmltools           0.5.8.1    2024-04-04 [1] CRAN (R 4.4.0)
#>  httr                1.4.7      2023-08-15 [1] CRAN (R 4.4.0)
#>  httr2               1.0.5      2024-09-26 [1] CRAN (R 4.4.1)
#>  iC10              * 2.0.2      2024-07-19 [1] CRAN (R 4.4.0)
#>  iC10TrainingData    2.0.1      2024-07-16 [1] CRAN (R 4.4.0)
#>  impute              1.78.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  IRanges             2.38.1     2024-07-03 [1] Bioconductor 3.19 (R 4.4.1)
#>  jquerylib           0.1.4      2021-04-26 [1] CRAN (R 4.4.0)
#>  jsonlite            1.8.9      2024-09-20 [1] CRAN (R 4.4.1)
#>  KEGGREST            1.44.1     2024-06-19 [1] Bioconductor 3.19 (R 4.4.0)
#>  KernSmooth          2.23-24    2024-05-17 [1] CRAN (R 4.4.0)
#>  knitr               1.48       2024-07-07 [1] CRAN (R 4.4.0)
#>  labeling            0.4.3      2023-08-29 [1] CRAN (R 4.4.0)
#>  lars                1.3        2022-04-13 [1] CRAN (R 4.4.0)
#>  lattice             0.22-6     2024-03-20 [1] CRAN (R 4.4.0)
#>  lava                1.8.0      2024-03-05 [1] CRAN (R 4.4.0)
#>  lifecycle           1.0.4      2023-11-07 [1] CRAN (R 4.4.0)
#>  limma               3.60.5     2024-09-29 [1] Bioconductor 3.19 (R 4.4.1)
#>  listenv             0.9.1      2024-01-29 [1] CRAN (R 4.4.0)
#>  lubridate         * 1.9.3      2023-09-27 [1] CRAN (R 4.4.0)
#>  magrittr            2.0.3      2022-03-30 [1] CRAN (R 4.4.0)
#>  MASS                7.3-61     2024-06-13 [1] CRAN (R 4.4.0)
#>  Matrix              1.7-0      2024-03-22 [1] CRAN (R 4.4.0)
#>  mclust              6.1.1      2024-04-29 [1] CRAN (R 4.4.0)
#>  mda                 0.5-4      2023-06-23 [1] CRAN (R 4.4.0)
#>  memoise             2.0.1      2021-11-26 [1] CRAN (R 4.4.0)
#>  munsell             0.5.1      2024-04-01 [1] CRAN (R 4.4.0)
#>  pamr                1.57       2024-07-01 [1] CRAN (R 4.4.0)
#>  parallelly          1.38.0     2024-07-27 [1] CRAN (R 4.4.0)
#>  pillar              1.9.0      2023-03-22 [1] CRAN (R 4.4.0)
#>  pkgconfig           2.0.3      2019-09-22 [1] CRAN (R 4.4.0)
#>  png                 0.1-8      2022-11-29 [1] CRAN (R 4.4.0)
#>  prettyunits         1.2.0      2023-09-24 [1] CRAN (R 4.4.0)
#>  prodlim           * 2024.06.25 2024-06-24 [1] CRAN (R 4.4.0)
#>  progress            1.2.3      2023-12-06 [1] CRAN (R 4.4.0)
#>  proxy               0.4-27     2022-06-09 [1] CRAN (R 4.4.0)
#>  purrr             * 1.0.2      2023-08-10 [1] CRAN (R 4.4.0)
#>  R6                  2.5.1      2021-08-19 [1] CRAN (R 4.4.0)
#>  rappdirs            0.3.3      2021-01-31 [1] CRAN (R 4.4.0)
#>  Rcpp                1.0.13     2024-07-17 [1] CRAN (R 4.4.0)
#>  readr             * 2.1.5      2024-01-10 [1] CRAN (R 4.4.0)
#>  rlang               1.1.4      2024-06-04 [1] CRAN (R 4.4.0)
```

```
#>  rmarkdown           2.28       2024-08-17 [1] CRAN (R 4.4.0)
#>  rmeta               3.0        2018-03-20 [1] CRAN (R 4.4.0)
#>  RSQLite             2.3.7      2024-05-27 [1] CRAN (R 4.4.0)
#>  rstudioapi          0.16.0     2024-03-24 [1] CRAN (R 4.4.0)
#>  S4Vectors           0.42.1     2024-07-03 [1] Bioconductor 3.19 (R 4.4.1)
#>  sass                0.4.9      2024-03-15 [1] CRAN (R 4.4.0)
#>  scales              1.3.0      2023-11-28 [1] CRAN (R 4.4.0)
#>  sessioninfo         1.2.2      2021-12-06 [1] CRAN (R 4.4.0)
#>  sparseLDA         * 0.1-9      2016-09-22 [1] CRAN (R 4.4.0)
#>  statmod             1.5.0      2023-01-06 [1] CRAN (R 4.4.0)
#>  stringi             1.8.4      2024-05-06 [1] CRAN (R 4.4.0)
#>  stringr           * 1.5.1      2023-11-14 [1] CRAN (R 4.4.0)
#>  SuppDists           1.1-9.8    2024-09-03 [1] CRAN (R 4.4.1)
#>  survcomp          * 1.54.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  survival          * 3.7-0      2024-06-05 [1] CRAN (R 4.4.0)
#>  survivalROC         1.0.3.1    2022-12-05 [1] CRAN (R 4.4.0)
#>  tibble            * 3.2.1      2023-03-20 [1] CRAN (R 4.4.0)
#>  tidyr             * 1.3.1      2024-01-24 [1] CRAN (R 4.4.0)
#>  tidyselect          1.2.1      2024-03-11 [1] CRAN (R 4.4.0)
#>  tidyverse         * 2.0.0      2023-02-22 [1] CRAN (R 4.4.0)
#>  timechange          0.3.0      2024-01-18 [1] CRAN (R 4.4.0)
#>  tzdb                0.4.0      2023-05-12 [1] CRAN (R 4.4.0)
#>  UCSC.utils          1.0.0      2024-05-06 [1] Bioconductor 3.19 (R 4.4.0)
#>  utf8                1.2.4      2023-10-22 [1] CRAN (R 4.4.0)
#>  vctrs               0.6.5      2023-12-01 [1] CRAN (R 4.4.0)
#>  withr               3.0.1      2024-07-31 [1] CRAN (R 4.4.0)
#>  xfun                0.47       2024-08-17 [1] CRAN (R 4.4.0)
#>  xml2                1.3.6      2023-12-04 [1] CRAN (R 4.4.0)
#>  XVector             0.44.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>  yaml                2.3.10     2024-07-26 [1] CRAN (R 4.4.0)
#>  zlibbioc            1.50.0     2024-04-30 [1] Bioconductor 3.19 (R 4.4.0)
#>
#>  [1] /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library
#>
#> ------------------------------------------------------------------------------
```