Variable selection in microbiome compositional data analysis: tutorial

Antoni Susin[1], Yiwen Wang[2], Kim-Anh Lê Cao[2], M.Luz Calle[3]
[1]Mathematical Department, UPC-Barcelona Tech, Barcelona, Spain
[2]Melbourne Integrative Genomics, School of Mathematics and Statistics, The University of Melbourne, Parkville VIC, Australia
[3]Biosciences Department, Faculty of Sciences and Technology, University of Vic - Central University of Catalonia, Vic, Spain

2020-03-28

Contents

Chapter 1

Introduction

This vignette supports the paper "Variable selection in microbiome compositional data analysis" by Susin *et al.* (2020) that assesses three compositional data analysis (CoDA) algorithms for microbiome variable selection:

- *selbal*: a forward selection method for the identification of two groups of taxa whose balance is most associated with the response variable (Rivera-Pinto et al., 2018).
- *clr-lasso*: penalized regression after the centered log-ratio (clr) transformation (Zou and Hastie, 2005; Tibshirani, 1996; Le Cessie and Van Houwelingen, 1992);
- *coda-lasso*: penalized log-contrast regression (log-transformed abundances and a zero-sum constraint on the regression coefficients) (Lu et al., 2019; Lin et al., 2014);

Among them, *coda-lasso* is not yet available as an R package, but the R code for implementing the algorithm is available on Github: https://github.com/UVic-omics/CoDA-Penalized-Regression. Therefore, let us copy the repository first. We only need to copy once, after that, we can update it by fetching the last modified version.

```
# copy the repository from https://github.com/UVic-omics/CoDA-Penalized-Regression
# system('git clone https://github.com/UVic-omics/CoDA-Penalized-Regression')

# fetch the last modified repository from
# https://github.com/UVic-omics/CoDA-Penalized-Regression
system('git pull https://github.com/UVic-omics/CoDA-Penalized-Regression')
```

This vignette only displays the application of all methods on the case studies. Paper related codes and datasets including simulations are all available on GitHub: https://github.com/UVic-omics/Microbiome-Variable-Selection

1.1 Packages installation and loading

Install then load the following packages:

```
# cran.packages <- c('knitr', 'glmnet', 'ggplot2', 'gridExtra',
#                    'UpSetR', 'ggforce')
# install.packages(cran.packages)
# devtools::install_github(repo = 'UVic-omics/selbal')

library(knitr) # rbookdown, kable
library(glmnet) # glmnet
library(selbal) # selbal
library(ggplot2) # draw selbal
library(gridExtra) # grid.arrange
library(UpSetR) # upset
library(ggforce) # selbal-like plot
library(grid) # grid.draw
# source coda-lasso functions
source(file = './CoDA-Penalized-Regression/R/functions_coda_penalized_regression.R')

# build in functions
source(file = 'functions.R')
```

1.2 Example datasets

1.2.1 Crohn's disease

Crohn's disease (CD) is an inflammatory bowel disease that has been linked to microbial alterations in the gut. The pediatric CD study (Gevers et al., 2014) includes 975 individuals from 662 patients with Crohn's disease and 313 without any symptoms. The processed data, from 16S rRNA gene sequencing after QIIME 1.7.0, were downloaded from Qiita (Gonzalez et al., 2018) study ID 1939. The OTU table was agglomerated to the genus level, resulting in a matrix with 48 genera and 975 samples (see Table 1.1).

Load the data as follows:

```
load('./datasets/Crohn_data.RData')
```

File "Crohn_data.RData" contains three objects:

x_Crohn: the abundance table, a data frame of counts with 975 rows (individuals) and 48 columns (genera)

```
class(x_Crohn)
```

```
## [1] "data.frame"
```

```
dim(x_Crohn)
```

```
## [1] 975  48
```

y_Crohn: a factor variable, indicator of disease status (CD vs. not CD)

```
class(y_Crohn)
```

```
## [1] "factor"
```

```
summary(y_Crohn)
```

```
##  CD  no
## 662 313
```

y_Crohn_numeric: a numerical variable with values 1 (CD) and 0 (not CD)

```
class(y_Crohn_numeric)
```

```
## [1] "numeric"
```

```
table(y_Crohn_numeric)
```

```
## y_Crohn_numeric
##   1   2
## 662 313
```

Note: **x_Crohn** contains no zero. The original matrix of counts (**X**) was transformed by adding one count to each matrix cell: $x\_Crohn = X + 1$. The original matrix of counts can easily be recovered and other imputation methods can be applied.

1.2.2 High fat high sugar diet in mice

The study was conducted by Dr Lê Cao at the University of Queensland Diamantina Institute that investigated the effect of diet in mice. C57/B6 female black mice were housed in cages (3 animals per cage and fed with a high fat high sugar diet (HFHS) or a normal diet). Stool sampling was performed at Day 0, 1, 4 and 7. Illumina MiSeq sequencing was used to obtain the 16S rRNA sequencing data. The sequencing data were then processed with QIIME 1.9.0. For our analysis, we considered Day 1 only (HFHSday1 data). The OTU (Operational Taxonomy Units) table after OTU filtering included 558 taxa and 47 samples (24 HFHS diet and 23 normal diet) (see Table 1.1). Taxonomy information is also available and reported here.

```
load('./datasets/HFHSday1.RData')
```

File "HFHSday1.RData" contains three objects:

x_HFHSday1: the abundance table, a matrix of proportions with 47 rows (samples) and 558 columns (OTUs)

```
class(x_HFHSday1)
```

```
## [1] "matrix"
```

```
dim(x_HFHSday1)
```

```
## [1]  47 558
```

y_HFHSday1: a factor variable, indicator of diet (HFHS vs. normal)

```
class(y_HFHSday1)
```

```
## [1] "factor"
```

```
summary(y_HFHSday1)
```

```
##   HFHS Normal
##     24     23
```

taxonomy_HFHS: taxonomy table

Note: **x_HFHSday1** contains no zero. Zero imputation was performed on the original abundance matrix.

Table 1.1: **A summary of the number of samples and number of taxa in each case study**

Crohn data		HFHSday1 data	
No. of genera	48	No. of OTUs	558
No. of samples	975	No. of samples	47
No. of patients with CD	662	No. of mice with HFHS diet	24
No. of healthy patients	313	No. of mice with normal diet	23

Chapter 2

Selbal: selection of balances

Selbal is a forward selection algorithm for the identification of two groups of variables whose balance is most associated with the response variable (Rivera-Pinto et al., 2018). *Selbal* R package is available on GitHub (https://github.com/UVic-omics/selbal) and can be installed with *devtools*:

for non-Windows users:

```
devtools::install_github(repo = "UVic-omics/selbal")
```

for Windows users:

```
devtools::install_url(url="https://github.com/UVic-omics/selbal/archive/master.zip",
                      INSTALL_opt= "--no-multiarch")
```

For a detailed description of *selbal* see the vignette: https://htmlpreview.github.io/?https://github.com/UVic-omics/selbal/blob/master/vignettes/vignette.html

We generated a wrapper function called *selbal_wrapper()* that will help us to handle the output of *selbal*. *selbal_wrapper()* will call *selbal()* function within the wrapper. The *selbal_wrapper()* function is uploaded via **functions.R**.

2.1 Crohn case study

For binary outcomes, *selbal()* requires that dependent variable **Y** is given as a factor and it implements logistic regression. If **Y** is numeric, *selbal()* implements linear regression.

The dependent variable in Crohn dataset is a factor:

```
class(y_Crohn)
```

```
## [1] "factor"
```

The performance measure (**logit.acc**) of the selected balance for binary outcomes is the **AUC** (default) or the proportion of explained deviance (**Dev**). For comparison with the other methods we will use **Dev** and will set the maximum number of variables (**maxV**) to be selected equal to 12 (**maxV = 12**).

```
selbal_Crohn <- selbal(x = x_Crohn, y = y_Crohn, maxV = 12,
                       logit.acc = 'Dev', draw = F)
```

The output of *selbal()* is a list and we can get the different elements of the list by indexing.

To visualise the results of *selbal*, we recommend the new balance representation (**global.plot2**):

```
# dev.off() # clean plots window when you run in Console
grid.draw(selbal_Crohn$global.plot2)
```

selbal() is the original *selbal* function. To improve the readability of codes and to compare more easily with the other two methods, we use *selbal_wrapper()* with the same input as *selbal()*. *selbal_wrapper()* will call *selbal()* function within the wrapper. Thus in this tutorial, we use *selbal_wrapper()* instead of the original *selbal()* function:

```
Crohn.results_selbal <- selbal_wrapper(X = x_Crohn, Y = y_Crohn,
                                       maxV = 12, logit.acc = 'Dev')
```

The number of selected variables:

```
Crohn.results_selbal$numVarSelect
```

```
## [1] 12
```

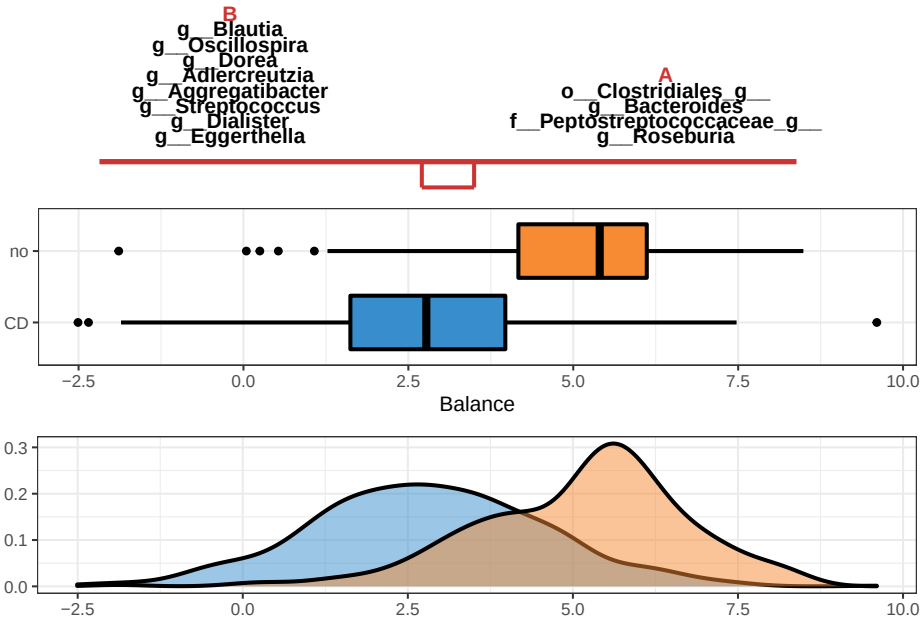
The names of selected variables:

```
Crohn.results_selbal$varSelect
```

```
##  [1] "g__Roseburia"                 "g__Eggerthella"
##  [3] "g__Dialister"                 "g__Streptococcus"
##  [5] "f__Peptostreptococcaceae_g__" "g__Bacteroides"
##  [7] "g__Aggregatibacter"           "g__Adlercreutzia"
##  [9] "g__Dorea"                     "g__Oscillospira"
## [11] "o__Clostridiales_g__"         "g__Blautia"
```

For visualisation, we can use *selbal_like_plot()* which can also be used in other two methods (see Chapter 5).

```
Crohn.selbal_pos <- Crohn.results_selbal$posVarSelect
Crohn.selbal_neg <- Crohn.results_selbal$negVarSelect
selbal_like_plot(pos.names = Crohn.selbal_pos, neg.names = Crohn.selbal_neg,
                 Y = y_Crohn, selbal = TRUE,
                 FINAL.BAL = Crohn.results_selbal$finalBal)
```



2.2 HFHS-Day1 case study

The analysis on HFHSday1 data is similar to Crohn data.

First, we need to check if the dependent variable **Y** is a factor.

```
class(y_HFHSday1)
```

```
## [1] "factor"
```

In HFHSday1 data, we use *selbal_wrapper()* directly and set the maximum number of variables to be selected equal to 2 (**maxV = 2**):

```
HFHS.results_selbal <- selbal_wrapper(X = x_HFHSday1, Y = y_HFHSday1,
                                      maxV = 2, logit.acc = 'Dev')
```

The number of selected variables:

```
HFHS.results_selbal$numVarSelect
```

```
## [1] 2
```

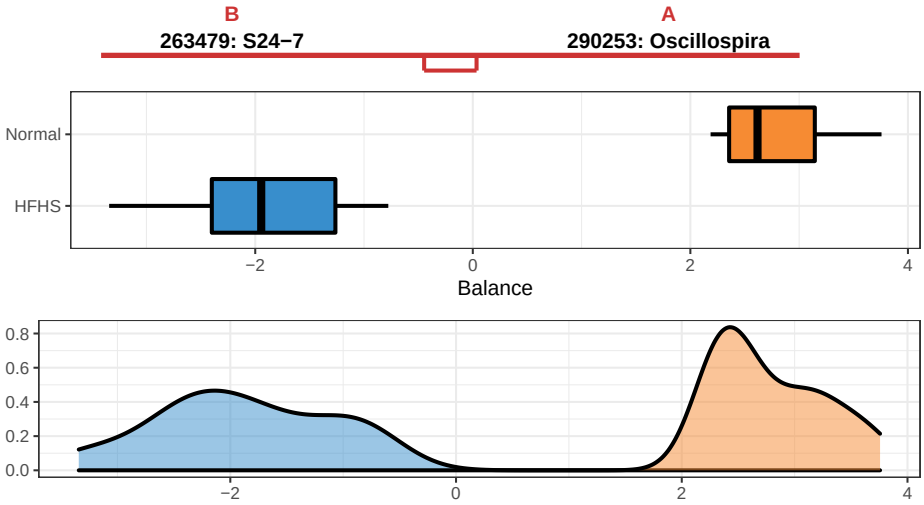
The names of selected variables:

```
HFHS.results_selbal$varSelect
```

```
## [1] "290253" "263479"
```

For visualisation, we then use *selbal_like_plot()*.

```
HFHS.selbal_pos <- HFHS.results_selbal$posVarSelect
HFHS.selbal_neg <- HFHS.results_selbal$negVarSelect
selbal_like_plot(pos.names = HFHS.selbal_pos, neg.names = HFHS.selbal_neg,
                 Y = y_HFHSday1, selbal = TRUE,
                 FINAL.BAL = HFHS.results_selbal$finalBal,
                 OTU = T, taxa = taxonomy_HFHS)
```



We also extract the taxonomic information of these selected OTUs.

```
HFHS.tax_selbal <- taxonomy_HFHS[which(rownames(taxonomy_HFHS) %in%
                                       HFHS.results_selbal$varSelect), ]
kable(HFHS.tax_selbal[ ,2:6], booktabs = T)
```

	Phylum	Class	Order	Family	Genus
290253	Firmicutes	Clostridia	Clostridiales	Ruminococcaceae	Oscillospira
263479	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	

Chapter 3

clr-lasso

Penalised regression is a powerful approach for variable selection in high dimensional settings (Zou and Hastie, 2005; Tibshirani, 1996; Le Cessie and Van Houwelingen, 1992). It can be adapted to compositional data analysis (CoDA) by previously transforming the compositional data with the centered log-ratio transformation (clr).

Clr-lasso implements penalised regression using the R package *glmnet* after clr transformation. Clr transformation corresponds to centering the log-transformed values:

$$clr(x) = clr(x_1, ..., x_k) = (log(x_1) - M, ..., log(x_k) - M)$$

Where $i = 1, ..., k$ microbial variables, x_k is the counts of variable k, M is the arithmetic mean of the log-transformed values.

$$M = \frac{1}{k} \sum_{i=1}^{k} log(x_i)$$

We also generated a wrapper function called *glmnet_wrapper()* that will call *glmnet()* function within the wrapper and help us to handle the output of *glmnet*. The *glmnet_wrapper()* function is uploaded via **functions.R**.

3.1 Crohn case study

First we perform the clr transformation of the abundance table **x_Crohn** as follows (the log-transformation requires a matrix of positive values and thus the zeros have been previously added with an offset of 1):

```
z_Crohn <- log(x_Crohn)
clrx_Crohn <- apply(z_Crohn, 2, function(x) x - rowMeans(z_Crohn))
```

We implement penalised regression with function *glmnet()*. This function requires the outcome **Y** to be numeric. The file "Crohn_data.RData" contains **y_Crohn_numeric**, a numerical vector of disease status with values 1 (CD) and 0 (not CD).

Penalised regression requires the specification of the penalization parameter λ. The larger the value of λ, the less variables will be selected. In previous analysis of this dataset with *selbal*, a balance with 12 variables was determined optimal to discriminate the CD status (Rivera-Pinto et al., 2018). For ease of comparison, we will specify the penalisation parameter that results in the selection of 12 variables for *clr-lasso*.

To identify the value of λ that corresponds to 12 variables selected we implement *glmnet()* function on the clr transformed values and with the specification that the response family is **binomial**:

```
Crohn.test_clrlasso <- glmnet(x = clrx_Crohn, y = y_Crohn_numeric,
                              family = 'binomial', nlambda = 30)
```

The output of *glmnet()* can be visualised with the selection plot:

```
plot(Crohn.test_clrlasso, xvar = 'lambda', label = T)
```

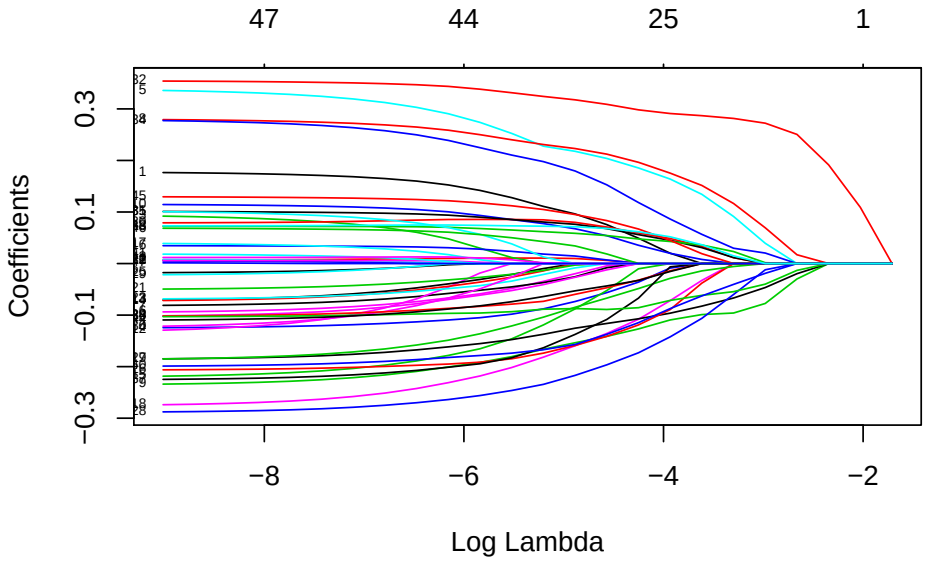


Figure 3.1: Selection plot of Crohn data

In Figure 3.1, each curve corresponds to a variable (i.e. genus). It shows the path of its coefficient against different $log(\lambda)$ values. At each $log(\lambda)$, the shown curves indicate the number of nonzero coefficients. In the plot command, if **label = T**, each curve will be

annotated with variable index.

The numerical output of *glmnet()* will help us to select the value of λ. It provides the number of selected variables or degrees of freedom of the model (Df), the proportion of explained deviance for a sequence of values of λ:

```
Crohn.test_clrlasso
```

```
##
## Call:  glmnet(x = clrx_Crohn, y = y_Crohn_numeric, family = "binomial",      nlambda = 30)
##
##    Df    %Dev   Lambda
## 1   0 0.00000 0.180900
## 2   1 0.05653 0.131600
## 3   1 0.08761 0.095820
## 4   5 0.12170 0.069750
## 5  10 0.16810 0.050770
## 6  14 0.20670 0.036950
## 7  18 0.24110 0.026900
## 8  23 0.26910 0.019580
## 9  25 0.29220 0.014250
## 10 30 0.30930 0.010370
## 11 32 0.32030 0.007551
## 12 36 0.32770 0.005496
## 13 39 0.33330 0.004001
## 14 41 0.33690 0.002912
## 15 44 0.33910 0.002120
## 16 45 0.34040 0.001543
## 17 46 0.34110 0.001123
## 18 46 0.34150 0.000818
## 19 46 0.34170 0.000595
## 20 47 0.34190 0.000433
## 21 47 0.34190 0.000315
## 22 47 0.34190 0.000230
## 23 47 0.34200 0.000167
## 24 47 0.34200 0.000122
```

We can see that the value of λ that will select 12 variables is between 0.037 and 0.051. We perform again *glmnet()* with the specification of a finer sequence of λ (between 0.03 and 0.05 and an increment of 0.001):

```
Crohn.test2_clrlasso <- glmnet(x = clrx_Crohn, y = y_Crohn_numeric,
                               family = 'binomial', lambda = seq(0.03, 0.05, 0.001))
Crohn.test2_clrlasso
```

```
##
## Call:  glmnet(x = clrx_Crohn, y = y_Crohn_numeric, family = "binomial",      lambda = seq(0.03
##
```

```
##    Df   %Dev Lambda
## 1  10 0.1702  0.050
## 2  10 0.1729  0.049
## 3  10 0.1755  0.048
## 4  10 0.1782  0.047
## 5  11 0.1808  0.046
## 6  11 0.1834  0.045
## 7  11 0.1860  0.044
## 8  12 0.1885  0.043
## 9  13 0.1915  0.042
## 10 14 0.1945  0.041
## 11 14 0.1976  0.040
## 12 14 0.2006  0.039
## 13 14 0.2036  0.038
## 14 14 0.2065  0.037
## 15 14 0.2094  0.036
## 16 14 0.2122  0.035
## 17 14 0.2150  0.034
## 18 15 0.2181  0.033
## 19 18 0.2217  0.032
## 20 18 0.2257  0.031
## 21 18 0.2295  0.030
```

According to this result, we select $\lambda = 0.043$ and implement *glmnet()*:

```
clrlasso_Crohn <- glmnet(x = clrx_Crohn, y = y_Crohn_numeric,
                         family = 'binomial', lambda = 0.043)
```

The same as *selbal*, instead of using the original *glmnet()* function, we use function *glmnet_wrapper()* that calls *glmnet()* function within the wrapper to handle the results of *clr-lasso* with the same input as *glmnet()*:

```
Crohn.results_clrlasso <- glmnet_wrapper(X = clrx_Crohn, Y = y_Crohn_numeric,
                                         family = 'binomial', lambda = 0.043)
```

We can get the number of selected variables:

```
Crohn.results_clrlasso$numVarSelect
```

```
## [1] 12
```

and the names of the selected variables:

```
Crohn.results_clrlasso$varSelect
```

```
##  [1] "g__Roseburia"                "g__Bacteroides"
##  [3] "g__Eggerthella"              "f__Peptostreptococcaceae_g__"
##  [5] "g__Dialister"                "g__Streptococcus"
##  [7] "g__Adlercreutzia"            "g__Aggregatibacter"
##  [9] "o__Clostridiales_g__"        "g__Lachnospira"
```

```
## [11] "o__Lactobacillales_g__"       "g__Bilophila"
```

3.2 HFHS-Day1 case study

The analysis on HFHSday1 data is similar to Crohn data. We first perform the clr transformation:

```
z_HFHSday1 <- log(x_HFHSday1)
clrx_HFHSday1 <- apply(z_HFHSday1, 2, function(x) x-rowMeans(z_HFHSday1))
```

The outcome **y_HFHSday1** is converted to a numeric vector.

```
y_HFHSday1_numeric <- as.numeric(y_HFHSday1)
```

We implement *glmnet()* and visualise the results (Figure 3.2).

```
HFHS.test_clrlasso <- glmnet(x = clrx_HFHSday1, y = y_HFHSday1_numeric,
                             family = 'binomial', nlambda = 30)
```

```
plot(HFHS.test_clrlasso, xvar = 'lambda', label = T)
```

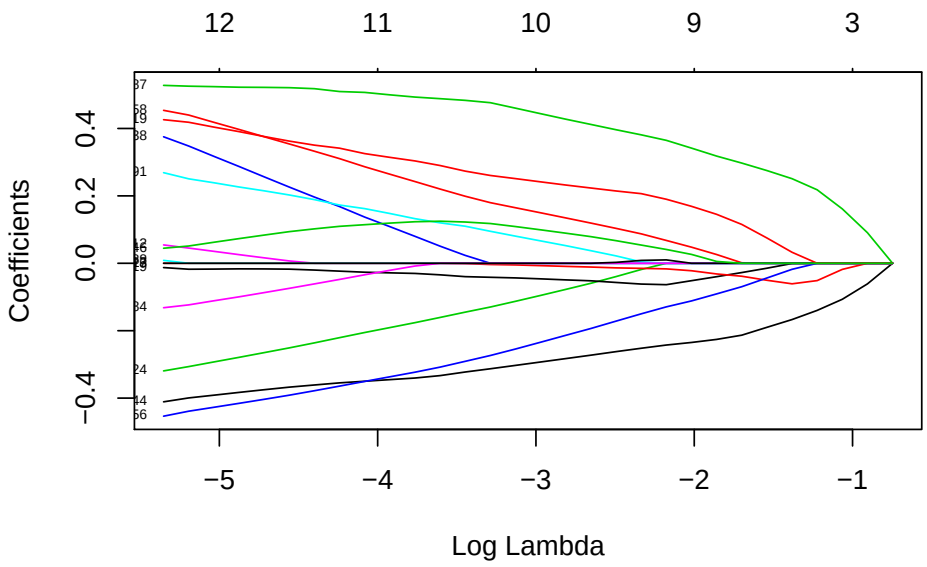


Figure 3.2: Lasso plot of HFHS-Day1 data

The explanation of Figure 3.2 is the same as Figure 3.1.

The numerical output of *glmnet()* will help us to decide the penalty term λ:

```
HFHS.test_clrlasso
```

```
##
## Call:  glmnet(x = clrx_HFHSday1, y = y_HFHSday1_numeric, family = "binomial",      r
##
##    Df   %Dev  Lambda
## 1   0 0.0000 0.47370
## 2   2 0.1862 0.40420
## 3   3 0.3290 0.34480
## 4   3 0.4417 0.29420
## 5   6 0.5335 0.25100
## 6   6 0.6089 0.21410
## 7   7 0.6707 0.18270
## 8   8 0.7221 0.15590
## 9   9 0.7649 0.13300
## 10  9 0.8007 0.11350
## 11 11 0.8309 0.09680
## 12 11 0.8564 0.08259
## 13 10 0.8778 0.07046
## 14 10 0.8959 0.06011
## 15 10 0.9113 0.05129
## 16 10 0.9244 0.04376
## 17 10 0.9354 0.03733
## 18 11 0.9449 0.03185
## 19 10 0.9530 0.02717
## 20 11 0.9599 0.02318
## 21 11 0.9657 0.01978
## 22 11 0.9707 0.01688
## 23 11 0.9750 0.01440
## 24 12 0.9786 0.01228
## 25 12 0.9818 0.01048
## 26 12 0.9844 0.00894
## 27 12 0.9867 0.00763
## 28 12 0.9886 0.00651
## 29 12 0.9903 0.00555
## 30 13 0.9917 0.00474
```

For comparison purposes we set the penalty term λ equal to 0.03 that results in the selection of 10 OTUs. In HFHSday1 data, we use *glmnet_wrapper()* directly and set the extra input $\lambda = 0.03$:

```
HFHS.results_clrlasso <- glmnet_wrapper(X = clrx_HFHSday1, Y = y_HFHSday1_numeric,
                                        family = 'binomial', lambda = 0.03)
```

Then we get the number of selected variables:

```
HFHS.results_clrlasso$numVarSelect
```

```
## [1] 10
```

and the names of selected variables:

```
HFHS.results_clrlasso$varSelect
```

```
##  [1] "400599"  "192222"  "348038"  "401384"  "290253"  "261265"  "300851"
##  [8] "462764"  "1108745" "265322"
```

We also extract the taxonomic information of these selected OTUs.

```
HFHS.tax_clrlasso <- taxonomy_HFHS[which(rownames(taxonomy_HFHS) %in%
                                           HFHS.results_clrlasso$varSelect), ]
kable(HFHS.tax_clrlasso[ ,2:6], booktabs = T)
```

	Phylum	Class	Order	Family	Genus
192222	Bacteroidetes	Bacteroidia	Bacteroidales	Prevotellaceae	
290253	Firmicutes	Clostridia	Clostridiales	Ruminococcaceae	Oscillospira
261265	Firmicutes	Clostridia	Clostridiales	Lachnospiraceae	
1108745	Firmicutes	Clostridia	Clostridiales	[Mogibacteriaceae]	
462764	Firmicutes	Clostridia	Clostridiales	Ruminococcaceae	Ruminococcus
265322	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	
400599	Firmicutes	Clostridia	Clostridiales		
348038	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	
401384	Firmicutes	Clostridia	Clostridiales	Ruminococcaceae	Oscillospira
300851	Firmicutes	Clostridia	Clostridiales	Ruminococcaceae	Oscillospira

Chapter 4

coda-lasso

Coda-lasso implements penalised regression on a log-contrast model (a regression model on log-transformed covariates and a zero-sum constraint on the regression coefficients, except the intercept) (Lu et al., 2019; Lin et al., 2014).

As we mentioned before, *coda-lasso* is not yet available as an R package. Our R code for implementing the algorithm includes two main functions that are *coda_logistic_lasso()* and *coda_logistic_elasticNet()* that implement LASSO or elastic-net penalisation on a logistic regression model for a binary outcome.

Details of function *coda_logistic_lasso*(**y**,**X**,**lambda**):

y is the binary outcome, can be numerical (values 0 and 1), factor (2 levels) or categorical (2 categories).

X is the matrix of microbiome abundances, either absolute abundances (counts) or relative abundances (proportions), the rows of **X** are individuals/samples, the columns are taxa.

lambda is the penalisation parameter, the larger the value of **lambda** the smaller the number of variables selected.

Notes:

- Imputation of zeros: The user should provide a matrix **X** of positive values without zeros. Zeros should be previously added with an offset of 1 by the user.
- Log-transformation: **X** should not be the matrix of log(counts) or log(proportions). The method itself performs the log-transformation of the abundances.
- Selection of λ: Functions *lambdaRange_codalasso()* and *lambdaRange_elasticnet()* are useful to find the optimum value of the penalisation parameter λ. Function *lambdaRange_codalasso*(**y**,**X**) provides the number of selected variables and the proportion of explained deviance for a sequence of values for the penalised parameter λ.

Bellow, we illustrate the use of *coda_logistic_lasso()* on the Crohn and HFHS-Day1 case

studies. We also generated a wrapper function called *coda_lasso_wrapper()* that will call *coda_logistic_lasso()* function within the wrapper and help us to handle the output of *coda-lasso*. The *coda_lasso_wrapper()* function is uploaded via **functions.R**.

4.1 Crohn case study

To run *coda-lasso*, we must specify a value of λ, the penalisation parameter: the larger the value of λ, the less variables will be selected. In previous analysis of this dataset with *selbal*, a balance with 12 variables was determined optimal to discriminate the CD status (Rivera-Pinto et al., 2018). For ease of comparison, we will specify the penalised parameter that results in the selection of 12 variables for *coda-lasso*. Function *lambdaRange_codalasso()* helps us to identify the value of λ that corresponds to 12 variables selected. To save space, we only consider a sequence of λ from 0.1 to 0.2 with an increment of 0.01, but you can start from a broader range:

```
lambdaRange_codalasso(X = x_Crohn, y = y_Crohn, lambdaSeq = seq(0.1, 0.2, 0.01))
```

```
## [1] "lambda"            "num.selected"      "prop.explained.dev"
## 0.1000  23  0.2473
## 0.1100  18  0.2398
## 0.1200  17  0.2355
## 0.1300  25  0.2288
## 0.1400  13  0.2185
## 0.1500  12  0.2130
## 0.1600  12  0.2043
## 0.1700  14  0.2010
## 0.1800  11  0.1908
## 0.1900  10  0.1903
## 0.2000  8  0.1775
```

In this output, the first column is the value of λ, the second column is the number of selected variables and the third column is the proportion of deviance explained. According to this results, we will take $\lambda = 0.15$.

```
codalasso_Crohn <- coda_logistic_lasso(X = x_Crohn, y = y_Crohn, lambda = 0.15)
```

The same as the previous two methods, we also use a wrapper function called *coda_lasso_wrapper()* instead of the original *coda_logistic_lasso()* function. This wrapper function calls *coda_logistic_lasso()* function within the wrapper and with the same input as *coda_logistic_lasso()*.

```
Crohn.results_codalasso <- coda_lasso_wrapper(X = x_Crohn, Y = y_Crohn,
                                              lambda = 0.15)
```

Then we get the number of selected variables:

```
Crohn.results_codalasso$numVarSelect
```

```
## [1] 12
```

and the names of selected variables:

```
Crohn.results_codalasso$varSelect
```

```
##  [1] "g__Roseburia"                "g__Streptococcus"
##  [3] "g__Dialister"                "f__Peptostreptococcaceae_g__"
##  [5] "g__Aggregatibacter"          "g__Eggerthella"
##  [7] "g__Prevotella"               "g__Dorea"
##  [9] "g__Bilophila"                "o__Lactobacillales_g__"
## [11] "g__Lachnospira"              "g__Clostridium"
```

4.2 HFHS-Day1 case study

The analysis on HFHSday1 data is similar to Crohn data.

Using function *lambdaRange_codalasso()*, we explore the number of selected OTUs and the proportion of explained variance for different values of λ. To save space, we only consider a sequence of λ from 1.3 to 1.8 with an increment of 0.01, but you can start from a broader range:

```
lambdaRange_codalasso(X = x_HFHSday1, y = y_HFHSday1, lambdaSeq = seq(1.3, 1.8, 0.01))
```

```
## [1] "lambda"            "num.selected"       "prop.explained.dev"
## 1.3000  8  1.0000
## 1.3100  8  1.0000
## 1.3200  8  1.0000
## 1.3300  8  1.0000
## 1.3400  8  1.0000
## 1.3500  8  1.0000
## 1.3600  8  1.0000
## 1.3700  8  1.0000
## 1.3800  9  1.0000
## 1.3900  9  1.0000
## 1.4000  9  1.0000
## 1.4100  9  1.0000
## 1.4200  9  1.0000
## 1.4300  9  1.0000
## 1.4400  7  1.0000
## 1.4500  7  1.0000
## 1.4600  7  1.0000
## 1.4700  7  1.0000
## 1.4800  6  0.6344
```

```
## 1.4900  6  0.6344
## 1.5000  6  0.6344
## 1.5100  6  0.6343
## 1.5200  6  0.6343
## 1.5300  6  0.6343
## 1.5400  6  0.6343
## 1.5500  6  0.6343
## 1.5600  6  0.6342
## 1.5700  3  0.0108
## 1.5800  3  0.0089
## 1.5900  3  0.0130
## 1.6000  3  0.0241
## 1.6100  3  0.0290
## 1.6200  3  0.0441
## 1.6300  3  0.0481
## 1.6400  3  0.0644
## 1.6500  3  0.0667
## 1.6600  3  0.0688
## 1.6700  3  0.0700
## 1.6800  3  0.0822
## 1.6900  3  0.0829
## 1.7000  3  0.0835
## 1.7100  3  0.0841
## 1.7200  3  0.0847
## 1.7300  3  0.0854
## 1.7400  3  0.0860
## 1.7500  3  0.0866
## 1.7600  3  0.0872
## 1.7700  3  0.0878
## 1.7800  3  0.0884
## 1.7900  3  0.0890
## 1.8000  3  0.0895
```

The largest λ that provides maximum proportion of explained deviance is $\lambda = 1.47$. Thus, we implement *coda-lasso* with this value of λ.

In HFHSday1 data, we use *coda_lasso_wrapper()* directly.

```
HFHS.results_codalasso <- coda_lasso_wrapper(X = x_HFHSday1, Y = y_HFHSday1, lambda = 1.47)
```

The number of selected variables is 7:

```
HFHS.results_codalasso$numVarSelect
```

```
## [1] 7
```

The proportion of explained deviance by the selected OTUs is 1 meaning that they completely discriminate the two diet groups:

```
HFHS.results_codalasso$explained_deviance_proportion
```

```
## [1] 1
```

The names of the selected OTUs are:

```
HFHS.results_codalasso$varSelect
```

```
## [1] "348038" "400599" "192222" "198339" "265322" "263479" "175272"
```

By using the taxomonic table, we extract the taxonomic information of these selected OTUs.

```
HFHS.tax_codalasso <- taxonomy_HFHS[which(rownames(taxonomy_HFHS) %in%
                                          HFHS.results_codalasso$varSelect), ]
kable(HFHS.tax_codalasso[ ,2:6], booktabs = T)
```

	Phylum	Class	Order	Family	Genus
192222	Bacteroidetes	Bacteroidia	Bacteroidales	Prevotellaceae	
265322	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	
263479	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	
400599	Firmicutes	Clostridia	Clostridiales		
348038	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	
198339	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	
175272	Bacteroidetes	Bacteroidia	Bacteroidales	S24-7	

Chapter 5

Concordance of variables selected by the three methods

In this chapter, we are going to use different visualisation approaches to display the variables selected by the three methods:

- *UpSet plot*: highlights overlap of the variables selected by the three methods.
- *Selbal-like plot*: lists the selected variables and displays their discriminating ability with respect to sample groups.
- *plotLoadings*: visualises the variable coefficients and sample groups each variable contributes to.
- *Trajectory plot*: represents the rank of the variables selected by *coda-lasso* and *clr-lasso*, and their corresponding regression coefficients
- *GraPhlAn*: displays the taxonomic tree of selected variables (HFHSday1 data only).

5.1 Crohn case study

5.1.1 UpSetR

UpSet is a visualisation technique for the quantitative analysis of sets and their intersections (Lex et al., 2014). Before we apply *upset()*, we take the list of variable vectors selected with the three methods and convert them into a data frame compatible with *upset()* using *fromList()*. We then assign different color shcemes for each variable selection.

```
Crohn.select <- list(selbal = Crohn.results_selbal$varSelect,
                     clr_lasso = Crohn.results_clrlasso$varSelect,
                     coda_lasso = Crohn.results_codalasso$varSelect)
```

```
Crohn.select.upsetR <- fromList(Crohn.select)

upset(as.data.frame(Crohn.select.upsetR), main.bar.color = 'gray36',
      sets.bar.color = color[c(1:2,5)], matrix.color = 'gray36',
      order.by = 'freq', empty.intersections = 'on',
      queries = list(list(query = intersects, params = list('selbal'),
                          color = color[5], active = T),
                     list(query = intersects, params = list('clr_lasso'),
                          color = color[2], active = T),
                     list(query = intersects, params = list('coda_lasso'),
                          color = color[1], active = T)))
```

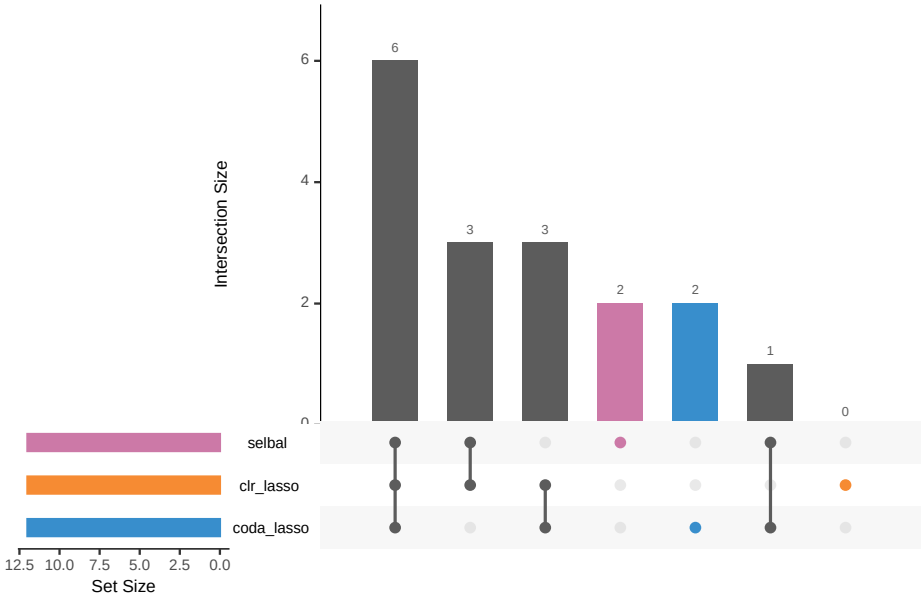


Figure 5.1: UpSet plot showing overlap between variables selected with different methods.

In Figure 5.1, the left bars show the number of variables selected by each method. The right bar plot combined with the scatterplot show different intersection and their aggregates. For example, in the first column, three points are linked with one line, and the intersection size of the bar is 6. This means that 6 variables are selected by all these three methods. While in the second column, 3 variables are only selected by the method *selbal* and *clr-lasso*.

5.1.2 Selbal-like plot

As mentioned in Chapter 2, *selbal* visualised the results using a mixture of selected variables, box plots and density plots:

```
# selbal
Crohn.selbal_pos <- Crohn.results_selbal$posVarSelect
Crohn.selbal_neg <- Crohn.results_selbal$negVarSelect
selbal_like_plot(pos.names = Crohn.selbal_pos,
                 neg.names = Crohn.selbal_neg,
                 Y = y_Crohn, selbal = TRUE,
                 FINAL.BAL = Crohn.results_selbal$finalBal)
```

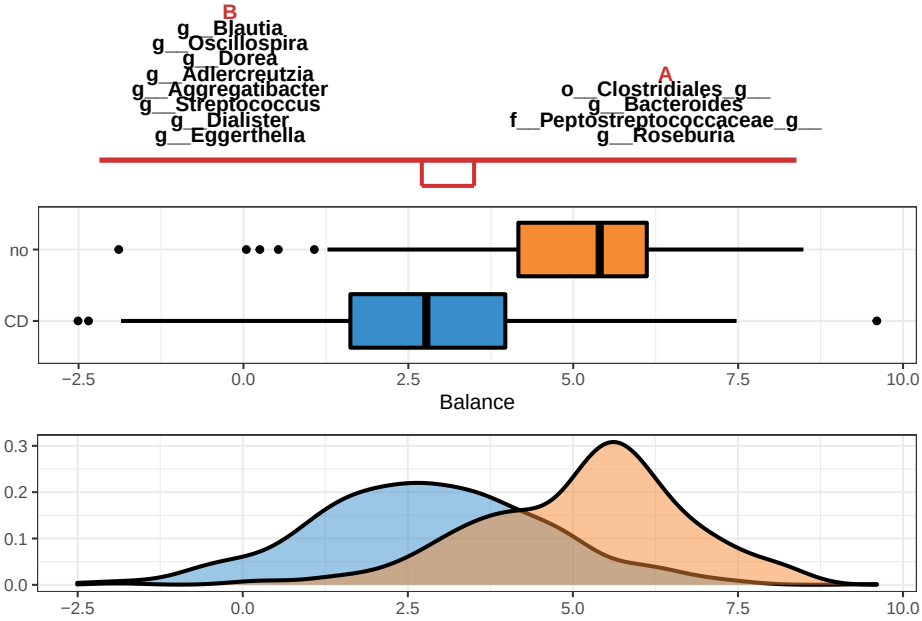


Figure 5.2: Selbal plot showing variables selected with method selbal and the ability of these variables to discriminate CD and non-CD individuals.

In Figure 5.2, the two groups of variables that form the global balance are specified at the top of the plot. They are equally important. The middle box plots represent the distribution of the balance scores for CD and non-CD individuals. The bottom part of the figure contains the density curve for each group.

Selbal-like plot is an extension of this kind of plots. Besides the visualisation of *selbal*, it can also be used to visualise the results from *clr-lasso* and *coda-lasso*, and generate similar plots as Figure 5.2.

```
# clr_lasso
Crohn.clr_pos <- Crohn.results_clrlasso$posCoefSelect
Crohn.clr_neg <- Crohn.results_clrlasso$negCoefSelect
selbal_like_plot(pos.names = names(Crohn.clr_pos),
                 neg.names = names(Crohn.clr_neg),
                 Y = y_Crohn, X = x_Crohn)
```

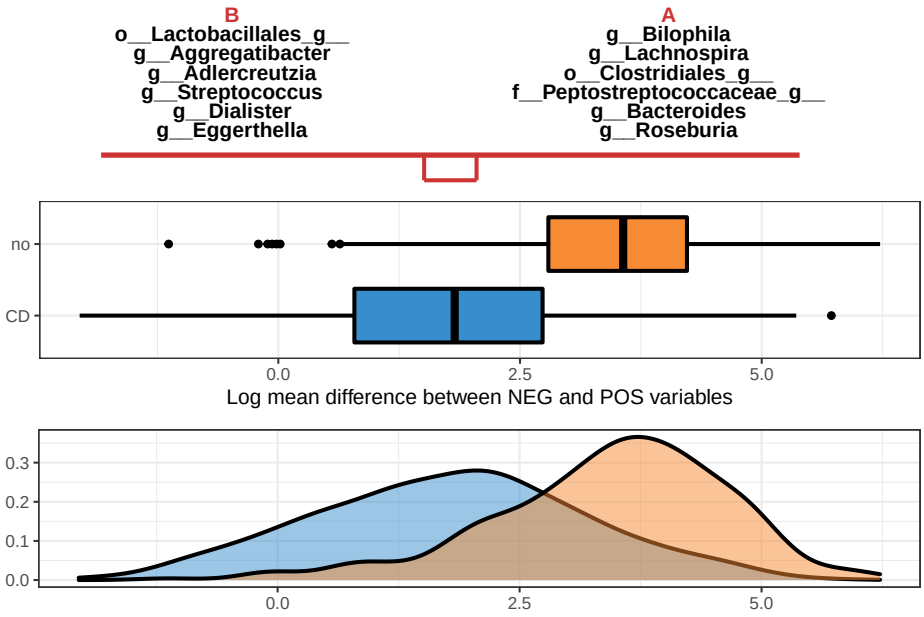


Figure 5.3: Selbal-like plot showing variables selected with method clr-lasso and the ability of these variables to discriminate CD and non-CD individuals.

In Figure 5.3, the top panel lists the selected variable names with either negative (B) or positive (A) coefficients. The names are ordered according to their importance (absolute coefficient values). The middle boxplots are based on the log mean difference between negative and positive variables: $\frac{1}{p_+}\sum_{i=1}^{p_+} logX_i - \frac{1}{p_-}\sum_{j=1}^{p_-} logX_j$. This log mean difference is calculated for each sample as a balance score, because it is proportionally equal to the balance mentioned in (Rivera-Pinto et al., 2018). The bottom density plots represent the distributions of the log mean difference scores for CD and non-CD individuals.

```
# coda_lasso
Crohn.coda_pos <- Crohn.results_codalasso$posCoefSelect
Crohn.coda_neg <- Crohn.results_codalasso$negCoefSelect
selbal_like_plot(pos.names = names(Crohn.coda_pos),
                 neg.names = names(Crohn.coda_neg),
                 Y = y_Crohn, X = x_Crohn)
```

The interpretation of Figure 5.4 is the same as Figure 5.3, but with variables selected with method *coda-lasso*.

5.1.3 plotLoadings

An easy way to visualise the coefficients of the selected variables is to plot them in a barplot in *mixMC* (Rohart et al., 2017). We have amended the *plotLoadings()* function from the

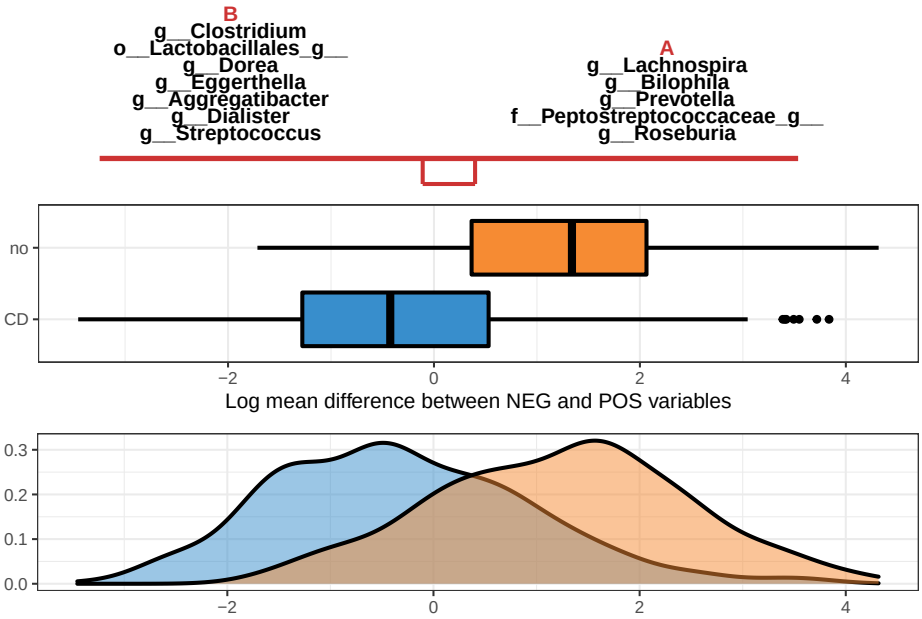


Figure 5.4: Selbal-like plot showing variables selected with method coda-lasso and the ability of these variables to discriminate CD and non-CD individuals.

package *mixOmics* to do so. The argument **Y** specified the sample class, so that the color assigened to each variable represents the class that has the larger mean value (**method = 'mean'** and **contrib.method = 'max'**).

```
# clr_lasso
Crohn.clr_coef <- Crohn.results_clrlasso$coefficientsSelect
Crohn.clr_data <- x_Crohn[ ,Crohn.results_clrlasso$varSelect]
Crohn.clr.plotloadings <- plotcoefficients(coef = Crohn.clr_coef,
                                           data = Crohn.clr_data,
                                           Y = y_Crohn,
                                           method = 'mean',
                                           contrib.method = 'max',
                                           title = 'Coefficients of clr-lasso on Crohn data')
```

Figure 5.5 shows that the variables colored in orange have an abundance greater in non-CD samples relative to CD samples (e.g. *Roseburia*), while the blue ones have a greater abundance in CD samples relative to non-CD samples (e.g. *Eggerthella*). It is based on their mean per group (CD vs. non-CDs). The bar indicates the coefficient. As we can see, all selected variables with a greater abundance in non-CD samples have been assigned a positive coefficient, and variables with a greater abundance in CD samples have been assigned a negative coefficient.

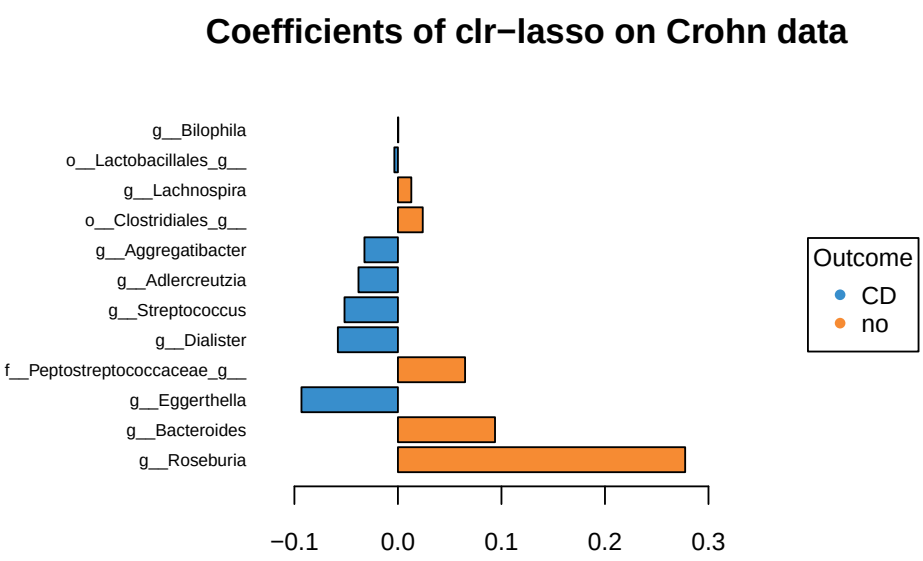


Figure 5.5: The plotLoadings of selected variables with clr-lasso.

```
# coda_lasso
Crohn.coda_coef <- Crohn.results_codalasso$coefficientsSelect
Crohn.coda_data <- x_Crohn[ ,Crohn.results_codalasso$varSelect]

Crohn.coda.plotloadings <- plotcoefficients(coef = Crohn.coda_coef,
                                data = Crohn.coda_data,
                                Y = y_Crohn,
                                method = 'mean',
                                contrib.method = 'max',
                                title = 'Coefficients of coda-lasso on Crohn data')
```

The same as Figure 5.5, we can interpret Figure 5.6 as follows. Both *Roseburia* and *Peptostreptococcaceae* selected by *clr-lasso* and *coda-lasso* have a greater abundance in non-CD group and were assigned with the same coeffcient rank. But both *Eggerthella* and *Dialister* selected by two methods were assigned with very different coefficient rank. Several variables have a greater abundance in CD group, but with a positive coefficient. It means this model is not optimal at some extent. This may suggest that *clr-lasso* is better at identifying discriminative variables than *coda-lasso*.

5.1.4 Trajectory plots

To visualise the change of variable coefficients and their ranks in the selection between different methods, we use *trajectory plots*.

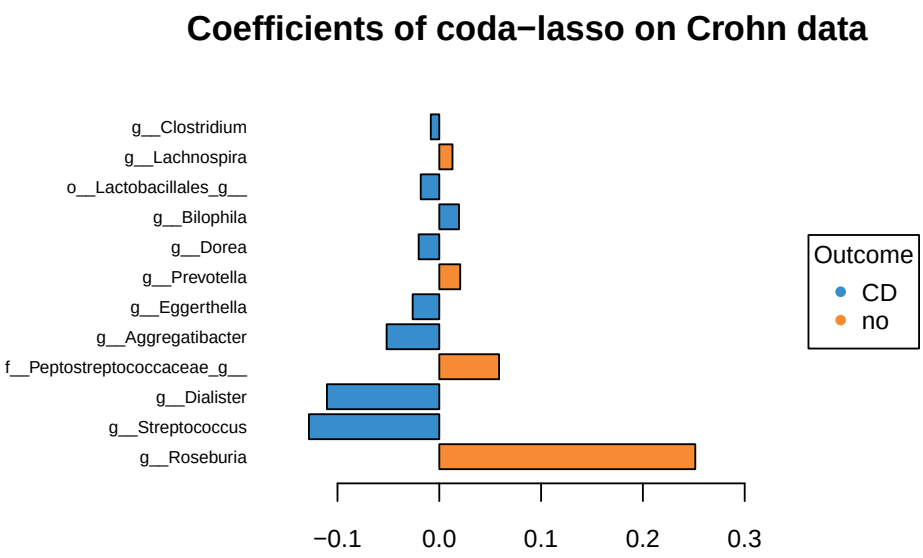


Figure 5.6: The plotLoadings of selected variables with coda-lasso.

```
TRAJ_plot(selectVar_coef_method1 = Crohn.coda_coef,
          selectVar_coef_method2 = Crohn.clr_coef,
          selectMethods = c('coda-lasso', 'clr-lasso'))
```

Figure 5.7 shows the selected variables ordered by their rank in the selection (according to their coefficient absolute values) between *coda-lasso* and *clr-lasso*, with the thickness of the lines representing the coefficient absolute values.

In this plot 5.7, we can visualise the rank change of each selected variable between *coda-lasso* and *clr-lasso* selection. For example, the rank of *Dialister* is lower in *clr-lasso* compared to *coda-lasso*. Moreover, we can detect the variables (e.g. *Bacteroides*) that are selected by one method (e.g. *clr-lasso*) with high coefficient rank, but not selected by the other method (e.g. *coda-lasso*).

5.2 HFHS-Day1 case study

Guidance on how to interpret the following plots is detailed in previous **section: Crohn case study**.

5.2.1 UpSetR

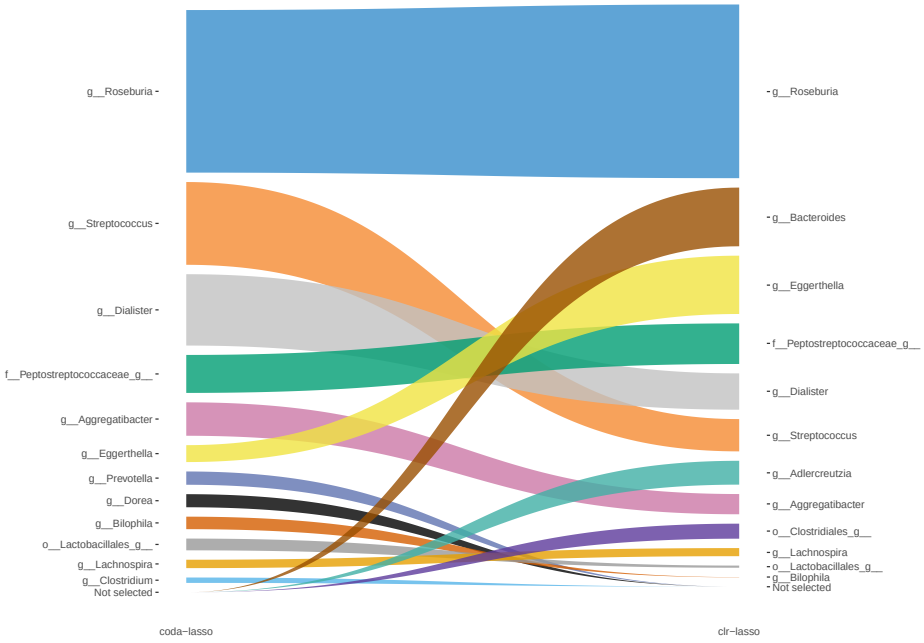


Figure 5.7: Trajectory plots of selected variables from both coda-lasso and clr-lasso in Crohn data.

```
HFHS.select <- list(selbal = HFHS.results_selbal$varSelect,
                    clr_lasso = HFHS.results_clrlasso$varSelect,
                    coda_lasso = HFHS.results_codalasso$varSelect)

HFHS.select.upsetR <- fromList(HFHS.select)

upset(as.data.frame(HFHS.select.upsetR), main.bar.color = 'gray36',
      sets.bar.color = color[c(1,2,5)], matrix.color = 'gray36',
      order.by = 'freq', empty.intersections = 'on',
      queries = list(list(query = intersects, params = list('selbal'),
                          color = color[5], active = T),
                     list(query = intersects, params = list('coda_lasso'),
                          color = color[2], active = T),
                     list(query = intersects, params = list('clr_lasso'),
                          color = color[1], active = T)))
```

Figure 5.8 shows that 5 OTUs are only selected with *clr-lasso*, 4 OTUs are selected both with *coda-lasso* and *clr-lasso*, 2 OTUs are only selected with *coda-lasso*, 1 OTUs is selected with both *selbal* and *coda-lasso*, and 1 is selected both with *selbal* and *clr-lasso*. Among three methods, *clr-lasso* selected the largest number of OTUs and *selbal* the smallest.

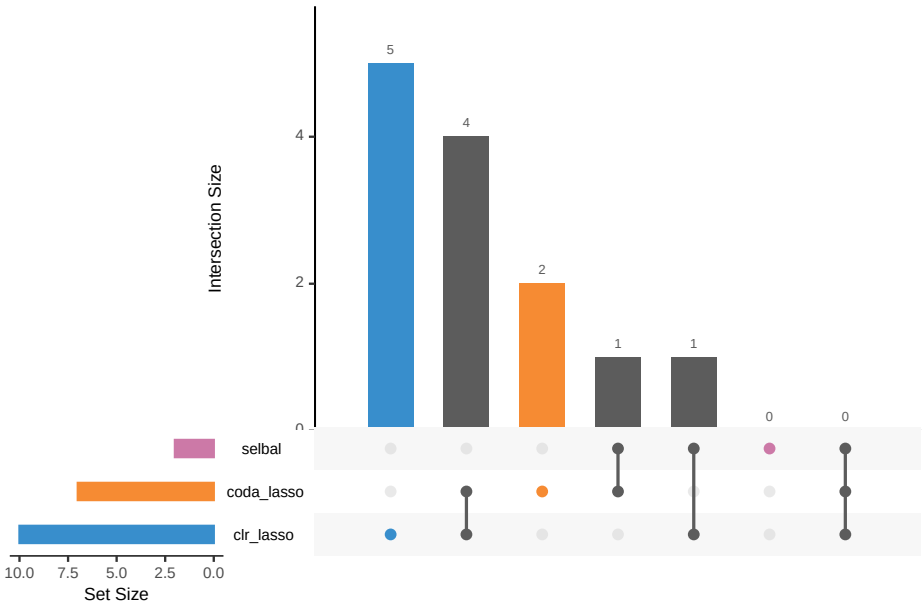


Figure 5.8: UpSet plot showing overlap between variables selected with different methods.

5.2.2 Selbal-like plot

```
# selbal
HFHS.selbal_pos <- HFHS.results_selbal$posVarSelect
HFHS.selbal_neg <- HFHS.results_selbal$negVarSelect
selbal_like_plot(pos.names = HFHS.selbal_pos,
                 neg.names = HFHS.selbal_neg,
                 Y = y_HFHSday1, selbal = TRUE,
                 FINAL.BAL = HFHS.results_selbal$finalBal,
                 OTU = T, taxa = taxonomy_HFHS)
```

```
# clr_lasso
HFHS.clr_pos <- HFHS.results_clrlasso$posCoefSelect
HFHS.clr_neg <- HFHS.results_clrlasso$negCoefSelect
selbal_like_plot(pos.names = names(HFHS.clr_pos),
                 neg.names = names(HFHS.clr_neg),
                 Y = y_HFHSday1, X = x_HFHSday1, OTU = T,
                 taxa = taxonomy_HFHS)
```

```
# coda_lasso
HFHS.coda_pos <- HFHS.results_codalasso$posCoefSelect
HFHS.coda_neg <- HFHS.results_codalasso$negCoefSelect
selbal_like_plot(pos.names = names(HFHS.coda_pos),
```

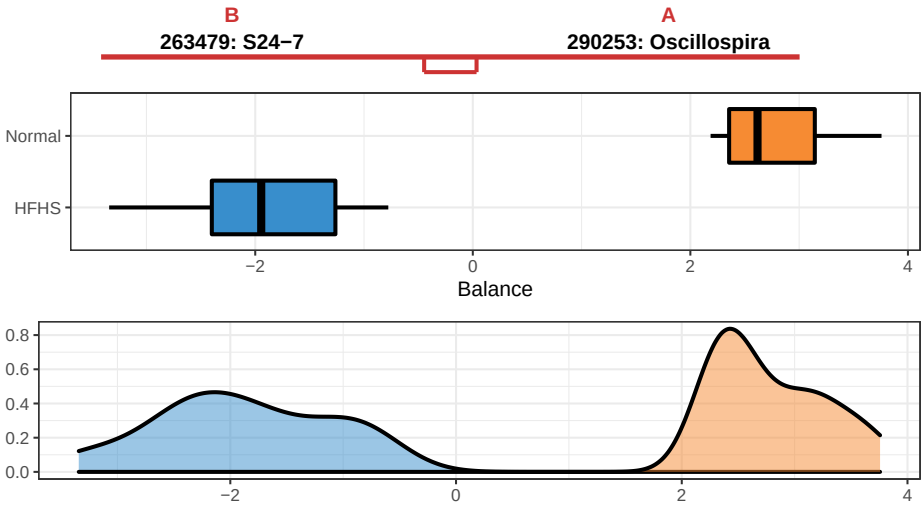


Figure 5.9: Selbal plot showing variables selected with methd selbal and the ability of these variables to discriminate HFHS and normal individuals.

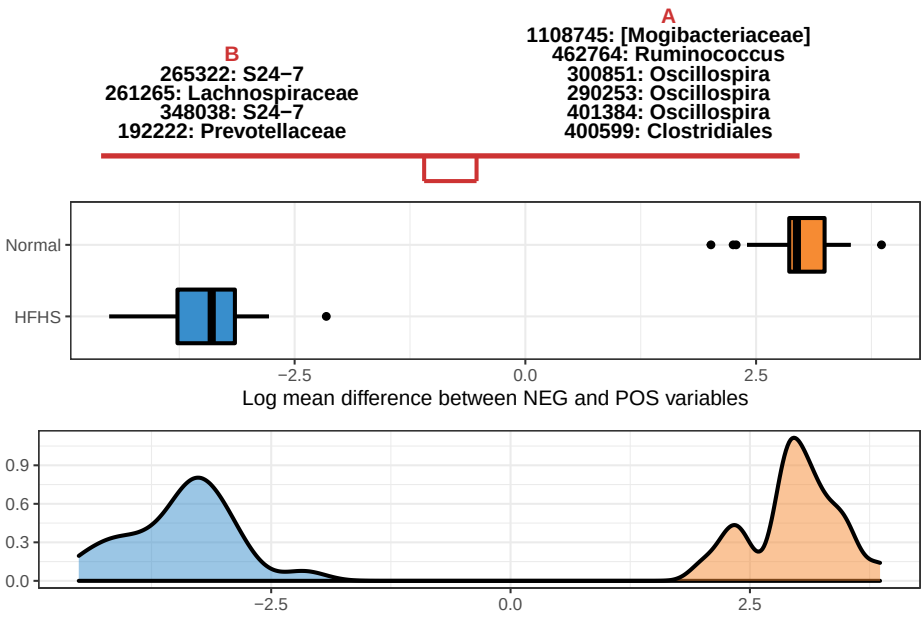


Figure 5.10: Selbal-like plot showing variables selected with method clr-lasso and the ability of these variables to discriminate HFHS and normal individuals.

```
                neg.names = names(HFHS.coda_neg),
                Y = y_HFHSday1, X = x_HFHSday1,
                OTU = T, taxa = taxonomy_HFHS)
```

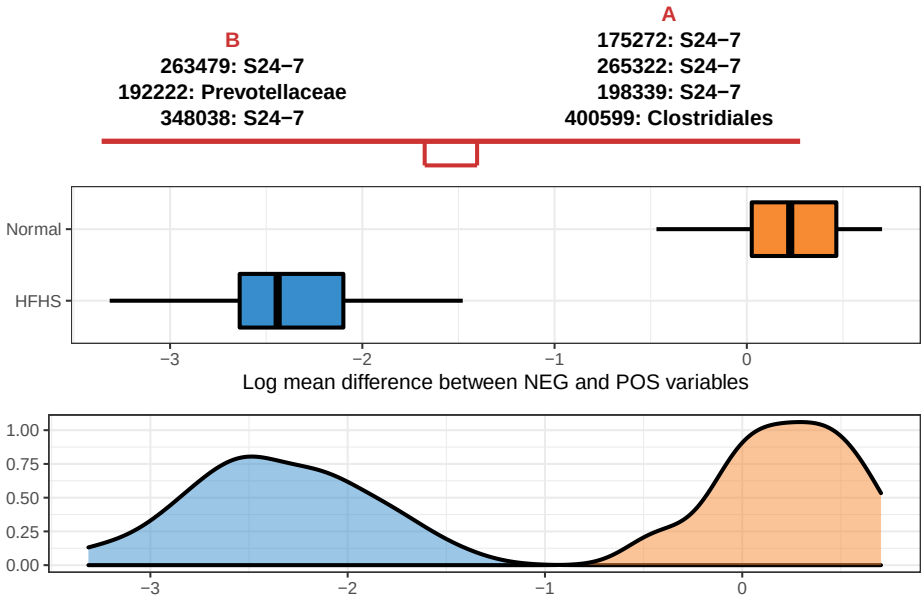


Figure 5.11: Selbal-like plot showing variables selected with method coda-lasso and the ability of these variables to discriminate HFHS and normal individuals.

Note: *S24-7* is a family from order *Bacteroidales*.

Among these methods, *selbal* only needs two OTUs to build a balance, it also means the association between microbiome composition and diet is very strong.

5.2.3 plotLoadings

```
# clr_lasso
HFHS.clr_coef <- HFHS.results_clrlasso$coefficientsSelect
HFHS.clr_data <- x_HFHSday1[ ,HFHS.results_clrlasso$varSelect]
HFHS.clr.plotloadings <- plotcoefficients(coef = HFHS.clr_coef,
                                data = HFHS.clr_data,
                                Y = y_HFHSday1,
                                title = 'Coefficients of clr-lasso on HFHS-Day1 data',
                                method = 'mean',
                                contrib.method = 'max',
                                OTU = T,
                                taxa = taxonomy_HFHS)
```

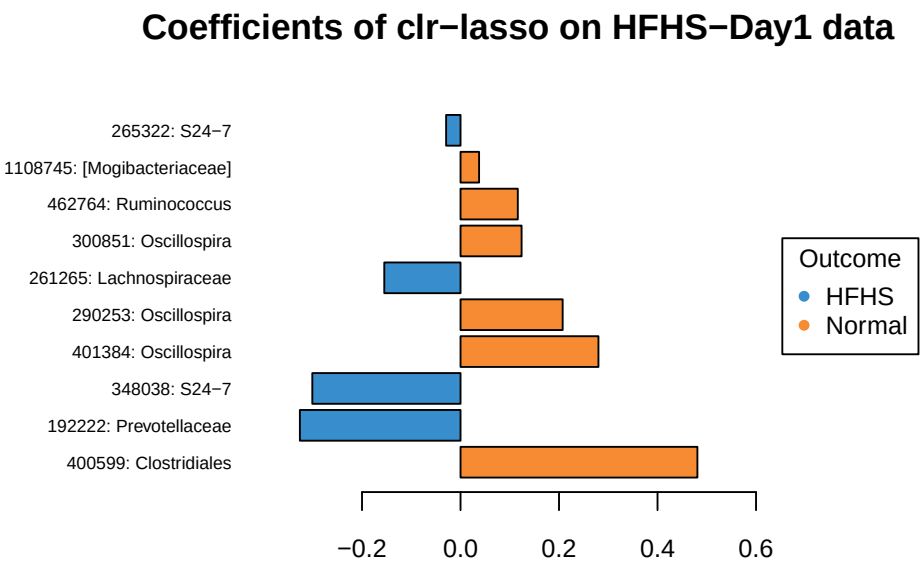


Figure 5.12: The plotLoadings of selected variables with clr-lasso.

```
# coda_lasso
HFHS.coda_coef <- HFHS.results_codalasso$coefficientsSelect
HFHS.coda_data <- x_HFHSday1[ ,HFHS.results_codalasso$varSelect]

HFHS.coda.plotloadings <- plotcoefficients(coef = HFHS.coda_coef,
                            data = HFHS.coda_data,
                            Y = y_HFHSday1,
                            method = 'mean',
                            contrib.method = 'max',
                            title = 'Coefficients of coda-lasso on HFHS-Day1 data',
                            OTU = T,
                            taxa = taxonomy_HFHS)
```

In Figure 5.13, three OTUs **175272: S24-7**, **265322: S24-7**, **198339: S24-7** have a greater abundance in HFHS group but were assigned with positive coefficients.

5.2.4 Trajectory plots

```
TRAJ_plot(selectVar_coef_method1 = HFHS.coda_coef,
          selectVar_coef_method2 = HFHS.clr_coef,
          selectMethods = c('coda-lasso', 'clr-lasso'),
          OTU = T, taxa = taxonomy_HFHS)
```

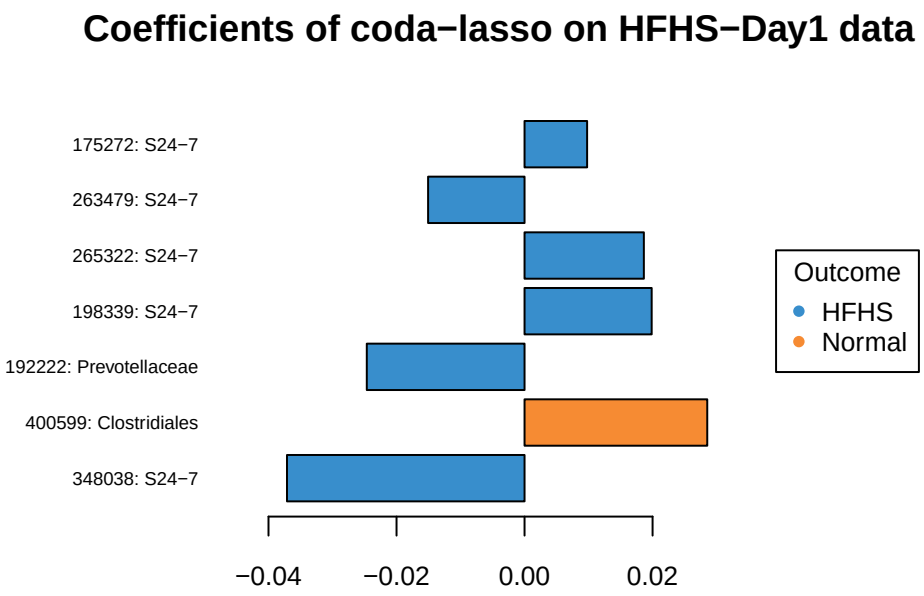


Figure 5.13: The plotLoadings of selected variables with coda-lasso.

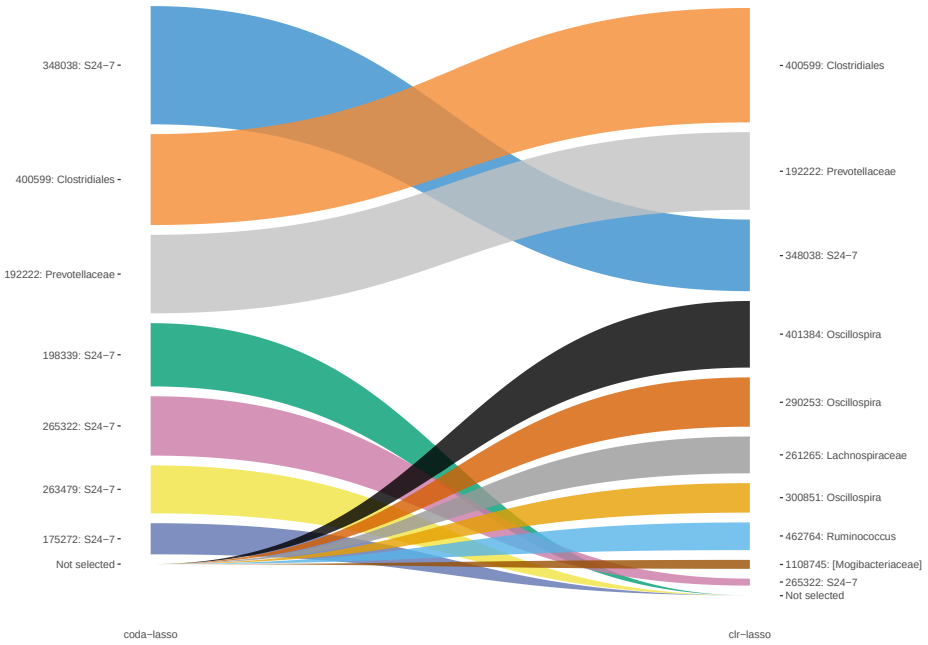


Figure 5.14: Trajectory plots of selected variables with both coda-lasso and clr-lasso in HFHSday1 data.

In Figure 5.14, top three OTUs selected with *clr-lasso* are also selected as top OTUs from *coda-lasso* but with different order. The other OTUs are either selected by *coda-lasso* or *clr-lasso*.

5.2.5 GraPhlAn

As we also have the taxonomic information of HFHSday1 data, we use *GraPhlAn* to visualise the taxonomic information of the selected OTUs. *GraPhlAn* is a software tool for producing high-quality circular representations of taxonomic and phylogenetic trees (https://huttenhower.sph.harvard.edu/graphlan). It is coded in Python.

We first remove empty taxa (e.g. species) and aggregate all these selected variables into a list. Then we use function *graphlan_annot_generation()* to generate the input files that graphlan python codes require. In the **save_folder**, there are two existing files: **annot_0.txt** and **graphlan_all.sh**. After we generate our input files **taxa.txt** and **annot_all.txt**, we only need to run the *./graphlan_all.sh* in the bash command line to generate the plot.

```
# remove empty columns
HFHS.tax_codalasso <- HFHS.tax_codalasso[,-7]
HFHS.tax_clrlasso <- HFHS.tax_clrlasso[,-7]
HFHS.tax_selbal <- HFHS.tax_selbal[,-7]

HFHS.select.tax <- list(selbal = HFHS.tax_selbal,
                        clr_lasso = HFHS.tax_clrlasso,
                        coda_lasso = HFHS.tax_codalasso)

graphlan_annot_generation(taxa_list = HFHS.select.tax, save_folder = 'graphlan/')
```

In Figure 5.15, the inner circle is a taxonomic tree of selected OTUs. The outside circles indicate different selection methods. If a proportion of a circle is coloured, it means that the corresponding OTU is selected by the method labeled on the circle. If the bottom nodes are coloured in gray, it indicates the OTUs are only selected by one method.

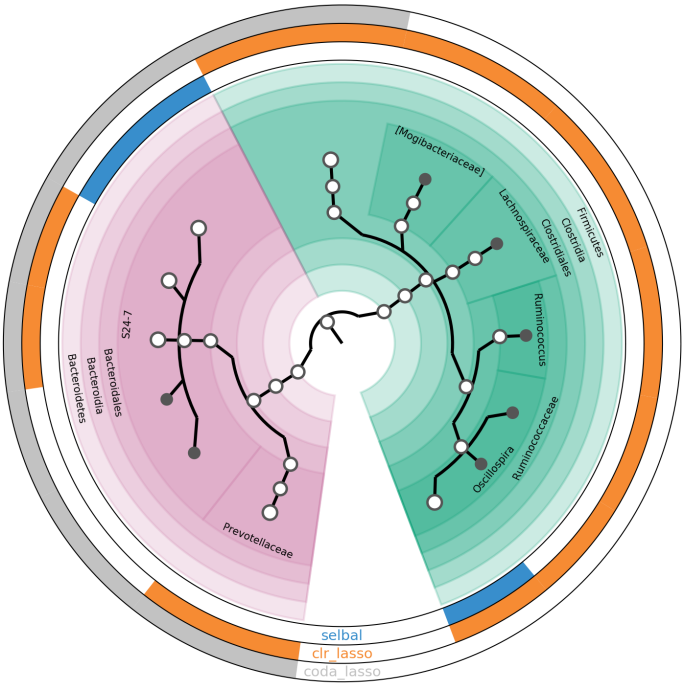


Figure 5.15: GraPhlAn of selected taxa from different methods in HFHS-Day1 data.

Bibliography

Gevers, D., Kugathasan, S., Denson, L. A., Vázquez-Baeza, Y., Van Treuren, W., Ren, B., Schwager, E., Knights, D., Song, S. J., Yassour, M., et al. (2014). The treatment-naive microbiome in new-onset crohn's disease. *Cell host & microbe*, 15(3):382–392.

Gonzalez, A., Navas-Molina, J. A., Kosciolek, T., McDonald, D., Vázquez-Baeza, Y., Ackermann, G., DeReus, J., Janssen, S., Swafford, A. D., Orchanian, S. B., et al. (2018). Qiita: rapid, web-enabled microbiome meta-analysis. *Nature methods*, 15(10):796–798.

Le Cessie, S. and Van Houwelingen, J. C. (1992). Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(1):191–201.

Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R., and Pfister, H. (2014). Upset: visualization of intersecting sets. *IEEE transactions on visualization and computer graphics*, 20(12):1983–1992.

Lin, W., Shi, P., Feng, R., and Li, H. (2014). Variable selection in regression with compositional covariates. *Biometrika*, 101(4):785–797.

Lu, J., Shi, P., and Li, H. (2019). Generalized linear models with linear constraints for microbiome compositional data. *Biometrics*, 75(1):235–244.

Rivera-Pinto, J., Egozcue, J., Pawlowsky-Glahn, V., Paredes, R., Noguera-Julian, M., and Calle, M. (2018). Balances: a new perspective for microbiome analysis. *MSystems*, 3(4):e00053–18.

Rohart, F., Eslami, A., Matigian, N., Bougeard, S., and Le Cao, K.-A. (2017). Mint: a multivariate integrative method to identify reproducible molecular signatures across independent experiments and platforms. *BMC bioinformatics*, 18(1):128.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.