

Methods Comparison

Malu Calle Rosingana

Coding: Yiwon Wang

2019-02-26

Contents

1	Introduction	5
1.1	Packages installation and loading	5
1.2	Simulations	5
1.3	Example datasets	5
1.3.1	Crohn disease data	5
1.3.2	BEME day1 data	6
2	Methods implementation	7
2.1	LOGISTIC LASSO	7
2.2	LOGISTIC ELASTIC NET	9
2.3	rangLambda	12
2.4	Soft thresholding	13
2.5	Other functions	14
3	CODA_LOGISTIC_LASSO	17
3.1	Crohn disease data	17
3.2	BEME day1 data	18
4	CLR_LOGISTIC_LASSO	19
4.1	Crohn disease data	19
4.2	BEME day1 data	24
5	Selbal: selection of balances	25
5.1	Crohn disease data	25
5.2	BEME day1 data	26
6	Concordance of variables selected by the three methods	27
6.1	Crohn disease data	27
6.2	BEME day1 data	29

Chapter 1

Introduction

This vignette provides a comparison of the 12 variables selected by the 3 methods: coda_logistic_lasso, clr+logistic_lasso, selbal.

First, let us install and load the packages.

1.1 Packages installation and loading

```
#cran.packages = c('knitr', 'MASS', 'VennDiagram', 'gplots', 'glmnet')
#install.packages(cran.packages)
#devtools::install_github(repo = "UVic-omics/selbal")

library(knitr)
library(MASS)
library(VennDiagram)
library(gplots)
library(glmnet)
library(selbal)
```

1.2 Simulations

The way we simulate datasets.

1.3 Example datasets

We have two case studies.

1.3.1 Crohn disease data

```
load("./datasets/Crohn_data.rda")
dim(x_Crohn)
```

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

The Crohn data have 975 samples and 48 taxa.

```
summary(y_Crohn)
```

```
##   CD   no
```

```
## 662 313
```

Among 975 samples, 662 samples are labeled as ‘CD’, the other 313 are labeled as ‘no’.

1.3.2 BEME day1 data

Chapter 2

Methods implementation

2.1 LOGISTIC LASSO

With linear constraint: $\sum \beta_j = 0$ (for $j > 1$)

```
# y: dependent variable, binary, vector of length n
# X: matrix of k covariates (positive values, taxa abundances in counts, proportions, intensities, ...),

coda_logistic_lasso<-function(y,X,lambda, maxiter=400, maxiter2=50, r=10,
                             tol=1.e-4, tol2=1.e-6){

  #install.packages("MASS")
  library(MASS) # for the computation of the generalized inverse ginv()
  # library(tictoc)

  if (!is.numeric(y)){
    y<-as.numeric(y)-1
  }

  #source("functions_coda_logistic_lasso.R")

  # Transform initial data
  ztransformation(X)

  #initial values for beta
  # p<-ncol(X) # p: number of covariates after filtering
  # n<-nrow(X); # both defined on ztransformation function
  #beta_ini<-rep(1,p+1)/(p+1) # uniform values
  beta_ini<-c(log(mean(y)/(1-mean(y))),rep(1/p,p)) # b0 related to mean(y) and uniform values for the

  #null deviance
  nulldev<-glm(y~1, family=binomial())[[10]]

  #initialization parameters

  k<-1
  epsilon<-0.1
```

```

beta_ant<-beta_ini
beta<-beta_ant
y_ant<-beta_ant
y_k<-y_ant
d_k<-y_ant
dev_explained_ant<-0

# Optimization with constraint
start_iter = Sys.time();
while((abs(epsilon)>tol)&(k<=maxiter)){
  #print(append("loop",k))
  k0<-0
  t_k<-10
  condition<-0.1
  while ((condition>0)&(k0<=maxiter2)){
    k0<-k0+1
    #print(append("k0",k0))
    d_k<-y_ant-t_k*grad_g(y_ant, z, y, n)

    # Soft thresholding:
    zproxi<-c(d_k[1],soft_thres(d_k[-1],lambda*t_k))

    # Projection:
    zproj<-projection(zproxi,p_c)

    # line search condition
    Gt<-(y_ant-zproj)/t_k
    condition<-g(y_ant-t_k*Gt,z, y, n)-g(y_ant,z, y, n)+t_k*t(grad_g(y_ant,z, y, n))%*%Gt-t_k/2*norm
    #print(append("condition",condition))
    #print(append("t_k",t_k))
    t_k<-t_k/2
  }
  beta<-zproj

  y_k<-beta+(k-1)/(k+r-1)*(beta-beta_ant)

  dev_explained<-1-(nrow(X)*2*g(beta,z, y, n)/nulldev)
  epsilon<-abs(dev_explained-dev_explained_ant)

  y_ant<-y_k
  beta_ant<-beta
  dev_explained_ant<-dev_explained
  k<-k+1
  #print(append("epsilon",epsilon))
}
end_iter = Sys.time();
sprintf("iter time = %f",end_iter-start_iter)

#Projection of the optimal beta to fulfil the constraint sum(beta[j])=0, for j>1

```



```

indx<-which(abs(zproxi)>tol2)
if (abs(zproxi[1])>0) indx<-indx[-1]
c0<-rep(1,(length(indx)))
c0<-c0/sqrt(normSqr(c0))
p_c0<-c0%*%t(c0)
#beta1<-as.numeric(projection(beta[indx],p_c0))
beta1<-projection(beta[indx],p_c0)
#beta1
beta_res<-c(beta[1],rep(0,p))
beta_res[indx]<-beta1

# print("beta coefficients:")
# beta_res

#print("taxa with non-zero coeff:")
# selec<-colnames(z)[abs(beta_res)>0]

#print("beta non-zero coefficients:")
# beta_res[abs(beta_res)>0]

#print("proportion of explained deviance")
# dev_explained
#print("proportion of explained deviance beta_res")

dev_explained_beta_res<-1-(nrow(X)*2*g(beta_res,z, y, n)/nulldev)
# dev_explained_beta_res

results<-list(
  "number of iterations"=k,
  "number of selected taxa"=
    sum(abs(beta_res)>0)-1,
  "indices of taxa with non-zero coeff"=
    which(abs(beta_res)>0)-1,
  "taxa with non-zero coeff"=
    colnames(z)[abs(beta_res)>0],
  "beta non-zero coefficients"=
    beta_res[abs(beta_res)>0],
  "proportion of explained deviance"=
    dev_explained_beta_res,
  "betas"=
    beta_res)

return(results)
} # END function coda_logistic_lasso

```

2.2 LOGISTIC ELASTIC NET

```
##-----
```

```

coda_logistic_elasticNet<-function(y,X,lambda, alpha=0.5, maxiter=1000, maxiter2=50, r=10,
                                  tol=1.e-4, tol2=1.e-6){

  #install.packages("MASS")
  library(MASS) # for the computation of the generalized inverse ginv()
  # library(tictoc)

  # source("functions_coda_logistic_lasso.R")

  # Transform initial data
  ztransformation(X)

  #initial values for beta
  #beta_ini<-rep(1,p+1)/(p+1) # uniform values
  beta_ini<-c(log(mean(y)/(1-mean(y))),rep(1/p,p)) # b0 related to mean(y) and uniform values for the

  #null deviance
  nulldev<-glm(y~1, family=binomial())[[10]]

  #initialization parameters

  k<-1
  epsilon<-0.1
  beta_ant<-beta_ini
  beta<-beta_ant
  y_ant<-beta_ant
  y_k<-y_ant
  d_k<-y_ant
  dev_explained_ant<-0
  # elastic parameters
  lambda=alpha*lambda;
  gamma=2*(1-alpha)/alpha;

  # Optimization with constraint
  start_iter = Sys.time()
  while((abs(epsilon)>tol)&(k<=maxiter)){
    #print(append("loop",k))
    k0<-0
    t_k<-10
    condition<-0.1
    while ((condition>0)&(k0<=maxiter2)){
      k0<-k0+1
      #print(append("k0",k0))
      d_k<-y_ant-t_k*grad_g(y_ant, z, y, n)

      # Soft thresholding:
      # zproxi<-c(d_k[1],soft_thres(d_k[-1], lambda*t_k))
      # zproxi = zproxi/(1+gamma*lambda);

```

```

zproxi<-c(d_k[1],soft_thres(d_k[-1],lambda*t_k)/(1+gamma*lambda*t_k))

# Projection:
zproj<-projection(zproxi,p_c)

# line search condition
Gt<-(y_ant-zproj)/t_k
condition<-g(y_ant-t_k*Gt,z, y, n)-g(y_ant,z, y, n)+t_k*t(grad_g(y_ant,z, y, n))%*%Gt-t_k/2*normS

#print(append("condition",condition))
#print(append("t_k",t_k))
t_k<-t_k/2
}
beta<-zproj

y_k<-beta+(k-1)/(k+r-1)*(beta-beta_ant)

dev_explained<-1-(nrow(X)*2*g(beta,z, y, n)/nulldev)
epsilon<-abs(dev_explained-dev_explained_ant)

y_ant<-y_k
beta_ant<-beta
dev_explained_ant<-dev_explained
k<-k+1
#print(append("epsilon",epsilon))
}
end_iter = Sys.time()
sprintf("iter time = %f",end_iter - start_iter)

#Projection of the optimal beta to fulfil the constraint sum(beta[j])=0, for j>1
indx<-which(abs(zproxi)>tol2)
if (abs(zproxi[1])>0) indx<-indx[-1]
c0<-rep(1,(length(indx)))
c0<-c0/sqrt(normSqr(c0))
p_c0<-c0%*%t(c0)
#beta1<-as.numeric(projection(beta[indx],p_c0))
beta1<-projection(beta[indx],p_c0)
#beta1
beta_res<-c(beta[1],rep(0,p))
beta_res[indx]<-beta1

#print("beta coefficients:")
# beta_res

#print("taxa with non-zero coeff:")
# selec<-colnames(z)[abs(beta_res)>0]
#return(selec)

```

```

# print("beta non-zero coefficients:")
# beta_res[abs(beta_res)>0]

# print("proportion of explained deviance")
# dev_explained
# print("proportion of explained deviance beta_res")

dev_explained_beta_res<-1-(nrow(X)*2*g(beta_res,z, y, n)/nulldev)
# dev_explained_beta_res

results<-list("number of iterations"=k,
             "number of selected taxa"=
               sum(abs(beta_res)>0)-1,
             "indices of taxa with non-zero coeff"=
               which(abs(beta_res)>0)-1,
             "name of taxa with non-zero coeff"=
               colnames(z)[abs(beta_res)>0],
             "beta non-zero coefficients"=
               beta_res[abs(beta_res)>0],
             "proportion of explained deviance"=
               dev_explained_beta_res,
             "betas"=
               beta_res)

return(results)

} # END function coda_logistic_ElasticNet

```

2.3 rangLambda

It provides a rang of lambda values corresponding to a given number of variables to be selected (numVar).

The default initial lambda is lambdaIni=1.

```

##-----
rangLambda <- function(y,X,numVar, lambdaIni =1){
  lambdaB = lambdaIni;
  lambdaA = 0;
  #lambda=0.5*(lambdaB+lambdaA);
  lambda=lambdaIni;
  results <- coda_logistic_lasso(y,X,lambda, maxiter = 100);
  numVarAct = results[[2]];
  if (numVarAct > numVar){
    lambda=lambda+0.5;
    #lambda=lambda+1;
    lambdaB =lambda;
  }else{
    lambdaB =lambda;
  }
  nIter=1;
  presentLambda = NULL;

```

```

presentnumVar = NULL;
presentLambda[nIter] = lambda;
presentnumVar[nIter] = numVarAct;
numvarA=ncol(X);
numvarB=numVarAct;
print(c(lambdaA, lambdaB, numvarA, numvarB))
diffNvar = abs(numvarB-numvarA);
while ((diffNvar>1) & (abs(numVarAct-numVar)>0) & (nIter < 6)){
  nIter=nIter+1;
  lambda=0.5*(lambdaB+lambdaA);
  results <- coda_logistic_lasso(y,X,lambda, maxiter = 100);
  numVarAct = results[[2]];
  if (numVarAct < numVar) {
    lambdaB = lambda;
    numvarB = numVarAct;
  }else{
    lambdaA = lambda;
    numvarA = numVarAct;
  }
  presentLambda[nIter] = lambda;
  presentnumVar[nIter] = numVarAct;
  diffNvar = abs(numvarB-numvarA);
  print(c(lambdaA, lambdaB, numvarA, numvarB))
}
indx=which(presentnumVar >= 0);
results = list( 'rang lambdas'=c(lambdaA,lambdaB), 'num selected variables'=c(numvarA,numvarB))
return(results);
}

```

2.4 Soft thresholding

<http://www.simonlucy.com/soft-thresholding/>

```

soft_thres<-function(b, lambda){
  x<-rep(0,length(b))
  # Set the threshold
  th = lambda/2;

  #First find elements that are larger than the threshold
  k <- which(b > th)
  x[k] <- b[k] - th

  # Next find elements that are less than abs
  k <-which(abs(b) <= th)
  x[k] <- 0

  # Finally find elements that are less than -th
  k <-which(b < -th)
  x[k] = b[k] + th

  return(x)
}

```

2.5 Other functions

```
##-----

norm1<-function(x){
  return(sum(abs(x)))
}

normSqr<-function(x){
  return(sum(x^2))
}

ztransformation<-function(X){
  p<-ncol(X)  # p: number of covariates after filtering
  n<-nrow(X);
  # log transformation Z=log(X)
  z<-log(X)

  # z=matrix of covariates: add a first column of 1's for beta0
  z<-cbind(rep(1,nrow(z)),z)
  z<-as.matrix(z)

  # c=linear constraint sum(betas)=0 (except beta0)
  c<-c(0,rep(1,ncol(X)))
  c<-c/sqrt(sum(c^2))
  c<-as.matrix(c)
  p_c<-c%*%t(c)

  # z transformation (centering) for improvement of optimization
  # this transformation does not affect the estimation since the linear predictor is the same
  z<- (z-(z%*%p_c))
  colnames(z)<-c("beta0",colnames(X))

  return(z)
}

A<-function(x){
  y<-log(1+exp(x))
  return(y)
}

mu_beta<-function(x, Z){
  zetabybeta<-Z%*%x
  res<-rep(1,length(zetabybeta))
  indzb<-which(zetabybeta<=100)
  res[indzb]<-exp(zetabybeta[indzb])/(1+exp(zetabybeta[indzb]))
  return(res)
}

g<-function(x,Z,Y,n){
```

```

res<- (t(Y))%*%Z%*%x
res<-res-sum(log(1+exp(Z%*%x)))
res<- as.vector((-res)/n)
return(res)
}

h<-function(x, lambda){
  lambda*norm1(x)
}

grad_g<-function(x, Z, Y, n){
  res<- (t(Y-mu_beta(x,Z)))%*%Z
  res<- as.vector((-res)/n)
  return(res)
}

# Projection

projection<-function(x, M){
  if (ncol(M)*nrow(M)>0){
    res<-x-ginv(M)%*%M%*%x
  } else {res<-rep(0,length(x))}
  return(res)
}

F<-function(x,t_k,d_k){
  lambda*t_k*norm1(x[-1])+normSqr(d_k-x)/2
}

F2<-function(x,Z,Y,n,lambda){
  g(x,Z,Y,n)+lambda*norm1(x[-1])
}

trapezInteg <- function(x,y) {
  # Compute AUC using trapezoid numerical method
  n = length(x);
  sumArea = 0;
  for (i in 1:(n-1)){
    h=x[i+1]-x[i];
    if (abs(h) > 1.e-7){
      sumArea = sumArea + 0.5*(y[i+1]+y[i])*h;
    }else{
      sumArea = sumArea;
    }
  }
  return(sumArea)
}

```


Chapter 3

CODA_LOGISTIC_LASSO

3.1 Crohn disease data

```
y<-y_Crohn
summary(y)
```

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

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

```
x<-x_Crohn
```

x is the matrix of microbiome abundances, either absolute abundances (counts) or relative abundances (proportions);

x should not be the matrix of log(counts) or log(proportions). The method itself performs the log-transformation of the abundances.

```
dim(x)
```

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

The rows of x are individuals/samples, the columns are taxa

```
rangLambda(y,x,numVar=12, lambdaIni =0.15)
```

```
## [1] 0.00 0.65 48.00 17.00
## [1] 0.000 0.325 48.000 2.000
## [1] 0.1625 0.3250 12.0000 2.0000
```

```
## $`rang lambdas`
## [1] 0.1625 0.3250
##
## $`num selected variables`
## [1] 12 2
```

It provides a rang of lambda values corresponding to a given number of variables to be selected (numVar).

The default initial lambda is lambdaIni=1.

```
results_codalasso<-coda_logistic_lasso(y,x,lambda=0.19)
```

lambda is the penalization parameter: the larger the value of lambda the fewer number of variables will be selected.

```
results_codalasso
```

```
## $`number of iterations`
## [1] 23
##
## $`number of selected taxa`
## [1] 11
##
## $`indices of taxa with non-zero coeff`
## [1] 0 2 5 9 19 27 31 32 33 39 40 48
##
## $`taxa with non-zero coeff`
## [1] "beta0" "g__Parabacteroides"
## [3] "f__Peptostreptococcaceae_g__" "g__Eggerthella"
## [5] "g__Dialister" "o__Lactobacillales_g__"
## [7] "g__Prevotella" "g__Roseburia"
## [9] "g__Lachnospira" "g__Streptococcus"
## [11] "g__Aggregatibacter" "g__Bilophila"
##
## $`beta non-zero coefficients`
## [1] -1.0023255478 -0.0057464016 0.0202907034 -0.0184875313 -0.0992393846
## [6] -0.0159418386 -0.0030842274 0.2530005235 -0.0085670215 -0.0818265310
## [11] -0.0406982632 0.0002999724
##
## $`proportion of explained deviance`
## [1] 0.1542276
##
## $betas
## [1] -1.0023255478 0.0000000000 -0.0057464016 0.0000000000 0.0000000000
## [6] 0.0202907034 0.0000000000 0.0000000000 0.0000000000 -0.0184875313
## [11] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [16] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 -0.0992393846
## [21] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [26] 0.0000000000 0.0000000000 -0.0159418386 0.0000000000 0.0000000000
## [31] 0.0000000000 -0.0030842274 0.2530005235 -0.0085670215 0.0000000000
## [36] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 -0.0818265310
## [41] -0.0406982632 0.0000000000 0.0000000000 0.0000000000 0.0000000000
## [46] 0.0000000000 0.0000000000 0.0000000000 0.0002999724
```

```
selected_codalasso<-results_codalasso[[4]][-1]
```

```
columns_selected_codalasso<-results_codalasso[[3]][-1]
```

```
#coef<-results[[5]][-1]
```

```
write.csv(data.frame(columns_selected_codalasso,selected_codalasso),"/Generated_datasets/results_codalasso.csv")
```

3.2 BEME day1 data

Chapter 4

CLR_LOGISTIC_LASSO

4.1 Crohn disease data

```
y<-y_Crohn_numeric
```

glmnet() requires y to be numeric.

```
x<-as.matrix(x_Crohn)
```

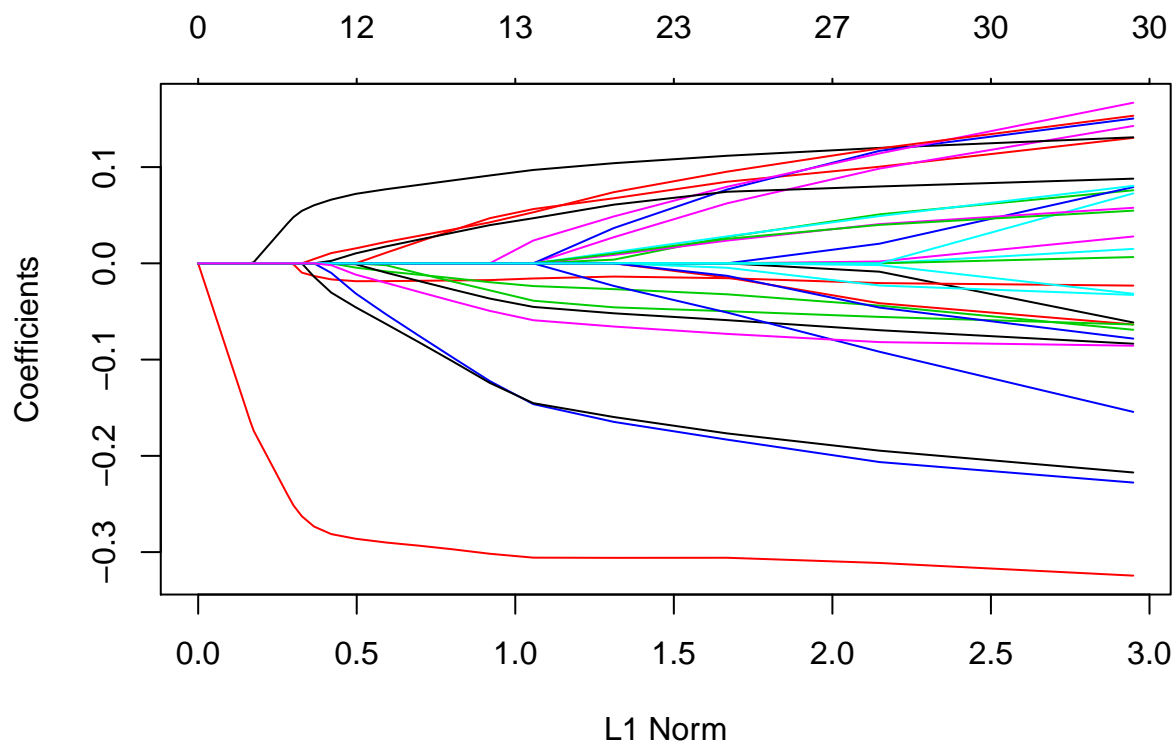
```
# CLR transformation  $Z=\log(x)$ 
```

```
z<-log(x)
```

```
clrx<- apply(z,2,function (x) x-rowMeans(z))
```

```
#rowMeans(clrx)
```

```
clrlasso <- glmnet(clrx, y, standardize=FALSE , alpha=1,family="binomial", lambda=seq(0.015,2,0.01))  
plot(clrlasso)
```



```
print(clrlasso)
```

```
##
## Call:  glmnet(x = clrx, y = y, family = "binomial", alpha = 1, lambda = seq(0.015,      2, 0.01), st
##
##           Df      %Dev Lambda
## [1,]  0 1.769e-15  1.995
## [2,]  0 1.769e-15  1.985
## [3,]  0 1.769e-15  1.975
## [4,]  0 1.769e-15  1.965
## [5,]  0 1.769e-15  1.955
## [6,]  0 1.769e-15  1.945
## [7,]  0 1.769e-15  1.935
## [8,]  0 1.769e-15  1.925
## [9,]  0 7.783e-15  1.915
## [10,] 0 1.769e-15  1.905
## [11,] 0 1.769e-15  1.895
## [12,] 0 1.769e-15  1.885
## [13,] 0 1.769e-15  1.875
## [14,] 0 1.769e-15  1.865
## [15,] 0 1.769e-15  1.855
## [16,] 0 1.769e-15  1.845
## [17,] 0 1.769e-15  1.835
## [18,] 0 1.769e-15  1.825
## [19,] 0 1.769e-15  1.815
## [20,] 0 1.769e-15  1.805
## [21,] 0 7.783e-15  1.795
## [22,] 0 1.769e-15  1.785
## [23,] 0 1.769e-15  1.775
## [24,] 0 1.769e-15  1.765
```

```

## [25,] 0 1.769e-15 1.755
## [26,] 0 1.769e-15 1.745
## [27,] 0 1.769e-15 1.735
## [28,] 0 1.769e-15 1.725
## [29,] 0 1.769e-15 1.715
## [30,] 0 1.769e-15 1.705
## [31,] 0 1.769e-15 1.695
## [32,] 0 1.769e-15 1.685
## [33,] 0 7.783e-15 1.675
## [34,] 0 1.769e-15 1.665
## [35,] 0 1.769e-15 1.655
## [36,] 0 1.769e-15 1.645
## [37,] 0 1.769e-15 1.635
## [38,] 0 1.769e-15 1.625
## [39,] 0 1.769e-15 1.615
## [40,] 0 1.769e-15 1.605
## [41,] 0 1.769e-15 1.595
## [42,] 0 1.769e-15 1.585
## [43,] 0 1.769e-15 1.575
## [44,] 0 1.769e-15 1.565
## [45,] 0 7.783e-15 1.555
## [46,] 0 1.769e-15 1.545
## [47,] 0 1.769e-15 1.535
## [48,] 0 1.769e-15 1.525
## [49,] 0 1.769e-15 1.515
## [50,] 0 1.769e-15 1.505
## [51,] 0 1.769e-15 1.495
## [52,] 0 1.769e-15 1.485
## [53,] 0 1.769e-15 1.475
## [54,] 0 1.769e-15 1.465
## [55,] 0 1.769e-15 1.455
## [56,] 0 1.769e-15 1.445
## [57,] 0 7.783e-15 1.435
## [58,] 0 1.769e-15 1.425
## [59,] 0 1.769e-15 1.415
## [60,] 0 1.769e-15 1.405
## [61,] 0 1.769e-15 1.395
## [62,] 0 1.769e-15 1.385
## [63,] 0 1.769e-15 1.375
## [64,] 0 1.769e-15 1.365
## [65,] 0 1.769e-15 1.355
## [66,] 0 1.769e-15 1.345
## [67,] 0 1.769e-15 1.335
## [68,] 0 1.769e-15 1.325
## [69,] 0 7.783e-15 1.315
## [70,] 0 1.769e-15 1.305
## [71,] 0 1.769e-15 1.295
## [72,] 0 1.769e-15 1.285
## [73,] 0 1.769e-15 1.275
## [74,] 0 1.769e-15 1.265
## [75,] 0 1.769e-15 1.255
## [76,] 0 1.769e-15 1.245
## [77,] 0 1.769e-15 1.235
## [78,] 0 1.769e-15 1.225

```

```
## [79,] 0 1.769e-15 1.215
## [80,] 0 1.769e-15 1.205
## [81,] 0 7.783e-15 1.195
## [82,] 0 1.769e-15 1.185
## [83,] 0 1.769e-15 1.175
## [84,] 0 1.769e-15 1.165
## [85,] 0 1.769e-15 1.155
## [86,] 0 1.769e-15 1.145
## [87,] 0 1.769e-15 1.135
## [88,] 0 1.769e-15 1.125
## [89,] 0 1.769e-15 1.115
## [90,] 0 1.769e-15 1.105
## [91,] 0 1.769e-15 1.095
## [92,] 0 1.769e-15 1.085
## [93,] 0 7.783e-15 1.075
## [94,] 0 1.769e-15 1.065
## [95,] 0 1.769e-15 1.055
## [96,] 0 1.769e-15 1.045
## [97,] 0 1.769e-15 1.035
## [98,] 0 1.769e-15 1.025
## [99,] 0 1.769e-15 1.015
## [100,] 0 1.769e-15 1.005
## [101,] 0 1.769e-15 0.995
## [102,] 0 1.769e-15 0.985
## [103,] 0 1.769e-15 0.975
## [104,] 0 1.769e-15 0.965
## [105,] 0 7.783e-15 0.955
## [106,] 0 1.769e-15 0.945
## [107,] 0 1.769e-15 0.935
## [108,] 0 1.769e-15 0.925
## [109,] 0 1.769e-15 0.915
## [110,] 0 1.769e-15 0.905
## [111,] 0 1.769e-15 0.895
## [112,] 0 1.769e-15 0.885
## [113,] 0 1.769e-15 0.875
## [114,] 0 1.769e-15 0.865
## [115,] 0 1.769e-15 0.855
## [116,] 0 1.769e-15 0.845
## [117,] 0 7.783e-15 0.835
## [118,] 0 1.769e-15 0.825
## [119,] 0 1.769e-15 0.815
## [120,] 0 1.769e-15 0.805
## [121,] 0 1.769e-15 0.795
## [122,] 0 1.769e-15 0.785
## [123,] 0 1.769e-15 0.775
## [124,] 0 1.769e-15 0.765
## [125,] 0 1.769e-15 0.755
## [126,] 0 1.769e-15 0.745
## [127,] 0 1.769e-15 0.735
## [128,] 0 1.769e-15 0.725
## [129,] 0 7.783e-15 0.715
## [130,] 0 1.769e-15 0.705
## [131,] 0 1.769e-15 0.695
## [132,] 0 1.769e-15 0.685
```

```
## [133,] 0 1.769e-15 0.675
## [134,] 0 1.769e-15 0.665
## [135,] 0 1.769e-15 0.655
## [136,] 0 1.769e-15 0.645
## [137,] 0 1.769e-15 0.635
## [138,] 0 1.769e-15 0.625
## [139,] 0 1.769e-15 0.615
## [140,] 0 1.769e-15 0.605
## [141,] 0 7.783e-15 0.595
## [142,] 0 1.769e-15 0.585
## [143,] 0 1.769e-15 0.575
## [144,] 0 1.769e-15 0.565
## [145,] 0 1.769e-15 0.555
## [146,] 0 1.769e-15 0.545
## [147,] 0 1.769e-15 0.535
## [148,] 0 1.769e-15 0.525
## [149,] 0 1.769e-15 0.515
## [150,] 0 1.769e-15 0.505
## [151,] 0 1.769e-15 0.495
## [152,] 0 1.769e-15 0.485
## [153,] 0 7.783e-15 0.475
## [154,] 0 1.769e-15 0.465
## [155,] 0 1.769e-15 0.455
## [156,] 0 1.769e-15 0.445
## [157,] 0 1.769e-15 0.435
## [158,] 0 1.769e-15 0.425
## [159,] 0 1.769e-15 0.415
## [160,] 0 1.769e-15 0.405
## [161,] 0 1.769e-15 0.395
## [162,] 0 1.769e-15 0.385
## [163,] 1 1.275e-03 0.375
## [164,] 1 7.497e-03 0.365
## [165,] 1 1.355e-02 0.355
## [166,] 1 1.944e-02 0.345
## [167,] 1 2.517e-02 0.335
## [168,] 1 3.074e-02 0.325
## [169,] 1 3.615e-02 0.315
## [170,] 1 4.141e-02 0.305
## [171,] 1 4.651e-02 0.295
## [172,] 1 5.146e-02 0.285
## [173,] 1 5.626e-02 0.275
## [174,] 1 6.091e-02 0.265
## [175,] 1 6.541e-02 0.255
## [176,] 1 6.976e-02 0.245
## [177,] 1 7.396e-02 0.235
## [178,] 1 7.802e-02 0.225
## [179,] 2 8.230e-02 0.215
## [180,] 2 8.806e-02 0.205
## [181,] 2 9.359e-02 0.195
## [182,] 2 9.891e-02 0.185
## [183,] 2 1.040e-01 0.175
## [184,] 2 1.089e-01 0.165
## [185,] 2 1.135e-01 0.155
## [186,] 3 1.181e-01 0.145
```

```
## [187,] 3 1.240e-01 0.135
## [188,] 5 1.319e-01 0.125
## [189,] 8 1.422e-01 0.115
## [190,] 10 1.557e-01 0.105
## [191,] 12 1.711e-01 0.095
## [192,] 12 1.860e-01 0.085
## [193,] 12 1.999e-01 0.075
## [194,] 12 2.125e-01 0.065
## [195,] 13 2.256e-01 0.055
## [196,] 20 2.457e-01 0.045
## [197,] 23 2.683e-01 0.035
## [198,] 27 2.910e-01 0.025
## [199,] 30 3.162e-01 0.015
```

```
clrlasso_coef<-coef(clrlasso,s=0.10)
sum(abs(clrlasso_coef)>0)
```

```
## [1] 13
```

```
selected_clrlasso<-which(as.numeric(abs(coef(clrlasso, s=0.1)))>0)[-1];
```

The indices of selected variables ($\text{abs}(\text{coef}) > 0$)

```
selected_clrlasso<-selected_clrlasso-1
taxa_id<-colnames(x_Crohn)[selected_clrlasso]
```

```
write.csv(data.frame(selected_clrlasso,taxa_id),"./Generated_datasets/results_clrlasso_Crohn12.csv")
```

4.2 BEME day1 data

Chapter 5

Selbal: selection of balances

5.1 Crohn disease data

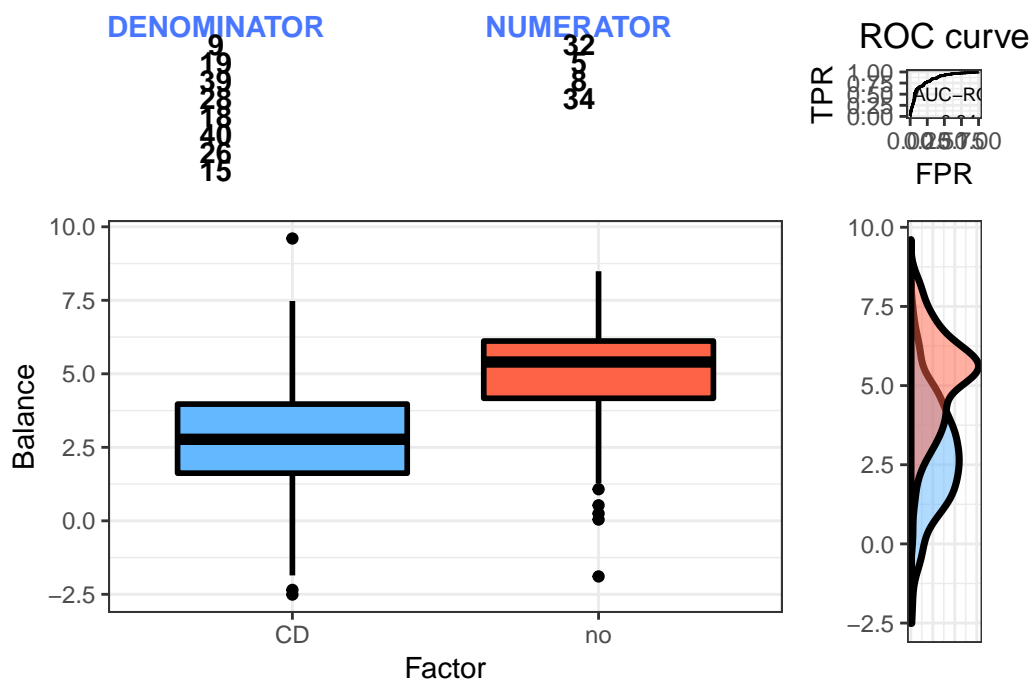
```
y<-y_Crohn
```

For binary outcomes (logistic regression), selbal requires y to be factor.

If y is numeric selbal implements linear regression.

```
x<-x_Crohn
colnames(x)<-(1:ncol(x))

selbal_Crohn<-selbal(x = x, y = y, logt=T, maxV=12)
```



```
selected_selbal<-as.numeric(c(selbal_Crohn[[6]][,1],selbal_Crohn[[6]][,2]))
```

```
## Warning: NAs introduced by coercion
```

```
id.na<-which(is.na(selected_selbal))
selected_selbal<-selected_selbal[-id.na]
selected_selbal<-as.character(selected_selbal)

columns_selected_selbal<-which(colnames(x)%in% selected_selbal)
taxa_id<-colnames(x_Crohn)[columns_selected_selbal]

write.csv(data.frame(columns_selected_selbal, taxa_id), "./Generated_datasets/results_selbal_Crohn12.csv")
```

5.2 BEME day1 data

Chapter 6

Concordance of variables selected by the three methods

6.1 Crohn disease data

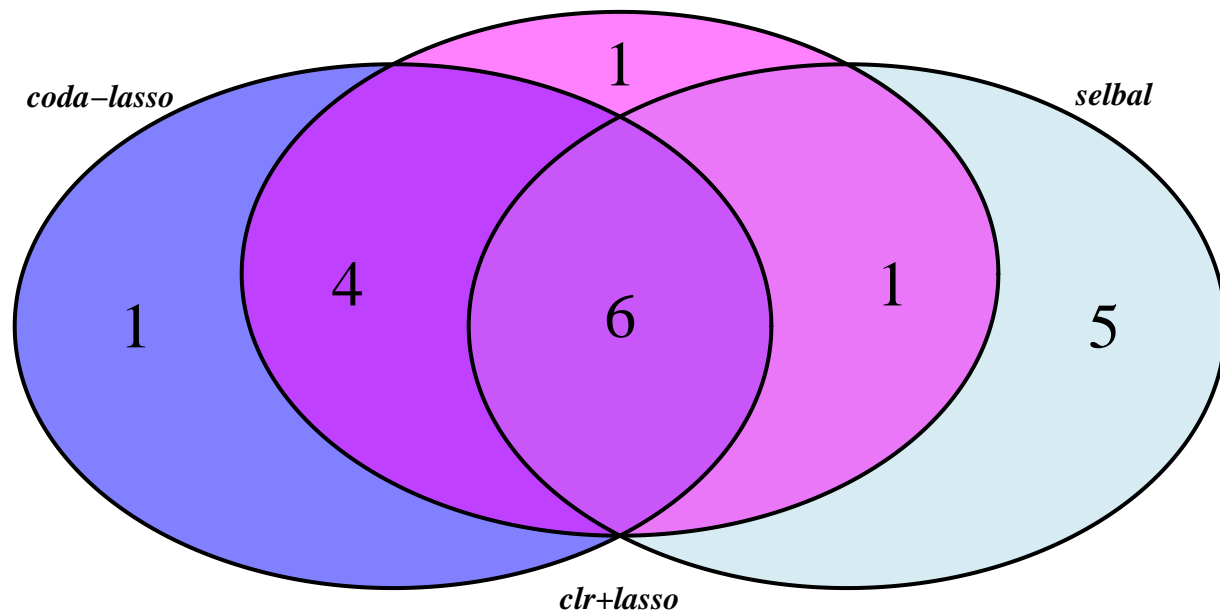
```
d_selbal<-read.csv("./Generated_datasets/results_selbal_Crohn12.csv", header = T)
d_clrlasso<-read.csv("./Generated_datasets/results_clrlasso_Crohn12.csv", header = T)
d_codalasso<-read.csv("./Generated_datasets/results_codalasso_Crohn12.csv", header = T)

#(d_selbal[,2], d_clrlasso[,2], d_codalasso[,2])

taxa_selected<-list(d_clrlasso[,2], d_codalasso[,2], d_selbal[,2])
taxa.id_selected<-list(d_clrlasso[,3], d_codalasso[,3], d_selbal[,3])

venn.plot <- venn.diagram(taxa_selected , NULL, fill=c("magenta", "blue", "lightblue"),
                          alpha=c(0.5,0.5,0.5), cex = 2, cat.fontface=4,
                          category.names=c("clr+lasso", "coda-lasso", "selbal"),
                          main="Concordance of selected taxa for Crohn data")
grid.draw(venn.plot)
```

Concordance of selected taxa for Crohn data



```
taxa.id <- venn(taxa.id_selected, show.plot=FALSE)
taxa <- venn(taxa_selected, show.plot=FALSE)

inters_taxa.id <- attr(taxa.id, "intersections")
inters_taxa <- attr(taxa, "intersections")

lapply(inters_taxa, head)
```

```
## $A
## [1] "10"
##
## $B
## [1] "27"
##
## $C
## [1] "15" "18" "26" "28" "34"
##
## $`A:B`
## [1] "2" "31" "33" "48"
##
## $`A:C`
## [1] "8"
##
## $`A:B:C`
## [1] "5" "9" "19" "32" "39" "40"
```

```
lapply(inters_taxa.id, head)
```

```
## $A
```

```
## [1] "g__Faecalibacterium"
##
## $B
## [1] "o__Lactobacillales_g__"
##
## $C
## [1] "g__Blautia"          "g__Dorea"          "g__Oscillospira"
## [4] "g__Adlercreutzia"    "o__Clostridiales_g__"
##
## $`A:B`
## [1] "g__Parabacteroides" "g__Prevotella"     "g__Lachnospira"
## [4] "g__Bilophila"
##
## $`A:C`
## [1] "g__Bacteroides"
##
## $`A:B:C`
## [1] "f__Peptostreptococcaceae_g__" "g__Eggerthella"
## [3] "g__Dialister"                  "g__Roseburia"
## [5] "g__Streptococcus"              "g__Aggregatibacter"
```

6.2 BEME day1 data