A study of bootstrapping on developing classification model

Xinlei Chen, Guojing Wu and Yujing Yao

April 19, 2019

Abstract

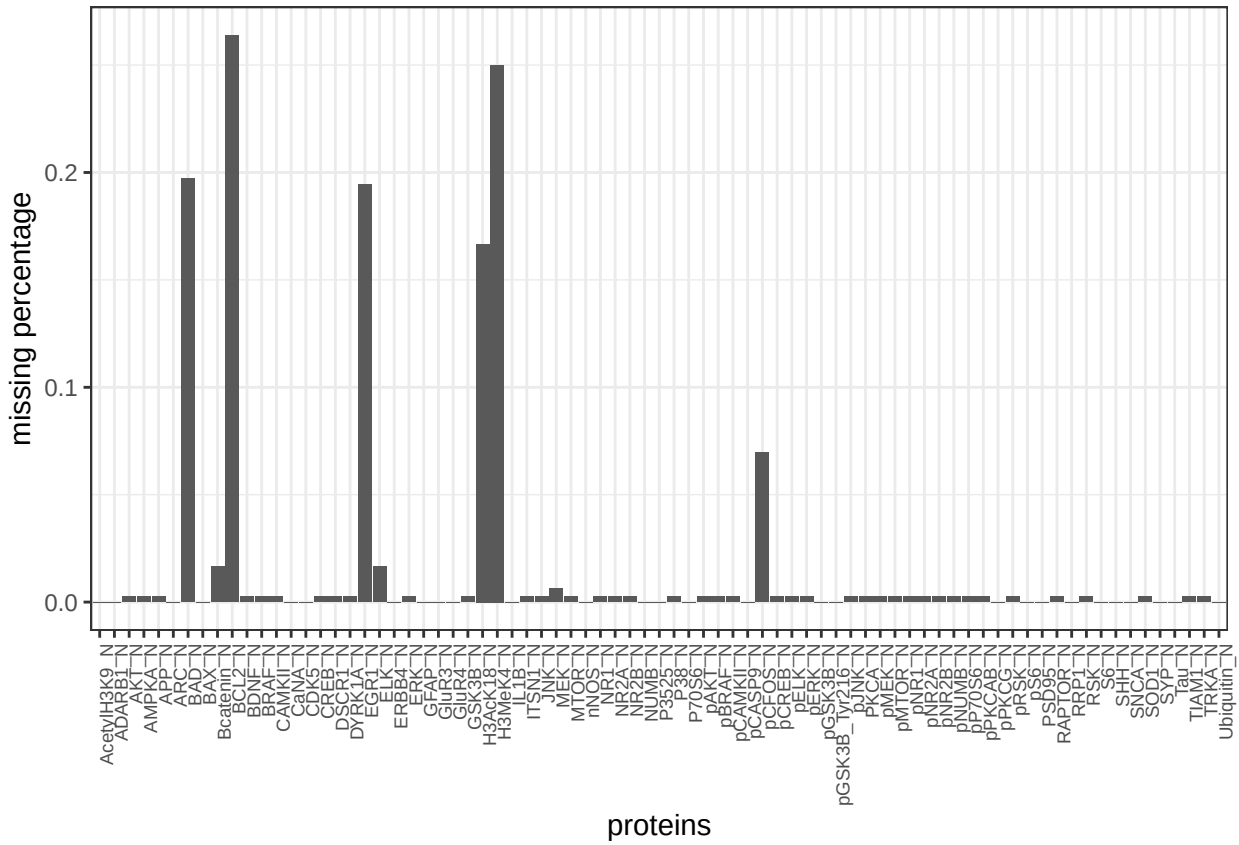
This report discusses a study of bootstrap to develop classification model based on the proteins expression levels. Our goal is to build a predictive model based on logistic regression to facilicate down syndrome diagnosis, and we compared methods including and Pathwise Coordinate Descent with regularized logistic regression and smoothed bootstrap estimation. Our result shows that .....

Background

The data *Down.csv* consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex. It has 1080 rows and 79 columns. The first column **MouseID** identifies individual mice; The column **2-78** are values of expression levels of 77 proteins. Column 79 indicates whether the mouse is a control or has Down syndrome. The goal is to develop classification model based on the proteins expression levels.

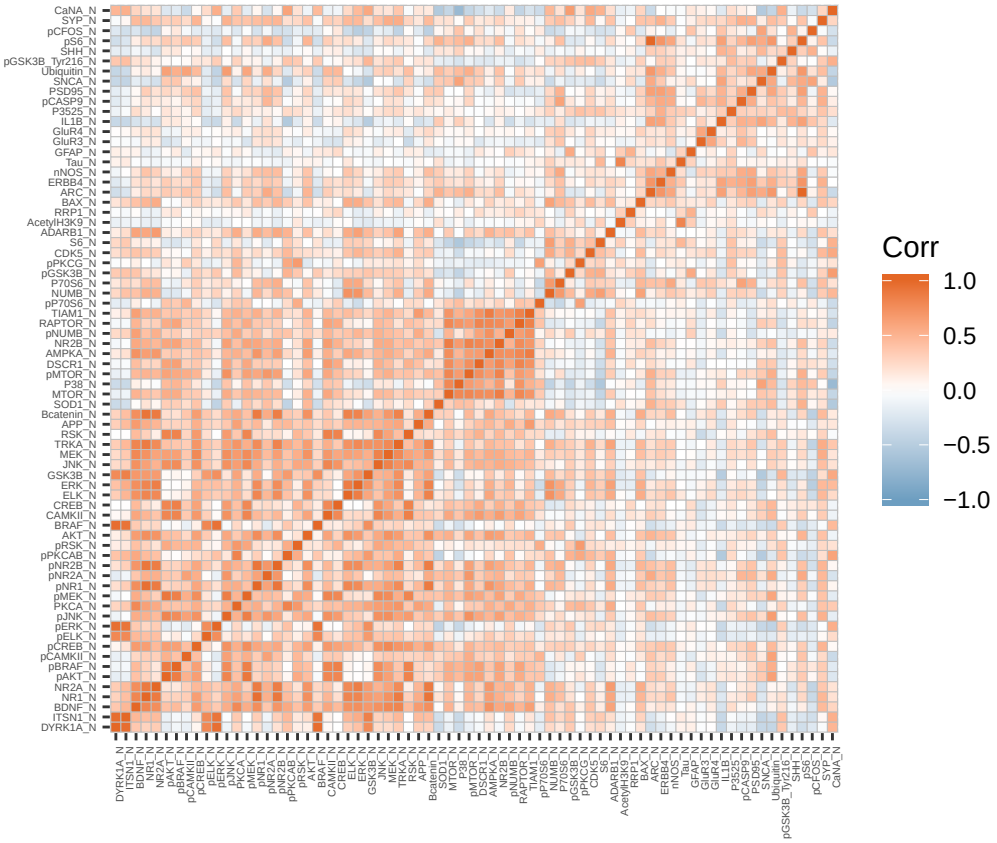
We found that data is missing for some of the covariates, so we deleted those high missing rate(>15%). For those covariates with missing rate < 15%, we assumed them to be missing completely at random(MCAR), so we would analyze the complete cases in the following (Fig 1).

Figure 1: proteins' missing percentage



!!! explain why we need to apply regulized method: singular

Figure 2: missingness plot and multicollinearity plot of the dataset



Method

Logistic Regression

Let y be the vector of n response random variable, X denote the $n \times p$ design matrix (X_i denote the ith row) and β denote the $p \times 1$ coefficient. Let $\eta = E(y) = X\beta$ and given the link function as $g(\eta) = \log \frac{\eta}{1-\eta}$, we have the logistic regression model written as:

$$\log(\frac{\eta}{1-\eta}) = X\beta$$

The likelihood of this logistic regression is:

$$L(\beta; X, y) = \prod_{i=1}^{n} \{(\frac{\exp(X_i\beta)}{1+\exp(X_i\beta)})^{y_i} (\frac{1}{1+\exp(X_i\beta)})^{1-y_i}\}$$

Maximizing the likelihood is equivalent to maximizing the log likelihood:

$$l(\beta) = \sum_{i=1}^{n} \{y_i(X_i\beta) - \log(1+\exp(X_i\beta))\}$$

Pathwise Coordinate Descent with regularized logistic regression

Regularization is the common variable selection approaches for high-dimensional covariates. The best known Regularization is called LASSO, which is to add $L1$-penalty to the objective function. In the context of logistic regression, we are aiming to maximize the penalized log likelihood:

$$\max_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^{n} \{y_i (X_i \beta) - \log(1 + \exp(X_i \beta))\} - \lambda \sum_{j=0}^{p} |\beta_j|$$

for some $\lambda \geq 0$. Here the $x_{i,j}$ are standardized so that $\sum_i x_{i,j}/n = 0$ and $\sum_i x_{i,j}^2/n = 0$.

The Newton algorithm for maximizing the log likelihood amounts to interatively reweighted least squares. Hence if the current estimate of the parameter is $\tilde{\beta}$, we can form a quardratic approximation to the negative log likelihood by taylor expansion around the current estimate, which is:

$$f(\beta) = -\frac{1}{2n} \sum_{i=1}^{n} w_i (z_i - \sum_{j=0}^{p} x_{i,j} \beta_j)^2 + C(\tilde{\beta})$$

where

$$z_i = \tilde{\beta}_0 + x_i^T \tilde{\beta} + \frac{y_i - \tilde{p}(x_i)}{\tilde{p}(x_i)(1 - \tilde{p}(x_i))}, \text{working response}$$
$$w_i = \tilde{p}(x_i)(1 - \tilde{p}(x_i)), \text{working weights}$$

and $\tilde{p}(x_i)$ is evaluated at the current parameters, the last term is constant. The Newton update is obtained by minimizing the $f(\beta)$

The coordinate descent algorithm to solve the penalized weighted least squares problem

$$\min_{\beta \in \mathbb{R}^{p+1}} \{-f(\beta) + \lambda P(\beta)\}$$

The above amounts to a sequence of nested loops:

- outer loop: start with λ that all the coefficients are forced to be zerp, then decrement λ;
- middle loop: update the quardratic $f(\beta)$ using the current estimates of parameters;
- inner loop: run the coordinate descent algorithm on the penalized weighted least square problem.

In our problem, care is taken to avoid coefficients diverging in order to achieve fitted probabilities of 0 or 1 which is the warning message by the R package. When a probability with $\epsilon = 1e - 5$ of 1, we set it to 1, and set the weights to ϵ. 0 is treated similarly.

Smoothing bootstrap and inference

Classical statistical theory ignores model selection in assessing estimation accuracy. Here we consider bootstrap methods for computing standard errors and confidence intervals that take model selection into account. The methodology involves bagging, also known as bootstrap smoothing, to tame the erratic discontinuities of selection-based estimators.

Point estimation:

- First we need to prepare a couple of candidate models
- for each bootstrap in bootstrap with B times, select the best model and get estimates for the coefficient denoted as $t(y^*)$
- smooth $\hat{\mu} = t(y)$ by averaging over the bootstrap replications, defining

$$\tilde{\mu} = s(y) = \frac{1}{B} \sum_{i=1}^{B} t(y^*)$$

Bootstrap smoothing (Efron and Tibshirani 1996), a form of model averaging, is better known as "bagging" in the prediction literature.

smoothed interval In addition to the percentile confidence interval, the nonparametric delta-method estimate of standard deviation for $s(y)$ in the nonideal case

$$\tilde{sd}_B = [\sum_{i=1}^{n} \hat{cov}_j^2]^{1/2}$$

where

$$\hat{cov}_j = \sum_{i=1}^{B} (Y_{ij}^* - Y_{\cdot j}^*)(t_i^* - t_\cdot^*)/B$$

with $Y_{\cdot j}^* = \sum_{i=1}^{B} Y_{ij}^*/B$ and $t_\cdot^* = \sum_{i=1}^{B} t_i^*/B = s(y)$.

Results

Pathwise Coordinate Descent Logistic Lasso

Figure 3: Pathwise Coordinate Descent Lasso

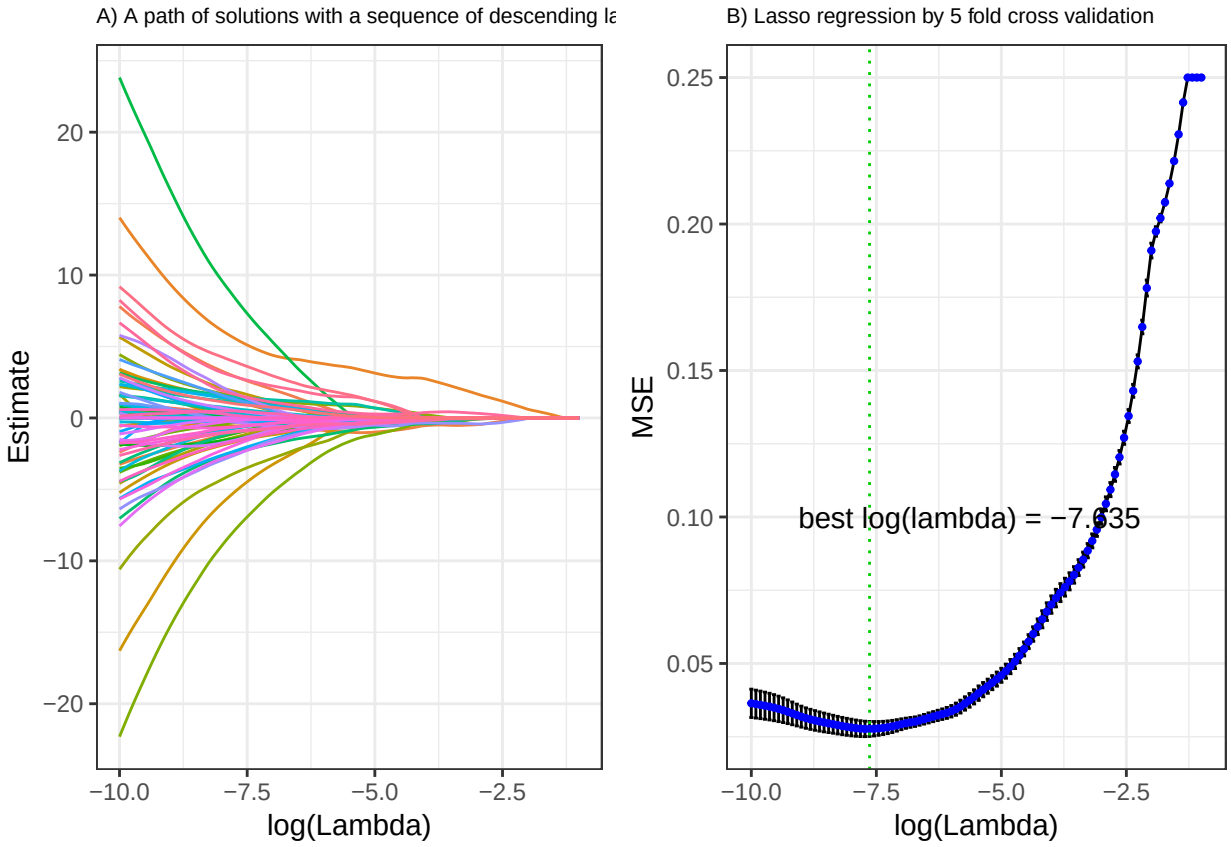


Fig. 3A shows us that as the λ increases, all the variable estimates of parameters shrink accordingly since we penalize all the parameters, though in some. When $\lambda = 0$, the result is the same as least square method and when λ is too large, all the estimates of parameters shrink to 0. **Fig. 3B** shows us the cross validation result for choosing the best λ.

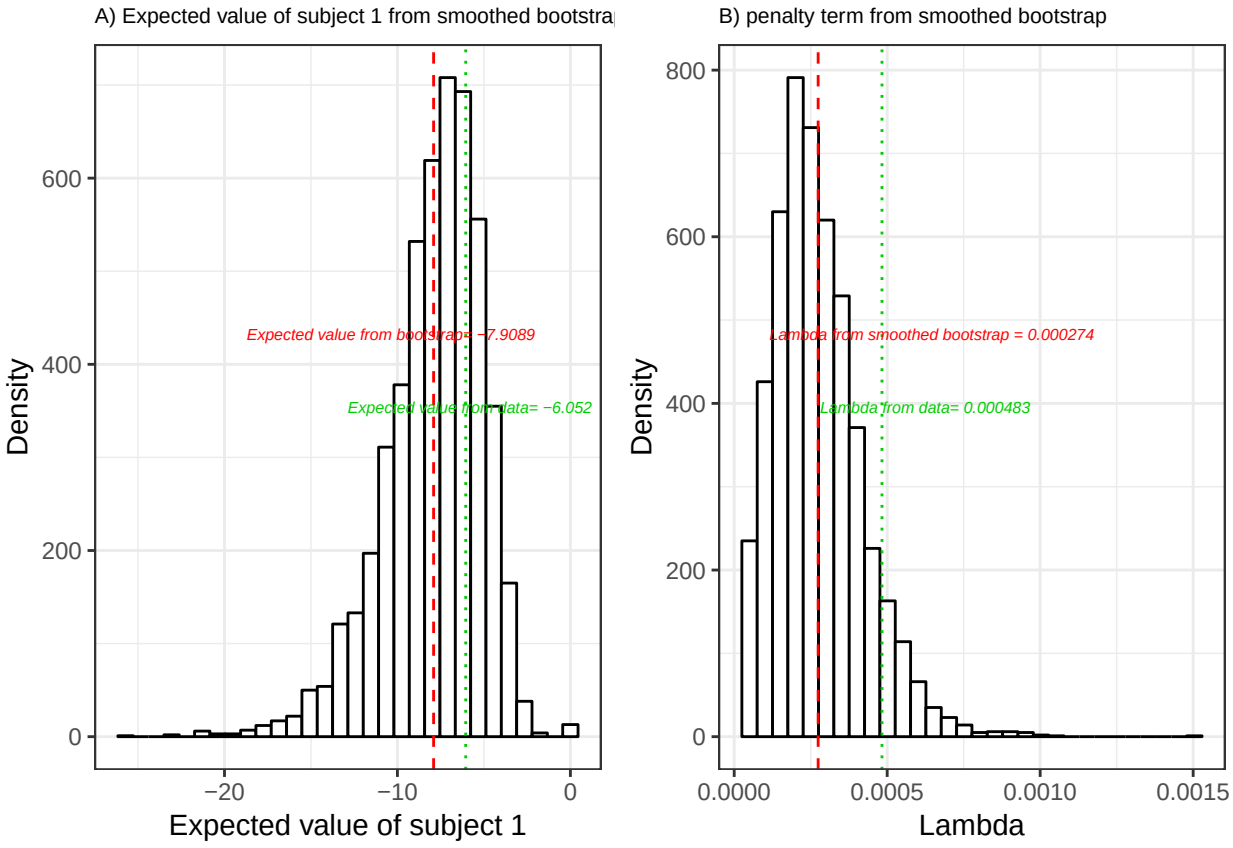
Model selection based on Smoothed Bootstrap for Logistic Lasso

We are still dealing with regularized lasso with this dataset, we wanted to apply smoothed bootstrap in order to get better estimates and inference result.

algorithm:

- bootstrap data from the original dataset
- do cross validation and select the best λ
- suggest $\lambda = \frac{1}{B}\sum_{i=1}^{B}\lambda_i^{(b)}$ bagging?

Figure 4: Lambda selection based on Smoothed Bootstrap for Logistic Lasso



Cross validation for model prediction comparison

We used 10 fold cross-validation to compare two different models, one is with λ selected from data, the other is selected from the bootstrap.

Table 1: The comparison of performance for two models

	Misclassification rate	Mean squred error
Penalty chosen by data	0.0322	0.0239
Penalty selected from smoothed bootstrap	0.0285	0.0244

[a] Dataset: Proteins expression levels of Down syndrome

Significant random variable selection from smoothed bootstrap

for question 4, two method, 1-select covariates based on number of chosen, 2-do hypothesis testing, select significant covariates.

Table 2: Smoothed bootstrap estimation for best model(1)

	origin	prob	coef	sd	lower	upper	lower.new	upper.new
Intercept	0.0949	1.00	0.3143	0.3546	-0.4132	1.2144	-0.3807	1.0093
DYRK1A_N	0.0000	0.04	0.0310	0.1390	0.0000	0.2384	-0.2414	0.3034
ITSN1_N	7.9040	1.00	9.6725	2.1862	4.4201	16.9149	5.3875	13.9575
BDNF_N	1.0655	0.86	1.0811	0.7540	0.0000	3.1864	-0.3967	2.5589
NR1_N	-0.9385	0.90	-2.0857	1.2323	-5.7961	0.0000	-4.5010	0.3296
NR2A_N	0.3742	0.54	0.4882	0.5696	0.0000	2.5152	-0.6282	1.6046
pAKT_N	0.6050	0.82	0.8438	0.6632	0.0000	2.7745	-0.4561	2.1437
pBRAF_N	0.0000	0.62	0.1922	0.4213	-0.7271	1.5147	-0.6335	1.0179
pCAMKII_N	-0.6756	0.93	-1.0979	0.6266	-2.7897	0.0000	-2.3260	0.1302
pCREB_N	1.2113	0.95	1.3620	0.6984	0.0000	3.3009	-0.0069	2.7309
pELK_N	-2.1788	0.99	-2.7419	1.1203	-6.1524	-0.4194	-4.9377	-0.5461
pERK_N	-0.2493	0.54	-0.5806	0.5946	-2.7898	0.0000	-1.7460	0.5848
pJNK_N	-0.0067	0.46	-0.2080	0.3361	-1.4612	0.3040	-0.8668	0.4508
PKCA_N	1.1962	0.82	1.2604	0.9599	0.0000	4.0332	-0.6210	3.1418
pMEK_N	0.0000	0.54	-0.0939	0.5630	-1.8812	1.2504	-1.1974	1.0096
pNR1_N	-2.4536	0.96	-2.6193	1.2315	-5.9502	0.0000	-5.0330	-0.2056
pNR2A_N	0.0000	0.28	-0.0891	0.2675	-1.1664	0.5175	-0.6134	0.4352
pNR2B_N	1.5568	0.90	1.8496	1.0888	0.0000	5.1242	-0.2844	3.9836
pPKCAB_N	-1.4553	0.94	-1.4310	0.7757	-3.7448	0.0000	-2.9514	0.0894
pRSK_N	-2.4102	1.00	-2.8974	0.7656	-5.2917	-1.1120	-4.3980	-1.3968
AKT_N	2.7674	1.00	3.3722	0.8799	1.2074	6.0927	1.6476	5.0968
BRAF_N	-4.8560	1.00	-5.6899	1.7241	-10.9610	-1.8681	-9.0691	-2.3107
CAMKII_N	-1.5901	0.99	-2.3111	0.8889	-4.9441	-0.3901	-4.0533	-0.5689
CREB_N	-1.2469	0.98	-1.3510	0.5539	-2.9474	-0.0163	-2.4366	-0.2654
ELK_N	-3.6872	1.00	-4.6751	1.0545	-8.0746	-2.1847	-6.7419	-2.6083
ERK_N	-7.4243	1.00	-8.7471	1.6856	-14.5871	-4.7828	-12.0509	-5.4433
GSK3B_N	-0.9853	0.88	-1.5983	1.0300	-4.6946	0.0000	-3.6171	0.4205
JNK_N	0.0000	0.56	-0.1163	0.5229	-1.7227	1.2068	-1.1412	0.9086
MEK_N	1.3308	0.98	1.6328	0.7152	0.0194	3.6599	0.2310	3.0346
TRKA_N	3.7756	1.00	5.5845	2.1337	2.0454	11.9326	1.4024	9.7666
RSK_N	-0.6061	0.72	-0.6804	0.6776	-2.6645	0.0600	-2.0085	0.6477
APP_N	5.3514	1.00	7.8719	1.4402	5.0187	13.0828	5.0491	10.6947
Bcatenin_N	0.0000	0.21	0.1476	0.3265	0.0000	1.6599	-0.4923	0.7875
SOD1_N	0.4636	0.62	0.4002	0.4175	0.0000	1.7697	-0.4181	1.2185
MTOR_N	-2.3190	0.99	-2.8751	0.9748	-5.7528	-0.7263	-4.7857	-0.9645

[a] Bootstap time=5000

Chosen probability greater than 80% will be considered as important; Chosen probability greater than 96% will be considered as significant(two exceptions); Chosen probability greater than 96% and confidence interval without zero might be a good choice.

Table 3: Smoothed bootstrap estimation for best model(2)

	origin	prob	coef	sd	lower	upper	lower.new	upper.new
P38_N	0.0000	0.44	-0.1663	0.3615	-1.4422	0.6667	-0.8748	0.5422
pMTOR_N	0.0000	0.42	0.1135	0.3941	-0.9596	1.4826	-0.6589	0.8859
DSCR1_N	1.2781	0.98	1.5514	0.6235	0.0022	3.3412	0.3293	2.7735
AMPKA_N	-0.9954	0.93	-1.4471	0.8265	-3.9602	0.0000	-3.0670	0.1728
NR2B_N	-0.1377	0.57	-0.2951	0.4053	-1.6332	0.3396	-1.0895	0.4993
pNUMB_N	-0.3484	0.74	-0.7193	0.6099	-2.5185	0.0000	-1.9147	0.4761
RAPTOR_N	-1.7634	0.96	-2.1489	1.0112	-4.9061	0.0000	-4.1309	-0.1669
TIAM1_N	2.7743	1.00	3.4095	1.0557	1.1476	6.4593	1.3403	5.4787
pP70S6_N	0.4770	0.83	0.9325	0.7488	0.0000	3.0487	-0.5351	2.4001
NUMB_N	1.4306	0.98	1.8409	0.8104	0.0385	4.1741	0.2525	3.4293
P70S6_N	0.2829	0.67	0.4033	0.4369	-0.1974	1.8139	-0.4530	1.2596
pGSK3B_N	0.4776	0.71	0.3922	0.5633	-0.7969	1.9720	-0.7119	1.4963
pPKCG_N	0.0000	0.50	-0.3093	0.4168	-1.9056	0.2988	-1.1262	0.5076
CDK5_N	0.0000	0.51	0.0790	0.2705	-0.6702	0.9113	-0.4512	0.6092
S6_N	-0.5883	0.87	-0.8153	0.5316	-2.3725	0.0000	-1.8572	0.2266
ADARB1_N	0.1880	0.69	0.3031	0.4307	-0.5048	1.6314	-0.5411	1.1473
AcetylH3K9_N	0.0000	0.33	-0.1398	0.2592	-1.3143	0.2063	-0.6478	0.3682
RRP1_N	-0.8838	0.98	-0.5174	1.0229	-2.2796	2.2449	-2.5223	1.4875
BAX_N	0.0000	0.61	-0.0985	0.3247	-1.1343	0.6766	-0.7349	0.5379
ARC_N	-0.5474	0.68	-0.2956	0.2802	-1.3687	0.0000	-0.8448	0.2536
ERBB4_N	1.4902	1.00	2.0181	0.5463	0.7995	3.6211	0.9474	3.0888
nNOS_N	-0.6199	0.88	-0.6207	0.3817	-1.7324	0.0000	-1.3688	0.1274
Tau_N	1.5522	1.00	2.2831	0.5607	1.0326	4.3017	1.1841	3.3821
GFAP_N	-0.2715	0.83	-0.4706	0.4336	-1.7065	0.2579	-1.3205	0.3793
GluR3_N	-1.3348	1.00	-1.7384	0.4746	-3.2161	-0.7186	-2.6686	-0.8082
GluR4_N	0.1356	0.82	0.3429	0.2973	-0.0463	1.3845	-0.2398	0.9256
IL1B_N	-1.4177	0.99	-1.9549	0.6874	-4.1387	-0.4795	-3.3022	-0.6076
P3525_N	1.0465	0.97	1.2122	0.5624	0.0000	2.7394	0.1099	2.3145
pCASP9_N	-0.1031	0.63	-0.1433	0.3663	-1.2589	0.6892	-0.8612	0.5746
PSD95_N	0.0000	0.45	0.0777	0.3431	-0.8004	1.3061	-0.5948	0.7502
SNCA_N	0.1480	0.75	0.4574	0.4959	-0.4088	1.9387	-0.5146	1.4294
Ubiquitin_N	0.9464	0.97	1.3435	0.6419	0.0000	3.1608	0.0854	2.6016
pGSK3B_Tyr216_N	0.0000	0.47	0.0178	0.3412	-0.9850	1.1540	-0.6510	0.6866
SHH_N	-1.5405	1.00	-1.9631	0.5122	-3.6381	-0.8420	-2.9670	-0.9592
pS6_N	-0.1735	0.93	-0.8120	0.4846	-2.3624	0.0000	-1.7618	0.1378
pCFOS_N	0.1882	0.76	0.3258	0.3444	-0.2652	1.3904	-0.3492	1.0008
SYP_N	-0.9364	0.99	-1.2874	0.4838	-2.6677	-0.1815	-2.2356	-0.3392
CaNA_N	1.7695	0.99	2.3003	0.7909	0.4918	4.8228	0.7501	3.8505

[a] Bootstap time=5000

Conclusions

References

1 Efron, Bradley. "Estimation and accuracy after model selection." Journal of the American Statistical Association 109.507 (2014): 991-1007.

Appendix A

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, comment = "")
library(tidyverse)
# plot
library(ggcorrplot)
library(ggpubr)
library(grid)
library(gridExtra)
# table
library(kableExtra)
# lasso
library(glmnet)
# parallel program
library(parallel)
library(doParallel)
library(foreach)
library(iterators)
nCores <- 3
registerDoParallel(nCores)
set.seed(99999)
options(knitr.table.format = "latex")
theme_set(theme_bw())
oridata <- read.csv("Down.csv")
mydata <- oridata%>%
  dplyr::select(.,-c(BAD_N,BCL2_N,H3AcK18_N,EGR1_N,H3MeK4_N))%>%
  filter(complete.cases(.))
x <- apply(is.na(oridata[,2:78]), 2, sum) / nrow(oridata)
tibble(name = names(x), percentage = as.numeric(x)) %>%
  ggplot(aes(x = name, y = percentage)) +
  geom_bar(stat = "identity") +
  labs(x = "proteins", y = "missing percentage") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 7))
### missing-------
n <- dim(mydata)[1]
ncol <- dim(mydata)[2]
list <- c(2:(ncol - 1))
namesX <- names(mydata)[-c(1,ncol)]
# standardize
dataX <- do.call(cbind, lapply(list, function(x) (mydata[,x] - mean(mydata[,x]))/sd(mydata[,x])))
colnames(dataX) <- namesX
# design matrix
X <- data.frame(dataX) %>%
  mutate(., intercept = 1)
# response
y <- as.vector(ifelse(mydata[,ncol] == "Down syndrome", 1, 0))
dat <- list(y = y, X = as.matrix(X))
# check correlation
corplot <- cor(dataX) %>%
  ggcorrplot(.,ggtheme = ggplot2::theme_gray,
             colors = c("#6D9EC1", "white", "#E46726"),
             tl.cex = 4,
             tl.srt = 90)
```

```r
grid.arrange(corplot,ncol=1)
### regulized logistic regression: middle loop and inner loop
sfun <- function(beta,lambda) sign(beta) * max(abs(beta)-lambda, 0)
reglogitlasso <- function(lambda, dat, start, tol=1e-10, maxiter = 200,...){
  p <- dim(dat$X)[2]
  n <- length(dat$y)
  betavec <- start
  i <- 0
  loglik <- 0
  prevloglik <- Inf
  res <- c(0, loglik, betavec)
  while (i < maxiter && abs(loglik - prevloglik) > tol && loglik < Inf) {
    i <- i + 1
    prevloglik <- loglik
    for (j in 1:p) {
      u <- dat$X %*% betavec
      expu <- exp(u)
      prob <- expu/(expu+1)
      w <- prob*(1-prob)
      w <- ifelse(abs(w-0) < 1e-5, 1e-5, w)
      z <- u + (dat$y-prob)/w
      znoj <- dat$X[,-j] %*% betavec[-j]
      betavec[j] <- sfun(mean(w*(dat$X[,j])*(z - znoj)), lambda)/(mean(w*dat$X[,j]*dat$X[,j]))
    }
    loglik <- sum(w*(z-dat$X %*% betavec)^2)/(2*n) + lambda * sum(abs(betavec))
    res <- rbind(res, c(i, loglik, betavec))}
  return(res)
}
# intial = rep(0, dim(dat$X)[2])
# corres <- reglogitlasso(lambda = exp(-10), dat, start = intial,tol=1e-5)
# start from -1 to -10

## pathwise update: outer loop
path <- function(data,grid){
  start <- rep(0, dim(data$X)[2])
  betas <- NULL
  for (x in 1:length(grid)){
    cor.result <- reglogitlasso(lambda = grid[x],dat = data,start= start)
    lasbeta <- cor.result[nrow(cor.result),3:dim(cor.result)[2]]
    start <- lasbeta
    betas <- rbind(betas,c(lasbeta))
  }
  return(data.frame(cbind(grid,betas)))
}
# path.out <- path(dat,grid=exp(seq(-1,-10, length=100)))
# colnames(path.out) <- c("grid",colnames(X))
path.out <- read.csv("path.out.csv",header = T)%>%
  dplyr::select(.,-c(1))
path.plot <- path.out %>%
  gather(key = par, value = estimate, c(2:dim(path.out)[2])) %>%
  ggplot(aes(x = log(grid), y = estimate, group = par, col = par)) +
  geom_line()+
  ggtitle("A) A path of solutions with a sequence of descending lambda's") +
```

```
  theme(plot.title = element_text(size = 8),legend.position = "none")+
  xlab("log(Lambda)") +
  ylab("Estimate")
##### 5-fold cross-validation to choose beta lambda
cvresult <- function(dat,grid,K){
  n <- dim(dat$X)[1]
  folds <- sample(1:K, n, replace=TRUE)
  start <- rep(0, dim(dat$X)[2])
  cv.error <- vector()
  cv.se <- vector()
  for (x in 1:length(grid)) {
  cv.errors <- vector()
  for (i in 1:K){
    cor.result <- reglogitlasso(lambda = grid[x],dat = list(X=dat$X[folds!=i,],y=dat$y[folds!=i]),
                                start = start)
    lasbeta <- cor.result[nrow(cor.result),3:dim(cor.result)[2]]
    u <- as.matrix(dat$X[folds == i,]) %*% lasbeta
    expu <- exp(u)
    prob <- expu / (1 + expu)
    y <- as.vector(dat$y[folds==i])
    cv.errors[i] = mean((y-prob)^2)
  }
  start <- lasbeta
  cv.error[x] <- mean(cv.errors)
  cv.se[x] <- sqrt(var(cv.errors)/K)
  }
  return(cbind(grid,cv.error,cv.se))
}
# result <- cvresult(dat,grid=exp(seq(-1,-10, length=100)),K=5)
# best.lambda <- result[which.min(result[,2]),1]
best.lambda <- round(0.0004825801 ,6)
result <- read.csv("cvresult.csv",header = T)%>%
  dplyr::select(.,-c(1))
cv.plot <-
    ggplot(result, aes(x=log(result$grid), y=result$cv.error)) +
    geom_errorbar(aes(ymin=result$cv.error-result$cv.se, ymax=result$cv.error+result$cv.se),
                  colour=1) +
    geom_line() +
    geom_point(size=0.8,colour = 4) +
    ggtitle("B) Lasso regression by 5 fold cross validation")+
    theme(plot.title = element_text(size = 8))+
    xlab("log(Lambda)") + ylab("MSE") +
    geom_vline(xintercept = log(best.lambda),col=3,lty=3) +
    annotate("text", log(best.lambda)+2, 0.1, label = paste("best log(lambda) = ", round(log(best.lambda
grid.arrange(path.plot,cv.plot,ncol=2)
#### final estimation for beta
# finlasso <- as.matrix(path(dat,grid = exp(seq(-1,log(best.lambda), length=100))))
# colnames(finlasso) <- c("grid",colnames(X))
# lasso.beta <- finlasso[nrow(finlasso),2:dim(finlasso)[2]]
temp <- read.csv("coeff.csv",header = T)
lenb <- dim(temp)[1]
co.fin.beta <- rbind(temp[lenb,],temp[1:(lenb-1),])
# compare the result of our methods and glmnet
```

```r
logmod <- glmnet(dat$X[,-dim(dat$X)[2]], y = dat$y, alpha=1, family="binomial",lambda = best.lambda)
res.comp <- cbind(co.fin.beta, as.matrix(coef.glmnet(logmod)))
names(res.comp) = c("protein", "LASSO", "glmnet")
############ get the estimate and se by smmothing bootstrap
n <- length(y)
p <- dim(dataX)[2]+1
B <- 5000
taskFun <- function(){
  # interest in: lambda,u_1(plot),beta,betanum, ynum
  ynum <- vector()
  betanum <- vector()
  beta <- vector()
  bootid <- sample(c(1:n),replace = T)
  boot.x <- dataX[bootid,]
  boot.y <- y[bootid]
  for (j in 1:n) ynum[j] <- sum(j==bootid)
  min.lambda <- cv.glmnet(x=boot.x, y=boot.y,alpha=1, family="binomial")$lambda.min
  model <- glmnet(x=boot.x, y=boot.y,alpha=1, family="binomial",lambda = min.lambda)
  beta <- coef(model)
  for (k in 1:p) betanum[k] <- I(coef(model)[k]!=0)
  sub1 <- predict(model,dataX)[1] #,type = "response"
  return(cbind(min.lambda,sub1,t(beta),t(betanum),t(ynum)))
}
# out <- foreach(i = 1:B, .combine = rbind) %dopar% taskFun()
# get sparse matrix, transform into data frame
out.df <- read.csv("bootstrap.csv",header = T)%>%
  dplyr::select(.,-c(1))
colnames(out.df) <- c("min.lambda","subject.1","intercpt",colnames(dataX),
                      sprintf("b[%d]",seq(1:p)),sprintf("n[%d]",seq(1:n)))
best.lambda.s <- mean(out.df$min.lambda)
# plot of expected value of subject 1
model <- glmnet(x=dataX, y=y,alpha=1, family="binomial",lambda = best.lambda)
sub1 <- predict(model,dataX)[1]#,type = "response"
s1 <- ggplot(out.df,aes(x=out.df$subject.1)) +
  geom_histogram(colour="black", fill="white")+
  #geom_density()
  ggtitle("A) Expected value of subject 1 from smoothed bootstrap")+
  theme(plot.title = element_text(size = 8))+
  xlab("Expected value of subject 1") + ylab("Density") +
  geom_vline(xintercept = mean(out.df$subject.1),col=2,lty=2) +
  geom_vline(xintercept = sub1,col=3,lty=3)
s2 <- ggplot(out.df,aes(x=out.df$min.lambda)) +
  geom_histogram(colour="black", fill="white")+
  #geom_density()
  ggtitle("B) penalty term from smoothed bootstrap")+
  theme(plot.title = element_text(size = 8))+
  xlab("Lambda") + ylab("Density") +
  geom_vline(xintercept = best.lambda.s,col=2,lty=2) +
  geom_vline(xintercept = best.lambda,col=3,lty=3)
grid.arrange(s1+
  annotation_custom(grid.text(paste("Expected value from data= ", round(sub1, 4), sep = ""),
                              x=0.5,y=0.5, hjust=0,
                      gp=gpar(col=3,fontsize=6, fontface="italic")))+
```

```
  annotation_custom(grid.text(paste("Expected value from bootstrap= ",
                                    round(mean(out.df$subject.1), 4), sep = ""),
                              x=0.3,y=0.6, hjust=0,
                    gp=gpar(col=2,fontsize=6, fontface="italic"))),
            s2+
  annotation_custom(grid.text(paste("Lambda from data= ", round(best.lambda, 6), sep = ""),
                              x=0.2,y=0.5, hjust=0,
                            gp=gpar(col=3,fontsize=6, fontface="italic")))+
  annotation_custom(grid.text( paste("Lambda from smoothed bootstrap = ",
                                     round(best.lambda.s, 6),sep = ""),
                               x=0.1,y=0.6, hjust=0,
                            gp=gpar(col=2,fontsize=6, fontface="italic"))),
  ncol=2)
################ cross validation for comparison of prediction error
cvcomp <- function(dat,K){
  n <- dim(dat$X)[1]
  folds <- sample(1:K, n, replace=TRUE)
  start <- rep(0, dim(dat$X)[2])
  cv.error.1 <- vector()
  cv.error.2 <- vector()
  cv.mse.1 <- vector()
  cv.mse.2 <- vector()
  for (i in 1:K){
    cv.x <- dat$X[folds!=i,]
    cv.y <- dat$y[folds!=i]
    obs <- glmnet(x=dat$X[folds!=i,],y=dat$y[folds!=i], alpha=1,family="binomial",lambda = best.lambda)
    smooth <- glmnet(x=dat$X[folds!=i,],y=dat$y[folds!=i], alpha=1,family="binomial",lambda = best.lambd
    y.data <- predict(obs,dat$X[folds == i,],type = "response")>0.5
    y.smooth <- predict(smooth,dat$X[folds == i,],type = "response")>0.5
    y.data2 <- predict(obs,dat$X[folds == i,],type = "response")
    y.smooth2 <- predict(smooth,dat$X[folds == i,],type = "response")
    # misclassification rate
    cv.error.1[i] = sum(y.data!=dat$y[folds == i])/length(y[folds == i])
    cv.error.2[i] = sum(y.smooth!=dat$y[folds == i])/length(y[folds == i])
    # mse
    cv.mse.1[i] = mean((dat$y[folds == i]-y.data2)^2)
    cv.mse.2[i] = mean((dat$y[folds == i]-y.smooth2)^2)
  }
  return(list(error = cbind(mean(cv.error.1),mean(cv.error.2)),
              mse = cbind(mean(cv.mse.1),mean(cv.mse.2))))
}
cvresult <- rbind(round(cvcomp(dat,K=10)$error,4),round(cvcomp(dat,K=10)$mse,4))
colnames(cvresult) <- c("Penalty chosen by data","Penalty selected from smoothed bootstrap")
rownames(cvresult) <- c("Misclassification rate","Mean squred error")
kable(t(cvresult), "latex", caption = "The comparison of performance for two models", booktabs = T) %>%
  kable_styling(latex_options = c("hold_position", "scale_down")) %>%
  add_footnote(c("Dataset: Proteins expression levels of Down syndrome"),
               notation = "alphabet")
## select beta based on number of chosen
# colnames(out.df) <- c("min.lambda","subject.1","intercpt",colnames(dataX),
#                       sprintf("b[%d]",seq(1:p)),sprintf("n[%d]",seq(1:n)))
betanum <- out.df[,c((2+p+1):(2+p+p))]
betachosen <- round(t(apply(betanum,2,mean)),2)
```

```
## inference-hypothesis testing
betacoeff <- out.df[,c((2+1):(2+p))]
ynum <- out.df[,c((2+2*p+1):(2+2*p+n))]
meany <- apply(ynum, 2, mean)
coef <- round(apply(betacoeff, 2, mean),4)
sd <- vector()
for (k in 1:p) {
  covj <- (t(ynum)-(as.vector(meany)%*%t(rep(1,B))))%*%(betacoeff[,k]-rep(coef[k],B))
  sd[k] <- round(sqrt((t(covj)%*%covj))/B,4)
}
coeff <- cbind(coef,sd)
newcf <- round(cbind(lower.new = coef -1.96*sd,upper.new = coef +1.96*sd),4)
cf <- round(cbind(lower = apply(betacoeff,2,quantile,0.025),
                  upper = apply(betacoeff,2,quantile,0.975)),4)
est.result <- cbind(origin =round(as.vector(co.fin.beta[,2]),4),
                    prob = as.vector(betachosen),
                    coeff,cf,newcf)
rownames(est.result) <- c("Intercept", colnames(dataX))
kable(est.result[c(1:35),], "latex", caption = "Smoothed bootstrap estimation for best model(1)", bookta
  kable_styling(latex_options = c("hold_position", "scale_down"),
                font_size = 6,full_width = F) %>%
  add_footnote(c("Bootstap time=5000"),
               notation = "alphabet")

kable(est.result[-c(1:35),], "latex", caption = "Smoothed bootstrap estimation for best model(2)", book
  kable_styling(latex_options = c("hold_position", "scale_down"),
                font_size = 6,full_width = F) %>%
  add_footnote(c("Bootstap time=5000"),
               notation = "alphabet")
```