ORF525, Assignment 3, Problem 5

Igor Silin

Data preparation

Let's load the data and do all necessary steps:

```
macro = read.csv("macro-transformed.csv", header=T)
macro = macro[14:719,]                                    # Jan 1960 to Oct 2018
macro[ , "sasdate"] = list(NULL)                          # remove the feature "sasdate"

features_na = colnames(macro)[colSums(is.na(macro)) > 0]  # features with missing entries
macro[ , features_na] = list(NULL)                        # remove features with missing entries
```

```
n = dim(macro)[1]
p = dim(macro)[2] - 1

Y <- as.numeric(macro$UNRATE >= 0)
macro[ , "UNRATE"] = list(NULL)

X = as.matrix(macro)
```

Let's shift the data:

```
X = X[1:(n-1), ]
Y = Y[2:n]
n = n-1

X.saved = X   # save for part (c)
```

(a)

Let's standardize the data. We need to do that because one of the questions asks to find the most important variables. To be able to compare magnitudes of coefficients we need to standardize the predictors. Actually, the algorithms perform standardization themselves, but then they transform the obtained coefficients back to the original scale, so it is dishonest to compare them in this case. So, we have to standardize the predictors manually to be able to track the corresponding coefficients relative to each other.

```
meanX = rep(apply(X, 2 , mean))
stdX = rep(apply(X, 2, sd))

X = X - matrix(meanX, ncol=dim(X)[2], nrow=dim(X)[1], byrow=T)
X = X/matrix(stdX, ncol=dim(X)[2], nrow=dim(X)[1], byrow=T)
```

```
set.seed(525)
library("glmnet")
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

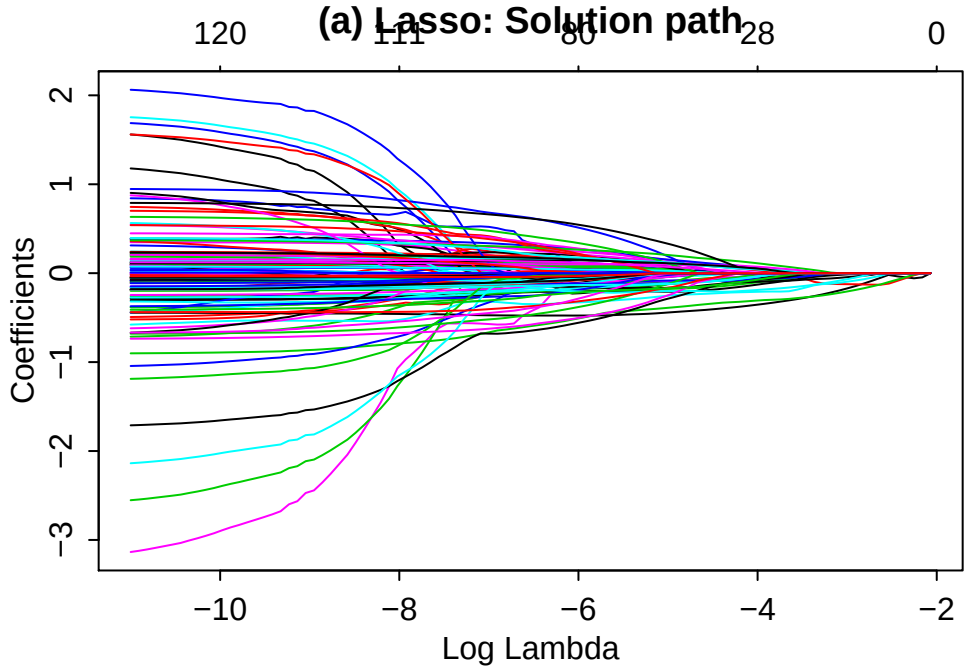
```
## Loaded glmnet 2.0-16
```

```
library("ncvreg")
```

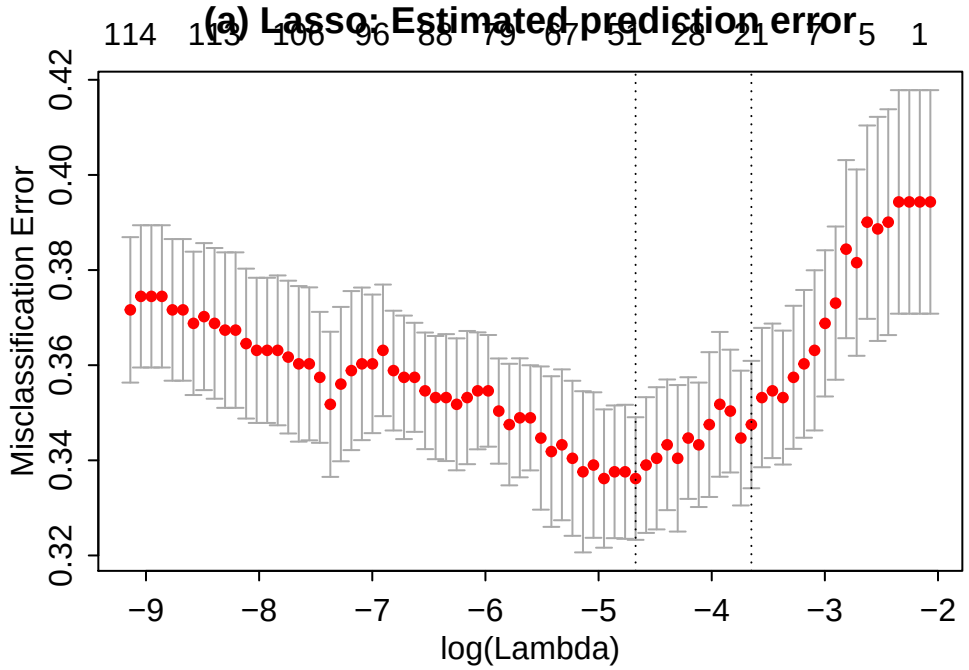
First, we consider Lasso. Let's plot the regularization paths as well as the prediction errors estimated by 10-fold cross-validation:

```
LASSO <- glmnet(X, Y, family="binomial", alpha = 1)
LASSOcv <- cv.glmnet(X, Y, family="binomial", alpha = 1, nfolds = 10, type.measure="class")

par(mfrow = c(2,1), mex=0.5)
par(fig=c(0,10,2,10)/10)
par(mai=c(0.5,1.5,0.5,0.5))
plot(LASSO, xvar=c("lambda"));
title('(a) Lasso: Solution path', line=2);
```




```
plot(LASSOcv)
title('(a) Lasso: Estimated prediction error', line=2)
```



The best model is described by λ that gives the smallest prediction error.

```
sprintf("(a) Lasso: Best lambda = %f", LASSOcv$lambda.min)
```

```
## [1] "(a) Lasso: Best lambda = 0.009347"
```

Let's find the two most influential variables. We output 5 largest coefficients in absolute value:

```
print(sort((abs(coef(LASSOcv, s="lambda.min")))[,1], decreasing=TRUE)[1:5])
```

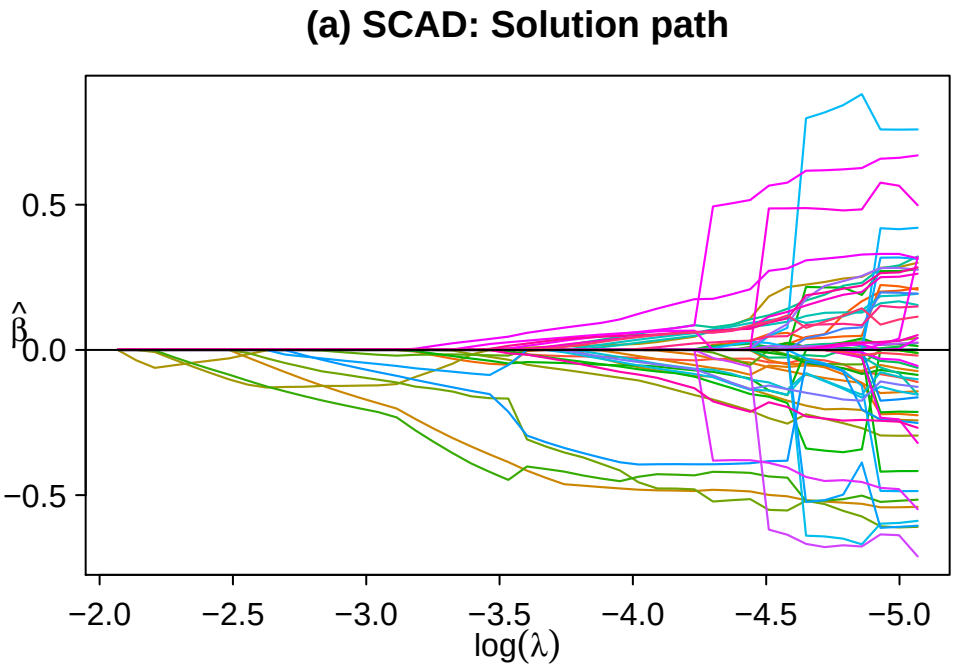
```
## (Intercept)     UEMPLT5       USTPU    CPIULFSL      MANEMP
##   0.5513445   0.4193307   0.3578109   0.2358987   0.2354687
```

So, two coefficients, corresponding to "UEMPLT5" and "USTPU", has absolute values slightly larger than all the other.

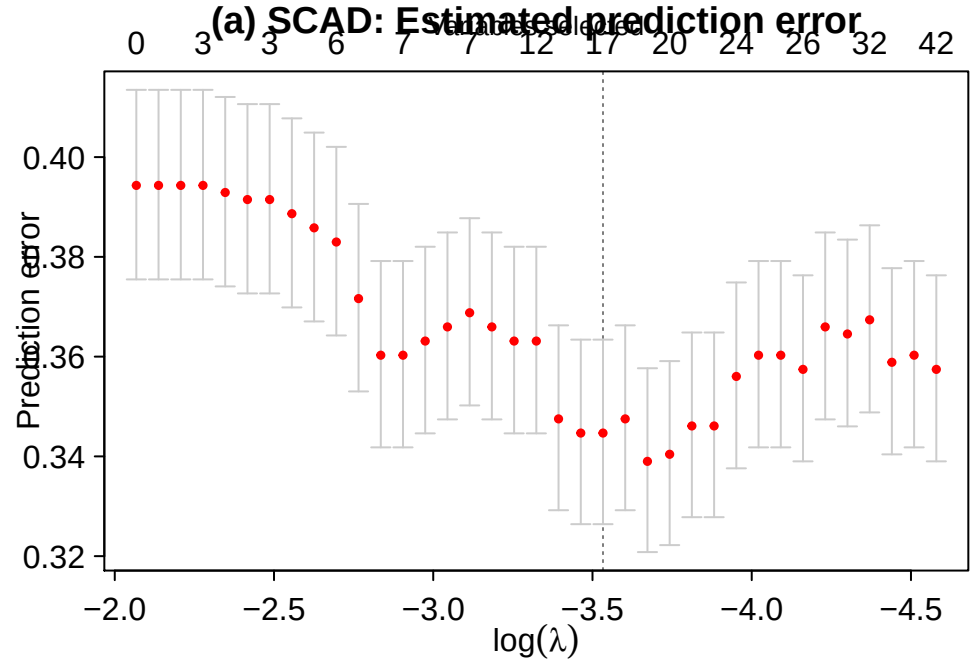
Now we do the same for SCAD. Note that the documentation on *ncvreg* says "The cross-validation error is based on the deviance", unlike *glmnet*, where we were able to specify that the cross-validation should be based on prediction error by typing *type* = "*class*".

```
SCAD = ncvreg(X, Y, penalty="SCAD", family="binomial")
SCADcv = cv.ncvreg(X, Y, penalty="SCAD", family="binomial", nfolds=10)

par(mfrow = c(2,1), mex=0.5)
par(fig=c(0,10,2,10)/10)
par(mai=c(0.5,1.5,0.5,0.5))
plot(SCAD, log.l=TRUE, shade=FALSE);
title('(a) SCAD: Solution path', line=2);
```




```
plot(SCADcv, log.l=TRUE, type="pred")
title('(a) SCAD: Estimated prediction error', line=2)
```



The best model is described by λ that gives the smallest prediction error.

```
sprintf("(a) SCAD: Best lambda = %f", SCADcv$lambda.min)
```

```
## [1] "(a) SCAD: Best lambda = 0.029217"
```

As we see from the regularization path, there are two most influential variables. Let's find them: for small λ, in particular, for the last fitted by the algorithm, we extract the coefficients, take its absolute values, sort them and output 5 largest:

```
print(sort(abs(coef(SCADcv)), decreasing=TRUE)[1:5])
```

```
## (Intercept)       USTPU     UEMPLT5     T10YFFM      MANEMP
##   0.4874592   0.4475877   0.3863470   0.2130463   0.1675136
```

Two coefficients, corresponding to "USTPU" and "UEMPLT5", has absolute values significantly larger than all the other.

(b)

Let's find the coefficients with absolute value > 0.01 and create a maxtix $X1$ with the corresponding features:

```
sign1 = abs(coef(LASSOcv, s=LASSOcv$lambda.min))[2:(p+1)] > 0.01
X1 = X[ , sign1]
```

Next, we run the logisitc regression and summarize the fit:

```
Data1 = data.frame(Y=Y, X1)
model1 = glm(Y ~ ., family=binomial, data=Data1)
summary(model1)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial, data = Data1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8409  -0.9513   0.4013   0.8407   2.2961
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.67135    0.09784   6.862 6.79e-12 ***
## DPCERA3M086SBEA -0.17867    0.11605  -1.540 0.123663
## CMRMTSPLx        0.07460    0.15238   0.490 0.624437
## IPMAT           -0.24543    0.15659  -1.567 0.117026
## IPB51222S       -0.06820    0.09249  -0.737 0.460899
## IPFUELS         -0.15690    0.10030  -1.564 0.117756
## UEMPLT5         -0.51134    0.09656  -5.295 1.19e-07 ***
## UEMP15OV        -0.18968    0.10952  -1.732 0.083291 .
## CLAIMSx          0.23838    0.11246   2.120 0.034038 *
## CES1021000001   -0.23996    0.12486  -1.922 0.054639 .
## MANEMP          -0.29304    0.20078  -1.459 0.144433
## NDMANEMP        -0.29861    0.16243  -1.838 0.066002 .
## USTPU           -0.46274    0.13781  -3.358 0.000786 ***
## USGOVT          -0.14005    0.10917  -1.283 0.199574
## AWOTMAN          0.13989    0.10255   1.364 0.172526
## AWHMAN          -0.05025    0.12256  -0.410 0.681782
## HOUSTMW          0.14162    0.13136   1.078 0.281018
## PERMITS         -0.26707    0.12203  -2.188 0.028635 *
## AMDMNOx          0.07229    0.10167   0.711 0.477101
## ISRATIOx        -0.15891    0.15833  -1.004 0.315523
## AMBSL            0.22381    0.11469   1.951 0.051017 .
## NONBORRES        0.30146    0.32312   0.933 0.350832
## REALLN           0.16932    0.10121   1.673 0.094358 .
## CONSPI          -0.15081    0.11211  -1.345 0.178571
```

```
## S.P.500         -0.17899    0.11079  -1.616 0.106183
## CP3Mx            0.13069    0.23300   0.561 0.574865
## GS1              0.05861    0.21786   0.269 0.787918
## COMPAPFFx        0.35535    0.18219   1.950 0.051124 .
## TB3SMFFM        -0.56839    0.24479  -2.322 0.020233 *
## T10YFFM         -0.19144    0.15228  -1.257 0.208689
## EXCAUSx         -0.17648    0.10430  -1.692 0.090650 .
## PPICMM           0.06930    0.09274   0.747 0.454908
## CPIAUCSL        -0.35836    0.16952  -2.114 0.034512 *
## CPITRNSL        -0.27396    0.17518  -1.564 0.117846
## CPIMEDSL         0.26080    0.09838   2.651 0.008024 **
## CUSR0000SAS      0.12338    0.11308   1.091 0.275205
## CPIULFSL         0.62936    0.18508   3.400 0.000673 ***
## DDURRG3M086SBEA  0.17681    0.09479   1.865 0.062153 .
## CES2000000008   -0.27769    0.10179  -2.728 0.006371 **
## MZMSL            0.20912    0.12111   1.727 0.084226 .
## DTCOLNVHFNM      0.12711    0.09985   1.273 0.203028
## DTCTHFNM         0.15882    0.12175   1.304 0.192084
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 945.61  on 704  degrees of freedom
## Residual deviance: 735.67  on 663  degrees of freedom
## AIC: 819.67
##
## Number of Fisher Scoring iterations: 6
```

Now, we select the features with z-value > 1.96:

```
sign2 = abs(summary(model1)$coefficients[,3])[2:(dim(X1)[2]+1)] > 1.96
X2 = X1[ , sign2]
```

Then, we run the logisitc regression and summarize the fit:

```
Data2 = data.frame(Y=Y, X2)
model2 = glm(Y ~ ., family=binomial, data=Data2)
summary(model2)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial, data = Data2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7580  -1.0911   0.5787   0.9468   1.9187
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.54574    0.08740   6.244 4.25e-10 ***
## UEMPLT5      -0.47089    0.08917  -5.281 1.28e-07 ***
## CLAIMSx       0.19602    0.09154   2.141  0.03226 *
## USTPU        -0.62448    0.10232  -6.103 1.04e-09 ***
## PERMITS      -0.17828    0.08808  -2.024  0.04297 *
## TB3SMFFM     -0.48064    0.11433  -4.204 2.62e-05 ***
```

```
## CPIAUCSL      -0.35075    0.14166  -2.476  0.01329 *
## CPIMEDSL       0.19146    0.08658   2.211  0.02702 *
## CPIULFSL       0.45162    0.13949   3.238  0.00121 **
## CES2000000008 -0.19207    0.09334  -2.058  0.03962 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 945.61  on 704  degrees of freedom
## Residual deviance: 820.87  on 695  degrees of freedom
## AIC: 840.87
##
## Number of Fisher Scoring iterations: 4
```

Then we compute p-values of Likelihood Ratio Tests based on Deviance:

```
pvalue = 1 - pchisq(model1$null.deviance - model1$deviance, df =  dim(X1)[2])
sprintf("Model1 vs Null: P-value of the Likelihood Ratio Test = %f", pvalue)
```

```
## [1] "Model1 vs Null: P-value of the Likelihood Ratio Test = 0.000000"
```

P-value is close to 0, which means that the response does depend on the features included in *model*1. *model*1 is statistically significant.

```
pvalue = 1 - pchisq(model2$null.deviance - model2$deviance, df =  dim(X2)[2])
sprintf("Model2 vs Null: P-value of the Likelihood Ratio Test = %f", pvalue)
```

```
## [1] "Model2 vs Null: P-value of the Likelihood Ratio Test = 0.000000"
```

Agian, P-value is close to 0, which means that the response does depend on the features included in *model*2. *model*2 is statistically significant.

```
pvalue = 1 - pchisq(model2$deviance - model1$deviance, df =  dim(X1)[2] - dim(X2)[2])
sprintf("Model1 vs Model2: P-value of the Likelihood Ratio Test: %f", pvalue)
```

```
## [1] "Model1 vs Model2: P-value of the Likelihood Ratio Test: 0.000001"
```

Such small p-value implies that *model*1 has important features that were removed for *model*2.

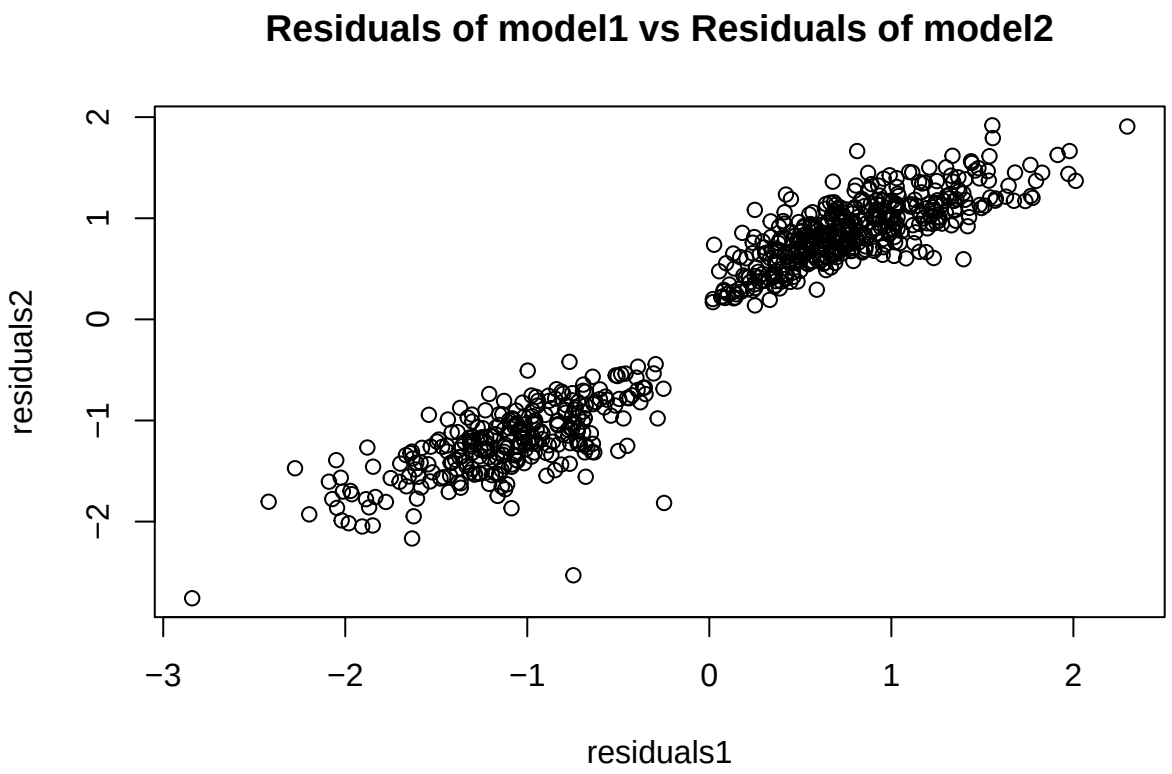
Now we compute residuals for *model*1 and *model*2:

```
theta1 = predict(model1, newdata=data.frame(X1))
mu1 = exp(theta1)/(1+exp(theta1))
residuals1 = (-2*log(mu1))^0.5*(Y==1) - (-2*log(1-mu1))^0.5*(Y==0)

theta2 = predict(model2, newdata=data.frame(X2))
mu2 = exp(theta2)/(1+exp(theta2))
residuals2 = (-2*log(mu2))^0.5*(Y==1) - (-2*log(1-mu2))^0.5*(Y==0)
```

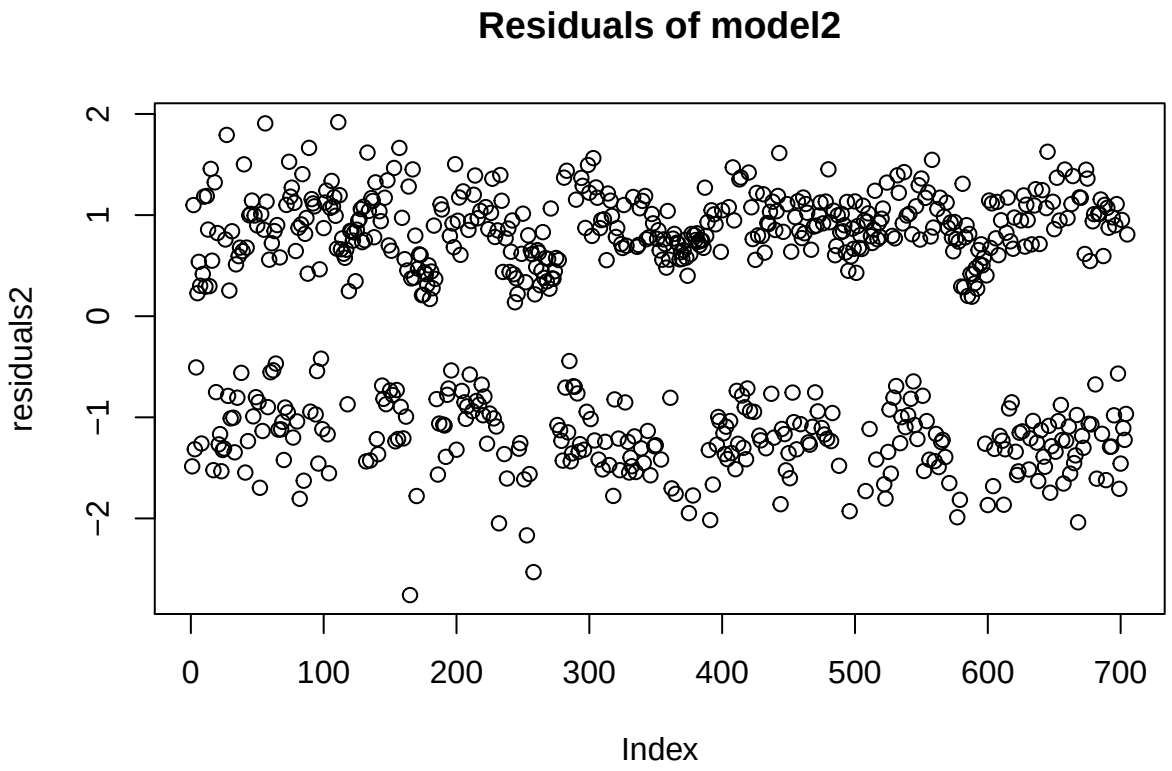
Let's plot the residuals of *model*1 vs residuals of *model*2:

```
plot(residuals1, residuals2)
title('Residuals of model1 vs Residuals of model2')
```



And let's plot the residuals of *model*2:

```
plot(residuals2)
title('Residuals of model2')
```



We do not see any patterns on both these plots.

(c)

Let's get back to the original data, because we need to standardize them using the train data only:

```
X = X.saved
```

Let's split data into the training and testing sets:

```
X.train = X[1:(n-120), ]
X.test = X[(n-120+1):n, ]
Y.train = Y[1:(n-120)]
Y.test = Y[(n-120+1):n]
```

And then standardize (based on the training set only as we would do in real problem setting):

```
meanX = rep(apply(X.train, 2 , mean))
stdX = rep(apply(X.train, 2, sd))

X.train = X.train - matrix(meanX, ncol=dim(X.train)[2], nrow=dim(X.train)[1], byrow=T)
X.train = X.train/matrix(stdX, ncol=dim(X.train)[2], nrow=dim(X.train)[1], byrow=T)

X.test = X.test - matrix(meanX, ncol=dim(X.test)[2], nrow=dim(X.test)[1], byrow=T)
X.test = X.test/matrix(stdX, ncol=dim(X.test)[2], nrow=dim(X.test)[1], byrow=T)
```

First, we look at out-of-sample quality of Lasso:

```
LASSOcv <- cv.glmnet(X.train, Y.train, family="binomial", alpha = 1, nfolds = 10, type.measure="class")
Linear.pred = predict(LASSOcv, newx=X.test, s="lambda.min")
Prob.pred = exp(Linear.pred)/(1+exp(Linear.pred))
Y.pred = as.numeric(Prob.pred > 0.5)

error = sum(Y.pred != Y.test)/120
sprintf('(c) Lasso: Out-of-sample proportion of errors = %f', error)
```

```
## [1] "(c) Lasso: Out-of-sample proportion of errors = 0.366667"
```

Then, we look at the performance of SCAD:

```
SCADcv = cv.ncvreg(X.train, Y.train, penalty="SCAD", family="binomial", nfolds=10)
Linear.pred = predict(SCADcv, X=X.test, lambda=SCADcv$lambda.min)
Prob.pred = exp(Linear.pred)/(1+exp(Linear.pred))
Y.pred = as.numeric(Prob.pred > 0.5)

error = sum(Y.pred != Y.test)/120
sprintf('(c) SCAD: Out-of-sample proportion of errors = %f', error)
```

```
## [1] "(c) SCAD: Out-of-sample proportion of errors = 0.416667"
```