Consider the `Hitters` dataframe from `library(ISLR)`. It includes the salary and the performance of 322 major league baseball players. It is of interest to predict a player's salary based on the player's performance. The dataset includes 19 player's performance measures. Fit the following shrinkage regression models

Ridge regression (alpha=0)

1. The dataset does not include the salary of some players (missing values or NA). Use function `na.omit` to remove all rows with missing values.

2. Use function `glmnet()` with `alpha` equal to zero, to fit $\underbrace{\text{one hundred}}_{\text{seq. length}}$ ridge regression models with $10^{-2} < \lambda < 10^{10}$.
   ↑ ln으로 나오기 때문

   λ=grid=10^(seq(from= , to=, ···)
   하면 안됨. 따로 따로

3. Plot the coefficients of the ridge regression predictors as a function of λ.

4. Split the data set into a training and test set (50%).

5. Fit a ridge regression model using $\lambda = 4$. Find its MSPE.

6. Use function `cv.glmnet` to perform cross validation. Use 10-fold cross validation to find the best value in terms of MSPE for λ. Find the MSPE of the ridge regression model with this value of λ.

7. Use the best λ value to fit a ridge regression model with the full data set.

Lasso Regression (alpha=1)

8. Use function `glmnet()` with `alpha` equal to one, to fit one hundred lasso regression models with $10^{-2} < \lambda < 10^{10}$.

9. Create a coefficients plot to see how much they vary as a function of λ.

10. Perform 10-fold cross validation to find the best value in terms of MSPE for λ. Find the MSPE of the lasso regression model with this value of λ.

11. Use the best λ value to fit a lasso regression model with the full data set.

```
d0 = read.table("example2b.txt",header=T)
d0
#  x1    x2     y
#1  S -0.10 19.19
#2  S  2.53 22.74
#3  S  4.86 23.91
#4  M  0.26  7.07
#5  M  2.55  7.93
#6  M  4.87  8.93
#7  L  0.08 20.63
#8  L  2.62 23.46
#9  L  5.09 25.75

str(d0)
# 'data.frame':   9 obs. of  3 variables:
#  $ x1: Factor w/ 3 levels "L","M","S": 3 3 3 2 2 2 1 1 1
#  $ x2: num  -0.1 2.53 4.86 0.26 2.55 4.87 0.08 2.62 5.09
#  $ y : num  19.19 22.74 23.91 7.07 7.93 ...
x = model.matrix(y~.,d0)
x
#  (Intercept) x1M x1S    x2
#1           1   0   1 -0.10
#2           1   0   1  2.53
#3           1   0   1  4.86
#4           1   1   0  0.26
#5           1   1   0  2.55
#6           1   1   0  4.87
#7           1   0   0  0.08
#8           1   0   0  2.62
#9           1   0   0  5.09

# model matrix converts a dataframe into a matrix
# converting categorical vars into binary columns
# removing response y

1. x = model.matrix(y~.,d0)[,-1]
#  x1M x1S    x2
#1   0   1 -0.10
#2   0   1  2.53
#3   0   1  4.86
#4   1   0  0.26
#5   1   0  2.55
#6   1   0  4.87
#7   0   0  0.08
#8   0   0  2.62
#9   0   0  5.09
```

```
library(ISLR)
dim(Hitters)
# [1] 322  20
sum(is.na(Hitters$Salary))  # missing 59 salaries
# [1] 59
d0=na.omit(Hitters)
dim(d0)
# [1] 263  20
n = nrow(d0)

x=model.matrix(Salary~.,d0)[,-1]  # removing Salary and intercept
y=d0$Salary

d0[1:6,11:20]
#                   CRuns CRBI CWalks League Division PutOuts Assists Errors Salary NewLeague
#-Alan Ashby          321  414    375      N        W     632      43     10  475.0         N
#-Alvin Davis         224  266    263      A        W     880      82     14  480.0         A
#-Andre Dawson        828  838    354      N        E     200      11      3  500.0         N
#-Andres Galarraga     48   46     33      N        E     805      40      4   91.5         N
#-Alfredo Griffin     501  336    194      A        W     282     421     25  750.0         A
#-Al Newman            30    9     24      N        E      76     127      7   70.0         A

dim(x)
# 263  19
x[1:6,11:19]
#                   CRuns CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
#-Alan Ashby          321  414    375       1         1     632      43     10          1
#-Alvin Davis         224  266    263       0         1     880      82     14          0
#-Andre Dawson        828  838    354       1         0     200      11      3          1
#-Andres Galarraga     48   46     33       1         0     805      40      4          1
#-Alfredo Griffin     501  336    194       0         1     282     421     25          0
#-Al Newman            30    9     24       1         0      76     127      7          0

# Salary is not in matrix x
# categoricals now are binary columns
```

1.
2.

3.

4.

```
# Ridge Regression
#============================================================

library(glmnet)

# lambdas from 10^10 to 10^{-2}
a = seq(from=10,to=-2,length=100)
head(a)
# [1] 10.000000  9.878788  9.757576  9.636364  9.515152  9.393939
tail(a)
# [1] -1.393939 -1.515152 -1.636364 -1.757576 -1.878788 -2.000000
grid=10^a

# 100 ridge regressions (one for each value of lambda)
models=glmnet(x,y,alpha=0,lambda=grid)
summary(models)
#          Length Class     Mode
#a0         100   -none-    numeric
#beta      1900   dgCMatrix S4
#df         100   -none-    numeric
#dim          2   -none-    numeric
#lambda     100   -none-    numeric
#dev.ratio  100   -none-    numeric
#nulldev      1   -none-    numeric
#npasses      1   -none-    numeric
#jerr         1   -none-    numeric
#offset       1   -none-    logical
#call         5   -none-    call
#nobs         1   -none-    numeric

# models$lambda = grid
head(grid)
# [1] 10000000000  7564633276  5722367659  4328761281  3274549163  2477076356
tail(grid)
# [1] 0.04037017 0.03053856 0.02310130 0.01747528 0.01321941 0.01000000
head(models$lambda)
# [1] 10000000000  7564633276  5722367659  4328761281  3274549163  2477076356
tail(models$lambda)
# [1] 0.04037017 0.03053856 0.02310130 0.01747528 0.01321941 0.01000000

# plot lambdas
plot(grid,ylim=c(0,20000))

# coef matrix
dim(coef(models))
# [1]  20 100
```

$\ominus$

```
# each col is a ridge regression model
coef(models)[1:20,1]
#  (Intercept)         AtBat          Hits         HmRun          Runs           RBI         Walks
# 5.359257e+02  5.443467e-08  1.974589e-07  7.956523e-07  3.339178e-07  3.527222e-07  4.151323e-07  1
#       CHmRun         CRuns          CRBI        CWalks       LeagueN     DivisionW       PutOuts
# 1.297171e-07  3.450846e-08  3.561348e-08  3.767877e-08 -5.800263e-07 -7.807263e-06  2.180288e-08  3
coef(models)[1:20,2]
#  (Intercept)         AtBat          Hits         HmRun          Runs           RBI         Walks
# 5.359256e+02  7.195940e-08  2.610289e-07  1.051805e-06  4.414196e-07  4.662778e-07  5.487803e-07  2
#       CHmRun         CRuns          CRBI        CWalks       LeagueN     DivisionW       PutOuts
# 1.714783e-07  4.561814e-08  4.707892e-08  4.980911e-08 -7.667601e-07 -1.032074e-05  2.882212e-08  4

# compare reg coef for two lambdas
models$lambda[50]
# [1] 11497.57
models$lambda[60]
# [1] 705.4802

options(digits=4)
coef(models)[,50]
#(Intercept)       AtBat        Hits       HmRun        Runs         RBI       Walks       Years
# 407.356050    0.036957    0.138180    0.524630    0.230702    0.239841    0.289619    1.107703    0
#     CHmRun       CRuns        CRBI      CWalks     LeagueN   DivisionW     PutOuts     Assists
#   0.087546    0.023380    0.024138    0.025015    0.085028   -6.215441    0.016483    0.002613   -0

coef(models)[,60]
#(Intercept)       AtBat        Hits       HmRun        Runs         RBI       Walks       Years
#   54.32520     0.11211     0.65622     1.17981     0.93770     0.84719     1.31988     2.59640
#     CHmRun       CRuns        CRBI      CWalks     LeagueN   DivisionW     PutOuts     Assists
#    0.33777     0.09356     0.09780     0.07190    13.68370   -54.65878     0.11852     0.01606    -

# smaller lambda, larger coefficients (exclude intercept)
```

\# L2 norms 참고 : $\sqrt{\Sigma(\beta_j)^2}$

```
sqrt(sum(coef(models)[-1,50]^2)) # [1] 6.361
sqrt(sum(coef(models)[-1,60]^2)) # [1] 57.11
# smaller lambda, larger L2 norm

# new ridge regression for new lambda
grid[60:100]
# [1] 705.48023 533.66992 403.70173 305.38555 231.01297 174.75284 132.19411 100.00000  75.64633  57.2
#[13]  24.77076  18.73817  14.17474  10.72267   8.11131   6.13591   4.64159   3.51119   2.65609   2.0
#[25]   0.86975   0.65793   0.49770   0.37649   0.28480   0.21544   0.16298   0.12328   0.09326   0.0
#[37]   0.03054   0.02310   0.01748   0.01322   0.01000

# lambda = 50 is not in the set of lambdas
```

```
options(digits=9)
predict(models,s=50,type="coefficients")
# (Intercept)   48.7661032922
#AtBat         -0.3580998594
#Hits           1.9693592865
#HmRun         -1.2782479815
#Runs           1.1458916321
#RBI            0.8038292284
#Walks          2.7161857962
#Years         -6.2183192173
#CAtBat         0.0054478372
#CHits          0.1064895140
#CHmRun         0.6244859561
#CRuns          0.2214984638
#CRBI           0.2186913803
#CWalks        -0.1500245485
#LeagueN       45.9258855144
#DivisionW   -118.2011368164
#PutOuts        0.2502321541
#Assists        0.1215664613
#Errors        -3.2785995446
#NewLeagueN    -9.4966803100

# coef plots
#================================================================

5. plot(models,xvar="lambda"); grid()
# xvar argument requests loglambda in the x-axis
# each curve is a regression coef

# left extreme is OLS regression coefs
options(digits=9)
predict(models,s=0,type="coefficients")
# (Intercept) 299.4446721950
# AtBat        -2.5353835506
# Hits          8.3358501910
# HmRun        11.5983081539
# Runs         -9.0597137055
# RBI           2.4532654580
# Walks         9.2177600598
# Years       -22.9823958271
# CAtBat       -0.1819165075
# CHits        -0.1056568836
# CHmRun       -1.3172135755
# CRuns         3.3115251855
# CRBI          0.0659068925
# CWalks       -1.0724447665
```

```
# LeagueN      59.7558727256
# DivisionW   -98.9439300481
# PutOuts       0.3408327575
# Assists       0.3415544534
# Errors       -0.6531247129
# NewLeagueN   -0.6588292982

# as lambda increases, coefficients shrink to zero

# MSPE
#==============================================================
set.seed(1)
train=sample(1:n, n/2)
test=(-train)
y.test=y[test]

# 100 ridge regressions (one for each value of lambda) using train set
models=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)

# MSPE for lambda=4
yhat=predict(models,s=4,x[test,])  # x must be matrix
mean((yhat-y.test)^2)
# 101036.833

# MSPE for lambda=10^10
yhat=predict(models,s=1e10,newx=x[test,])
mean((yhat-y.test)^2)
# 193253.057

# MSPE for lambda=0 (this is OLS)
yhat=predict(models,s=0,x[test,])
mean((yhat-y.test)^2)
# 114723.615


# 10-fold cross validation to select best lambda
#===================================================================
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=0,nfolds=10)

# Even though we are doing cross validation, we use train set
# to compare with previous MSPE with lambda = 4, 10^10

cv.out$lambda.min
# 211.741585   best lambda
# nfolds = 10 is default
```

```
summary(cv.out)
           Length Class  Mode
lambda     98     -none- numeric
cvm        98     -none- numeric
cvsd       98     -none- numeric
cvup       98     -none- numeric
cvlo       98     -none- numeric
nzero      98     -none- numeric
name        1     -none- character
glmnet.fit 12     elnet  list
lambda.min  1     -none- numeric
lambda.1se  1     -none- numeric
cv.out$lambda
 [1] 227043.5608975 206873.6367056 188495.5529883 171750.1275764 156492.3196057 142589.9732428 12992
 [9] 107864.0856352  98281.7375532  89550.6588648  81595.2251432  74346.5302274  67741.7893896  6172
[17]  51244.1794162  46691.7877517  42543.8180158  38764.3424790  35320.6251321  32182.8381431  2932
[25]  24345.1368319  22182.3819738  20211.7602964  18416.2032176  16780.1584810  15289.4554496  1393
[33]  11565.9123459  10538.4285701   9602.2236210   8749.1885394   7971.9347433   7263.7300323   661
[41]   5494.7453907   5006.6073543   4561.8341558   4156.5733823   3787.3148590   3450.8602452   314
[49]   2610.4492240   2378.5441097   2167.2408065   1974.7091064   1799.2813919   1639.4381921   149
[57]   1240.1748701   1130.0011527   1029.6149647    938.1468090    854.8044321    778.8659623    70
[65]    589.1835376    536.8420960    489.1505237    445.6957394    406.1013583    370.0244329    33
[73]    279.9099138    255.0434885    232.3861279    211.7415848    192.9310460    175.7915836    16
[81]    132.9798863    121.1663197    110.4022378    100.5944073     91.6578774     83.5152442     7
[89]     63.1762195     57.5638183     52.4500073     47.7904932     43.5449175     39.6765071     3
[97]     30.0138226     27.3474773
# train MSE values
cv.out$cvm  Train MSE
 [1] 214354.304 213164.709 212292.016 212085.980 211861.029 211615.551 211347.820 211055.990 210738.
[12] 209606.640 209162.508 208680.758 208158.758 207593.705 206982.973 206323.835 205613.445 204849.
[23] 202206.110 201200.811 200130.174 198992.847 197787.958 196515.199 195174.904 193768.128 192296.
[34] 187526.154 185832.259 184096.387 182325.622 180527.737 178711.409 176885.679 175059.910 173243.
[45] 167943.948 166255.899 164619.813 163042.063 161527.575 160080.311 158703.367 157399.333 156169.
[56] 152919.539 151979.518 151107.992 150302.664 149561.168 148880.595 148258.567 147695.308 147186.
[67] 145961.814 145648.394 145378.467 145155.580 144974.873 144831.441 144726.028 144654.900 144617.
[78] 144664.408 144728.060 144808.990 144907.143 145016.549 145135.627 145265.863 145397.121 145534.
[89] 145946.620 146078.021 146206.384 146327.728 146442.204 146550.186 146649.690 146741.094 146824.

# plot MSE values based on train set
plot(cv.out$cvm,type="l",ylab="train MSE",xlab="")
grid()

which.min(cv.out$cvm)
# 76
cv.out$lambda[76]
# 211.741585
```

```
# minimum train MSE
cv.out$cvm[76]
# 144606.302

abline(v=76,lty=2,col="red")

# MSPE for best lambda (now using test set)
```

10.

```
bestlam=cv.out$lambda.min
yhat=predict(models,s=bestlam,newx=x[test,])
mean((yhat-y.test)^2)
```

MSPE가 낮아졌는지 확인하는 과정.

```
# 96015.5127
# better than MSPE with lambda = 4

# refit with full dataset
```

11.

```
rmodel=glmnet(x,y,alpha=0)
predict(rmodel,type="coefficients",s=bestlam)[1:20,]

#   (Intercept)         AtBat          Hits         HmRun
#  9.8848715652  0.0314399123  1.0088287507  0.1392762360
#          Runs           RBI         Walks         Years
#  1.1132078099  0.8731899006  1.8041022920  0.1307438111
#        CAtBat         CHits        CHmRun         CRuns
#  0.0111397798  0.0648984332  0.4515854621  0.1290004905
#          CRBI        CWalks       LeagueN     DivisionW
#  0.1373771163  0.0290857160 27.1822753486 -91.6341129943
#       PutOuts       Assists        Errors    NewLeagueN
#  0.1914925199  0.0425453624 -1.8124447027  7.2120838997

# ridge model requires all predictors
```

```
# Lasso Regression (alpha=1)
#============================================================

# 100 lasso regressions (one for each value of lambda) using train set
lmodels=glmnet(x[train,],y[train],alpha=1,lambda=grid)

# coef plot
plot(lmodels,xvar="lambda"); grid()
# increasing lambda decreases coefs, some zero

# 10-fold cross validation to select best lambda
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1,nfolds=10)
cv.out$lambda.min
#  16.7801585   best lambda

# MSPE for best lambda
bestlam=cv.out$lambda.min
yhat=predict(lmodels,s=bestlam,newx=x[test,])
mean((yhat-y.test)^2)
# 100743.446

# close to MSPE of ridge regression 96016

# refit with full dataset
lasso.model = glmnet(x,y,alpha=1,lambda=grid)
lasso.coef = predict(lasso.model,type="coefficients",s=bestlam)[1:20,]
lasso.coef
   (Intercept)          AtBat           Hits          HmRun
  18.539484370    0.000000000    1.873538979    0.000000000
          Runs            RBI          Walks          Years
   0.000000000    0.000000000    2.217844394    0.000000000
        CAtBat          CHits         CHmRun          CRuns
   0.000000000    0.000000000    0.000000000    0.207125173
          CRBI         CWalks        LeagueN      DivisionW
   0.413013209    0.000000000    3.266667729 -103.484545814
       PutOuts        Assists         Errors     NewLeagueN
   0.220428413    0.000000000    0.000000000    0.000000000

# 12 coefs are zero

# predictors in lasso model
13. lasso.coef[lasso.coef!=0]   Lasso는 coef가 0인 경우가 있으므로
#   (Intercept)           Hits          Walks          CRuns
#  18.539484370    1.873538979    2.217844394    0.207125173
#          CRBI        LeagueN      DivisionW        PutOuts
#   0.413013209    3.266667729 -103.484545814    0.220428413
```

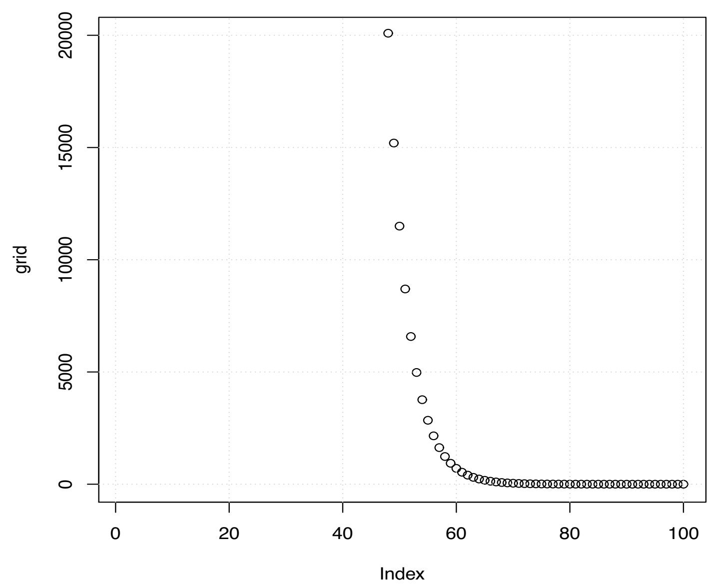

Figure 1: Sequence of lamba values in the y-axis

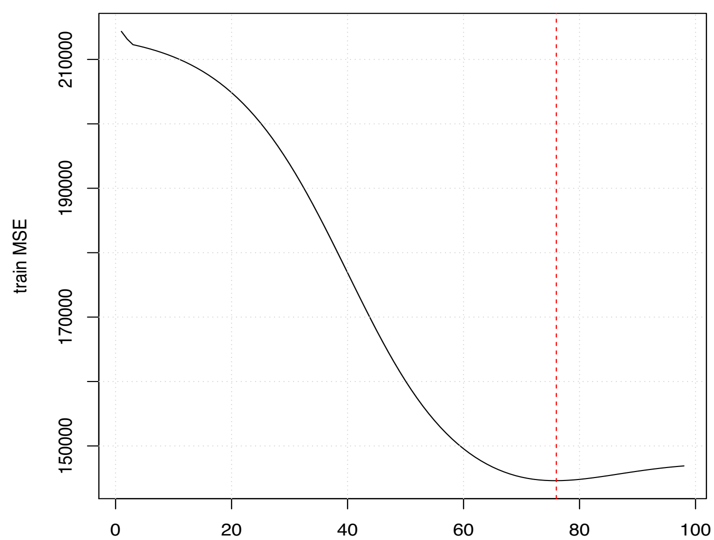

Figure 2: Ridge regression - train MSE values from 10-fold cross-validation

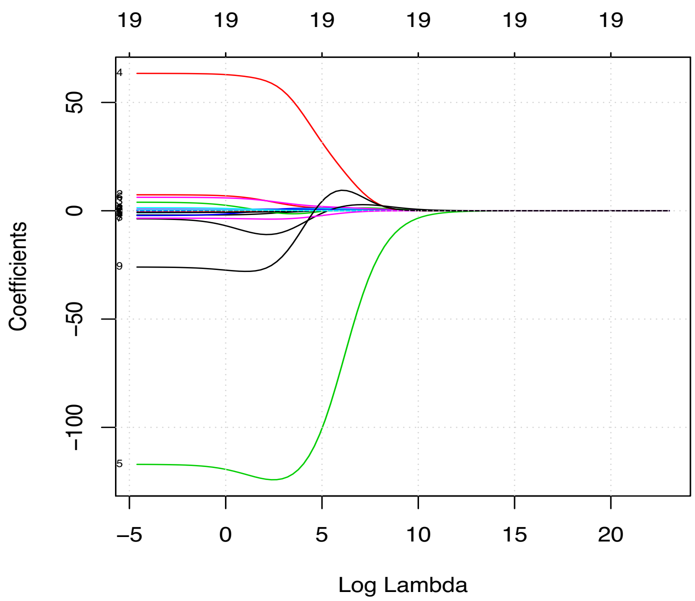

Figure 3: Ridge regression coefficients plot vs lamba

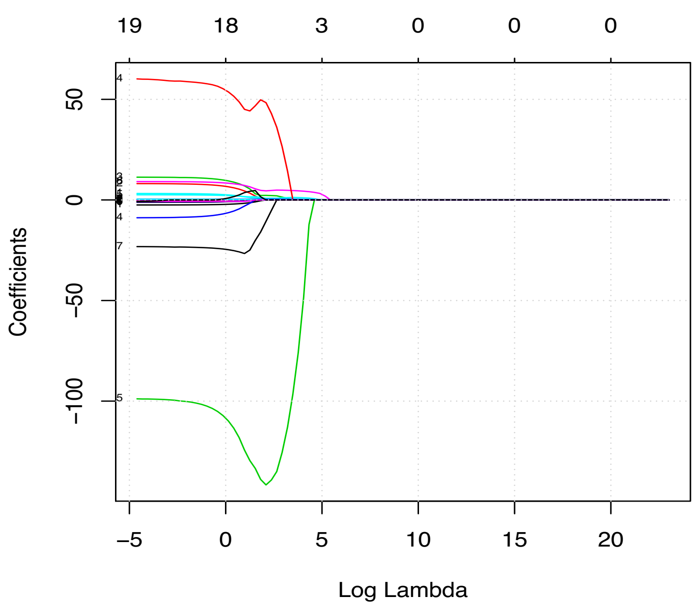

Figure 4: Lasso coefficients plot vs lamba