Homework 3

YutingMei

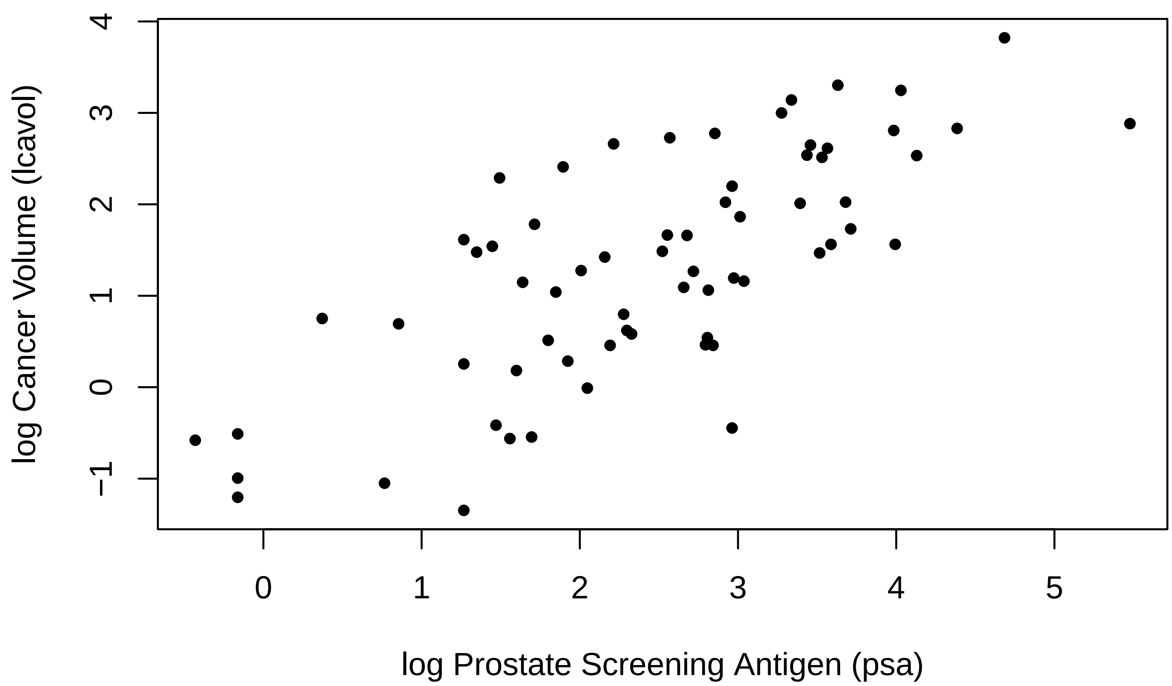
February 25, 2022

- Use the prostate cancer data.

```
## load prostate data
prostate <-
  read.table(url(
    'https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data'))
```

```
## subset to training examples
prostate_train <- subset(prostate, train==TRUE)
```

```
## plot lcavol vs lpsa
plot_psa_data <- function(dat=prostate_train) {
  plot(dat$lpsa, dat$lcavol,
       xlab="log Prostate Screening Antigen (psa)",
       ylab="log Cancer Volume (lcavol)",
       pch = 20)
}
plot_psa_data()
```



- Use the cor function to reproduce the correlations listed in HTF Table 3.1, page 50.

```
dt = prostate_train[, -which(colnames(prostate_train) %in% c('train','lpsa'))]
```

```
rec = cor(dt)
rec[upper.tri(rec, diag = T)] = ''
rec %>% data.frame()
```

```
##                     lcavol             lweight                 age
## lcavol
## lweight  0.300231986902163
## age      0.286324265575102   0.31672346842086
## lbph    0.0631677202229616  0.437041536580065   0.28734644573788
## svi      0.592949130529482   0.18105447828435 0.128902263031428
## lcp      0.692043075322634  0.156828594785416 0.172951397595152
## gleason  0.426414072448678 0.0235582072888314 0.365915122513229
## pgg45    0.483161357103663 0.0741663208573534 0.275805729124431
##                       lbph                svi                 lcp
## lcavol
## lweight
## age
## lbph
## svi      -0.139146799268104
## lcp     -0.0885345593690737 0.671240210303299
## gleason  0.0329921520469304 0.306875372378583 0.476436835735007
```

```
## pgg45   -0.0304038194388767 0.481357740933028 0.662533351565115
##                     gleason pgg45
## lcavol
## lweight
## age
## lbph
## svi
## lcp
## gleason
## pgg45   0.757056496400119
```

- Treat lcavol as the outcome, and use all other variables in the data set as predictors.

With the training subset of the prostate data, train a least-squares regression model with all predictors using the lm function.

```
ft = lm(lcavol ~., prostate_train[,!names(prostate_train) %in% c('train')])
ft
```

```
##
## Call:
## lm(formula = lcavol ~ ., data = prostate_train[, !names(prostate_train) %in%
##     c("train")])
##
## Coefficients:
## (Intercept)      lweight          age         lbph          svi          lcp
##   -2.173357    -0.113370     0.020102    -0.056981     0.035116     0.418455
##     gleason        pgg45         lpsa
##    0.224387    -0.009113     0.575455
```

```
test <- subset(prostate, train==FALSE)
pre = predict(ft, test[,!names(test) %in% c('train')])
```

- Use the testing subset to compute the test error (average squared-error loss) using the fitted least-squares regression model.

```
se = pre - test['lcavol'] %>% as.vector()
se = se^2
mse = se %>% sum / length(pre)
mse
```

```
## [1] 0.5084068
```

- Train a ridge regression model using the glmnet function, and tune the value of lambda (i.e., use guess and check to find the value of lambda that approximately minimizes the test error).

```
## functions to compute testing/training error w/lm
L2_loss <- function(y, yhat)
  (y-yhat)^2
```

```r
## use glmnet to fit ridge
## glmnet fits using penalized L2 loss
## first create an input matrix and output vector
form  <- lcavol ~  lweight + age + lbph + lcp + pgg45 + lpsa + svi + gleason
x_inp <- model.matrix(form, data=prostate_train)
y_out <- prostate_train$lcavol
lambdas = seq(1, 0, -0.05)
fit <- glmnet(x=x_inp, y=y_out, alpha = 0, lambda=lambdas)
# print(fit$beta)
```

```r
## functions to compute testing/training error with glmnet
error <- function(dat, fit, lam, form, loss=L2_loss) {
  x_inp <- model.matrix(form, data=dat)
  y_out <- dat$lcavol
  y_hat <- predict(fit, newx=x_inp, s=lam)  ## see predict.elnet
  mean(loss(y_out, y_hat))
}
```

```r
lam_f = function(l_list){
  sumup = c()
for (i in seq_along(l_list)){
  sumup[i] = error(test, fit, lam=l_list[i], form=form)
}
data.frame(test_error = sumup, l_list)
}
```

```r
ll = lam_f(lambdas)
```

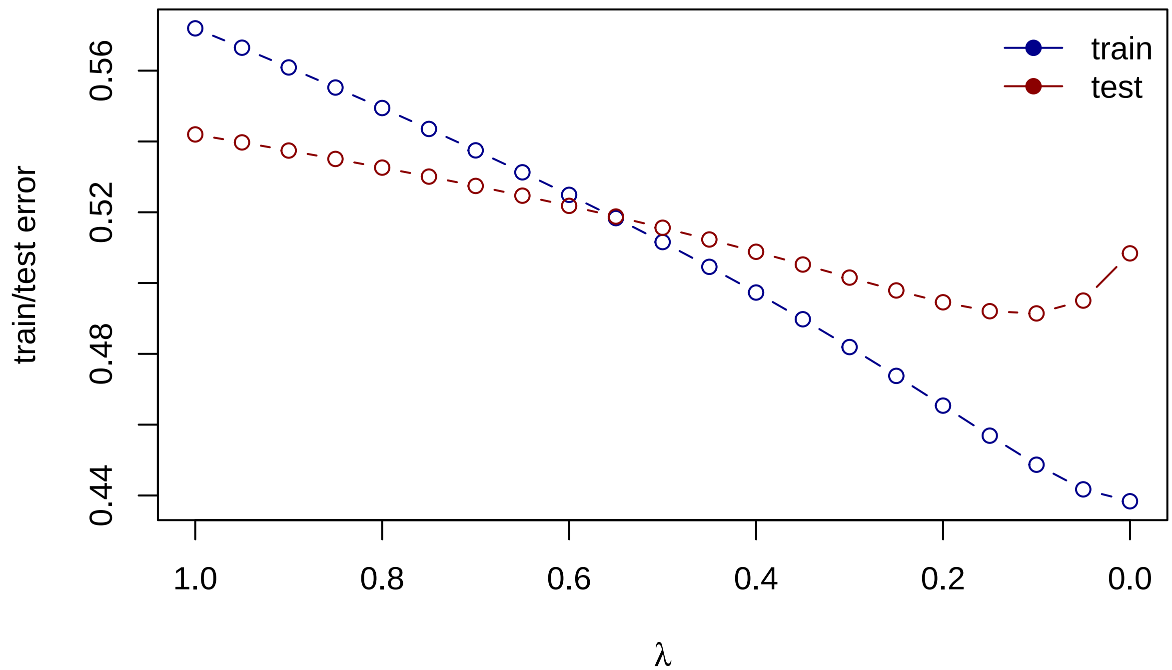
```r
# get the lambda which approximately makes the least test error
ll$l_list[which.min(ll$test_error)]
```

```
## [1] 0.1
```

- Create a figure that shows the training and test error associated with ridge regression as a function of lambda

```r
## compute training and testing errors as function of lambda
err_train_1 <- sapply(fit$lambda, function(lam)
  error(prostate_train, fit, lam, form))
err_test_1 <- sapply(fit$lambda, function(lam)
  error(test, fit, lam, form))
```

```r
## plot test/train error
plot(x=range(fit$lambda),
     y=range(c(err_train_1, err_test_1)),
     xlim=rev(range(fit$lambda)),
     type='n',
     xlab=expression(lambda),
     ylab='train/test error')
points(fit$lambda, err_train_1, type='b', col='darkblue')
points(fit$lambda, err_test_1, type='b', col='darkred')
legend('topright', c('train','test'), lty=1, pch=19,
       col=c('darkblue','darkred'), bty='n')
```




```r
colnames(fit$beta) <- paste('lam =', fit$lambda)
# print(fit$beta %>% as.matrix)
```

- Create a path diagram of the ridge regression analysis, similar to HTF Figure 3.8

```r
df_c = function(x, lambda){
  x^2 / (x^2 + lambda)
}
```

```r
df_all = NULL
s = svd(x_inp, nu = nrow(x_inp), nv = ncol(x_inp))$d
for (i in lambdas){
  df_all$lam = sapply(s, function(s, lam)
  df_c(x = s, lam = i))
  names(df_all) = paste0('lam_', df_all[i])
}
```

```r
df_all = df_all %>% data.frame()
cnames = paste("lambda_", lambdas, sep = '')
colnames(df_all) = cnames
```

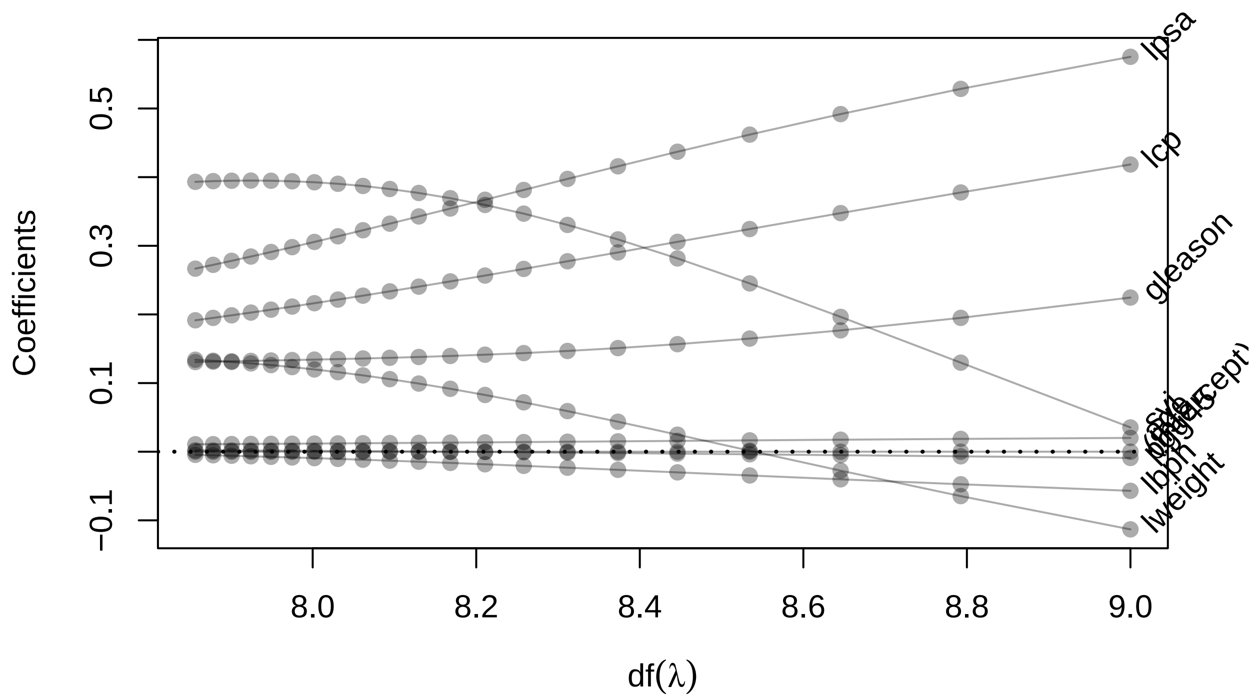
```r
# get degree of freedom of all variables with different lambdas
df_all = data.frame(df_all, row.names = colnames(x_inp))
df_all
```

```
##              lambda_1 lambda_0.95 lambda_0.9 lambda_0.85 lambda_0.8 lambda_0.75
## (Intercept) 0.9999971   0.9999973  0.9999974   0.9999976  0.9999977   0.9999978
## lweight     0.9999778   0.9999789  0.9999800   0.9999811  0.9999822   0.9999833
## age         0.9934849   0.9938086  0.9941325   0.9944567  0.9947811   0.9951057
## lbph        0.9898741   0.9903755  0.9908774   0.9913799  0.9918828   0.9923863
## lcp         0.9826525   0.9835056  0.9843601   0.9852161  0.9860737   0.9869327
## pgg45       0.9686083   0.9701310  0.9716585   0.9731908  0.9747280   0.9762700
## lpsa        0.9141939   0.9181330  0.9221062   0.9261139  0.9301566   0.9342347
## svi         0.8315183   0.8385826  0.8457679   0.8530775  0.8605145   0.8680823
## gleason     0.1762042   0.1837738  0.1920229   0.2010475  0.2109621   0.2219054
##             lambda_0.7 lambda_0.65 lambda_0.6 lambda_0.55 lambda_0.5
## (Intercept)  0.9999980   0.9999981  0.9999983   0.9999984  0.9999986
## lweight      0.9999844   0.9999856  0.9999867   0.9999878  0.9999889
## age          0.9954305   0.9957555  0.9960807   0.9964061  0.9967318
## lbph         0.9928903   0.9933947  0.9938997   0.9944053  0.9949113
## lcp          0.9877932   0.9886552  0.9895188   0.9903838  0.9912504
## pgg45        0.9778169   0.9793687  0.9809255   0.9824872  0.9840539
## lpsa         0.9383487   0.9424992  0.9466865   0.9509112  0.9551738
## svi          0.8757844   0.8836244  0.8916060   0.8997331  0.9080098
## gleason      0.2340460   0.2475920  0.2628024   0.2800039  0.2996149
##             lambda_0.45 lambda_0.4 lambda_0.35 lambda_0.3 lambda_0.25
## (Intercept)   0.9999987  0.9999988   0.9999990  0.9999991   0.9999993
## lweight       0.9999900  0.9999911   0.9999922  0.9999933   0.9999944
## age           0.9970576  0.9973837   0.9977100  0.9980365   0.9983632
## lbph          0.9954178  0.9959249   0.9964324  0.9969405   0.9974491
## lcp           0.9921184  0.9929880   0.9938591  0.9947318   0.9956060
## pgg45         0.9856256  0.9872023   0.9887840  0.9903709   0.9919628
## lpsa          0.9594748  0.9638146   0.9681939  0.9726132   0.9770730
## svi           0.9164401  0.9250285   0.9337794  0.9426974   0.9517874
## gleason       0.3221800  0.3484207   0.3793149  0.4162209   0.4610826
##             lambda_0.2 lambda_0.15 lambda_0.1 lambda_0.0499999999999999
## (Intercept)  0.9999994   0.9999996  0.9999997                 0.9999999
## lweight      0.9999956   0.9999967  0.9999978                 0.9999989
## age          0.9986901   0.9990173  0.9993446                 0.9996722
## lbph         0.9979583   0.9984679  0.9989781                 0.9994888
## lcp          0.9964817   0.9973589  0.9982377                 0.9991181
## pgg45        0.9935599   0.9951622  0.9967696                 0.9983822
## lpsa         0.9815739   0.9861165  0.9907013                 0.9953289
## svi          0.9610543   0.9705036  0.9801405                 0.9899706
## gleason      0.5167833   0.5877909  0.6814201                 0.8105293
##             lambda_0
## (Intercept)        1
## lweight            1
## age                1
## lbph               1
## lcp                1
## pgg45              1
## lpsa               1
## svi                1
## gleason            1
```

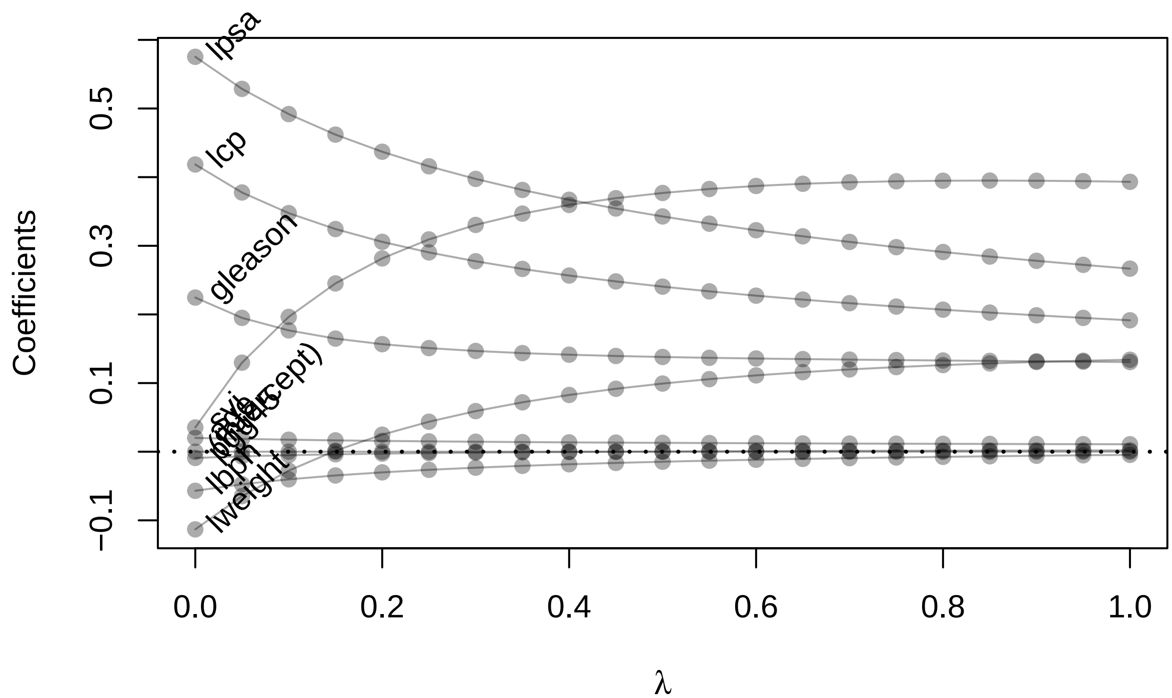
```
# sumup df of all variables
df_l = colSums(df_all) %>% data.frame()
df_l
```

```
##                                  .
## lambda_1                  7.856511
## lambda_0.95               7.878286
## lambda_0.9                7.900903
## lambda_0.85               7.924461
## lambda_0.8                7.949079
## lambda_0.75               7.974898
## lambda_0.7                8.002092
## lambda_0.65               8.030873
## lambda_0.6                8.061504
## lambda_0.55               8.094317
## lambda_0.5                8.129733
## lambda_0.45               8.168303
## lambda_0.4                8.210753
## lambda_0.35               8.258065
## lambda_0.3                8.311604
## lambda_0.25               8.373318
## lambda_0.2                8.446097
## lambda_0.15               8.534414
## lambda_0.1                8.645589
## lambda_0.0499999999999999 8.792489
## lambda_0                  9.000000
```

```
# plot the path of df as x axis
plot(x=range(as.matrix((df_l$.))),
     y=range(as.matrix(fit$beta)),
     type='n',
     xlab=expression(df(lambda)),
     ylab='Coefficients')
for(i in 1:nrow(fit$beta)) {
  points(x=df_l$., y=fit$beta[i,], pch=19, col='#00000055')
  lines(x=df_l$., y=fit$beta[i,], col='#00000055')
}
text(x=9, y=fit$beta[,ncol(fit$beta)],
     labels=rownames(fit$beta),
     xpd=NA, pos=4, srt=45)
abline(h=0, lty=3, lwd=2)
```




```r
plot(x=range(fit$lambda),
     y=range(as.matrix(fit$beta)),
     type='n',
     xlab=expression(lambda),
     ylab='Coefficients')
for(i in 1:nrow(fit$beta)) {
  points(x=fit$lambda, y=fit$beta[i,], pch=19, col='#00000055')
  lines(x=fit$lambda, y=fit$beta[i,], col='#00000055')
}
text(x=0, y=fit$beta[,ncol(fit$beta)],
     labels=rownames(fit$beta),
     xpd=NA, pos=4, srt=45)
abline(h=0, lty=3, lwd=2)
```


Coefficients
λ
0.0
0.2
0.4
0.6
0.8
1.0
−0.1
0.1
0.3
0.5
lpsa
lcp
gleason
svi
lbph
lweight