343 HW 4

Madhuri Raman

11/2/2021

```
library(faraway)
df <- data.frame(divusa)
```

1. (Faraway 7.3)

a)

```
lm.out <- lm(divorce ~ unemployed + femlab + marriage + birth + military, data = df)
# summary(lm.out)
```

```
# eigendecomposition of XTX (not including intercept in X)

x <- model.matrix(lm.out)[,-1] # design matrix
e <- eigen(t(x) %*% x)

e$values # eigenvalues for each covariate
```

```
## [1] 1174600.548   21261.741   16133.842    6206.181    1856.894
```

```
# (go largest to smallest where smaller = more collinearity)

sqrt(e$values[1]/e$values) # condition numbers
```

```
## [1]  1.000000  7.432684  8.532498 13.757290 25.150782
```

Condition numbers measure the relative sizes of the eigenvalues, where zero-valued eigenvalues imply exact collinearity and pretty small eigenvalues denote multi-collinearity.

None of these condition numbers are particularly large (greater than 30). This means that there is likely not an issue of multi-collinearity between any of the covariates. Specifically, when we scale all eigenvalues by the first covariate's eigenvalue, we see that none of them are relatively large compared to the others.

b)

```
(vifX1 <- 1/(1-summary(lm(x[,1] ~ x[,-1]))$r.squared))
```

```
## [1] 2.252888
```

```
(vifX2 <- 1/(1-summary(lm(x[,2] ~ x[,-2]))$r.squared))
```

```
## [1] 3.613276
```

```
(vifX3 <- 1/(1-summary(lm(x[,3] ~ x[,-3]))$r.squared))
```

```
## [1] 2.864864
```

```
(vifX4 <- 1/(1-summary(lm(x[,4] ~ x[,-4]))$r.squared))
```

```
## [1] 2.585485
```

```
(vifX5 <- 1/(1-summary(lm(x[,5] ~ x[,-5]))$r.squared))
```

```
## [1] 1.249596
```

We calculate the variance inflation factors for each covariate. Note that for perfectly orthogonal covariates, the VIF is 1. Here we see that all the VIFs are quite small; in fact they are all less than 4. This suggests that the R-squareds for each covariate were all quite small and that the covariates were reasonably orthogonal. However, since the VIFs are not exactly equal to one, so there may be a bit of collinearity between some covariates. But there is not enough evidence to conclude that collinearity is causing some predictors not to be significant.

c)

Does the removal of insignificant predictors from the model reduce the collinearity?

In the original model, `military` and `unemployed` are not significant at the $\alpha = 0.05$ level. We will rerun the model excluding these two variables and see how the collinearity looks.

```
lm.out.2 <- lm(divorce ~ femlab + marriage + birth, data = df)

x2 <- model.matrix(lm.out.2)[,-1] # design matrix
e2 <- eigen(t(x2) %*% x2)
e2$values
```

```
## [1] 1158413.60   20972.53    6208.46
```

```
sqrt(e2$values[1]/e2$values)
```

```
## [1]  1.000000  7.432012 13.659660
```

```
(vif2X1 <- 1/(1-summary(lm(x2[,1] ~ x2[,-1]))$r.squared))
```

```
## [1] 1.89339
```

```
(vif2X2 <- 1/(1-summary(lm(x2[,2] ~ x2[,-2]))$r.squared))
```

```
## [1] 2.201891
```

```
(vif2X3 <- 1/(1-summary(lm(x2[,3] ~ x2[,-3]))$r.squared))
```

```
## [1] 2.008469
```

After rerunning the model without the originally insignificant covariates, we see that the VIFs are slightly smaller than before but not by much. The condition numbers are also a bit smaller than before, but even in the original model, none of them were over 30, only near 30. It does appear that removing `military` and `unemployed` reduced the collinearity between covariates, but not by very much.

2. (Faraway 8.5)

```
library(MASS)
library(quantreg)
df.sl <- data.frame(stackloss)
```

```
# Least Squares
lm.out.sl <- lm(stack.loss ~ ., data = df.sl)
summary(lm.out.sl)$coefficients
```

```
##                Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -39.9196744 11.8959969 -3.3557234 3.750307e-03
## Air.Flow      0.7156402  0.1348582  5.3066130 5.799025e-05
## Water.Temp    1.2952861  0.3680243  3.5195672 2.630054e-03
## Acid.Conc.   -0.1521225  0.1562940 -0.9733098 3.440461e-01
```

```
# Least absolute deviations
lad.out.sl <- rq(stack.loss ~ ., data = df.sl)
summary(lad.out.sl)$coefficients
```

```
##             coefficients    lower bd     upper bd
## (Intercept) -39.68985507 -41.6197317 -29.67753515
## Air.Flow      0.83188406   0.5127787   1.14117115
## Water.Temp    0.57391304   0.3218235   1.41089812
## Acid.Conc.   -0.06086957  -0.2134829  -0.02891341
```

```
# Huber method
hub.out.sl <- rlm(stack.loss ~ ., data = df.sl)
summary(hub.out.sl)$coefficients
```

```
##                   Value Std. Error    t value
## (Intercept) -41.0265311  9.8073472 -4.1832445
## Air.Flow      0.8293739  0.1111803  7.4597166
## Water.Temp    0.9261082  0.3034081  3.0523516
## Acid.Conc.   -0.1278492  0.1288526 -0.9922126
```

```
# Least trimmed squares
(lts.out.sl <- ltsreg(stack.loss ~ ., data = df.sl))
```

```
## Call:
## lqs.formula(formula = stack.loss ~ ., data = df.sl, method = "lts")
##
## Coefficients:
## (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
##  -3.631e+01    7.292e-01    4.167e-01   -3.589e-16
##
## Scale estimates 0.9149 1.0148
```

When we compare the four methods, we see that the standard errors of the three covariate coefficients for LAD are higher than for least squares, but the standard errors for Huber are all lower than least squares. For LAD and Huber, the coefficients on air flow and acid conc are also higher compared to in least squares. Interestingly, the coefficient for acid conc in LTS is extremely close to 0 and the intercept term coefficient is much smaller than the other methods as well, but we don't have access to standard errors for this method.

The largest studentized residual is -3.3305. This is more negative than the Bonferroni critical value of -3.2451, so we conclude that data point #21 is an outlier. Now we will use least squares but excluding this data point.

```r
lm.out.sl.2 <- lm(stack.loss ~ ., data = df.sl[-21,])
summary(lm.out.sl.2)$coefficients
```

```
##                Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -43.7040310  9.4915652 -4.6045125 2.930291e-04
## Air.Flow      0.8891082  0.1188476  7.4810750 1.309021e-06
## Water.Temp    0.8166199  0.3250294  2.5124489 2.308829e-02
## Acid.Conc.   -0.1071414  0.1245414 -0.8602872 4.023381e-01
```

When we rerun least squares on the data without this outlier point, we notice that the standard errors of the coefficient estimates are all smaller than they previously were for this method. The coefficients are slightly smaller but pretty much the same as before, but at least now we can feel more comfortable comparing this model to the more robust methods of LAD, LTS, and Huber since we removed the outlier point.

3.

a)

Yes. For the ordinary least squares estimator $\hat{\beta}$, although there are unequal variances in the sample, $\hat{\beta}$ is still going to be an unbiased estimator for β because the expectation of $\hat{\beta}$ is only dependent on fixed X values.

b)

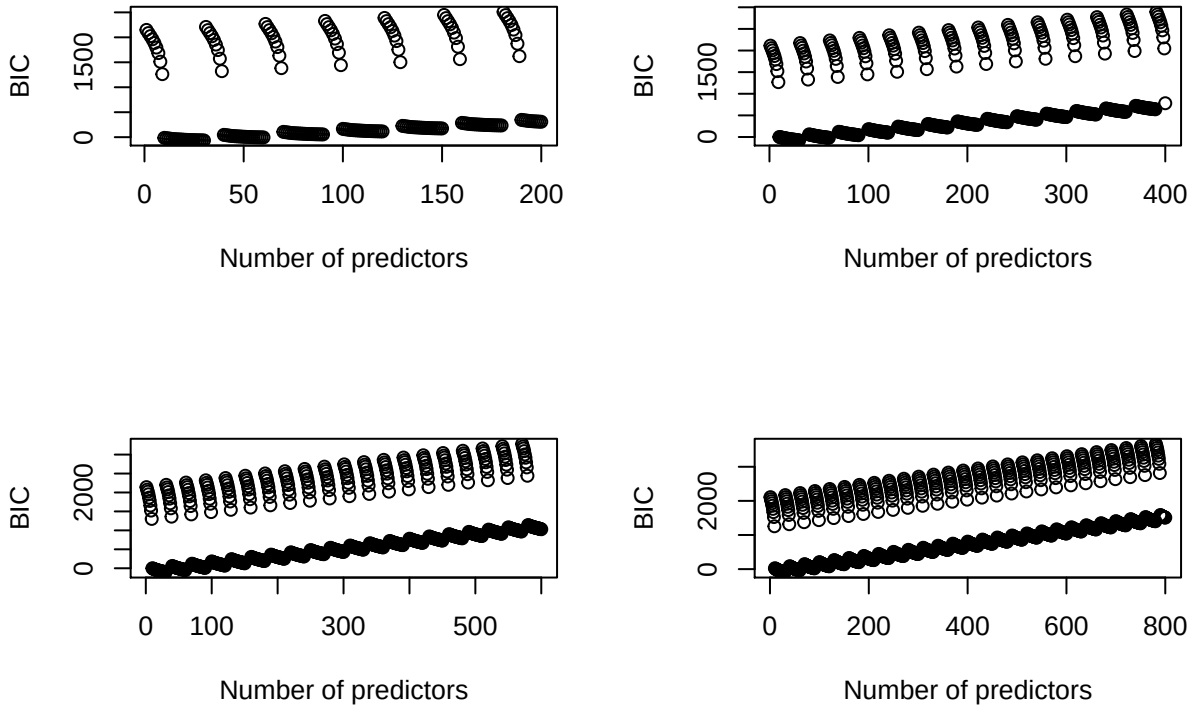
No. When the variance differs among different portions of the sample, specifically here when there is some $\sigma_1^2 \neq \sigma_2^2$, we can no longer write a singular $\sigma^2 = ||U^T Y||_2^2$ where U has dimension $n \times (n-p)$ and is an orthonormal basis for the subspace $span(x)^{\perp}$. Instead, Y is now distributed normally (we assume) with mean $X\beta$ and variance σ_1^2 with probability n_1/n and σ_2^2 with probability n_2/n. Thus, $U^T[(X^TX)^{-1}X^T]^T \neq 0$ anymore so U^TY is no longer orthogonal to $(X^TX)^{-1}X^TY$ since the U's are no longer constant for all data points. Thus, $\hat{\sigma}^2$ and $\hat{\beta}$ are not independent in this situation.

4.

a)

```r
library(leaps)
par(mfrow = c(2,2))
set.seed(819)

n <- 400
possible_ps <- c(200, 400, 600, 800) # aka number of covariates

for (p in possible_ps){
  X <- matrix(data = rnorm(n*p,mean = 0, sd = 1), nrow = n, ncol = p)
  norm_vec <- function(x) {sqrt(sum(x^2))}
  Xnorms <- apply(X, MARGIN = 2, FUN = norm_vec)
  Xnormed <- sweep(X, 2, Xnorms, FUN = '/')
  true_beta_vector <- c(rep(5, 10), rep(0, p - 10))
  Y <- X %*% true_beta_vector + rnorm(n = n, mean = 0, sd = 1)
```

```
  b <- regsubsets(x = X, y = Y, nvmax = 30, method = "forward")
  rs <- summary(b)
  # summary(rs)
  # rs$which
  BIC <- n*log(rs$rss/n) + (1:p + 1)*2
  plot(BIC ~ c(1:p) , ylab = "BIC", xlab = "Number of predictors")
}
```



The plots we see look similar in terms of shape and pattern, but that is due to the nature of only being interested in up to 30-predictor models in the forward selection process. However, the plots differ in terms of the maximum BIC values that are reached. When there are less predictors overall (smaller p), the BICs stay quite low and do not surpass 3000, even at the high end of the plot for p = 200. When there are more predictors, up to p = 800, we notice that the maximum value of the BICs surpasses 3000 and the individual trends look more and more vertical. Both within one plot and between the plots for various values of p, as p increases, BIC naturally grows as well, which is problematic because it indicates a multiple testing issue.

b)

```
set.seed(819)
possible_ps <- c(200, 400, 600, 800)
par(mfrow = c(2,2))
for (p in possible_ps){
  X <- matrix(data = rnorm(n*p,mean = 0, sd = 1), nrow = n, ncol = p)
  norm_vec <- function(x) {sqrt(sum(x^2))}
  Xnorms <- apply(X, MARGIN = 2, FUN = norm_vec)
  Xnormed <- sweep(X, 2, Xnorms, FUN = '/')
  true_beta_vector <- c(rep(5, 10), rep(0, p - 10))
  Y <- X %*% true_beta_vector + rnorm(n = n, mean = 0, sd = 1)

  s <- sample(1:n, 200, replace = FALSE)
```

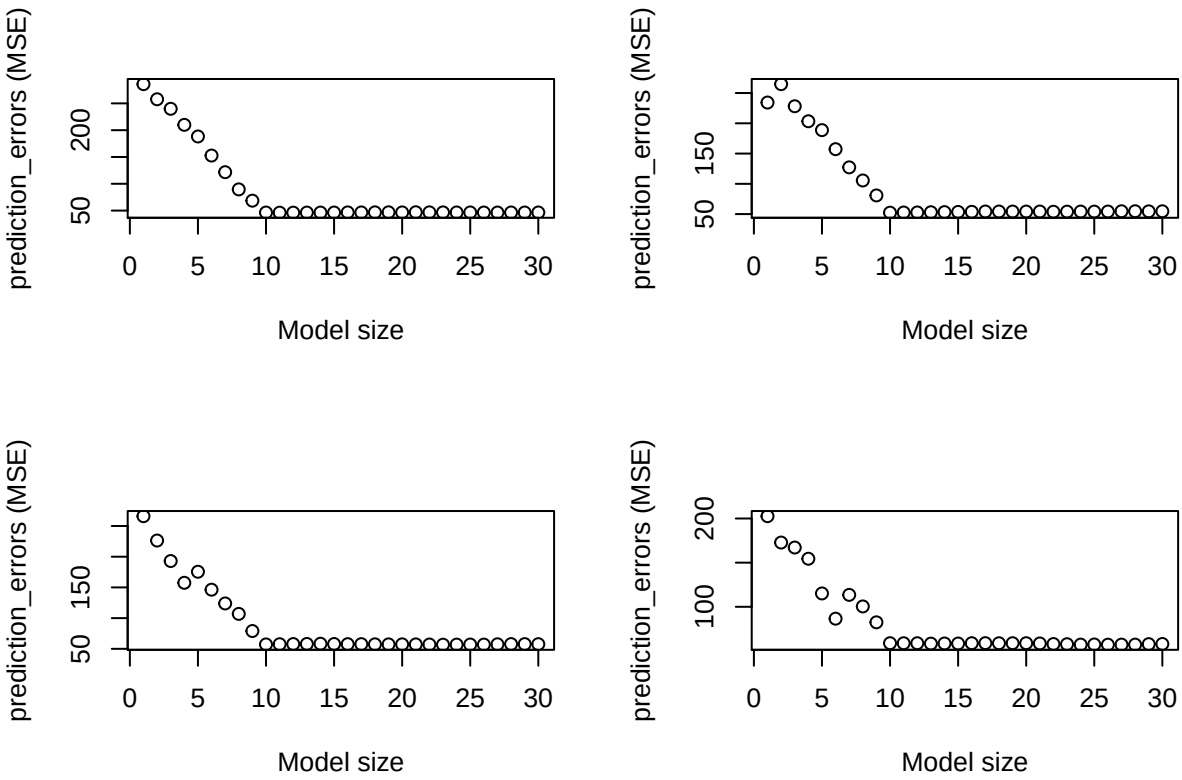
```
  train_X <- X[s,]
  valid_X <- X[-s,]
  train_Y <- Y[s]
  valid_Y <- Y[-s]

  b <- regsubsets(x = train_X, y = train_Y, nvmax = 30, method = "forward",
                  id = c(1:30), vcov = TRUE)
  rs <- summary(b)
  prediction_errors <- c()
  for (ii in 1:30){
    rcoefs_ii <- coef(b, id = ii)
    thispred <- valid_X[,which(rs$which[ii,])] %*% rcoefs_ii
    pred_error_model_ii <- mean((valid_Y - thispred)^2)
    prediction_errors <- c(prediction_errors, pred_error_model_ii)
  }
  print(prediction_errors)
  plot(y = prediction_errors, x = 1:30 , ylab = "prediction_errors (MSE)",
       xlab = "Model size")

}
```

```
##  [1] 285.87276 257.65449 239.87635 209.96888 188.26221 152.66222 121.64631
##  [8]  89.52050  68.48019  46.33485  46.12887  46.14766  46.31812  46.30169
## [15]  46.34902  46.62537  46.52865  46.69368  46.82405  46.56225  46.92045
## [22]  46.68868  46.59639  46.50750  46.48888  46.38165  46.36853  46.47310
## [29]  46.43131  46.44216
```

```
##  [1] 234.29159 264.66823 228.15765 203.38752 188.69164 157.24436 127.21773
##  [8] 105.54797  80.81167  52.36337  52.59478  52.68907  53.06842  53.29065
## [15]  53.56127  53.74198  54.25826  54.19650  54.17979  54.16029  54.18235
## [22]  53.83852  54.02288  54.16417  54.06332  54.21822  54.68748  54.70085
## [29]  54.51945  54.65239
```

```
##  [1] 265.98197 226.32199 192.95293 157.51005 175.49955 146.27747 123.85797
##  [8] 106.89288  78.98826  57.16609  57.49933  57.76269  57.93836  58.29825
## [15]  57.85736  57.60842  57.65350  57.34647  57.24790  57.31331  57.20078
## [22]  57.03913  56.74359  56.85264  56.96814  56.91424  57.36627  57.72689
## [29]  57.66631  57.60320
```

```
##  [1] 202.63225 172.76521 167.11519 154.43563 115.12448  86.53233 113.50094
##  [8] 100.40434  82.43136  58.78384  58.70910  58.75172  58.29993  58.45589
## [15]  58.40404  58.78081  58.83422  58.70863  58.77609  58.78961  58.46591
## [22]  57.85166  57.64317  57.16357  57.32089  57.09685  57.16894  57.18059
## [29]  57.71324  57.90319
```



In the plots of prediction error (MSE) versus model size, we see that the MSEs decrease as model size increases. When we print out the prediction errors for the 30 models for each p, it is even more clear that smaller model sizes result in higher prediction error, especially when the total number of predictors p is also lower. For example, the null (intercept) model gives smaller prediction error for p = 200 compared to the higher values of p (400, 600, 800), especially when we rerun this simulation a few times. This supports the conclusions we drew from BIC plots that there is a multiple testing problem since we see the natural reduction in prediction error as p increases.