# MAST90083 Assignment 1

Haonan Zhong 867492

### Question 1.1

```r
library("ISLR")
# Load the Hitters dataset
data(Hitters)
# Remove all rows from Hitters dataset that have entry NA in the Salary column
hitters <- Hitters[!is.na(Hitters$Salary),]
```

### Question 1.2

```r
# Construct design matrix and response variable
x <- model.matrix(Salary~., data = hitters)[,-1]
y <- hitters$Salary
lambda <- 10^seq(10, -2, length = 100)

suppressMessages(library("glmnet"))
# Estimate ridge coefficients for 100 lambda values
ridge_model <- glmnet(x, y, lambda = lambda, alpha = 0)
# Observe the coefficients for the largest lambda
coef(ridge_model)[,1]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  5.359257e+02  5.443467e-08  1.974589e-07  7.956523e-07  3.339178e-07
##           RBI         Walks         Years        CAtBat         CHits
##  3.527222e-07  4.151323e-07  1.697711e-06  4.673743e-09  1.720071e-08
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  1.297171e-07  3.450846e-08  3.561348e-08  3.767877e-08 -5.800263e-07
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -7.807263e-06  2.180288e-08  3.561198e-09 -1.660460e-08 -1.152288e-07
```

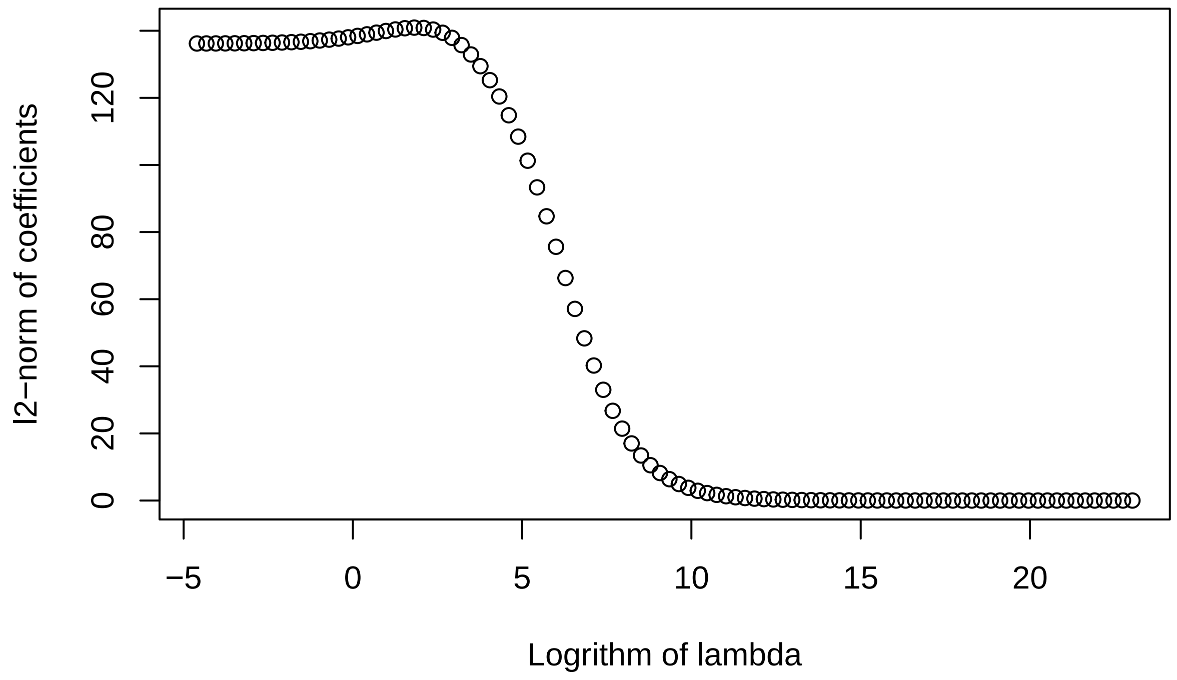
```r
# Observe the coefficients for the smallest lambda
coef(ridge_model)[,100]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  164.11321606   -1.97386151    7.37772270    3.93660219   -2.19873625
##           RBI         Walks         Years        CAtBat         CHits
##   -0.91623008    6.20037718   -3.71403424   -0.17510063    0.21132772
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##    0.05629004    1.36605490    0.70965516   -0.79582173   63.40493257
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -117.08243713    0.28202541    0.37318482   -3.42400281  -25.99081928
```

As we can see from the output above, coefficients of the largest lambda is much more closer to 0, which well reflects the effect of shrinkage penalization, as $\lambda$ increases, the ridge regression coefficients will shrink closer to 0.

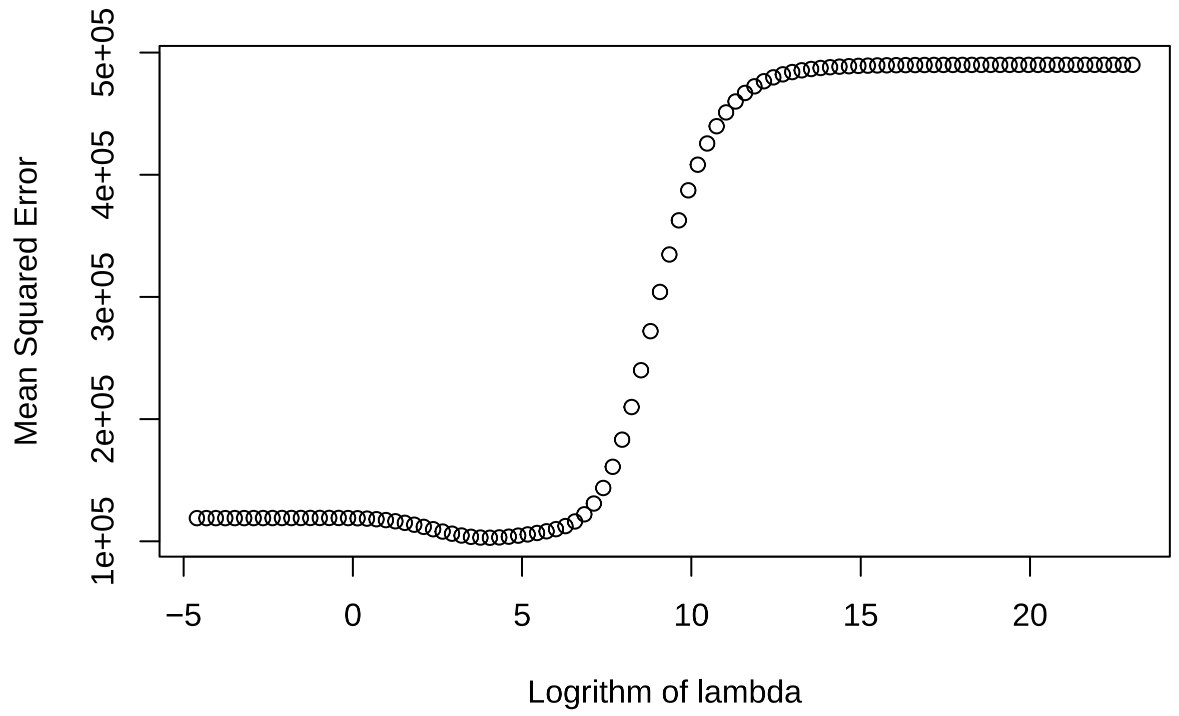
**Question 1.3**

```r
l2norm <- rep(0, length(lambda))
for (i in 1:length(lambda)) {
  l2norm[i] <- norm(ridge_model$beta[,i], type="2")
}
plot(log(lambda), l2norm, xlab = "Logrithm of lambda", ylab = "l2-norm of coefficients")
```



```r
# Computing the MSE for each lambda
mse <- rep(0, length(lambda))
for (i in 1:length(lambda)) {
  prediction <- rep(0, length(y))
  for (j in 1:length(y)) {
    prediction[j] <- t(x[j,]) %*% matrix(ridge_model$beta[, i])
  }
  mse[i] <- mean((y - prediction)^2)
}

plot(log(lambda), mse, xlab = "Logrithm of lambda", ylab = "Mean Squared Error")
```



We cannot really say anything about the optimal value of $\lambda$ with $l2$-norm, since it only tells us the change of size of the coefficients. On the other hand, mean squared error can tell us the model accuracy in terms of estimating the response variable, *Salary*.

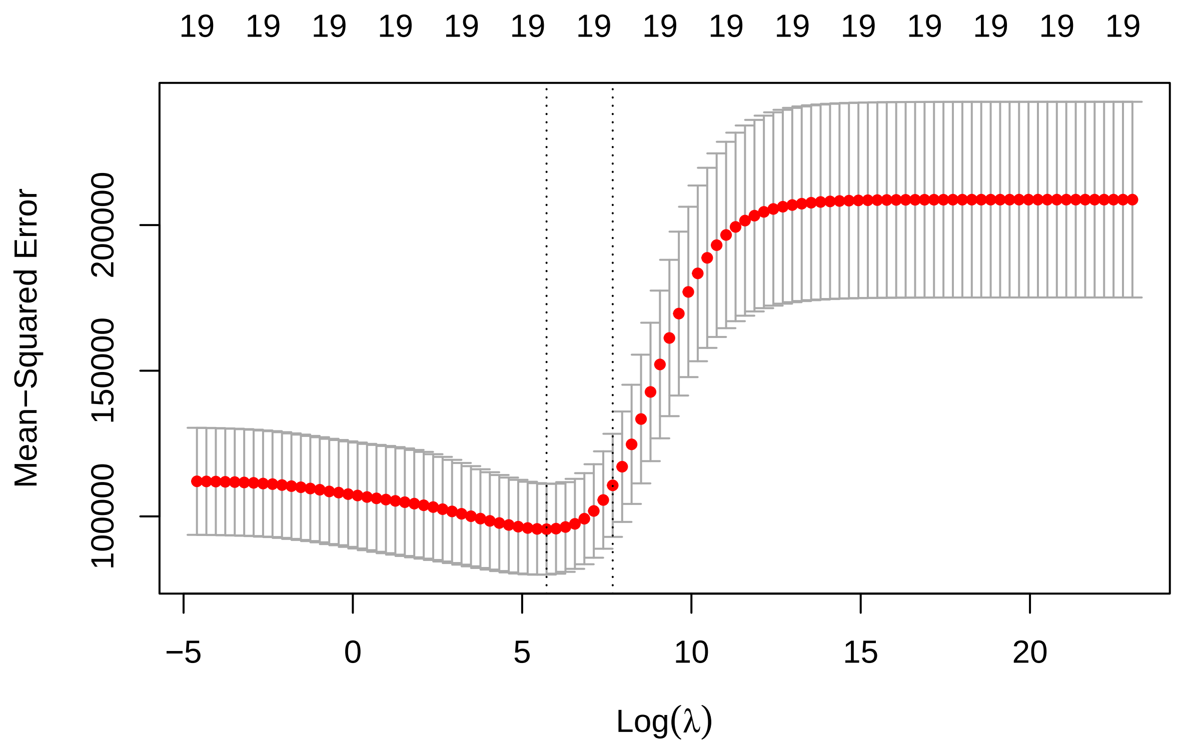
**Question 1.4**

```
# Set the seed equal to 10 for random number generator
set.seed(10)
# Sample the training set
n_train <- sample(seq_len(length(y)), size = 131)
x_train <- x[n_train, ]
y_train <- y[n_train]
# Sample the testing set
x_test <- x[-n_train, ]
y_test <- y[-n_train]

# Performing 10-fold cross validation
(ridge_cv <- cv.glmnet(x_train, y_train, lambda = lambda, type.measure = "mse", alpha = 0))
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, lambda = lambda, type.measure = "mse",      alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  305.4     63    95606 15565       19
## 1se 2154.4     56   110658 17696       19
```

```
# Plot the cross-validation result
plot(ridge_cv)
```



As the result above depicted, the model has the lowest mean squared error at 95606 when $\lambda = 305.4$. Next, we will evaluate the test mean squared error.

```
test_pred <- predict(ridge_cv$glmnet.fit, ridge_cv$lambda.min, newx = x_test)
mean((y_test - test_pred)^2)
```

```
## [1] 143198.3
```

The corresponding MSE for $\lambda = 305.4$ on the testing set is 143198.3.

```
# Refit the ridge regression model on the full data set using the lambda chosen by CV
ridge_new_model <- glmnet(x, y, alpha = 0, lambda = ridge_cv$lambda.min)
coef(ridge_new_model)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  13.82836042
## AtBat         0.07185690
## Hits          0.87923686
## HmRun          0.53787861
## Runs           1.07209934
## RBI            0.87940924
## Walks          1.65046005
## Years          1.20215837
## CAtBat         0.01135929
## CHits          0.05850049
## CHmRun         0.41307137
## CRuns          0.11642426
## CRBI           0.12329787
## CWalks         0.05014536
## LeagueN       22.83614180
## DivisionW    -81.02283953
## PutOuts        0.17012394
## Assists        0.03105466
## Errors        -1.42556422
## NewLeagueN     8.95856315
```

```
# Fit a comparison model using ordinary least square
ols_model <- lm(Salary ~ ., data = Hitters)
ols_model$coefficients
```

```
##   (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##   163.1035878   -1.9798729    7.5007675    4.3308829   -2.3762100   -1.0449620
##         Walks        Years       CAtBat        CHits       CHmRun        CRuns
##     6.2312863   -3.4890543   -0.1713405    0.1339910   -0.1728611    1.4543049
##          CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##     0.8077088   -0.8115709   62.5994230 -116.8492456    0.2818925    0.3710692
##        Errors   NewLeagueN
##    -3.3607605  -24.7623251
```

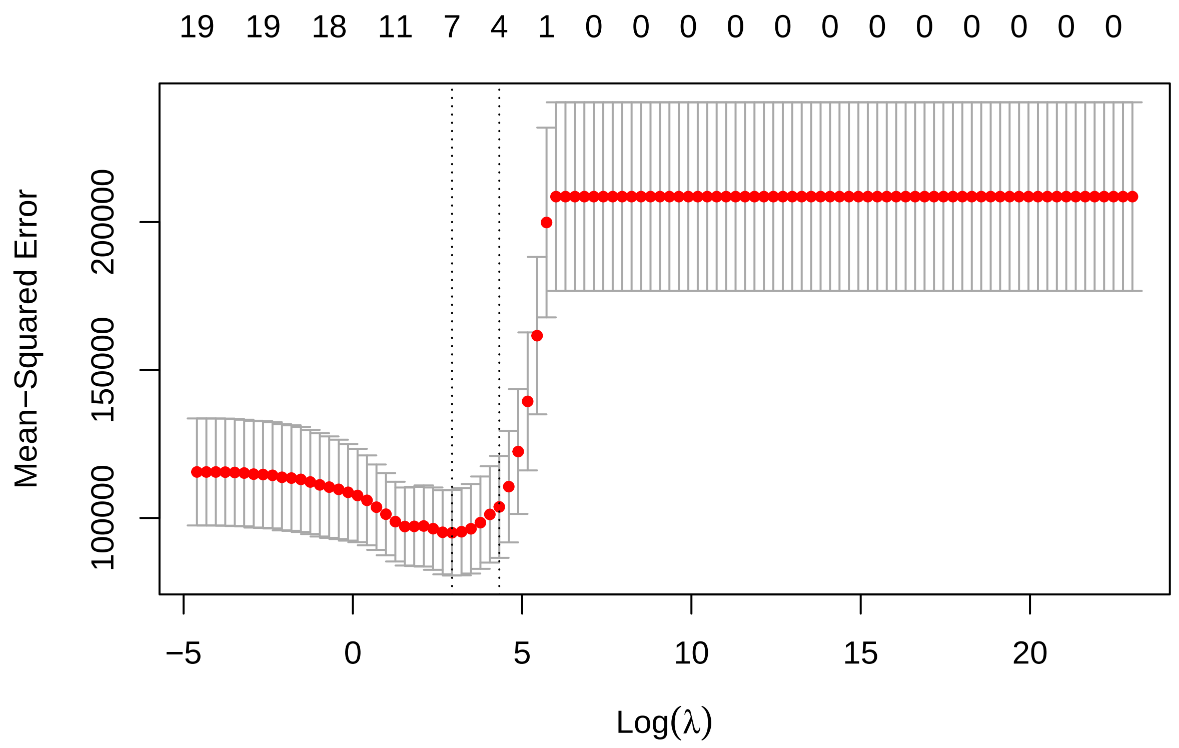
As we can see from the output above, the coefficients of the ridge regression model are much smaller compare to the one from ordinary least square model. In some cases, the coefficients in ridge regression are shrinked close to zero, but ridge regression still retains all the variables.

### Question 1.5

```
set.seed(10)
# Plot the cross-validation result
(lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1, lambda = lambda, type.measure = "mse"))
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, lambda = lambda, type.measure = "mse",      alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  18.74    73   95029 14464       7
## 1se  75.65    68  103753 17212       4
```

```
plot(lasso_cv)
```



```
lasso_test_pred <- predict(lasso_cv$glmnet.fit, lasso_cv$lambda.min, newx = x_test)
mean((y_test - lasso_test_pred)^2)
```

```
## [1] 142270.1
```

As the output above suggests, the optimal $\lambda$ value for lasso regression is 18.74, and the corresponding mean squared error for the testing set is 142270.1.

```
lasso_new_model <- glmnet(x, y, alpha = 1, lambda = lasso_cv$lambda.min)
coef(lasso_new_model)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept)  24.6396990
## AtBat          .
## Hits          1.8499834
## HmRun          .
## Runs           .
## RBI            .
## Walks         2.1959762
## Years          .
## CAtBat         .
## CHits           .
## CHmRun          .
## CRuns          0.2058514
## CRBI           0.4095910
## CWalks          .
## LeagueN         .
## DivisionW   -99.9733911
## PutOuts        0.2158910
## Assists         .
## Errors          .
## NewLeagueN      .
```

Finally, we refitted the lasso regression model using the $\lambda = 18.74$ selected from cross-validation. As we can see, most of the coefficients were shrinked to zero, thus, less important variables are eliminated when penalized, resulted in a sparse model compare to ordinary least square and ridge regression model.

**Question 2.1**

As we can see from the model, the first $p$ samples are used as the predictors for the first response, and the response starts from $p+1$. Therefore, the number of effective sample size $T$ is $n-p$.

**Question 2.2**

The model can be view as a variation of $y = X\phi + \eta$,

$$\begin{bmatrix} y_{p+1} \\ y_{p+2} \\ . \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} y_p & y_{p-1} & \cdots & y_1 \\ y_{p+1} & y_p & \cdots & y_2 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ y_{n-1} & y_{n-2} & \cdots & y_{n-p} \end{bmatrix} \times \begin{bmatrix} \phi_1 \\ \phi_2 \\ . \\ . \\ . \\ \phi_p \end{bmatrix}$$

.

Therefore, to obtain the least square estimator, we can apply the normal equation $\hat{\phi} = (X^T X)^{-1} X^T y$, where,

$$X = \begin{bmatrix} y_p & y_{p-1} & \cdots & y_1 \\ y_{p+1} & y_p & \cdots & y_2 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ y_{n-1} & y_{n-2} & \cdots & y_{n-p} \end{bmatrix} \quad and \ \ y = \begin{bmatrix} y_{p+1} \\ y_{p+2} \\ . \\ . \\ . \\ y_n \end{bmatrix}$$

**Question 2.3**

Given that

$$\sigma_p^2 = \frac{RSS^2}{T} = \frac{||Y - \hat{Y}||^2}{T}$$

Therefore, replacing $T$ with the effective sample size we have,

$$\sigma_p^2 = \frac{||Y - \hat{Y}||^2}{T} = \frac{||Y - X\hat{\phi}||^2}{n-p}$$

**Question 2.4**

```r
# Here we assume the first p samples to be zero
set.seed(10)
n_samples <- 100
M1 <- rep(0, n_samples)
M2 <- rep(0, n_samples)

# Generate samples for model M1
for (i in 6:n_samples){
  M1[i] <- 0.434 * M1[i-1] + 0.217 * M1[i-2] + 0.145 * M1[i-3] + 0.108 * M1[i-4] +
    0.087 * M1[i-5] + rnorm(1)
}

# Generate samples for model M2
for (i in 3:n_samples) {
  M2[i] <- 0.682 * M2[i-1] + 0.346 * M2[i-2] + rnorm(1)
}
```

**Question 2.5**

```r
# Here we will construct X using the samples
Create_X <- function(n, p, model) {
  X <- matrix(0, n-p, p)
  for (i in 0:(p-1)) {
    for (j in 1:(n-p)) {
      X[j, p-i] <- model[i+j]
    }
  }
  return(X)
}

# Compute the ICs
Compute_IC <- function(model, n, IC1, IC2, IC3, P) {
  for (p in P) {
    T <- n-p
    tmp_y <- model[-c(1:p)]
    tmp_X <- Create_X(n_samples, p, model)
    phi_hat <- solve(t(tmp_X) %*% tmp_X) %*% t(tmp_X) %*% matrix(tmp_y)
    sigma2 <- sum((model[c((p+1):n)] - (tmp_X %*% phi_hat))^2)/T

    # Compute the criteria
    IC1[p] <- log(sigma2) + (2 * (p+1)/T)
    IC2[p] <- log(sigma2) + ((T + p)/(T - p - 2))
    IC3[p] <- log(sigma2) + (p * log(T)/T)
  }
  return(cbind(IC1, IC2, IC3))
}
```

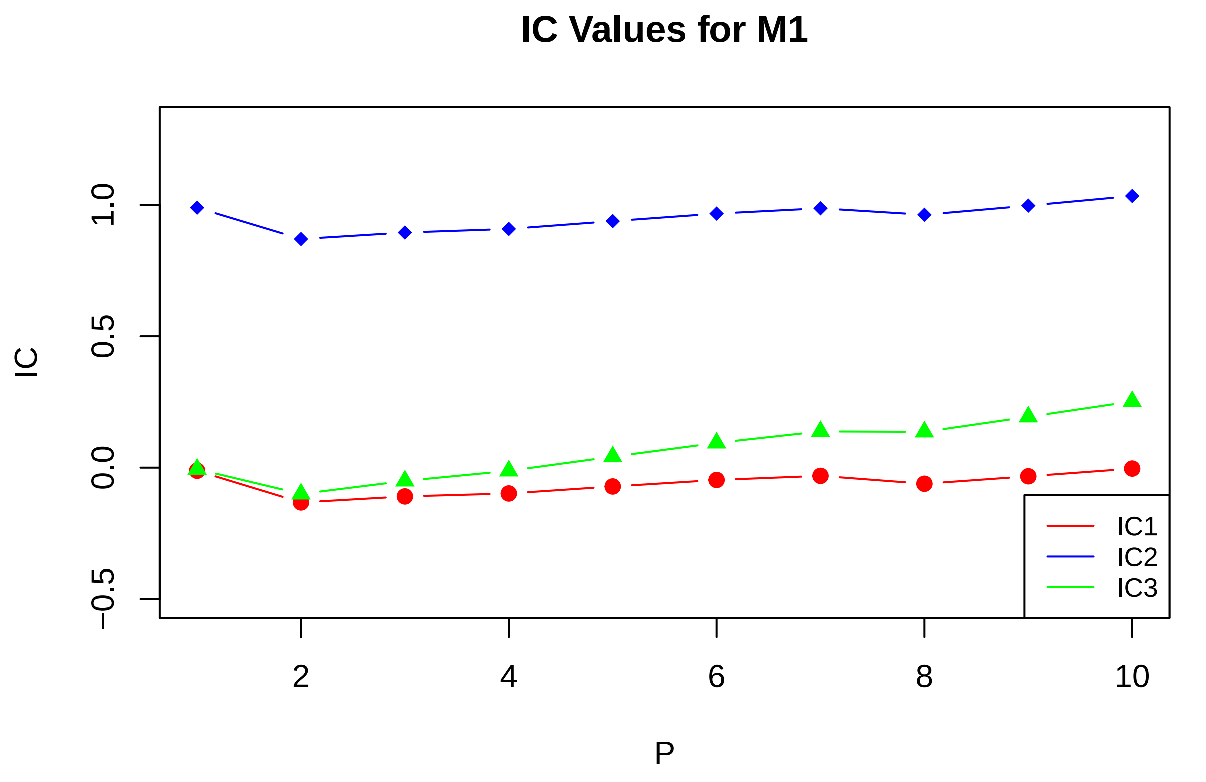
```r
P <- c(1:10)
n_samples <- 100
# Values of IC for model M1
(M1_IC <- Compute_IC(M1, n_samples, rep(0, 10), rep(0, 10), rep(0, 10), P))
```

```
##               IC1       IC2          IC3
##  [1,] -0.011763558 0.9894991 -0.005752247
##  [2,] -0.132347501 0.8702578 -0.100001226
##  [3,] -0.109525572 0.8949567 -0.050513892
##  [4,] -0.098316908 0.9086275 -0.012302400
##  [5,] -0.071641003 0.9384068  0.041720939
##  [6,] -0.046897938 0.9669566  0.094163431
##  [7,] -0.031286564 0.9871466  0.137833828
##  [8,] -0.061324068 0.9625360  0.136222765
##  [9,] -0.032933141 0.9972866  0.193415601
## [10,] -0.003615629 1.0339912  0.251918778
```

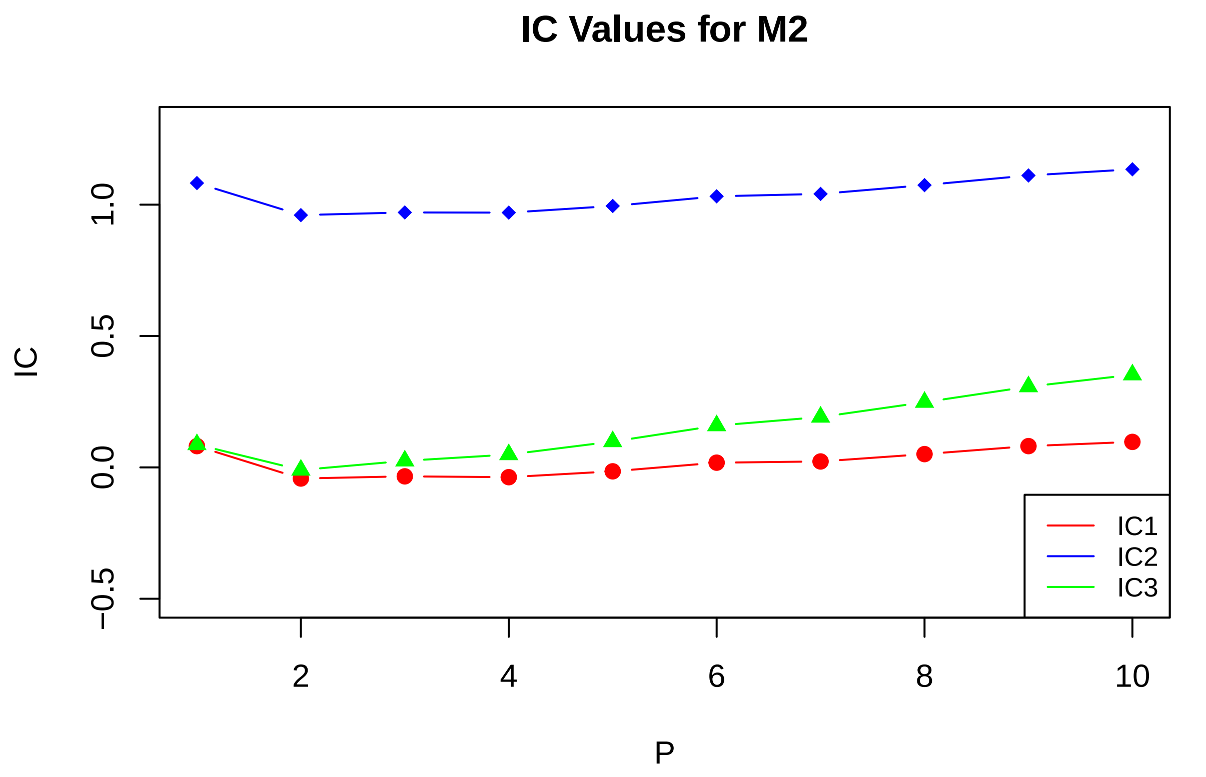
```r
# Values of IC for model M2
(M2_IC <- Compute_IC(M2, n_samples, rep(0, 10), rep(0, 10), rep(0, 10), P))
```

```
##              IC1       IC2         IC3
##  [1,]  0.08099436 1.0822570  0.08700568
##  [2,] -0.04238235 0.9602229 -0.01003608
##  [3,] -0.03413800 0.9703443  0.02487368
##  [4,] -0.03725311 0.9696913  0.04876140
##  [5,] -0.01505449 0.9949934  0.09830746
##  [6,]  0.01790558 1.0317601  0.15896695
##  [7,]  0.02249788 1.0409311  0.19161827
##  [8,]  0.05056833 1.0744284  0.24811517
##  [9,]  0.08095538 1.1111752  0.30730412
## [10,]  0.09701027 1.1346171  0.35254468
```

```r
# Plotting the three criteria for model M1
plot(P, M1_IC[,1], type = "b", pch = 19, col = 'red', ylim = c(-0.5, 1.3),
     main = "IC Values for M1", ylab = "IC")
lines(P, M1_IC[,2], pch = 18, col = 'blue', type = "b")
lines(P, M1_IC[,3], pch = 17, col = 'green', type = "b")
legend("bottomright", legend=c("IC1", "IC2", "IC3"),
       col=c("red", "blue", "Green"), lty = 1, cex=0.8)
```



```r
# Plotting the three criteria for model M2
plot(P, M2_IC[,1], type = "b", pch = 19, col = 'red', ylim = c(-0.5, 1.3),
     main = "IC Values for M2", ylab = "IC")
lines(P, M2_IC[,2], pch = 18, col = 'blue', type = "b")
lines(P, M2_IC[,3], pch = 17, col = 'green', type = "b")
legend("bottomright", legend=c("IC1", "IC2", "IC3"),
       col=c("red", "blue", "Green"), lty = 1, cex=0.8)
```



**Question 2.6**

```r
set.seed(10)
# Generate 1000 sets of size 100 using model M1
n_samples <- 100
n_sets <- 1000
M1_set_100 <- matrix(0, n_samples, n_sets)
for (i in 1:n_sets) {
  M1 <- rep(0, n_samples)
  for (j in 6:n_samples) {
    M1[j] <- 0.434 * M1[j-1] + 0.217 * M1[j-2] + 0.145 * M1[j-3] + 0.108 * M1[j-4] +
      0.087 * M1[j-5] + rnorm(1)
  }
  M1_set_100[, i] <- M1
}
```

```r
IC_count_func <- function(model_set, n_samples, n_sets, P) {
  IC_count <- matrix(0, 3, length(P))
  for (i in 1:1000) {
    IC <- Compute_IC(model_set[, i], n_samples,
                     rep(0, length(P)), rep(0, length(P)), rep(0,  length(P)), P)
    IC_count[1, which(IC[,1] == min(IC[,1]))] <- IC_count[1, which(IC[,1] == min(IC[,1]))] + 1
    IC_count[2, which(IC[,2] == min(IC[,2]))] <- IC_count[2, which(IC[,2] == min(IC[,2]))] + 1
    IC_count[3, which(IC[,3] == min(IC[,3]))] <- IC_count[3, which(IC[,3] == min(IC[,3]))] + 1
  }

  IC_count <- data.frame(IC_count, row.names = c("IC1", "IC2", "IC3"))
  colnames(IC_count) <- P
  return(IC_count)
}

P <- c(1:10)
# Print table of counts of the selected model by IC1, IC2 and IC3
IC_count_func(M1_set_100, n_samples, n_sets, P)
```

```
##       1   2   3   4  5  6  7  8  9 10
## IC1 15 204 370 180 78 43 34 29 17 30
## IC2 16 232 395 177 78 36 25 19 12 10
## IC3 72 502 335  69 14  5  1  1  1  0
```

**Question 2.7**

```
set.seed(10)
# Generate 1000 sets of size 15 using model M1
n_samples <- 15
M1_set_15 <- matrix(0, n_samples, n_sets)
for (i in 1:n_sets) {
  M1 <- rep(0, n_samples)
  for (j in 6:n_samples) {
    M1[j] <- 0.434 * M1[j-1] + 0.217 * M1[j-2] + 0.145 * M1[j-3] + 0.108 * M1[j-4] +
      0.087 * M1[j-5] + rnorm(1)
  }
  M1_set_15[, i] <- M1
}

P <- c(1:6)
# Print table of counts of the selected model by IC1, IC2 and IC3
IC_count_func(M1_set_15, n_samples, n_sets, P)
```

```
##       1   2  3  4  5   6
## IC1 677 119 33 26 27 118
## IC2 904  85 10  1  0   0
## IC3 704  97 31 23 21 124
```

**Question 2.8**

```
set.seed(10)
# Generate 1000 sets of size 100 using model M2
n_samples <- 100
M2_set_100 <- matrix(0, n_samples, n_sets)
for (i in 1:n_sets) {
  M2 <- rep(0, n_samples)
  for (j in 3:n_samples) {
    M2[j] <- 0.682 * M2[j-1] + 0.346 * M2[j-2] + rnorm(1)
  }
  M2_set_100[, i] <- M2
}

P <- c(1:10)
# Print table of counts of the selected model by IC1, IC2 and IC3
IC_count_func(M2_set_100, n_samples, n_sets, P)
```

```
##       1   2   3  4  5  6  7  8  9 10
## IC1  42 570 127 79 48 42 25 23 22 22
## IC2  52 613 137 72 44 32 15 14 13  8
## IC3 170 758  61  9  2  0  0  0  0  0
```

```
set.seed(10)
# Generate 1000 sets of size 15 using model M2
n_samples <- 15
M2_set_15 <- matrix(0, n_samples, n_sets)
for (i in 1:n_sets) {
  M2 <- rep(0, n_samples)
  for (j in 3:n_samples) {
    M2[j] <- 0.682 * M2[j-1] + 0.346 * M2[j-2] + rnorm(1)
  }
  M2_set_15[, i] <- M2
}

P <- c(1:6)
# Print table of counts of the selected model by IC1, IC2 and IC3
IC_count_func(M2_set_15, n_samples, n_sets, P)
```

```
##       1  2  3  4  5   6
## IC1 527 90 69 45 72 197
## IC2 884 84 26  6  0   0
## IC3 561 75 54 39 63 208
```

### Question 2.9

As we can see from the four tables above, the suggested value of $P$ given by the three criteria tend to be small when $n = 100$ for both $M1$ and $M2$, as most of the count are clustered between $p = 2$ and $p = 3$, and the number of count decreases as the order of $p$ increases, which some what resembles the trend of the plots from question 2.5.

On the other hand, the maximum order of $p$ when $n = 15$ is set to 6. We can observe that most of the count are concentrated at $p = 1$, and the count decreases as $p$ increases. But for $IC_1$ and $IC_3$, we can see there's a slight increase of count towards the end, when $p = 6$.

### Question 2.10

Assuming the true model order is $p_0$ and we fit a candidate model of order $p_0 + L$, then the criterion selects the overfitted model if $IC_{p_0+L} < IC_{p_0}$.

Therefore, the expression of the probability of overfitting for $IC_1$ is,

$$P(IC_{1p_0+L} < IC_{1p_0}) = P(log(\sigma^2_{p_0+L}) + \frac{2(p_0+L+1)}{n-p_0-L} < log(\sigma^2_{p_0}) + \frac{2(p_0+1)}{n-p_0})$$

$$\sigma^2_p = \frac{RSS}{T} \Rightarrow log(\sigma^2_p) = log(RSS) - log(T)$$

$$P(log(RSS_{p_0+L}) - log(n-p_0-L) + \frac{2(p_0+L+1)}{n-p_0-L} < log(RSS_{p_0}) - log(n-p_0) + \frac{2(p_0+1)}{n-p_0})$$

$$= P(log(\frac{RSS_{p_0+L}}{RSS_{p_0}}) < log(\frac{n-p_0-L}{n-p_0}) + \frac{-2L(1+n)}{(n-p_0-L)(n-p_0)})$$

$$= P(log(\frac{RSS_{p_0}}{RSS_{p_0+L}}) > log(\frac{n-p_0}{n-p_0-L}) + \frac{2L(1+n)}{(n-p_0-L)(n-p_0)})$$

$$= P(\frac{RSS_{p_0}}{RSS_{p_0+L}} - 1 > \frac{n-p_0}{n-p_0-L} \cdot exp(\frac{2L(1+n)}{(n-p_0-L)(n-p_0)}) - 1)$$

$$= P(\frac{RSS_{p_0} - RSS_{p_0+L}}{RSS_{p_0+L}} > \frac{n-p_0}{n-p_0-L} \cdot exp(\frac{2L(1+n)}{(n-p_0-L)(n-p_0)}) - 1)$$

$$= P(F_{L,n-p_0-L} > \frac{n-p_0-L}{L} \cdot (\frac{n-p_0}{n-p_0-L} \cdot exp(\frac{2L(1+n)}{(n-p_0-L)(n-p_0)}) - 1))$$

The expression of the probability of overfitting for $IC_2$ is,

$$P(IC_{2p_0+L} < IC_{2p_0}) = P(log(\sigma^2_{p_0+L}) + \frac{n-p_0-L+p_0+L}{n-p_0-L-p_0-L-2} < log(\sigma^2_{p_0}) + \frac{n-p_0+p_0}{n-p_0-p_0-2})$$

$$P(log(RSS_{p_0+L}) - log(n-p_0-L) + \frac{n}{n-2p_0-2L-2} < log(RSS_{p_0}) - log(n-p_0) + \frac{n}{n-2p_0-2})$$

$$= P(log(\frac{RSS_{p_0+L}}{RSS_{p_0}}) < log(\frac{n-p_0-L}{n-p_0}) + \frac{-2nL}{(n-2p_0-2)(n-2p_0-2L-2)})$$

$$= P(log(\frac{RSS_{p_0}}{RSS_{p_0+L}}) > log(\frac{n-p_0}{n-p_0-L}) + \frac{2nL}{(n-2p_0-2)(n-2p_0-2L-2)})$$

$$= P(\frac{RSS_{p_0}}{RSS_{p_0+L}} - 1 > \frac{n-p_0}{n-p_0-L} \cdot exp(\frac{2nL}{(n-2p_0-2)(n-2p_0-2L-2)}) - 1)$$

$$= P(\frac{RSS_{p_0} - RSS_{p_0+L}}{RSS_{p_0+L}} > \frac{n-p_0}{n-p_0-L} \cdot exp(\frac{2nL}{(n-2p_0-2)(n-2p_0-2L-2)}) - 1)$$

$$= P(F_{L,n-p_0-L} > \frac{n-p_0-L}{L} \cdot (\frac{n-p_0}{n-p_0-L} \cdot exp(\frac{2nL}{(n-2p_0-2)(n-2p_0-2L-2)}) - 1))$$

The expression of the probability of overfitting for $IC_3$ is,

$$P(IC_{3p_0+L} < IC_{3p_0}) = P(log(\sigma^2_{p_0+L}) + \frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} < log(\sigma^2_{p_0}) + \frac{p_0 log(n-p_0)}{n-p_0})$$

$$P(log(RSS_{p_0+L}) - log(n-p_0-L) + \frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} < log(RSS_{p_0}) - log(n-p_0) + \frac{p_0 log(n-p_0)}{n-p_0})$$

$$= P(log(\frac{RSS_{p_0+L}}{RSS_{p_0}}) < log(\frac{n-p_0-L}{n-p_0}) + \frac{p_0 log(n-p_0)}{n-p_0} - \frac{(p_0+L)log(n-p_0-L)}{n-p_0-L})$$

$$= P(log(\frac{RSS_{p_0}}{RSS_{p_0+L}}) > log(\frac{n-p_0}{n-p_0-L}) - \frac{p_0 log(n-p_0)}{n-p_0} + \frac{(p_0+L)log(n-p_0-L)}{n-p_0-L})$$

$$= P(\frac{RSS_{p_0}}{RSS_{p_0+L}} - 1 > \frac{n-p_0}{n-p_0-L} \cdot exp(\frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} - \frac{p_0 log(n-p_0)}{n-p_0}) - 1)$$

$$= P(\frac{RSS_{p_0} - RSS_{p_0+L}}{RSS_{p_0+L}} > \frac{n-p_0}{n-p_0-L} \cdot exp(\frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} - \frac{p_0 log(n-p_0)}{n-p_0}) - 1)$$

$$= P(F_{L,n-p_0-L} > \frac{n-p_0-L}{L} \cdot (\frac{n-p_0}{n-p_0-L} \cdot exp(\frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} - \frac{p_0 log(n-p_0)}{n-p_0}) - 1))$$

**Question 2.11**

```
Compute_F <- function(n, p, L) {
  F1 <- ((n-p-L)/L)*(((n-p)/(n-p-L))*exp((2*L*(1+n))/((n-p-L)*(n-p))) - 1)
  F2 <- ((n-p-L)/L)*(((n-p)/(n-p-L))*exp((2*n*L)/((n-2*p-2)*(n-2*p-2*L-2))) - 1)
  F3 <- ((n-p-L)/L)*(((n-p)/(n-p-L))*exp(((p+L)*log(n-p-L))/(n-p-L)-(p*log(n-p))/(n-p)) - 1)

  return(cbind(F1, F2, F3))
}

Compute_prob <- function(n, p, L) {
  prob <- matrix(0, 3, length(L))
   for (l in L) {
    F_stat <- Compute_F(n, p, l)
    prob[1, l] <- pf(F_stat[1], l, n-p-l, lower.tail = F)
    prob[2, l] <- pf(F_stat[2], l, n-p-l, lower.tail = F)
    prob[3, l] <- pf(F_stat[3], l, n-p-l, lower.tail = F)
   }
  prob <- data.frame(prob, row.names = c("IC1", "IC2", "IC3"))
  colnames(prob) <- L
  return(prob)
}
```

```
L <- c(1:8)
# tables of the calculated probabilities for M1 in the cases n = 25
n_samples <- 25
Compute_prob(n_samples, 5, L)
```

```
##                1            2            3            4            5
## IC1 0.061993039 0.0287751060 1.256454e-02 5.237465e-03 2.074740e-03
## IC2 0.006425828 0.0001767892 6.552481e-07 2.979619e-11 2.962378e-21
## IC3 0.038130449 0.0132448275 4.554776e-03 1.575617e-03 5.504635e-04
##                6            7            8
## IC1 7.763016e-04 2.728411e-04 8.967531e-05
## IC2 2.073606e-70 1.000000e+00 1.000000e+00
## IC3 1.954053e-04 7.115851e-05 2.694694e-05
```

```
# tables of the calculated probabilities for M1 in the cases n = 100
n_samples <- 100
Compute_prob(n_samples, 5, L)
```

```
##              1           2            3            4            5            6
## IC1 0.07807509 0.044346796 0.0246301746 0.0136734985 7.593488e-03 4.212596e-03
## IC2 0.06285268 0.030027766 0.0137227425 0.0060971939 2.625621e-03 1.090322e-03
## IC3 0.01701753 0.003317702 0.0006529802 0.0001298958 2.596594e-05 5.194280e-06
##                7            8
## IC1 2.331189e-03 1.285272e-03
## IC2 4.343283e-04 1.650842e-04
## IC3 1.036975e-06 2.062236e-07
```

**Question 2.12**

As the tables from the above question suggest, when the number of samples are small ($n = 25$), model will be overfitted when we fit 1 extra parameters $L = 1$ suggested by the expression derived from $IC_1$ and $IC_2$, as the p-value is less than the significance level of 0.05, while the model will be overfitted when we add 2 extra parameters $L = 2$ suggested by $IC_1$. And as we add more parameters the p-value will decreases even more, suggesting more serious overfitting.

On the other hand, when the number of samples are 100, model will be overfitted when we fit 2 extra parameters $L = 2$ suggested by $IC_1$ and $IC_2$, as the p-value is smaller than the significance level of 0.05, while $IC_3$ suggest that model will be overfitted when we add 1 extra parameters $L = 1$. However, comparing to $n = 25$, the p-values decreases slower when $n = 100$.

**Question 2.13**

Here we will apply limits to the expressions that we derived above.
For $IC_1$,

$$\begin{aligned}
&\lim_{n\to\infty} \frac{n-p_0-L}{L} \cdot \left(\frac{n-p_0}{n-p_0-L} \cdot exp\left(\frac{2L(1+n)}{(n-p_0-L)(n-p_0)}\right) - 1\right) \\
=&\lim_{n\to\infty} \frac{n-p_0}{L} \cdot \lim_{n\to\infty} exp\left(\frac{2L(1+n)}{(n-p_0-L)(n-p_0)}\right) - \lim_{n\to\infty} \frac{n-p_0-L}{L} \\
=&\frac{1}{L}\lim_{n\to\infty} n-p_0 \cdot exp\left(\lim_{n\to\infty} \frac{2L(1+n)}{(n-p_0-L)(n-p_0)}\right) - \frac{1}{L}\lim_{n\to\infty} n-p_0-L \\
=&\frac{1}{L}\lim_{n\to\infty} n-p_0 \cdot exp(0) - \frac{1}{L}\lim_{n\to\infty} n-p_0-L \\
=&\frac{1}{L}\lim_{n\to\infty} n-p_0-n+p_0+L \\
=&1
\end{aligned}$$

For $IC_2$,

$$\begin{aligned}
&\lim_{n\to\infty} \frac{n-p_0-L}{L} \cdot \left(\frac{n-p_0}{n-p_0-L} \cdot exp\left(\frac{2nL}{(n-2p_0-2)(n-2p_0-2L-2)}\right) - 1\right) \\
=&\lim_{n\to\infty} \frac{n-p_0}{L} \lim_{n\to\infty} exp\left(\frac{2nL}{(n-2p_0-2)(n-2p_0-2L-2)}\right) - \lim_{n\to\infty} \frac{n-p_0-L}{L} \\
=&\frac{1}{L}\lim_{n\to\infty} n-p_0 \cdot exp\left(\lim_{n\to\infty} \frac{2nL}{(n-2p_0-2)(n-2p_0-2L-2)}\right) - \frac{1}{L}\lim_{n\to\infty} n-p_0-L \\
=&\frac{1}{L}\lim_{n\to\infty} n-p_0 \cdot exp(0) - \frac{1}{L}\lim_{n\to\infty} n-p_0-L \\
=&\frac{1}{L}\lim_{n\to\infty} n-p_0-n+p_0+L \\
=&1
\end{aligned}$$

For $IC_3$,

$$\begin{aligned}
&\lim_{n\to\infty} \frac{n-p_0-L}{L} \cdot \left(\frac{n-p_0}{n-p_0-L} \cdot exp\left(\frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} - \frac{p_0 log(n-p_0)}{n-p_0}\right) - 1\right) \\
=& \lim_{n\to\infty} \frac{n-p_0}{L} \lim_{n\to\infty} exp\left(\frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} - \frac{p_0 log(n-p_0)}{n-p_0}\right) - \lim_{n\to\infty} \frac{n-p_0-L}{L} \\
=& \frac{1}{L} \lim_{n\to\infty} n-p_0 \cdot exp\left(\lim_{n\to\infty}\left(\frac{(p_0+L)log(n-p_0-L)}{n-p_0-L} - \frac{p_0 log(n-p_0)}{n-p_0}\right)\right) - \frac{1}{L} \lim_{n\to\infty} n-p_0-L \\
=& \frac{1}{L} \lim_{n\to\infty} n-p_0 \cdot exp(0) - \frac{1}{L} \lim_{n\to\infty} n-p_0-L \\
=& \frac{1}{L} \lim_{n\to\infty} n-p_0-n+p_0+L \\
=& 1
\end{aligned}$$

**Question 2.14**

The important observation we can make from above is that, as the sample size $n$ tends to infinity, for all three information criteria, the critical value of the F distribution will converge to 1, in which the chance of overfitting is reduced when the sample size $n$ tends to infinity.