Problem Set 3

MACS 30100 Winter 2020

Nak Won Rim

Conceptual Excercises

Training/test error for subset selection

1. Generate a data set with $p = 20$ features, $n = 1000$ observations, and an associated quantitative response vector generated according to the model

$$Y = X\beta + \epsilon$$

where β has some elements that are exactly equal to zero.

I simulated the data where the $X \sim \mathcal{N}(4, 4), \beta \sim \mathcal{N}(1, 1)$ and $\epsilon \sim \mathcal{N}(0, 4)$. Then, I chose four random betas to become zero.

```r
set.seed(25)
create_observations <- function(i){
  df <- data.frame(matrix(NA, nrow = 1000, ncol = i))
  for(j in 1:i){
    df[,j] <- rnorm(1000, mean=4, sd=2)
  }
  return(df)
}

df <- create_observations(20)
betas <- rnorm(20, mean=1, sd=1)
beta_converts <- sample(1:20, 4, replace = FALSE, prob = NULL)
for (i in beta_converts){
  betas[i] <- 0
}
epsilon <- rnorm(1000, mean=0, sd=2)
df$y <- as.matrix(df) %*% betas + epsilon
# sanity check
dim(df)
```

```
## [1] 1000   21
```

2. Split your data set into a training set containing 100 observations and a test set containing 900 observations.

```r
split <- rsample::initial_split(df, prop = .1)
train <- rsample::training(split)
test <- rsample::testing(split)
# sanity check
dim(train)
```

```
## [1] 100  21
```

```r
dim(test)
```

```
## [1] 900  21
```

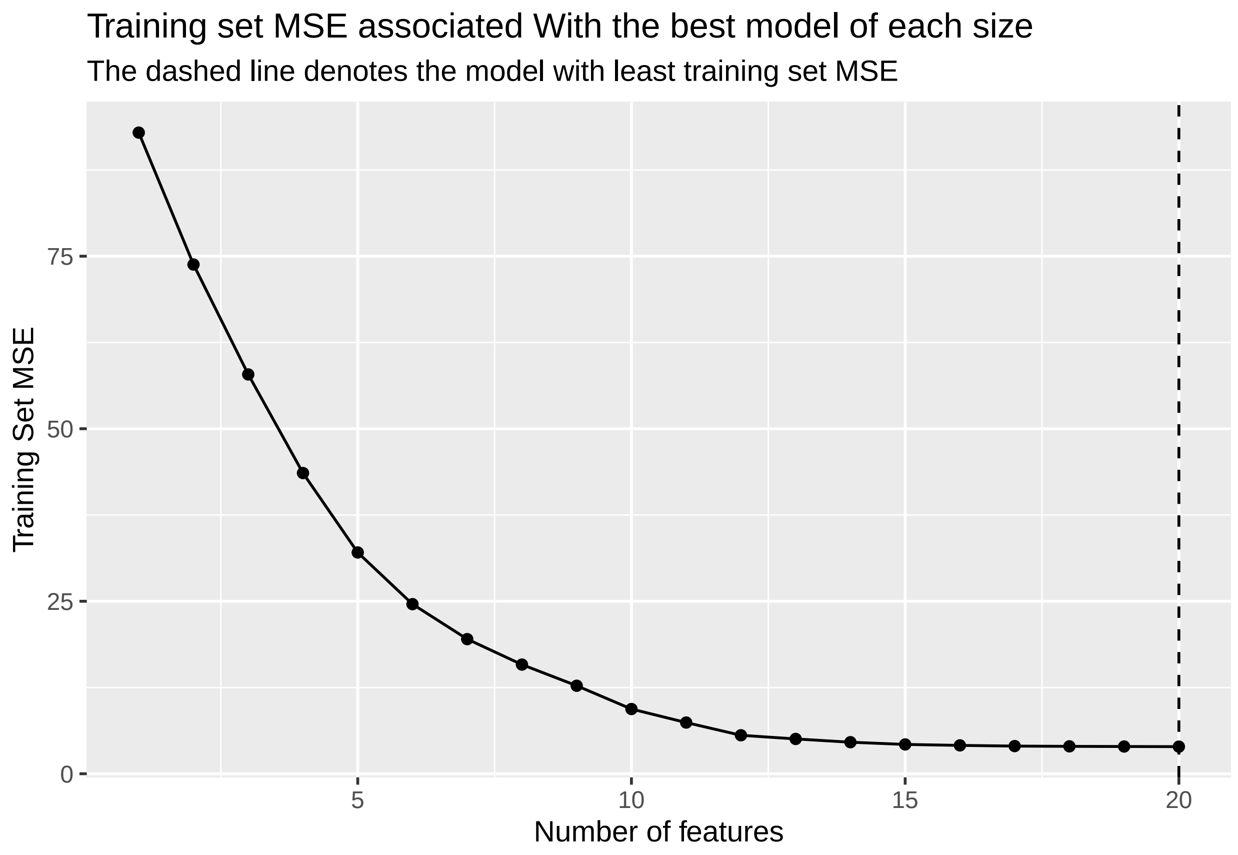
3. Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size. For which model size does the training set MSE take on its minimum value?

```r
best_subset <- leaps::regsubsets(y ~ ., data = train, nvmax = 20)

# using codes from the lab to get the MSE
# object: the regsubsets model itself, newdata: data to predict
# id: the number of predictors
predict.regsubsets <- function(object, newdata, id ,...) {
  form <- as.formula(y ~ .) # store the formula itself
  mat <- model.matrix(form, newdata) # create a matrix without y
  coefi <- coef(object, id = id) # get the coefficients
  xvars <- names(coefi) # get the names of coefficients
  # matrix multiply the selected coefficients to get the predicted value
  return(as.vector(mat[, xvars] %*% coefi))
}

predict_all <- function(object, newdata, k){
  map(k, ~ predict.regsubsets(object, newdata, id = .x)) %>%
    set_names(k)
}
train_predicted <- as.data.frame(predict_all(best_subset, train, 1:20))
train_mse_df <- data.frame(matrix(NA, ncol=2, nrow=20))
for (i in 1:20){
  train_mse_df[i,1] <- i
  train_mse_df[i,2] <- rcfss::mse_vec(as.vector(train$y), train_predicted[,i])
}

ggplot(data=train_mse_df, aes(X1, X2)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = which.min(train_mse_df$X2), linetype = 2) +
  labs(title = "Training set MSE associated With the best model of each size",
       subtitle = "The dashed line denotes the model with least training set MSE",
       x = "Number of features",
       y = "Training Set MSE")
```

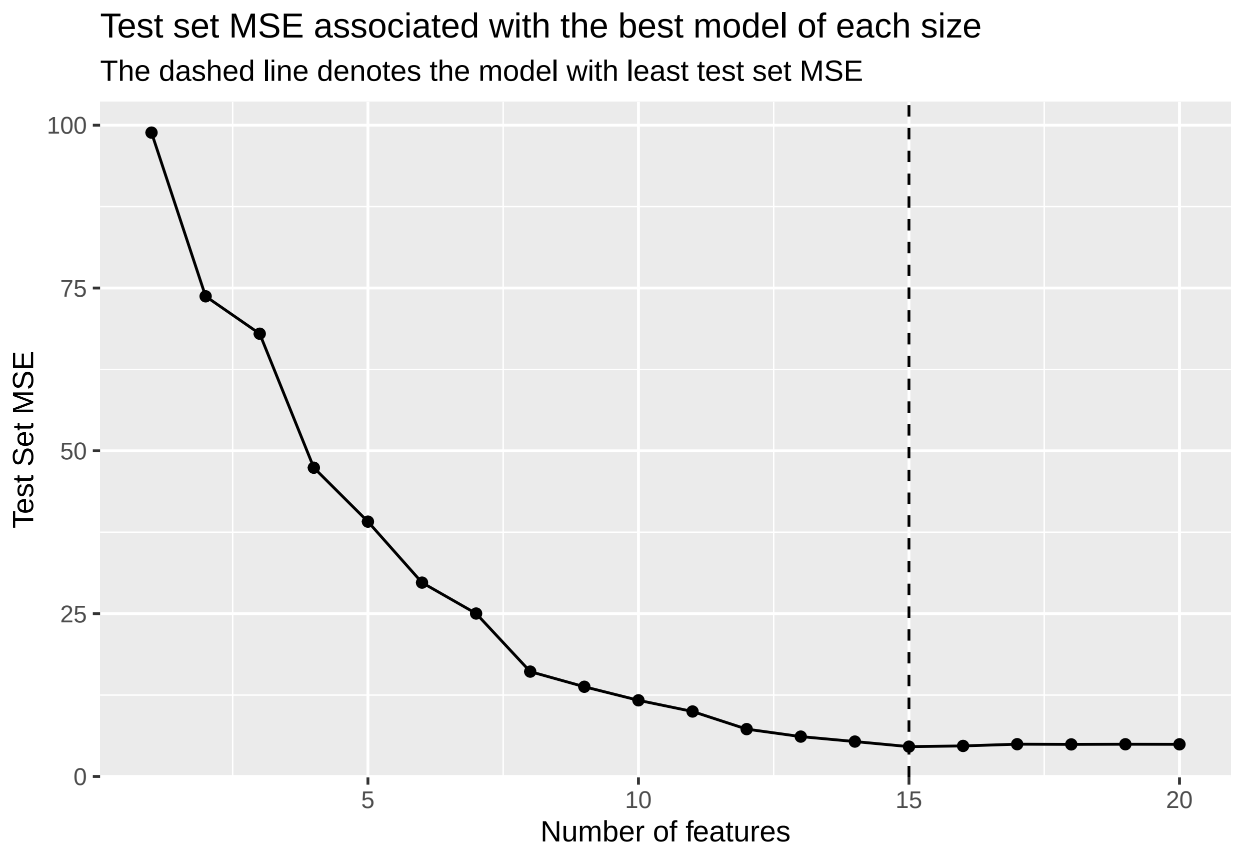


The training set MSE takes on its minimum value in the model that contains all the features, or the model that has a model size of 20.

4. Plot the test set MSE associated with the best model of each size.

```r
test_predicted <- as.data.frame(predict_all(best_subset, test, 1:20))
test_mse_df <- data.frame(matrix(NA, ncol=2, nrow=20))
for (i in 1:20){
  test_mse_df[i,1] <- i
  test_mse_df[i,2] <- rcfss::mse_vec(as.vector(test$y),test_predicted[,i])
}

ggplot(data=test_mse_df, aes(X1, X2)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = which.min(test_mse_df$X2), linetype = 2) +
  labs(title = "Test set MSE associated with the best model of each size",
       subtitle = "The dashed line denotes the model with least test set MSE",
       x = "Number of features",
       y = "Test Set MSE")
```



5. For which model size does the test set MSE take on its minimum value? Comment on your results.

The test set MSE takes on its minimum value in the model that contains 15 features. I suspect that the four features in which the associated true betas were zero and a feature in which the associated true beta was very small were excluded from the best fitting model. The difference between the model size that shows the least training set MSE (20) and the model size that shows the least test set MSE (15) is most likely because of overfitting. The model that contained all the features most likely has overfitted the errors in the training set using the redundant features. This would have decreased the MSE in the training set, but have increased MSE in the test set.

6. How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient sizes.

```r
betas_df <- as.data.frame(betas)
betas_df$predictor <- c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10",
                        "X11","X12","X13","X14","X15","X16","X17","X18","X19","X20")
estimated <- coef(best_subset, id = 15) %>% as.data.frame()
names(estimated) <- c("estimate_beta")
estimated$predictor = rownames(estimated)
merged_df <- merge(x=betas_df, y=estimated, by="predictor", all.x=TRUE)
merged_df[is.na(merged_df)] <- 0
rownames(merged_df) <- merged_df$predictor
```

```r
merged_df <- merged_df[c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10",
                         "X11","X12","X13","X14","X15","X16","X17","X18","X19","X20"),]
rownames(merged_df) <- NULL
merged_df$difference <- abs(merged_df$betas - merged_df$estimate_beta)
knitr::kable(merged_df, align="cccc",
             col.names=c("Features", "True Beta", "Estimated Beta", "Difference"),
             caption = "True Beta and Estimated Beta of the best model")
```

Table 1: True Beta and Estimated Beta of the best model

Features	True Beta	Estimated Beta	Difference
X1	1.4464050	1.5035028	0.0570978
X2	2.6417009	2.5537496	0.0879513
X3	0.0000000	0.0000000	0.0000000
X4	0.8111750	0.7725921	0.0385828
X5	-0.7880117	-0.8737966	0.0857848
X6	0.0000000	0.0000000	0.0000000
X7	0.0000000	0.0000000	0.0000000
X8	0.9293478	1.1439091	0.2145613
X9	-0.0001735	0.0000000	0.0001735
X10	1.5184232	1.4247405	0.0936826
X11	1.6013397	1.5728276	0.0285121
X12	2.3336978	2.3770862	0.0433884
X13	0.6763620	0.6418784	0.0344836
X14	1.7173109	1.6596318	0.0576791
X15	0.3756528	0.2975237	0.0781291
X16	0.4810138	0.4627064	0.0183073
X17	0.0000000	0.0000000	0.0000000
X18	-0.7353492	-0.8498718	0.1145226
X19	0.3694964	0.3682467	0.0012497
X20	1.9952304	1.9231722	0.0720583

From the above table, we can see that the features that were excluded from the model that showed the least test set MSE were the features in which the associated true betas were zero (X3, X6, X7, and X17) or nearly zero (X9, true beta = -0.0001735). This shows that subset selection successfully found features that were most relevant to the response and ruled out features that were not very relevant. Furthermore, the differences between the true betas and the estimated betas were smaller than 0.1 except two features. This small difference shows that the model that showed the least test set MSE successfully estimated the true beta values.

7. Create a plot displaying

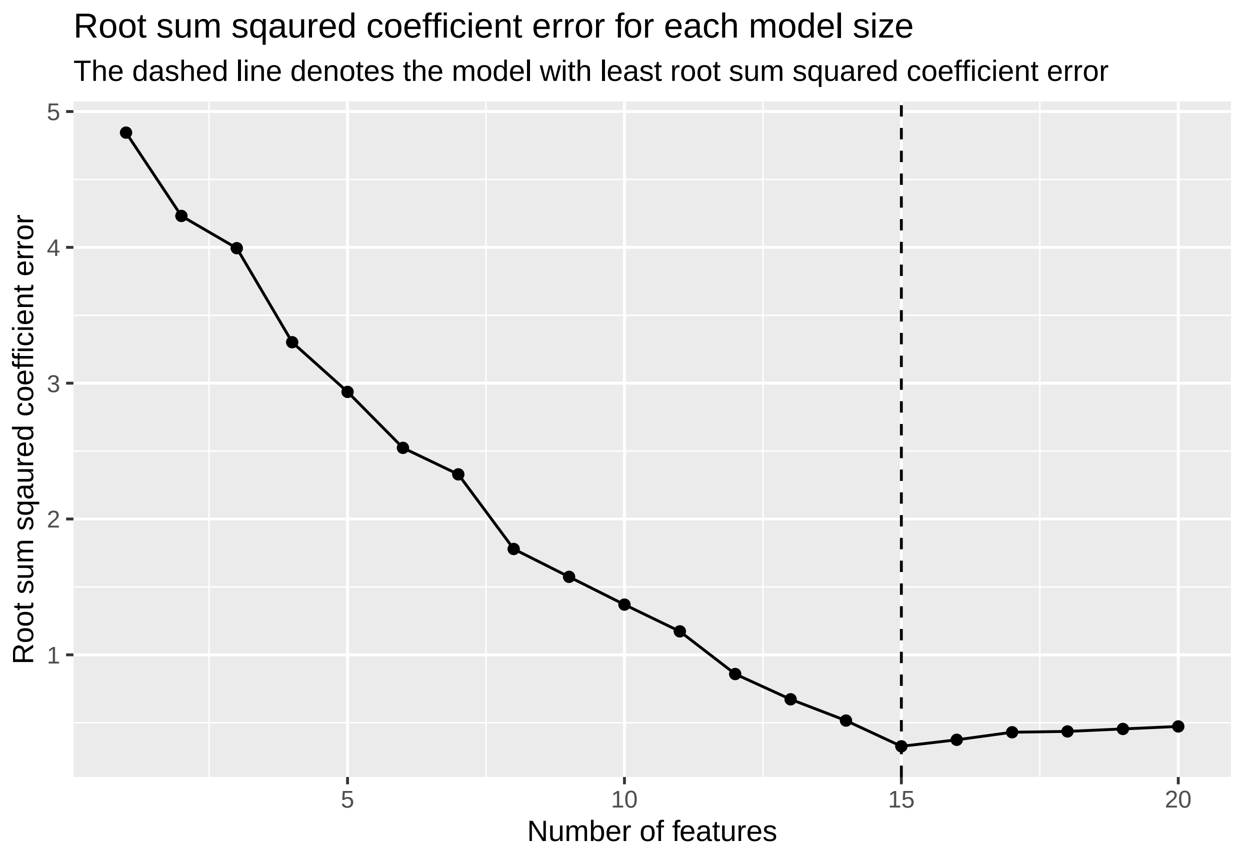
$$\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$$

for a range of values of r, where $\hat{\beta}_j^r$ is the jth coefficient estimate for the best model containing r coefficients. Comment on what you observe. How does this compare to the test MSE plot?

```r
rssce_df_1 <- data.frame(matrix(NA, ncol=1, nrow=20))
colnames(rssce_df_1) <- "rssce"
```

```r
for (i in 1:20){
  estimated_tmp <- coef(best_subset, id = i) %>% as.data.frame()
  names(estimated_tmp) <- c("estimate_beta")
  estimated_tmp$predictor = rownames(estimated_tmp)
  merged_df_tmp <- merge(x=betas_df, y=estimated_tmp, by="predictor", all.x=TRUE)
  merged_df_tmp[is.na(merged_df_tmp)] <- 0
  merged_df_tmp$difference <- (merged_df_tmp$betas - merged_df_tmp$estimate_beta) ^ 2
  rssce_df_1[i,] <- sqrt(sum(merged_df_tmp$difference))
}
rssce_df_1$num_predictor <- as.numeric(rownames(rssce_df_1))

ggplot(data=rssce_df_1, aes(num_predictor, rssce))+
  geom_vline(xintercept = which.min(rssce_df_1$rssce), linetype = 2) +
  geom_point()+
  geom_line()+
  labs(title = "Root sum sqaured coefficient error for each model size",
       subtitle = "The dashed line denotes the model with least root sum squared coefficient error",
       x = "Number of features",
       y = "Root sum sqaured coefficient error")
```

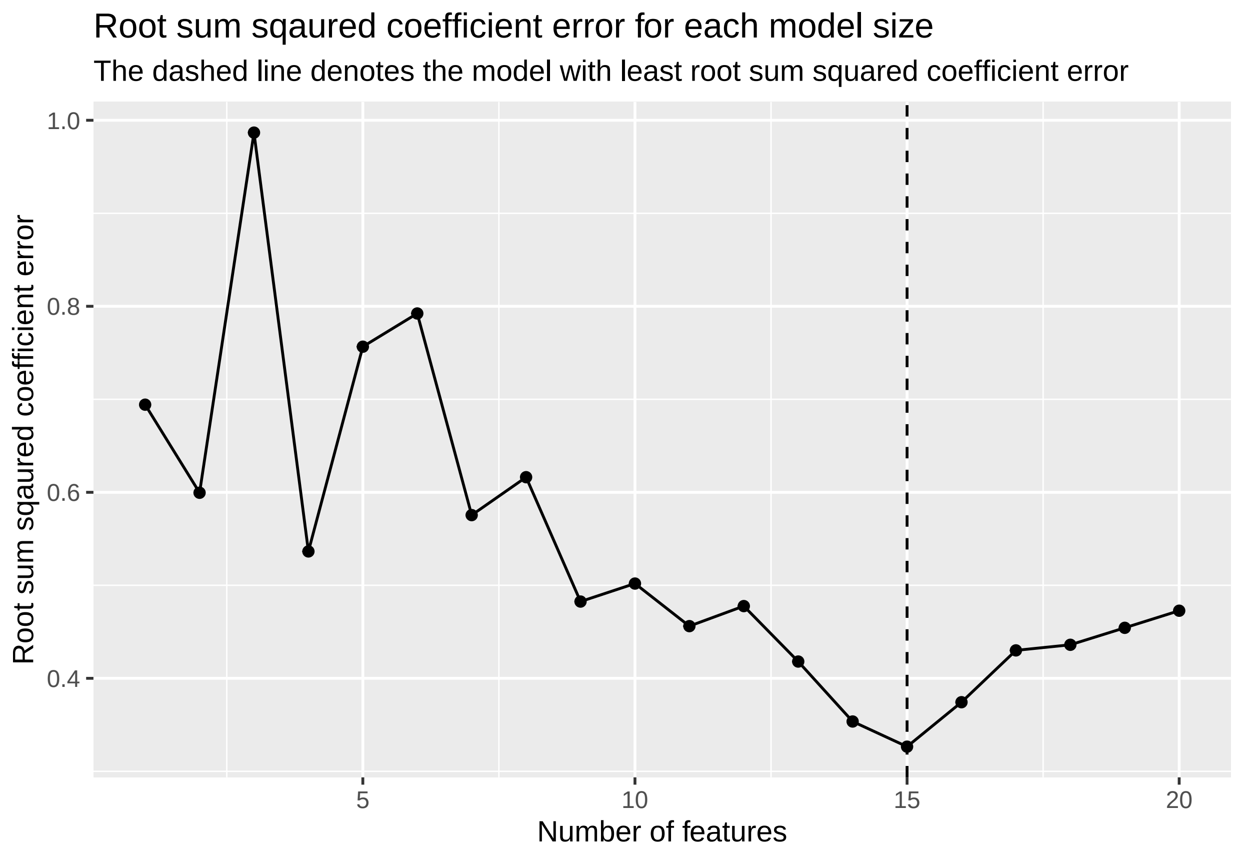


As the above plot shows, the root sum squared coefficient error shows a pattern that is very similar to the pattern shown in the test MSE plot. The root sum squared coefficient error decreases as the number of included features increases until it reaches the optimal model size (15). After that, the root sum squared coefficient error shows flattening or a slight increase as the model size get increased. This makes sense because a smaller difference between the estimated betas and true betas means that the model is accounting for the true model more accurately. Since the test MSE is a measure of how accurate a model is, it is natural that the test MSE and root sum coefficient error exhibits similar patterns.

However, comparing the root sum squared coefficient error this way could be unfair to the models that have a lower number of features. In a model that has a low number of included features, the square of all the betas associated with unincluded features will be added to the root sum squared coefficient error. This will increase the root sum squared coefficient error of the models with smaller model sizes. A way to account for this will be to calculate the root sum squared coefficient error only on the included features of each model, which is shown below.

```r
rownames(betas_df) <- betas_df$predictor
rssce_df_2 <- data.frame(matrix(NA, ncol=1, nrow=20))
colnames(rssce_df_2) <- "rssce"
for(i in 1:20){
  coefi <- coef(best_subset, id = i)[-1]
  xvars <- names(coefi)
  rssce_df_2[i,1] <- sqrt(sum((as.numeric(betas_df[xvars,]$betas) - as.numeric(coefi)) ^ 2))
}
rssce_df_2$num_predictor <- as.numeric(rownames(rssce_df_2))

ggplot(data=rssce_df_2, aes(num_predictor, rssce))+
  geom_vline(xintercept = which.min(rssce_df_2$rssce), linetype = 2) +
  geom_point()+
  geom_line()+
  labs(title = "Root sum sqaured coefficient error for each model size",
       subtitle = "The dashed line denotes the model with least root sum squared coefficient error",
       x = "Number of features",
       y = "Root sum sqaured coefficient error")
```



This plot shows that the root sum squared coefficient error calculated this way gradually decreases as model size increases until it reaches the optimal model size 15. After that, the root sum squared coefficient error increases as the model size gets larger. I think this shows a quite similar pattern with the above plot and the test MSE plot. It is true that this plot looks less smooth than the above plot and the test MSE plot, but I think the general tendency - gradual decrease until the model size reaches the optimal size (15) and slight increase afterward - remains the same. Also, this plot shares the same optimal model size with the other two plots, which could be another evidence that the patterns are similar.

The reason why this plot is less smooth than the test MSE plot is probably that when the number of features is smaller than the optimal size, the model tries to explain the response with the limited number of variables. This will cause the beta values to fluctuate, so the plot is not as smooth as the test MSE plot.

Application Excercises

```
gss_train <- read.csv("data/gss_train.csv")
gss_test <- read.csv("data/gss_test.csv")
```

1. Fit a least squares linear model on the training set, and report the test MSE.

```
lslm <- lm(egalit_scale ~ ., data=gss_train)
lslm_predict <- predict(lslm, gss_test)
lslm_mse <- rcfss::mse_vec(as.vector(gss_test$egalit_scale), lslm_predict)
lslm_mse
```

```
## [1] 63.21363
```

The test MSE is 63.21363 for the least squares linear model.

2. Fit a ridge regression model on the training set, with λ chosen by 10-fold cross-validation. Report the test MSE.

I note that I will use the same fold id for all models in problem 2,3,4. This will let the models use the same folds across models, leading to better comparisons.

```r
set.seed(163)
# getting the model matrix so I do not incldue egalit_scale
# add -1 to exclude the intercepts
gss_train_x <- model.matrix(egalit_scale ~ ., data=gss_train)[, -1]
gss_test_x <- model.matrix(egalit_scale ~ ., data=gss_test)[, -1]

# generating fold id to maintain the same folds across all models
# note that since x=1:10, this is roughly equivalent to 10-fold CV when we use foldid=fold_id
fold_id <- sample(1:10, size = length(gss_train$egalit_scale), replace = TRUE)

gss_ridge_cv <- glmnet::cv.glmnet(
  x = gss_train_x,
  y = gss_train$egalit_scale,
  alpha = 0,
  foldid = fold_id
)

gss_ridge_predict <- predict(gss_ridge_cv, gss_test_x, s="lambda.min")
rcfss::mse_vec(as.vector(gss_test$egalit_scale), as.vector(gss_ridge_predict))
```

```
## [1] 60.96434
```

The test MSE is 60.96434 for the ridge regression model.

3. Fit a lasso regression on the training set, with λ chosen by 10-fold cross-validation. Report the test MSE, along with the number of non-zero coefficient estimates.

```r
gss_lasso_cv <- glmnet::cv.glmnet(
  x = gss_train_x,
  y = gss_train$egalit_scale,
  alpha = 1,
  foldid = fold_id
)

gss_lasso_predict <- predict(gss_lasso_cv, gss_test_x, s="lambda.min")
rcfss::mse_vec(as.vector(gss_test$egalit_scale), as.vector(gss_lasso_predict))
```

```
## [1] 61.22341
```

```r
coeff_estimates_df_lasso <- gss_lasso_cv %>%
  coef(s = "lambda.min") %>%
  tidy() %>%
  filter("1" != "." & row != "(Intercept)")
length(coeff_estimates_df_lasso$row)
```

```
## [1] 29
```

The test MSE is 61.22341 for the lasso regression model. The number of non-zero coefficient estimates was 29 out of 78 features.

4. Fit an elastic net regression model on the training set, with α and λ chosen by 10-fold cross-validation. That is, estimate models with $\alpha = 0, 0.1, 0.2, \ldots, 1$ using the same values for λ across each model. Select the combination of α and λ with the lowest cross-validation MSE. For that combination, report the test MSE along with the number of non-zero coefficient estimates.

```r
# search across a range of alphas
tuning_grid <- data.frame(
  alpha      = seq(0, 1, by = .1),
  mse_min    = NA,
  lambda_min = NA
)

for(i in seq_along(tuning_grid$alpha)) {
  # fit CV model for each alpha value
  fit <- glmnet::cv.glmnet(gss_train_x,
                           gss_train$egalit_scale,
                           alpha = tuning_grid$alpha[i],
                           foldid = fold_id)
  # extract MSE and lambda values
  tuning_grid$mse_min[i]    <- fit$cvm[fit$lambda == fit$lambda.min]
  tuning_grid$lambda_min[i] <- fit$lambda.min
}

tuning_grid$alpha[which.min(tuning_grid$mse_min)]
```

```
## [1] 0.5
```

```r
tuning_grid$lambda_min[which.min(tuning_grid$mse_min)]
```

```
## [1] 0.4255691
```

```r
gss_elnet_cv <- glmnet::cv.glmnet(gss_train_x, gss_train$egalit_scale,
                                  alpha = tuning_grid$alpha[which.min(tuning_grid$mse_min)],
                                  foldid = fold_id)

# sanity check
# gss_elnet_cv$lambda.min == tuning_grid$lambda_min[which.min(tuning_grid$mse_min)]

gss_elnet_predict <- predict(gss_elnet_cv, gss_test_x, s="lambda.min")
rcfss::mse_vec(as.vector(gss_test$egalit_scale), as.vector(gss_elnet_predict))
```

```
## [1] 61.14024
```

```
coeff_estimates_df_elnet <- gss_elnet_cv %>%
  coef(s = "lambda.min") %>%
  tidy() %>%
  filter("1" != "." & row != "(Intercept)")
length(coeff_estimates_df_elnet$row)
```

```
## [1] 29
```

```
identical(coeff_estimates_df_elnet$row, coeff_estimates_df_lasso$row)
```

```
## [1] TRUE
```

The test MSE is 61.14024 for the best elastic regression model, which had α of 0.5 and λ of 0.4255691. The number of non-zero coefficient estimates was 29 out of 78 features. The selected features were identical to the selected features of lasso regression.

5. Comment on the results obtained. How accurately can we predict an individual's egalitarianism? Is there much difference among the test errors resulting from these approaches?

```
sqrt(rcfss::mse_vec(as.vector(gss_test$egalit_scale), as.vector(gss_ridge_predict)))
```

```
## [1] 7.807967
```

```
sqrt(rcfss::mse_vec(as.vector(gss_test$egalit_scale),
                    rep(median(gss_test$egalit_scale),length(gss_test$egalit_scale))))
```

```
## [1] 9.506697
```

Ridge, lasso, and elastic regression models all showed similar test MSE around 61. The least squared linear model performed slightly worse in terms of test MSE, showing MSE around 63. This makes sense since there were a lot of features in the data and it is likely that some of them were not very relevant to the individual's egalitarianism. The least square linear model does not regularize the coefficient estimates, so the model could overfit the variance caused by the errors in the training set using some redundant features. However, I note that the difference seems marginal at best.

The best performing model in terms of test MSE was the ridge regression model, which had test MSE of 60.96434. If we take the square root of it to scale it, RMSE becomes around 7.8. This could be seen that the prediction by the model was about 7.8 off on average from the actual value. I do not think this is great in terms of accuracy since the feature ranges from 1 to 35. Of course, it is better than the null model which predicts all value to be the median value of the egalitarian score (19), which shows RMSE around 9.5. However, I think a model that is a bit more flexible could predict the egalitarian scale better.

As previously mentioned, the models that utilize shrinkage methods to regularize the coefficient estimate shows very similar test MSEs. I think this result could be seen that there is no shrinkage method that is strictly superior to other shrinkage methods.

This article was written using R Markdown and was knitted to pdf using the tinytex package