Assaf Bitton

You can run the following code to prepare the analysis.

```
library(tidyverse)  #INSTALL IF NECESSARY

## Warning: package 'ggplot2' was built under R version 4.4.3

## Warning: package 'tidyr' was built under R version 4.4.2

## Warning: package 'dplyr' was built under R version 4.4.2

n <- 300
p <- 20
set.seed(111)
x <- matrix(rnorm(n*p), n, p)
beta <- rep(0, p)
beta[seq(1, 19, 2)] = seq(from = 0.1, to = 1, by = 0.1)
y <- x %*% beta + rnorm(n)
dat <- data.frame(x = x, y = y)
colnames(dat) <- c(paste0("x", 1:p), "y")
tr_dat <- dat[1:(n/2), ]
te_dat <- dat[-(1:(n/2)), ]
```

Suppose we want to build a linear regression model to predict `y`. Please answer the following questions.

Q1

Use the training data `tr_dat` using the following methods. For each method, output its regression coefficients, and compute its test error on `te_dat`. Looking at the true regression coefficients `beta`, discuss your findings on the following methods.

a. Best Subset Selection with BIC
b. Forward Stepwise Selection with Adjusted R^2
c. Backward Stepwise Selection with Cp
d. Ridge Regression with 10-fold CV
e. Lasso with 10-fold CV

Note: Use `set.seed(0)` before calling `cv.glmnet` to ensure reproducibility.

Solution:

```
library(leaps)
# Define a 'predict' method for 'regsubsets' objects
predict.regsubsets <- function(object, newdata, id) {
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefs <- coef(object, id = id)
  vars <- names(coefs)
  mat[, vars] %*% coefs
```

```
}
###################################
# a. Best Subset Selection with BIC
###################################
best_subset <- regsubsets(y ~ .,
                          data = tr_dat,
                          nvmax = 20)
best_subset_sum <- summary(best_subset)

best_id <- which.min(best_subset_sum$bic)

best_coef <- coef(best_subset, best_id)

# Best Subset Selection (BIC) Coefficients
best_coef
```

```
## (Intercept)          x3          x5          x7          x9         x11
##  0.05054974  0.25487147  0.30175393  0.40122277  0.67319974  0.47436124
##         x13         x15         x17         x19
##  0.68123969  0.85593997  0.88167860  1.06175205
```

```
pred_bic <- predict(best_subset,
                    newdata = te_dat,
                    id = best_id)
# Compute test errors
mse_bic <- mean((te_dat$y - pred_bic)^2)
mse_bic
```

```
## [1] 1.086088
```

```
########################################
# b. Forward Stepwise Selection with R^2
########################################
forward_fit <- regsubsets(y ~ .,
                          data = tr_dat,
                          method = "forward",
                          nvmax = 20)
forward_sum <- summary(forward_fit)

best_id_adjr2 <- which.max(forward_sum$adjr2)

forward_coef <- coef(forward_fit, best_id_adjr2)

# Forward Stepwise (Adj R^2) Coefficients
forward_coef
```

```
## (Intercept)          x1          x3          x4          x5          x7
##  0.01542060  0.12327731  0.24252397  0.13498448  0.27747364  0.42375748
##          x8          x9         x11         x12         x13         x15
##  0.10679685  0.67132119  0.50256514  0.08732112  0.71537591  0.82484632
```

```
##         x16         x17         x19
## -0.13172896  0.91630167  1.03904912
```

```
pred_forward <- predict(forward_fit,
                    newdata = te_dat,
                    id = best_id_adjr2)

# Compute test errors
mse_forward <- mean((te_dat$y - pred_forward)^2)
mse_forward
```

```
## [1] 1.170967
```

```
########################################
# c. Backward Stepwise Selection with Cp
########################################
backward_fit <- regsubsets(y ~ .,
                           data = tr_dat,
                           method = "backward",
                           nvmax = 20)
backward_sum <- summary(backward_fit)

best_id_cp <- which.min(backward_sum$cp)

backward_coef <- coef(backward_fit, best_id_cp)

# Backward Stepwise (Cp) Coefficients
backward_coef
```

```
## (Intercept)          x1          x3          x4          x5          x7
##  0.04186763  0.12887942  0.26095900  0.11548569  0.28745533  0.41191305
##          x9         x11         x13         x15         x17         x19
##  0.67575383  0.46927310  0.68605540  0.85010635  0.89619917  1.03873904
```

```
pred_backward <- predict(backward_fit,
                    newdata = te_dat,
                    id = best_id_cp)
# Compute test errors
mse_backward <- mean((te_dat$y - pred_backward)^2)
mse_backward
```

```
## [1] 1.148156
```

```
#####################################
# d. Ridge Regression with 10-fold CV
#####################################
library(glmnet)
set.seed(0)

x_tr <- model.matrix(y ~ .,
                     data = tr_dat)[, -1]
```

```r
x_te <- model.matrix(y ~ .,
                     data = te_dat)[, -1]
y_tr <- tr_dat$y
y_te <- te_dat$y

cv_fit_ridge <- cv.glmnet(x_tr,
                          y_tr,
                          alpha = 0)
ridge_coef <- coef(cv_fit_ridge)

# Ridge Regression Coefficients
ridge_coef
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  0.004352261
## x1           0.133801223
## x2           0.004905727
## x3           0.210883573
## x4           0.080544175
## x5           0.243223769
## x6          -0.039344950
## x7           0.334102040
## x8           0.084926232
## x9           0.557687958
## x10         -0.029847927
## x11          0.429092121
## x12          0.049250177
## x13          0.614787255
## x14          0.029007781
## x15          0.732111047
## x16         -0.109724668
## x17          0.818725283
## x18         -0.065487653
## x19          0.888469041
## x20          0.059139977
```

```r
pred_ridge <- predict(cv_fit_ridge,
                      newx = x_te)
# Compute test errors
mse_ridge <- mean((te_dat$y - pred_ridge)^2)
mse_ridge
```

```
## [1] 1.197607
```

```r
##################################
# e. Lasso with with 10-fold CV
##################################
set.seed(0)
cv_fit_lasso <- cv.glmnet(x_tr,
```

```
                        y_tr,
                        alpha = 1)

lasso_coef <- coef(cv_fit_lasso)

# Lasso Coefficients
lasso_coef
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.034096298
## x1           0.076722776
## x2           .
## x3           0.178684124
## x4           0.010367352
## x5           0.206348007
## x6           .
## x7           0.266272778
## x8           0.009337447
## x9           0.567724182
## x10          .
## x11          0.362222236
## x12          .
## x13          0.588806671
## x14          .
## x15          0.762683920
## x16         -0.013345077
## x17          0.847191874
## x18          .
## x19          0.946453113
## x20          .
```

```
pred_lasso <- predict(cv_fit_lasso,
                      newx = x_te)

# Compute test errors
mse_lasso <- mean((te_dat$y - pred_lasso)^2)
mse_lasso
```

```
## [1] 1.158921
```

- Using "**Best Subset Selection**" with method "**BIC**" results in a model with **9** predictors. Those **predictors** are **x3, x5, x7, x9, x11, x13, x15, x17, and x19**. The model **correctly identified** almost all **true predictors** and **correctly excluded** all **noise variables**. The model **dropped x1** since it's **true coefficient** is 0.1, which is **very small**. The **Test Error** is **1.086088**, which is the **lowest** test error achieved among the **subset/stepwise methods**.
- Using "**Forward Stepwise Selection**" with method "**Adjusted** $R2$" results in a model with **14** predictors. Those **predictors** are **x1, x3, x4, x5, x7, x8, x9, x11, x12,**

x13, x15, x16, x17, x19. The model **successfully** included **x1**, detecting the weak signla that **BIC missed**. However, because adjusted $R2$ is less strict than **BIC**, it **included 4 noise variables (x4, x8, x12, x16)** that should have been **zero**. The **Test Error** is 1.170967, which is **higher than BIC**. Therefore, the benefit of **including x1** was outweighed by the **variance introduced by the 4 noise variables**.

- Using "**Backward Stepwise Selection**" with method "**Cp**" results in a model with **11** predictors. Those **predictors** are **x1, x3, x4, x5, x7, x9, x11, x13, x15, x17, x19**. This model also correctly identified **x1** as a signal. The model **included 1 noise variable (x4)**. This resulted in a model that fit somewhere between **BIC** and adjusted $R2$. The **Test Error** is 1.148156, which fits **better than Forward Stepwise with adjusted** $R2$ but **worse than Best Subset with BIC**.
- Using "**Ridge Regression**" results in a model with **all 20** predictors. Ridge regression shrinks coefficients to zero, but does not perform variable selection. Because of this, all noise variables are still included in the model. The **Test Error** is 1.197607, which is the **highest error** among all of the methods. Since the true model only includes half the predictors, Ridge is at a disadvantage because **it cannot zero out** the **noise variables**.
- Using "**Lasso**" results in a model with **13** predictors. Those **predictors** are **x1, x3, x4, x5, x7, x8, x9, x11, x13, x15, x16, x17, x19**. Since lasso performs variable selection, it **successfully** zeroed out **most** of the **noise variables**. It managed to keep **x1** (with a coefficient of 0.077), despite its very weak signal, but also **kept x4 and x8** which are small **noise variables**. The **Test Error** is 1.158921, which **beat out Ridge** and **Forward Stepwise with adjusted** $R2$, but fell below **Best Subset with BIC** and **Backward stepwise with Cp**.

Q2

For the Simple Special Case for Ridge Regression, prove (6.14) on Page 248 of ISLRv2. Hint: Use (6.12) and the quadratic vertex formula.

Solution: