1 Part I

(a) Which of the three selection methods with k predictors has the smallest training RSS?

Best subset selection.

- Best subset selection examines all possible models of size k, ensuring the one with the lowest training RSS is chosen.
- Forward and backward stepwise selection are greedy algorithms and may miss the best subset, resulting in a higher training RSS.

(b) Which of the three selection methods with k predictors has the smallest test RSS?

It depends, but typically forward or backward stepwise selection may have lower test RSS than best subset selection.

- Best subset selection may overfit due to considering a larger number of models, leading to lower training RSS but potentially higher test RSS.
- Stepwise methods, by being more constrained, may generalize better (i.e., lower variance), leading to improved test performance in some cases.

(c) True or False: the predictors in Model 1 are a subset of the predictors in Model 2

Scenario	Model 1	Model 2	Answer
i.	Forward selection, *k* variables	Forward selection, *k+1* variables	**True** — Forward selection adds predictors one at a time.
ii.	Backward selection, *k* variables	Backward selection, *k+1* variables	**False** — Backward selection removes predictors one at a time, so no guarantee of strict nesting.
iii.	Backward selection, *k* variables	Forward selection, *k+1* variables	**False** — Different selection strategies can yield different sets.
iv.	Forward selection, *k* variables	Backward selection, *k+1* variables	**False** — Again, different methods and not necessarily nested.
v.	Best subset selection, *k* variables	Best subset selection, *k+1* variables	**False** — Best subset independently chooses the best model at each size; no nesting guaranteed.

1 Part II

(a) The lasso, relative to least squares, is:

iii. "Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance."

- The lasso introduces a penalty on the absolute value of the coefficients (L1 penalty), which shrinks some coefficients to zero.
- This leads to less flexibility than least squares (which fits the training data exactly).
- By reducing variance more than it increases bias, lasso can improve test prediction accuracy.

(b) Repeat (a) for ridge regression relative to least squares.

iii. "Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance."

- Ridge regression uses an L2 penalty (squares of the coefficients), which also shrinks coefficients but doesn't zero them out like lasso.
- Like lasso, ridge regression is **less flexible** than least squares and reduces variance at the cost of increased bias.
- If the reduction in variance outweighs the increase in bias, prediction accuracy improves.

2

(a) As we increase *s* from 0, the training RSS will:

iv. Steadily decrease.

- When $s = \mathit{0}$, all $\beta_j = 0$, so the model is just the intercept.
- As s increases, more coefficients can become non-zero, allowing a better fit.
- Therefore, the training RSS **decreases monotonically** as s increases.

(b) Repeat (a) for test RSS.

ii. Decrease initially, and then eventually start increasing in a U shape.

- Initially, increasing s allows the model to capture more true structure $\rightarrow$ **test RSS decreases**.
- After a point, model becomes too complex and overfits $\rightarrow$ **test RSS increases**.
- This yields a **U-shaped curve** for test RSS.

(c) Repeat (a) for variance.

iv. Steadily increase.

- As more parameters are added (increasing s), the model becomes more flexible.
- This flexibility leads to higher **variance**.

(d) Repeat (a) for (squared) bias.

iv. Steadily decrease.

- With small s, the model is constrained, and **underfits**, resulting in high bias.
- As s increases, the model becomes more flexible and **bias decreases**.

3

(a) Write out the ridge regression optimization problem in this setting.

We minimize:

$$\sum_{i=1}^{2} \left(y_i - (\beta_1 x_{i1} + \beta_2 x_{i2})\right)^2 + \lambda(\beta_1^2 + \beta_2^2)$$

Given that $x_{i1} = x_{i2}$, we can rewrite as:

$$\sum_{i=1}^{2} \left(y_i - (\beta_1 + \beta_2) x_{i1}\right)^2 + \lambda(\beta_1^2 + \beta_2^2)$$

(b) Argue that in this setting, the ridge coefficient estimates satisfy $\hat{\beta}_1 = \hat{\beta}_2$

Since $x_{i1} = x_{i2}$ for both observations, any linear combination where $\beta_1 + \beta_2 = c$ produces the same fit. However, the ridge penalty $\lambda(\beta_1^2 + \beta_2^2)$ is minimized when $\beta_1 = \beta_2$. Therefore, ridge regression will choose $\hat{\beta}_1 = \hat{\beta}_2$.

(c) Write out the lasso optimization problem in this setting.

We minimize:

$$\sum_{i=1}^{2} (y_i - (\beta_1 x_{i1} + \beta_2 x_{i2}))^2 + \lambda(|\beta_1| + |\beta_2|)$$

Again, since $x_{i1} = x_{i2}$, this becomes:

$$\sum_{i=1}^{2} (y_i - (\beta_1 + \beta_2) x_{i1})^2 + \lambda(|\beta_1| + |\beta_2|)$$

(d) Argue that in this setting, the lasso coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique

The objective function depends only on the sum $\beta_1 + \beta_2$, not the individual values. For a fixed sum, say $\beta_1 + \beta_2 = c$, the lasso penalty $|\beta_1| + |\beta_2|$ is minimized by making one of the coefficients zero if possible.

Hence, multiple combinations of β_1 and β_2 achieve the same fit but different penalties, and some may tie for the minimum. For example:

- $\beta_1 = c, \beta_2 = 0$
- $\beta_1 = 0, \beta_2 = c$
- Or any combination on the line $\beta_1 + \beta_2 = c$ that minimizes the L1 norm.

Therefore, **the solution is not unique.**

4

(a)

```r
# Load necessary libraries
library(ISLR2)
library(glmnet)
library(caret)

# Load the College data
data(College)

# (a) Split the data into a training set and a test set
set.seed(123)
train_index <- createDataPartition(College$Apps, p = 0.7, list = FALSE)
train_data <- College[train_index, ]
test_data <- College[-train_index, ]

# Prepare matrices for glmnet
x_train <- model.matrix(Apps ~ ., train_data)[, -1]
y_train <- train_data$Apps
x_test <- model.matrix(Apps ~ ., test_data)[, -1]
y_test <- test_data$Apps
```

(b)

```r
# (b) Fit a linear model using least squares
lm_fit <- lm(Apps ~ ., data = train_data)
lm_pred <- predict(lm_fit, newdata = test_data)
lm_mse <- mean((lm_pred - y_test)^2)
cat("(b) Linear Model Test MSE:", lm_mse, "\n")
```

```
## (b) Linear Model Test MSE: 1882074
```

(c)

```r
# (c) Ridge regression with cross-validation
ridge_cv <- cv.glmnet(x_train, y_train, alpha = 0)
ridge_best_lambda <- ridge_cv$lambda.min
ridge_pred <- predict(ridge_cv, s = ridge_best_lambda, newx = x_test)
ridge_mse <- mean((ridge_pred - y_test)^2)
cat("(c) Ridge Regression Test MSE:", ridge_mse, "\n")
```

```
## (c) Ridge Regression Test MSE: 3265646
```

(d)

```r
# (d) Lasso regression with cross-validation
lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1)
lasso_best_lambda <- lasso_cv$lambda.min
lasso_pred <- predict(lasso_cv, s = lasso_best_lambda, newx = x_test)
lasso_mse <- mean((lasso_pred - y_test)^2)
cat("(d) Lasso Regression Test MSE:", lasso_mse, "\n")
```

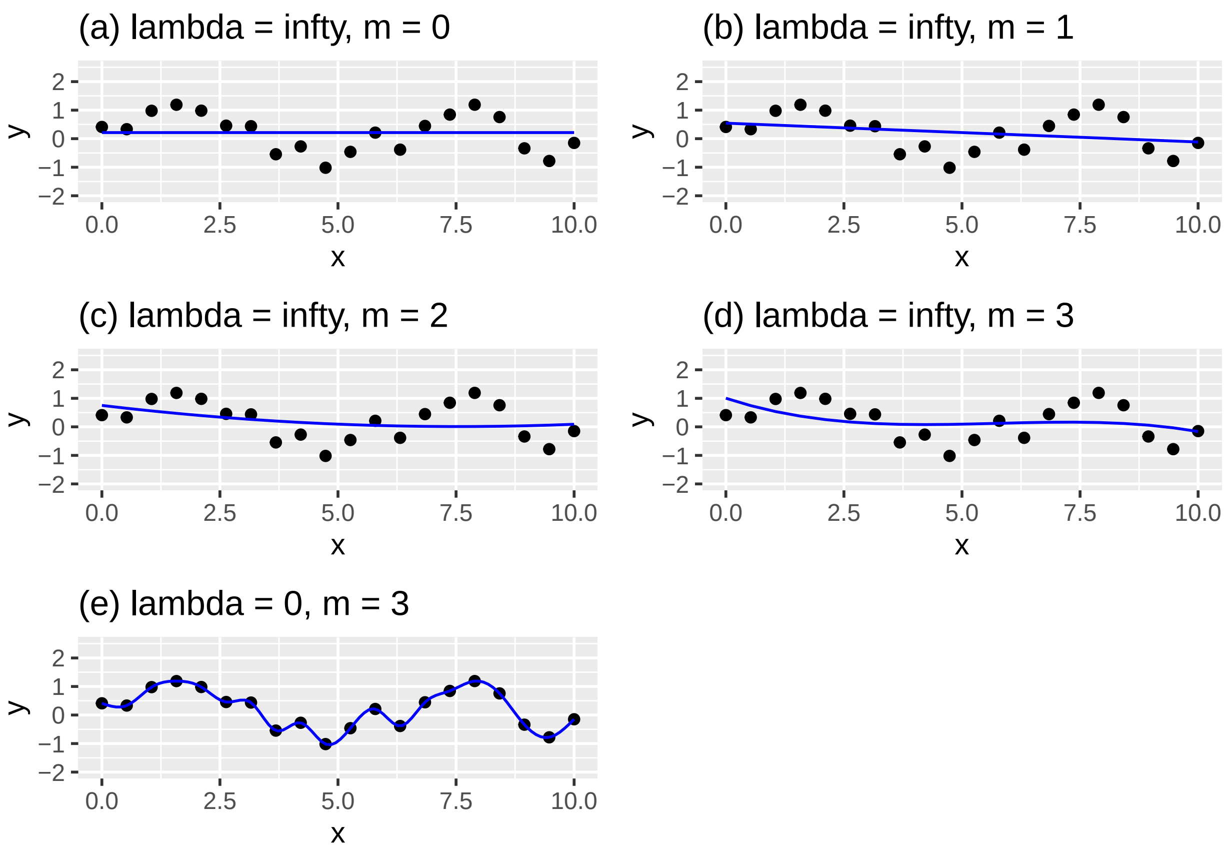
```
## (d) Lasso Regression Test MSE: 1942428
```

```r
# Number of non-zero coefficients
lasso_model <- glmnet(x_train, y_train, alpha = 1, lambda = lasso_best_lambda)
num_nonzero <- sum(coef(lasso_model) != 0) - 1  # exclude intercept
cat("(d) Number of non-zero Lasso coefficients:", num_nonzero, "\n")
```

```
## (d) Number of non-zero Lasso coefficients: 16
```

5

```r
# Load necessary libraries
library(ggplot2)
library(gridExtra)
set.seed(42)

# Generate sample data
n <- 20
x <- seq(0, 10, length.out = n)
y <- sin(x) + rnorm(n, sd = 0.3)
df <- data.frame(x = x, y = y)

# (a) lambda = infty, m = 0 → g(x) = constant
fit_a <- mean(y)
df_a <- data.frame(x = x, y = rep(fit_a, n))

# (b) lambda = infty, m = 1 → g(x) = straight line
fit_b <- lm(y ~ x)
df_b <- data.frame(x = x, y = predict(fit_b))

# (c) lambda = infty, m = 2 → g(x) = quadratic curve
fit_c <- lm(y ~ poly(x, 2))
df_c <- data.frame(x = x, y = predict(fit_c))

# (d) lambda = infty, m = 3 → g(x) = cubic curve
fit_d <- lm(y ~ poly(x, 3))
df_d <- data.frame(x = x, y = predict(fit_d))

# (e) lambda = 0, m = 3 → interpolate all points (overfit)
fit_e <- spline(x, y, n = 100, method = "natural")
df_e <- data.frame(x = fit_e$x, y = fit_e$y)

# Plotting
p_base <- ggplot(df, aes(x = x, y = y)) + geom_point() + ylim(-2, 2.5)

p_a <- p_base + geom_line(data = df_a, color = "blue") + ggtitle("(a) lambda = infty, m = 0")
p_b <- p_base + geom_line(data = df_b, color = "blue") + ggtitle("(b) lambda = infty, m = 1")
p_c <- p_base + geom_line(data = df_c, color = "blue") + ggtitle("(c) lambda = infty, m = 2")
p_d <- p_base + geom_line(data = df_d, color = "blue") + ggtitle("(d) lambda = infty, m = 3")
p_e <- p_base + geom_line(data = df_e, color = "blue") + ggtitle("(e) lambda = 0, m = 3")

# Display all plots
grid.arrange(p_a, p_b, p_c, p_d, p_e, ncol = 2)
```



(a) lambda = infty, m = 0
(b) lambda = infty, m = 1
(c) lambda = infty, m = 2
(d) lambda = infty, m = 3
(e) lambda = 0, m = 3
y
x
2
1
0
−1
−2
0.0
2.5
5.0
7.5
10.0