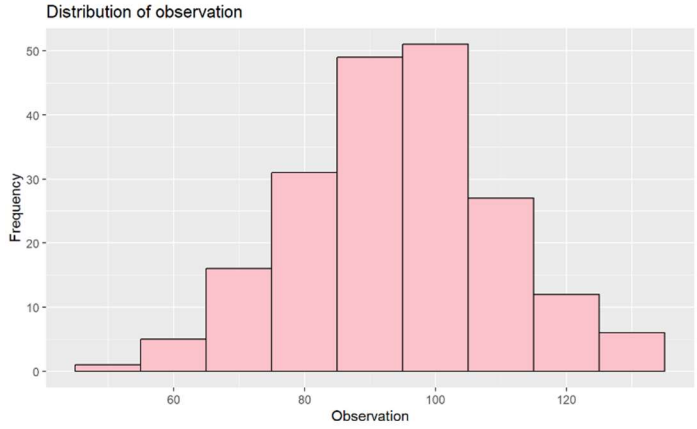
STAT 6340 Mini Project 4

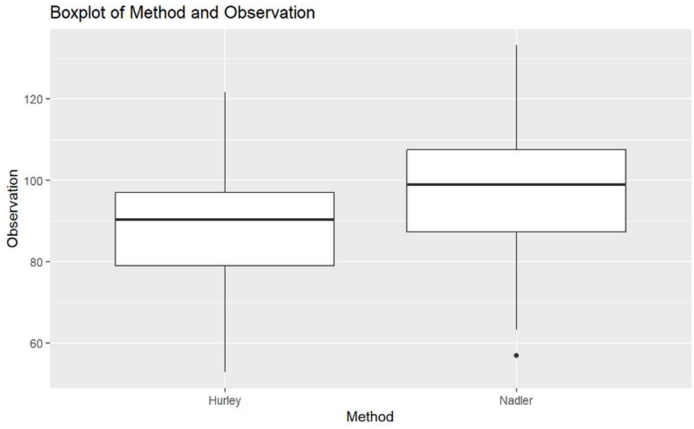
Lakshmipriya Narayanan

SECTION 1: Observations and Answers

1

(a): Exploratory analysis of data:





The histogram above indicates that the subject observations approximately form a normal distribution. From the boxplot we can see that subjects have higher plasma observations when measured by the Nadler method. This method also has an outlier which may or may not be influential. It needs further analysis.

(b): Point estimate = 0.9902. Since this value for correlation between the two methods is very close to 1, we can conclude that there is a strong, positive, and linear relationship between these two methods. Therefore, this implies that the two different methods produce very similar observations/results. We can use bootstrap to compute bias and standard error of correlation estimate. It is done in (c) and (d) below.

(c):
Bias estimate = 4.042×10^{-5} = 0.00004042. This Is very small => the bootstrap estimates are approximately unbiased.
Standard error estimate = 0.00184. Also very small indicating that the bootstrap estimates of the correlation have little variability and are consistent.
95% CI = [0.9863, 0.9936]. This interval is narrow, indicating that it captures the true value of the estimate (0.9902 belongs to this interval).

(d):
Bias estimate = 5.788×10^{-5} = 0.00005788
Standard error estimate = 0.00179
95% CI = [0.9865, 0.9935].

All of the estimates are extremely close to what we obtained in (c) above.

(e): The two methods for measuring plasma content are highly correlated implying that they both provide similar measurements. Therefore, one can use either of the methods to measure plasma content in a person.

2

(a) – (c):

Included in (d) below

(d):

Question	Method	Penalty parameter	Test error rate
2(a)	Logistic Regression	-	0.2226
2(b)	Ridge regression	0.01	0.2200
2(c)	Lasso	0.01	0.2265

Comparing test error rates, we can recommend the **ridge regression** method because it has the lowest test error rate amongst the three methods albeit their similarity in the error rates. It is also valuable to notice that the penalty parameters for the ridge regression and the lasso are the same when estimated by LOOCV. This contradicts the conclusion from Mini Project 3 where we recommended a logistic regression model.

3

(a) – (f): included in (g) below:

(b): Dropped predictors according to best subset selection:
Y, months (Jan, Feb, Apr, May, Jul, Aug, Nov), RH, ISI, FFMC, rain, days (Mon, Wed, Thurs, Fri, Sun)

(c): Dropped predictors according to forward stepwise selection:
Y, months (Jan, Mar, Apr, May, Jun, Jul, Aug, Nov), RH, ISI, FFMC, rain, days (Mon, Wed, Thurs, Fri, Sun).
The model in (b) above also has these predictors dropped but a few changes in the qualitative variable month (here, Feb, Mar, Jun are also dropped).

(d): Dropped predictors according to backward stepwise selection is the same as best subset selection above.

(g):

Question	Method	R^2_{adj}	Penalty parameter	Test MSE
3(a)	Linear Regression (all predictors)	0.022	-	2.108
3(b)	Best Subset Selection	0.0429	-	1.888
3(c)	Forward Stepwise Selection	0.0414	-	1.889
3(d)	Backward Stepwise Selection	0.0429	-	1.888
3(e)	Ridge Regression	-	10.722	1.929
3(f)	Lasso	-	0.0932	1.914

We notice that the results for best subset selection match the backward stepwise selection method. Hence, we recommend either **best subset selection** or **backward stepwise selection**. It is worth noting that the forward stepwise selection method's MSE is extremely very close to the recommended methods above.

Section 2: Code

```
#Load Required Libraries
library(ggplot2)
library(boot)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(leaps)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
set.seed(1)
```

Problem 1

(a):

```r
plasma <- read.csv("plasma_volume.csv")

#---------DATA ANALYSIS : ------------------------
#Check for missing values
sum(is.na(plasma))
```

```
## [1] 0
```

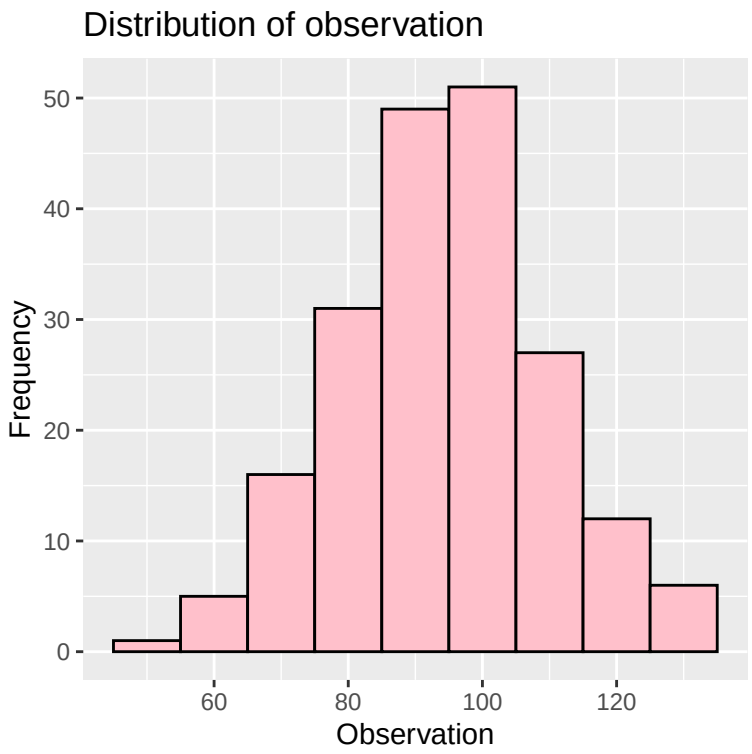
```r
#Check the structure of the data we are working with to know what variables we have, etc.
str(plasma)
```

```
## 'data.frame':    198 obs. of  3 variables:
##  $ method     : chr  "Nadler" "Nadler" "Nadler" "Nadler" ...
##  $ subject    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ observation: num  56.9 63.2 65.5 73.6 74.1 77.1 77.3 77.5 77.8 78.9 ...
```

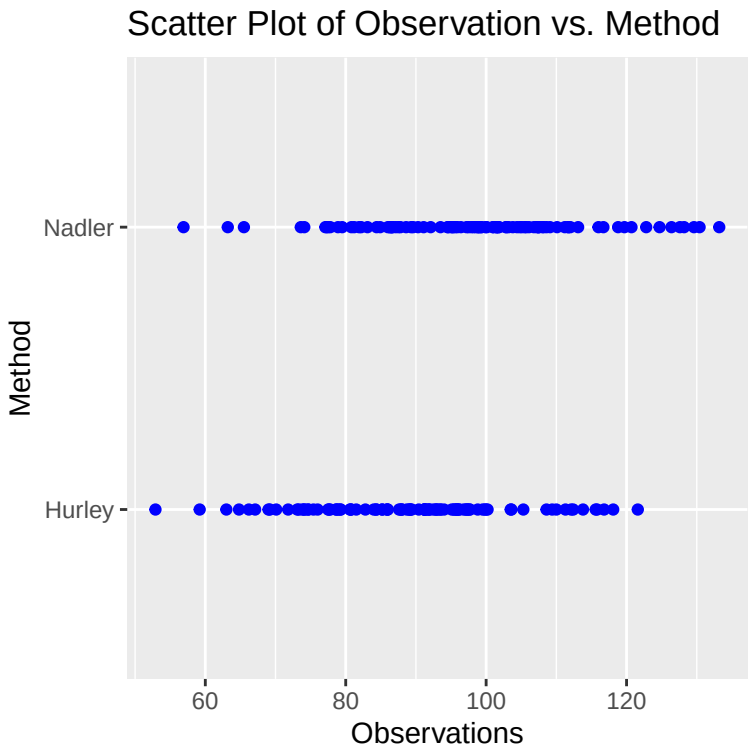
```r
summary(plasma)
```

```
##     method             subject       observation
##  Length:198         Min.   : 1.00   Min.   : 52.90
##  Class :character   1st Qu.:25.25   1st Qu.: 84.33
##  Mode  :character   Median :50.00   Median : 94.25
##                     Mean   :50.00   Mean   : 93.87
##                     3rd Qu.:74.75   3rd Qu.:103.42
##                     Max.   :99.00   Max.   :133.20
```

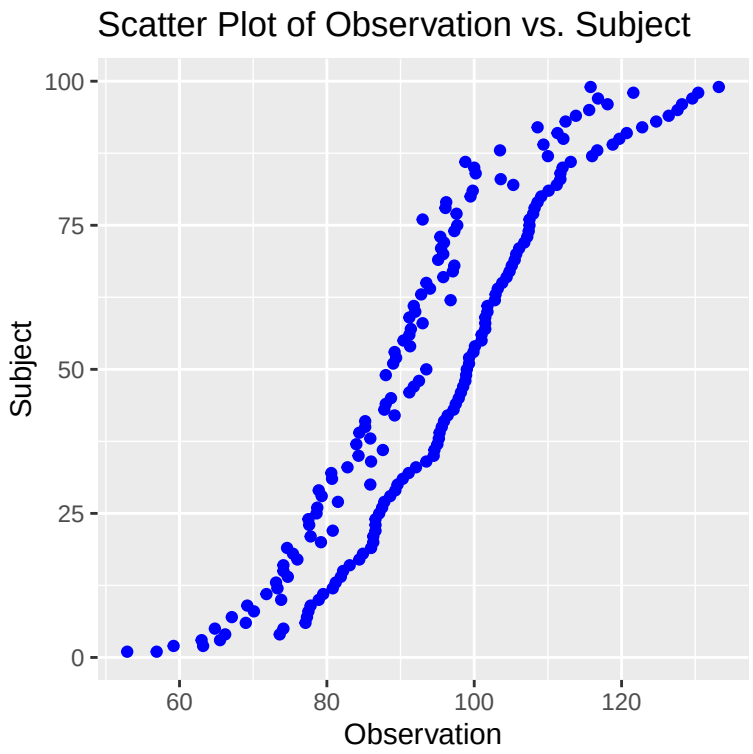
```r
#-------------------------PLOTS : ---------------------------------------

ggplot(plasma, aes(x = observation)) +
  geom_histogram(binwidth = 10, fill = "pink", color = "black") +
  labs(title = "Distribution of observation", x = "Observation", y = "Frequency")
```



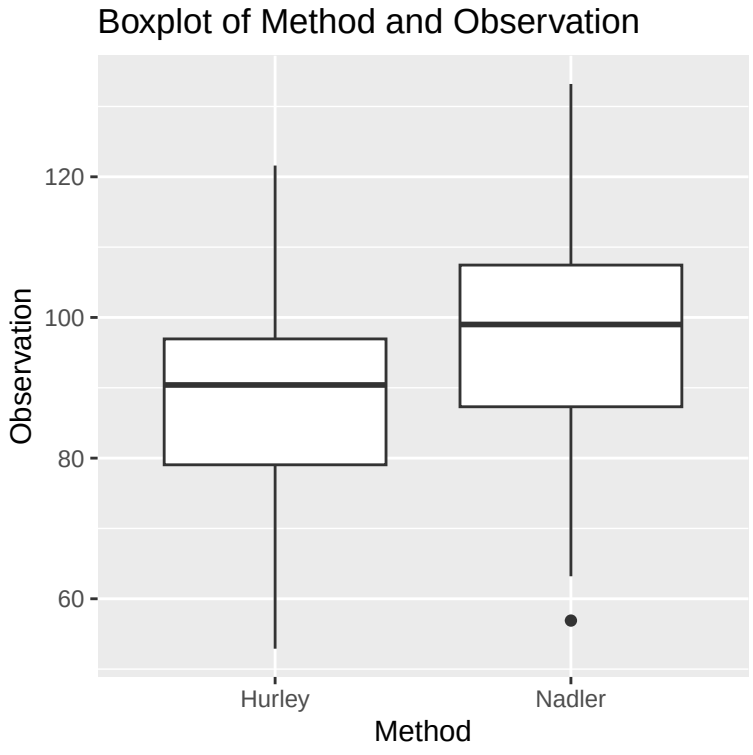

```
ggplot(plasma, aes(x=observation, y=method)) + geom_point(color="blue") +
  labs(title="Scatter Plot of Observation vs. Method", x="Observations", y="Method")
```




```
ggplot(plasma, aes(x=observation, y=subject)) + geom_point(color="blue") +
  labs(title="Scatter Plot of Observation vs. Subject", x="Observation", y="Subject")
```




```
ggplot(plasma, aes(x=method, y=observation)) + geom_boxplot() +
  labs(title="Boxplot of Method and Observation", x="Method", y="Observation")
```




```
#observation from boxplot: Nadler method observation are higher. also has an outlier.
#Observations histogram implies normality
```

(b):

```
# Split the data by subject to obtain correlation between them
nadler <- plasma[plasma$method == "Nadler", "observation"]
hurley <- plasma[plasma$method == "Hurley", "observation"]

point_est <- cor(nadler, hurley)
point_est
```

```
## [1] 0.9902389
```

(c):

```
# Ensure both vectors are of the same length for calculating bootstrap estimates
min_length <- min(length(nadler), length(hurley))
#min_length
nadler <- nadler[1:min_length]
hurley <- hurley[1:min_length]

# Number of bootstrap samples, B = 1000
B <- 1000
bootstrap_cor <- numeric(B)#Initialize a vector of zeroes of size 1000 for replication
#bootstrap_cor


#Bootstrap from scratch
for (i in 1:B) {
  # Generate bootstrap sample indices
  indices <- sample(1:min_length, min_length, replace = TRUE)

  # Calculate correlation for the bootstrap sample for comparison
  bootstrap_cor[i] <- cor(nadler[indices], hurley[indices])
}

#Bias = mean - correaltion
bias_est <- mean(bootstrap_cor) - point_est
cat("Bias =", bias_est, "\n")
```

```
## Bias = 4.042549e-05
```

```
#Standard error
se_est <- sd(bootstrap_cor)
cat("Standard Error = ", se_est, "\n")
```

```
## Standard Error =  0.001846272
```

```r
#95% confidence interval using the percentile method
ci_lower <- quantile(bootstrap_cor, 0.025)
ci_upper <- quantile(bootstrap_cor, 0.975)
cat("95% CI: [",ci_lower, ",", ci_upper,"]")
```

```
## 95% CI: [ 0.9863098 , 0.993677 ]
```

#(d):

```r
# Define a function to calculate the correlation
cor_func <- function(data, indices) {
  data_updated <- data[indices, ]
  return(cor(data_updated$nadler, data_updated$hurley))
}

# Combine the data into a data frame
data_method <- data.frame(nadler = nadler, hurley = hurley)

# Perform bootstrapping
results_bootstrap <- boot(data_method, statistic = cor_func, R = 1000)
results_bootstrap
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data_method, statistic = cor_func, R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.9902389 5.788359e-06 0.001790789
```

```r
# Calculate 95% confidence interval
ci_cor <- boot.ci(results_bootstrap, type = "perc")
ci_cor
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results_bootstrap, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 0.9865,  0.9935 )
## Calculations and Intervals on Original Scale
```

Problem 2:

```
# TAKEN COMPLETELY FROM MY Mini Project 3 (Problem 2)
diab_data <- read.csv("C:/Users/lakpr/Documents/Working Directory STAT ML 6340/diabetes.csv")

#Convert outcome to a factor variable because it is qualitative
diab_data$Outcome <- as.factor(diab_data$Outcome)

#------------------STANDARDIZING ALL PREDICTORS--------------------------------
# Function to standardize the training variable
standardize <- function(x) {
  (x - mean(x)) / sd(x)
}

# Standardize all columns except "Outcome"
standardize_Diab <- diab_data %>%
  mutate(across(-Outcome, standardize))
```

(a):

```
# TAKEN COMPLETELY FROM MY Mini Project 3 (Problem 2a and 2c)

#set.seed(1)
#Logistic regression model with all predictors
diab_log_reg_all <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure +
                    SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                        family = binomial, data = standardize_Diab)
#summary(diab_log_reg_all)

# Define a cost function for error rate
cost_loocv <- function(r, pi = 0) mean(abs(r - pi) > 0.5)

# Perform LOOCV using the boot package
cv_error_new <- cv.glm(standardize_Diab, diab_log_reg_all, cost = cost_loocv,
                       K = nrow(standardize_Diab))

# Calculate the misclassification error rate
test_error_rate <- cv_error_new$delta[1]
cat("Test Error Rate using LOOCV:", test_error_rate, "\n")
```

```
## Test Error Rate using LOOCV: 0.2226563
```

(b):

```r
# Set response and predictor matrices. The design matrix X includes 1's in the
# first column. We consider the predictor matrix without these 1's
y_resp <- standardize_Diab$Outcome
x_pred <- model.matrix(Outcome ~ ., standardize_Diab)[, -1]

# Define a grid of lambda values
grid <- 10^seq(10, -2, length = 100)

# Fit ridge regression model for each lambda (penalty parameter) on the grid above.
ridge_reg_diab_all <- glmnet(x_pred, y_resp, alpha = 0, lambda = grid,
                             family = "binomial")

# Perform LOOCV to find the optimal/best lambda
loocv_ridge <- cv.glmnet(x_pred, y_resp, alpha = 0, nfolds = nrow(standardize_Diab),
                         lambda = grid, family = "binomial", type.measure = "class")
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```r
# Penalty parameter is the minimum value of the lambdas
penalty_par <- loocv_ridge$lambda.min
penalty_par
```

```
## [1] 0.01
```

```r
# Fit the ridge regression model with the optimal lambda to help calculate
# the test MSE for the best lambda
ridge_reg_diab <- glmnet(x_pred, y_resp, alpha = 0, lambda = penalty_par,
                         family = "binomial")

#Same method followed as in previous mini projects
predictions <- predict(ridge_reg_diab, newx = x_pred, type = "response")
predicted_classes <- ifelse(predictions > 0.5, "1", "0")
test_error_rate <- mean(predicted_classes != y_resp)
cat("Test MSE using Ridge Regression:", test_error_rate, "\n")
```

```
## Test MSE using Ridge Regression: 0.2200521
```

(c):

```r
# Fit lasso model for each lambda (penalty parameter) on the grid created in (b)
lasso_diab_all <- glmnet(x_pred, y_resp, alpha = 1, lambda = grid,
                         family = "binomial")

# Perform LOOCV to find the optimal lambda for the lasso method
loocv_lasso <- cv.glmnet(x_pred, y_resp, alpha = 1, nfolds = nrow(standardize_Diab),
                         lambda = grid, family = "binomial", type.measure = "class")
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```
# Best lambda
penalty_par_lasso<- loocv_lasso$lambda.min
penalty_par_lasso
```

```
## [1] 0.01
```

```
# Fit the ridge regression model with the best lambda to obtain test MSE
lasso_diab <- glmnet(x_pred, y_resp, alpha = 1, lambda = penalty_par,
                     family = "binomial")

predictions <- predict(lasso_diab, newx = x_pred, type = "response")
predicted_classes <- ifelse(predictions > 0.5, "1", "0")
test_error_rate <- mean(predicted_classes != y_resp)
cat("Test Error Rate using Ridge Regression:", test_error_rate, "\n")
```

```
## Test Error Rate using Ridge Regression: 0.2265625
```

Problem 3:

```
# TAKEN COMPLETELY FROM MY Mini Project 2 (Problem 2a and 2c)

#Read the data and convert qualitative variables 'day' and 'month' to factors
trainforest <- read.csv("C:/Users/lakpr/Documents/Working Directory STAT ML 6340/forestfires.csv")

trainforest$day <- as.factor(trainforest$day)
trainforest$month <- as.factor(trainforest$month)

#transformation of area variable
trainforest$area_new <- log(trainforest$area + 1)

#sum(is.na(trainforest))
```

(a):

```
train_ctrl <- trainControl(method = "LOOCV")

# Train the regression model using the train function defined above
multi_lm_forest <- train(area_new ~ temp + X + Y + month  + DMC + DC + RH + ISI +
                           day + FFMC + wind + rain , data = trainforest,
                         method = "lm", trControl = train_ctrl)
summary(multi_lm_forest)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9959 -1.0299 -0.4997  0.8434  5.2001
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.4918597  1.6460189  -0.299  0.76521
## temp         0.0340979  0.0221701   1.538  0.12469
## X            0.0520233  0.0321987   1.616  0.10681
## Y           -0.0355402  0.0609138  -0.583  0.55986
## monthaug     0.2719539  0.8189967   0.332  0.73999
## monthdec     2.1904025  0.7942663   2.758  0.00604 **
## monthfeb     0.1842463  0.5577891   0.330  0.74130
## monthjan    -0.3390164  1.2125469  -0.280  0.77991
## monthjul     0.0845391  0.7105195   0.119  0.90534
## monthjun    -0.2845657  0.6521446  -0.436  0.66277
## monthmar    -0.3385076  0.5033235  -0.673  0.50156
## monthmay     0.7101212  1.0944982   0.649  0.51677
## monthoct     0.7603841  0.9766298   0.779  0.43660
## monthsep     0.9360814  0.9176266   1.020  0.30818
## DMC          0.0039404  0.0018681   2.109  0.03543 *
## DC          -0.0018874  0.0012629  -1.494  0.13571
## RH           0.0007592  0.0062007   0.122  0.90260
## ISI         -0.0120540  0.0178924  -0.674  0.50083
## daymon       0.1479392  0.2253147   0.657  0.51175
## daysat       0.3170128  0.2163143   1.466  0.14342
## daysun       0.2188758  0.2104396   1.040  0.29881
## daythu       0.0824648  0.2387867   0.345  0.72998
## daytue       0.3244884  0.2339426   1.387  0.16606
## daywed       0.1337699  0.2462862   0.543  0.58728
## FFMC         0.0076552  0.0165489   0.463  0.64387
## wind         0.0564641  0.0382512   1.476  0.14055
## rain         0.0382507  0.2133982   0.179  0.85782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.373 on 488 degrees of freedom
## Multiple R-squared:  0.07149,    Adjusted R-squared:  0.02202
## F-statistic: 1.445 on 26 and 488 DF,  p-value: 0.07353
```

```
# Calculate the test MSE = Root MSE * Root MSE
test_mse <- multi_lm_forest$results$RMSE^2
test_mse
```

```
## [1] 2.108706
```

```
#adjusted R^2 = 0.02202
```

(b):

```
# Total number of predictors in the data = number of columns - area - area_new
totpred <- ncol(trainforest) - 2
totpred
```

```
## [1] 12
```

```
# Fit the full model with best subset selection method using 'regsubsets' function.
fit_full <- regsubsets(area_new ~ temp + X + Y + month  + DMC + DC + RH + ISI +
                day + FFMC + wind + rain , data = trainforest, nvmax = totpred)

# Get the maximum adjusted R^2
summary_full <- summary(fit_full)
summary_full$adjr2        # All the adjusted R^2 values for each predictor
```

```
##  [1] 0.01797010 0.02613555 0.03257891 0.03561858 0.03739573 0.03871298
##  [7] 0.03958968 0.04034044 0.04156734 0.04246959 0.04259422 0.04296864
```

```
max(summary_full$adjr2)   # Finds the maximum Adj. R^2 value from the above list
```

```
## [1] 0.04296864
```

```
max_adjr2 <- which.max(summary_full$adjr2)

# Print the coefficients of all predictors selected in the best subset selection model.
print(coef(fit_full, max_adjr2))
```

```
##  (Intercept)         temp            X     monthdec     monthjun     monthmar
##  0.285138803  0.031908549  0.041435495  1.982703951 -0.483896795 -0.458031877
##     monthoct     monthsep          DMC           DC       daysat       daytue
##  0.520631256  0.649283726  0.003832441 -0.001595714  0.198367898  0.204355275
##         wind
##  0.053429096
```

```
#dropped predictors: y, month(JF AM JA  N ), RH, ISI, FFMC,rain, day(M WTF S)

# Since we are unable to dro qualitative predictors like specific months or days,
# make a new dataset that splits all the levels/ classes of a categorical variable.

ff_split <- data.frame(model.matrix(~ . -1, data = trainforest))
str(ff_split)
```

```
## 'data.frame':    515 obs. of  29 variables:
##  $ X       : num  7 7 7 8 8 8 8 8 7 ...
##  $ Y       : num  5 4 4 6 6 6 6 6 5 ...
##  $ monthapr: num  0 0 0 0 0 0 0 0 0 ...
##  $ monthaug: num  0 0 0 0 0 1 1 1 0 0 ...
##  $ monthdec: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthfeb: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthjan: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthjul: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthjun: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthmar: num  1 0 0 1 1 0 0 0 0 0 ...
##  $ monthmay: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ monthoct: num  0 1 1 0 0 0 0 0 0 0 ...
##  $ monthsep: num  0 0 0 0 0 0 0 0 1 1 ...
##  $ daymon  : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ daysat  : num  0 0 1 0 0 0 0 0 0 1 ...
##  $ daysun  : num  0 0 0 0 1 1 0 0 0 0 ...
##  $ daythu  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ daytue  : num  0 1 0 0 0 0 0 0 1 0 ...
##  $ daywed  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ FFMC    : num  86.2 90.6 90.6 91.7 89.3 92.3 92.3 91.5 91 92.5 ...
##  $ DMC     : num  26.2 35.4 43.7 33.3 51.3 ...
##  $ DC      : num  94.3 669.1 686.9 77.5 102.2 ...
##  $ ISI     : num  5.1 6.7 6.7 9 9.6 14.7 8.5 10.7 7 7.1 ...
##  $ temp    : num  8.2 18 14.6 8.3 11.4 22.2 24.1 8 13.1 22.8 ...
##  $ RH      : num  51 33 33 97 99 29 27 86 63 40 ...
##  $ wind    : num  6.7 0.9 1.3 4 1.8 5.4 3.1 2.2 5.4 4 ...
##  $ rain    : num  0 0 0 0.2 0 0 0 0 0 0 ...
##  $ area    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ area_new: num  0 0 0 0 0 0 0 0 0 0 ...
```

```
# Fit the new model with the dropped predictors
loocv_best <- train(area_new ~ temp + X + monthdec + monthjun + monthmar +
                      monthoct + monthsep + DMC + DC + daysat + daytue + wind,
                    data = ff_split, method = "lm", trControl = train_ctrl)

# Obtain the test MSE of the above best model. Test MSE = (Root MSE)^2
loocv_best$results$RMSE^2
```

```
## [1] 1.888273
```

(c):

```
# Fit the full model with forward stepwise selection method using 'regsubsets'
# function with method "forward"
fit_full_forward <- regsubsets(area_new ~ temp + X + Y + month  + DMC + DC +
                                 RH + ISI + day + FFMC + wind + rain ,
                       data = trainforest, nvmax = totpred, method = "forward")
```

```r
# Get the maximum adjusted R^2 just like we did in best subset selection
summary_full_forward <- summary(fit_full_forward)
summary_full_forward$adjr2
```

```
##  [1] 0.01797010 0.02613555 0.03257891 0.03561858 0.03739573 0.03871298
##  [7] 0.03958968 0.04034044 0.04080571 0.04139024 0.04145558 0.04143922
```

```r
max(summary_full_forward$adjr2)
```

```
## [1] 0.04145558
```

```r
max_adjr2_forw <- which.max(summary_full_forward$adjr2)

# Print the coefficients of all predictors selected in the forward stepwise
# selection model.
print(coef(fit_full_forward, max_adjr2_forw))
```

```
##   (Intercept)          temp             X      monthdec      monthfeb
## -0.1319491365  0.0326326959  0.0398729648  2.1581635453  0.4544597542
##      monthoct      monthsep           DMC            DC        daysat
##  0.4370604351  0.5662872361  0.0032794913 -0.0008325099  0.2116267407
##        daytue          wind
##  0.2180113689  0.0521637867
```

```r
#dropped predictors: y, month(J MAMJJA  N ), RH, ISI, FFMC,rain, day(M WTF S)

# Fit the new model with the dropped predictors
loocv_best_forw <- train(area_new ~ temp + X + monthdec + monthfeb + monthoct
                         + monthsep + DMC + DC + daysat + daytue +  wind,
                         data = ff_split, method = "lm", trControl = train_ctrl)

# Get the test MSE of the model
loocv_best_forw$results$RMSE^2
```

```
## [1] 1.889174
```

(d):

```r
# Fit the full model with backward stepwise selection method using 'regsubsets'
# function with method "backward"
fit_full_backward <- regsubsets(area_new ~ temp + X + Y + month  + DMC + DC + RH
                       + ISI + day + FFMC + wind + rain , data = trainforest,
                       nvmax = totpred, method = "backward")

# Get the maximum adjusted R^2 just like we did in the above two methods
summary_full_backward <- summary(fit_full_backward)
summary_full_backward$adjr2
```

```
##  [1] 0.01797010 0.02613555 0.03257891 0.03561858 0.03542850 0.03792023
##  [7] 0.03904498 0.03990050 0.04156734 0.04246959 0.04259422 0.04296864
```

```
max(summary_full_backward$adjr2)
```

```
## [1] 0.04296864
```

```
max_adjr2_back <- which.max(summary_full_backward$adjr2)

# Print the coefficients of all predictors selected in the backward stepwise
# selection model.
print(coef(fit_full_backward, max_adjr2_back))
```

```
##  (Intercept)         temp            X     monthdec     monthjun     monthmar
##  0.285138803  0.031908549  0.041435495  1.982703951 -0.483896795 -0.458031877
##     monthoct     monthsep          DMC           DC       daysat       daytue
##  0.520631256  0.649283726  0.003832441 -0.001595714  0.198367898  0.204355275
##         wind
##  0.053429096
```

```
#dropped predicctors: y, month(JF AM JA  N ), RH, ISI, FFMC,rain, day(M WTF S)

# Fit the new model with the dropped predictors
loocv_best_back <- train(area_new ~ temp + X + monthdec + monthjun + monthmar +
                         monthoct + monthsep + DMC + DC + daysat + daytue + wind,
                          data = ff_split, method = "lm", trControl = train_ctrl)

# Compute test MSE
loocv_best_back$results$RMSE^2
```

```
## [1] 1.888273
```

```
# Notice that the results of this method exactly match the best subset selection method.
```

(e):

```
# Set response and predictor matrices. The design matrix X includes 1's in the
# first column. We consider the predictor matrix without these 1's
y_resp_ff <- trainforest$area_new
x_pred_ff <- model.matrix(area_new ~ temp + X + Y + month  + DMC + DC + RH + ISI +
                day + FFMC + wind + rain, trainforest)[, -1]

# Fit ridge regression model for each lambda (penalty parameter) on the grid.
ridge_reg_ff_all <- glmnet(x_pred_ff, y_resp_ff, alpha = 0, lambda = grid)

# Perform LOOCV to find the penalty parameter lambda
loocv_ridge_ff <- cv.glmnet(x_pred_ff, y_resp_ff, alpha = 0, nfolds = nrow(trainforest),
```

```
                        lambda = grid, grouped = FALSE, type.measure = "mse")

# Penalty parameter = minimum value of lambda
penalty_par_ff <- loocv_ridge_ff$lambda.min
penalty_par_ff
```

```
## [1] 10.72267
```

```
# Calculate test MSE using built-in function "cvm" which stands for
# Cross Validation MSE which is very easy to calculate (it is included in the
# glmnet package)
test_err_new <- min(loocv_ridge_ff$cvm)
test_err_new
```

```
## [1] 1.929397
```

(f):

```
# Fit lasso model for each lambda (penalty parameter) on the grid.
lasso_ff_all <- glmnet(x_pred_ff, y_resp_ff, alpha = 1, lambda = grid)

# Perform LOOCV to find the best lambda from the grid
loocv_lasso_ff <- cv.glmnet(x_pred_ff, y_resp_ff, alpha = 1, nfolds = nrow(trainforest),
                        lambda = grid, grouped = FALSE, type.measure = "mse")

# Best lambda
penalty_par_ff_lasso <- loocv_lasso_ff$lambda.min
penalty_par_ff_lasso
```

```
## [1] 0.09326033
```

```
# Calculate test MSE using built-in function "cvm"
test_err_new_lasso <- min(loocv_lasso_ff$cvm)
test_err_new_lasso
```

```
## [1] 1.914636
```