# Homework #4: Boosting, Dimension reduction, Clustering

We will analyze data from the Investigation of Serial Studies to Predict Your Therapeutic Response with Imaging and moLecular Analysis (I-SPY TRIAL) breast cancer trial. This dataset contains patients with measurements derived from MRI scans. The outcome of interest is to predict the final diameter of the tumor as measured through the MRI, which is stored in the column `MRI_LD_Tfinal`. We have longitudinal measurements from each patient, collected at timepoints T0, T1, T2, and Tfinal. T0 is pre-treatment. T1 is early treatment. T2 is inter-regimen. Tfinal is the final measurement.

1. Load the iSPY1 dataset by running the following. Notice that there is a `fakeSite` column with values ranging from 1-8. We've added this column to simulate combining data from 8 different sites.

```
ispy_dat <- read.csv("ispy1doctored_site.csv")
ispy_dat$HR_HER2status <- as.factor(ispy_dat$HR_HER2status)
```

## Boosting

We will create and evaluate a gradient boosted model that predicts `MRI_LD_Tfinal` using all the predictors measured before Tfinal.

2. (1 point) Hold out sites 7 and 8 for testing. Store the training data in `datTrain` and the test data in `datTest`.

```
datTest <- ispy_dat[ispy_dat["fakeSite"]==7|ispy_dat["fakeSite"]==8,]
datTrain <- ispy_dat[ispy_dat["fakeSite"]!=7&ispy_dat["fakeSite"]!=8,]
```

3. (2 points) In this homework, we will manually implement site-wise 3-fold cross-validation (i.e. two sites per fold) rather than using the `caret` package. Create vector with True and False values to split the rows in `datTrain` into 3 folds, where sites 1 and 2 are in the first fold, sites 3 and 4 are in the second fold, and sites 5 and 6 are in the third fold. Create a list with these three True/False vectors. It may be helpful to reference this list later in this homework.

```
cv1 <- datTrain$fakeSite %in% c(1,2)
cv2 <- datTrain$fakeSite %in% c(3,4)
cv3 <- datTrain$fakeSite %in% c(5,6)

cv_list <- list(as.numeric(cv1),as.numeric(cv2),as.numeric(cv3))
```

4. Load the `gbm` package.

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.2
```

```
## Loaded gbm 2.1.9
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://g
ithub.com/gbm-developers/gbm3
```

5. (3 points) We will select the hyperparameters for a gradient boosted model using site-wise three-fold CV. Create a function named `fit_fold` that takes as input the fold number `fold_idx` , number of trees, and interaction depth. The function will fit a gradient boosted model using `gbm` using the aforementioned predictors. Also exclude `fakeSite` . Train on all the folds except for the `fold_idx` -th one. The function should output the mean squared error of the fitted model on the held out fold. Use the folds you made in question 3. Fix the shrinkage hyperparameter in `gbm` as 0.01.

```
fit_fold <- function(fold_idx, n_trees, interaction_depth){
  X <- datTrain[,!(names(datTrain) %in% c('fakeSite', 'MRI_LD_Tfinal'))]
  y <- datTrain$MRI_LD_Tfinal

  train_row <- !cv_list[[fold_idx]]
  test_row <- cv_list[[fold_idx]]

  train_data <- X[train_row,]
  test_data <- X[test_row,]

  train_outcome <- y[train_row]
  test_outcome <- y[test_row]

  gbm_mdl <- gbm(
  formula = train_outcome ~ .,
  data = train_data,
  distribution = "gaussian",
  n.trees = n_trees,
  interaction.depth = interaction_depth,
  shrinkage = 0.01,
  n.minobsinnode = 10,
  verbose = FALSE
  )

  pred <- predict(gbm_mdl, newdata= test_data, n.trees = n_trees)

  mse <- mean((test_outcome - pred)^2)
  return(mse)
}
```

6. (4 points) Using the function you made in question 5, tune the number of trees and interaction depth using site-wise three-fold CV. Search over the values `n.trees=100, 200, 400, 800, 1600` and `interaction.depth=1, 2` . Which hyperparameter values minimize the cross-validated mean squared error?

```
# Define the sets of potential hyperparameters
set.seed(7)
n_trees_values <- c(100, 200, 400, 800, 1600)
interaction_depth_values <- c(1, 2)

# Initialize a variable to store the results
best_mse <- Inf
results <- data.frame(n_trees = integer(), interaction_depth = integer(), avg_mse = numeric())
best_params <- list(n_trees = NA, interaction_depth = NA)

# Evaluate all combinations of hyperparameters
for (n_trees in n_trees_values) {
  for (interaction_depth in interaction_depth_values) {
    # Initialize a vector to store MSEs
    mse_values <- numeric(length(cv_list))

    # Perform 3-fold CV
    for (fold_idx in seq_along(cv_list)) {
      mse_values[fold_idx] <- fit_fold(fold_idx, n_trees, interaction_depth)
    }

    # Calculate the average MSE
    avg_mse <- mean(mse_values)
    results <- rbind(results, data.frame(n_trees = n_trees, interaction_depth = interaction_dept
h, avg_mse = avg_mse))

    # Update best hyperparameters if the current combination has a lower average MSE
    if (avg_mse < best_mse) {
      best_mse <- avg_mse
      best_params <- list(n_trees = n_trees, interaction_depth = interaction_depth)
    }
  }
}

# Print the best hyperparameters and the lowest MSE
print(paste("Best n_trees:", best_params$n_trees))
```

```
## [1] "Best n_trees: 1600"
```

```
print(paste("Best interaction_depth:", best_params$interaction_depth))
```

```
## [1] "Best interaction_depth: 2"
```

```
print(paste("Lowest cross-validated MSE:", best_mse))
```

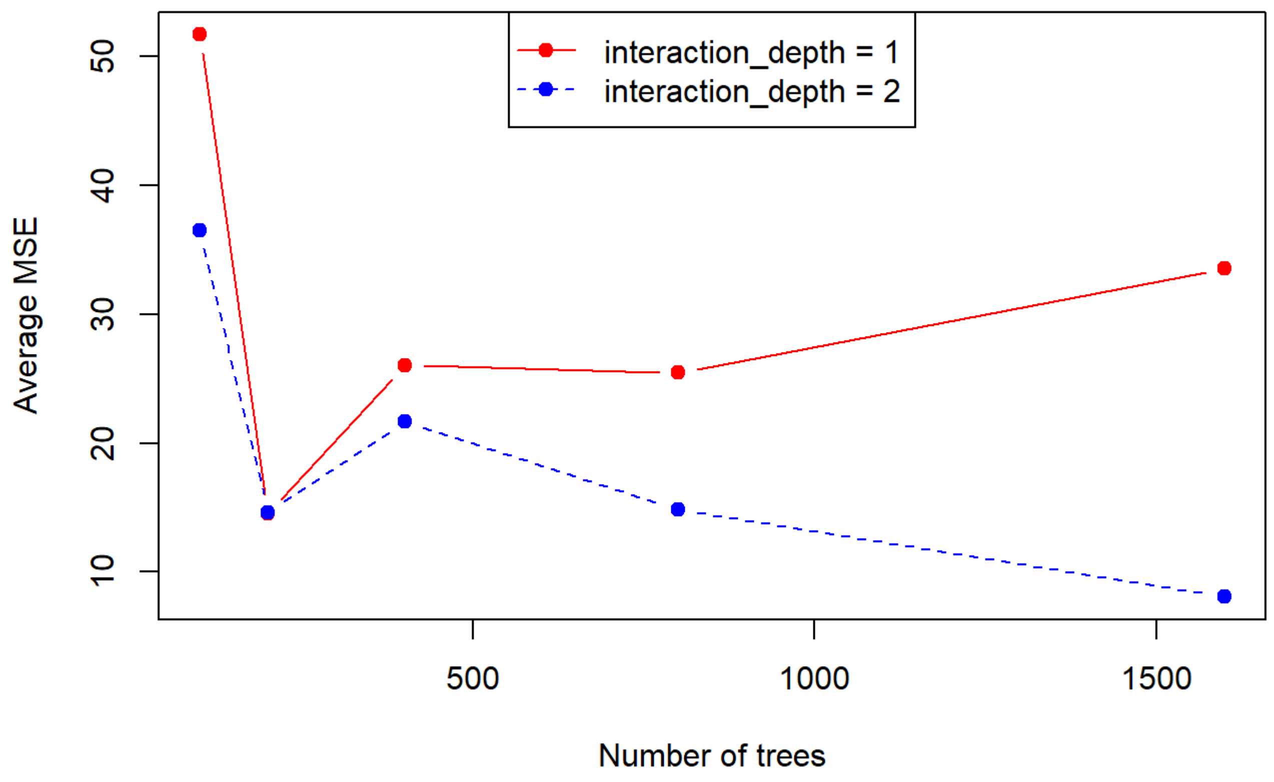
```
## [1] "Lowest cross-validated MSE: 8.07412546170622"
```

7. (2 points) Plot the cross-validated error with respect to `n.trees` for the interaction depth that attained the lowest CV error.

```
plot(results$n_trees[results$interaction_depth==1], results$avg_mse[results$interaction_depth==
1],
     type = "b", col="red",xlab="Number of trees", ylab = "Average MSE", xlim = c(min(results$n_
trees),max(results$n_trees)), ylim = c(min(results$avg_mse), max(results$avg_mse)), pch=19, lty=
1)

points(results$n_trees[results$interaction_depth == 2], results$avg_mse[results$interaction_dept
h == 2],
       type = "b", col = "blue", pch = 19, lty = 2)

legend("top", legend = c("interaction_depth = 1", "interaction_depth = 2"),
       col = c("red", "blue"), pch = 19, lty = 1:2)
```



8. (1 point) Refit the gradient boosted model on all the training data ( `datTrain` ) using the hyperparameters that minimized the CV error.

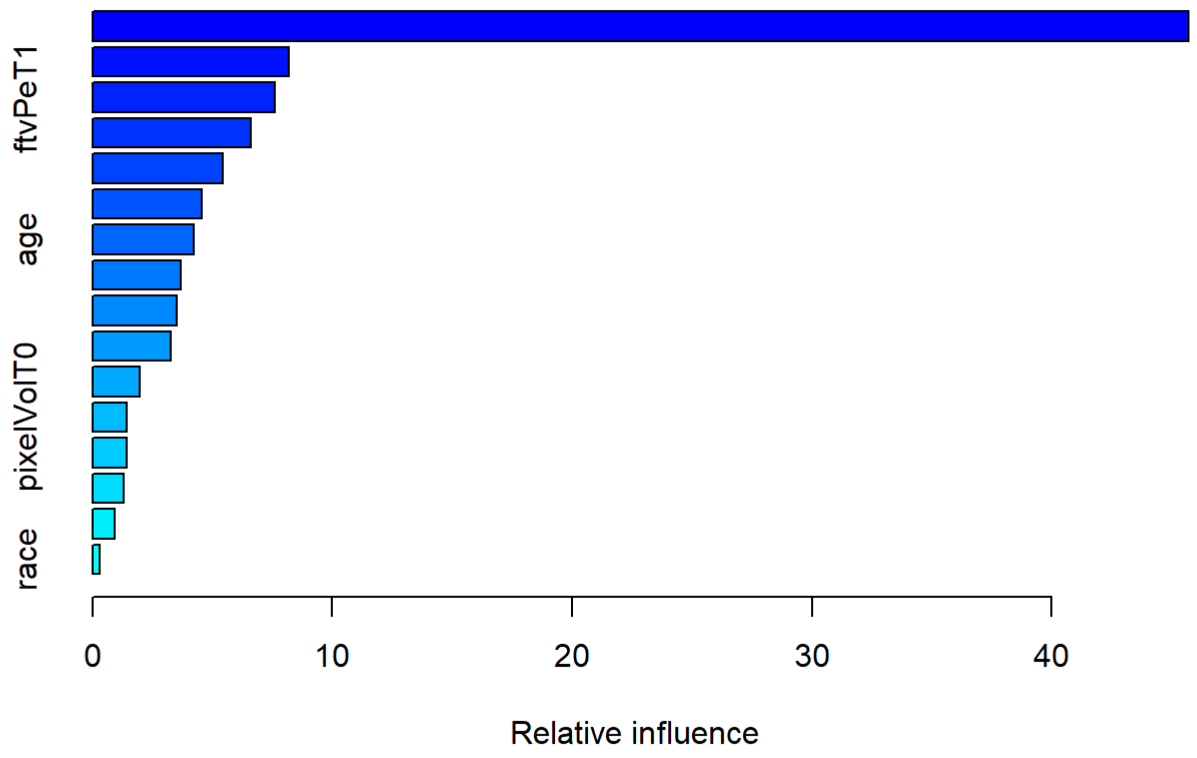
```r
# Check the best parameters and set them
opt_n_trees <- 1600
opt_interaction_depth <- 2

# Set the target and features
X <- datTrain[, !(names(datTrain) %in% c('fakeSite', 'MRI_LD_Tfinal'))]
y <- datTrain$MRI_LD_Tfinal

# Fit the model on the all train data
best_gbm_model <- gbm(
  formula = y ~ .,
  data = X,
  distribution = "gaussian",
  n.trees = opt_n_trees,
  interaction.depth = opt_interaction_depth,
  shrinkage = 0.01,
  n.minobsinnode = 10,
  verbose = TRUE
)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        806.4364             nan     0.0100    5.1388
##      2        796.6885             nan     0.0100    8.5387
##      3        786.8402             nan     0.0100    9.6934
##      4        777.5080             nan     0.0100    9.0974
##      5        770.1237             nan     0.0100    5.3455
##      6        760.8756             nan     0.0100    8.0408
##      7        753.1938             nan     0.0100    7.4499
##      8        743.4184             nan     0.0100    7.2395
##      9        735.5992             nan     0.0100    6.3347
##     10        729.5958             nan     0.0100    5.8279
##     20        653.9902             nan     0.0100    5.1653
##     40        542.4794             nan     0.0100    2.4892
##     60        453.6630             nan     0.0100    2.9926
##     80        391.8812             nan     0.0100    2.0529
##    100        347.6759             nan     0.0100    0.7893
##    120        311.9355             nan     0.0100    1.1754
##    140        290.0601             nan     0.0100   -0.1587
##    160        270.7253             nan     0.0100    1.0622
##    180        253.2096             nan     0.0100    0.4521
##    200        240.3764             nan     0.0100   -0.0919
##    220        228.5378             nan     0.0100   -0.9384
##    240        218.7614             nan     0.0100    0.0132
##    260        211.2403             nan     0.0100    0.1441
##    280        204.3398             nan     0.0100   -0.5538
##    300        199.8679             nan     0.0100   -0.4728
##    320        194.3170             nan     0.0100   -0.4069
##    340        188.7469             nan     0.0100   -0.3721
##    360        184.7131             nan     0.0100   -0.1238
##    380        180.7707             nan     0.0100   -0.2276
##    400        175.8266             nan     0.0100   -0.1436
##    420        172.1787             nan     0.0100   -0.2954
##    440        168.2319             nan     0.0100    0.1000
##    460        164.9999             nan     0.0100   -0.4498
##    480        161.7935             nan     0.0100   -0.1838
##    500        159.2395             nan     0.0100   -0.1074
##    520        155.9049             nan     0.0100   -0.1099
##    540        153.1775             nan     0.0100   -0.4268
##    560        150.2953             nan     0.0100   -0.2861
##    580        147.8680             nan     0.0100   -0.4435
##    600        144.6684             nan     0.0100   -0.1660
##    620        142.3697             nan     0.0100   -0.3934
##    640        140.2654             nan     0.0100   -0.3305
##    660        138.2703             nan     0.0100   -0.0986
##    680        136.3267             nan     0.0100   -0.2439
##    700        134.2245             nan     0.0100   -0.3809
##    720        131.6922             nan     0.0100   -0.3878
##    740        129.6761             nan     0.0100   -0.2734
##    760        127.8547             nan     0.0100   -0.2613
##    780        125.9649             nan     0.0100   -0.0802
##    800        123.7761             nan     0.0100   -0.2306
##    820        121.7287             nan     0.0100   -0.0667
##     840       119.9784             nan     0.0100   -0.2200
##     860       118.2015             nan     0.0100   -0.1626
##     880       116.5736             nan     0.0100   -0.2371
##     900       114.5591             nan     0.0100   -0.1656
##     920       112.9851             nan     0.0100   -0.1196
##     940       111.2883             nan     0.0100   -0.2304
##     960       109.5806             nan     0.0100   -0.1310
##     980       107.8985             nan     0.0100   -0.2377
##    1000       106.5801             nan     0.0100   -0.1231
##    1020       104.7583             nan     0.0100   -0.3479
##    1040       103.0241             nan     0.0100   -0.0623
##    1060       101.7254             nan     0.0100   -0.1908
##    1080       100.0340             nan     0.0100   -0.1160
##    1100        98.7143             nan     0.0100   -0.1534
##    1120        97.4235             nan     0.0100   -0.2574
##    1140        96.5062             nan     0.0100   -0.2059
##    1160        95.1669             nan     0.0100   -0.2085
##    1180        93.9834             nan     0.0100   -0.1881
##    1200        92.9486             nan     0.0100   -0.2596
##    1220        91.8643             nan     0.0100   -0.2631
##    1240        90.4865             nan     0.0100   -0.1568
##    1260        89.3648             nan     0.0100   -0.2702
##    1280        88.1566             nan     0.0100   -0.1274
##    1300        86.9108             nan     0.0100   -0.1341
##    1320        85.8337             nan     0.0100   -0.0220
##    1340        84.8670             nan     0.0100   -0.2727
##    1360        83.6808             nan     0.0100   -0.0818
##    1380        82.5478             nan     0.0100   -0.1121
##    1400        81.6271             nan     0.0100   -0.0705
##    1420        79.9900             nan     0.0100   -0.2341
##    1440        79.1057             nan     0.0100   -0.1325
##    1460        78.3485             nan     0.0100   -0.3535
##    1480        77.5392             nan     0.0100   -0.1955
##    1500        76.4349             nan     0.0100   -0.1368
##    1520        75.5284             nan     0.0100   -0.1067
##    1540        74.5213             nan     0.0100   -0.2209
##    1560        73.4533             nan     0.0100   -0.1799
##    1580        72.6055             nan     0.0100   -0.1235
##    1600        71.4229             nan     0.0100   -0.2966
```

```
summary(best_gbm_model)
```



```
##                                       var    rel.inf
## MRI_LD_T2                       MRI_LD_T2 45.7025693
## ftvPeT2                           ftvPeT2  8.1709222
## ftvPeT1                           ftvPeT1  7.5837720
## ftvPeTfinal                   ftvPeTfinal  6.5793962
## MRI_LD_T1                       MRI_LD_T1  5.4166433
## MRI_LD_T0                       MRI_LD_T0  4.5408821
## age                                   age  4.2300717
## ftvPeT0                           ftvPeT0  3.6673401
## ftvPePctChgT0_T1         ftvPePctChgT0_T1  3.5291705
## HR_HER2status               HR_HER2status  3.2675751
## pixelVolT1                     pixelVolT1  1.9706736
## pixelVolT0                     pixelVolT0  1.4228526
## pixelVolTfinal             pixelVolTfinal  1.4040006
## pixelVolT2                     pixelVolT2  1.3099910
## pixelVolPctChgT0_T1 pixelVolPctChgT0_T1  0.9202991
## race                                 race  0.2838404
```

9. (1 point) Evaluate the MSE of the fitted model on the test data. How much of the variance have we explained using the GBM?

```
test_predictions <- predict(best_gbm_model, newdata = datTest[, !(names(datTest) %in% c('fakeSit
e', 'MRI_LD_Tfinal'))], n.trees = opt_n_trees)

# Observed data in the test dataset
test_actual <- datTest$MRI_LD_Tfinal

# Calculate MSE and R^2 score
mse_test <- mean((test_actual - test_predictions)^2)
print(paste("Test MSE:", mse_test))
```

```
## [1] "Test MSE: 383.480561993256"
```

```
ss_total <- sum((test_actual - mean(test_actual))^2)
ss_res <- sum((test_actual - test_predictions)^2)
r_squared <- 1 - (ss_res / ss_total)
print(paste("R^2 Score:", r_squared))
```

```
## [1] "R^2 Score: 0.534967769074254"
```

# Kmeans

Let's perform K-means on the iSPY data.

10. (1 point) Create a new data frame named `ispy_subdat` that only contains the continuous variables measured before Tfinal.

```
drop_col <- c('age', 'race', 'HR_HER2status', 'fakeSite')
ispy_subdat <- ispy_dat[, !(names(ispy_dat) %in% drop_col)]
```

11. (1 point) Before we run K-means, center and scale all the variables so that they have mean 0 and variance 1 (hint: use the `scale` command).

```
ispy_subdat_scaled <- scale(ispy_subdat)
```

12. (3 points) Tune the number of clusters used in K-means. To do this, use the function `fviz_nbclust` from the `factoextra` library. The function `fviz_nbclust` determines and visualizes the optimal number of clusters using different methods (within cluster sums of squares, average silhouette and gap statistics). Plot the average silhouette with respect to the number of clusters by passing in the argument `method="silhouette"`. What is the optimal number of clusters according to the silhouette statistic?

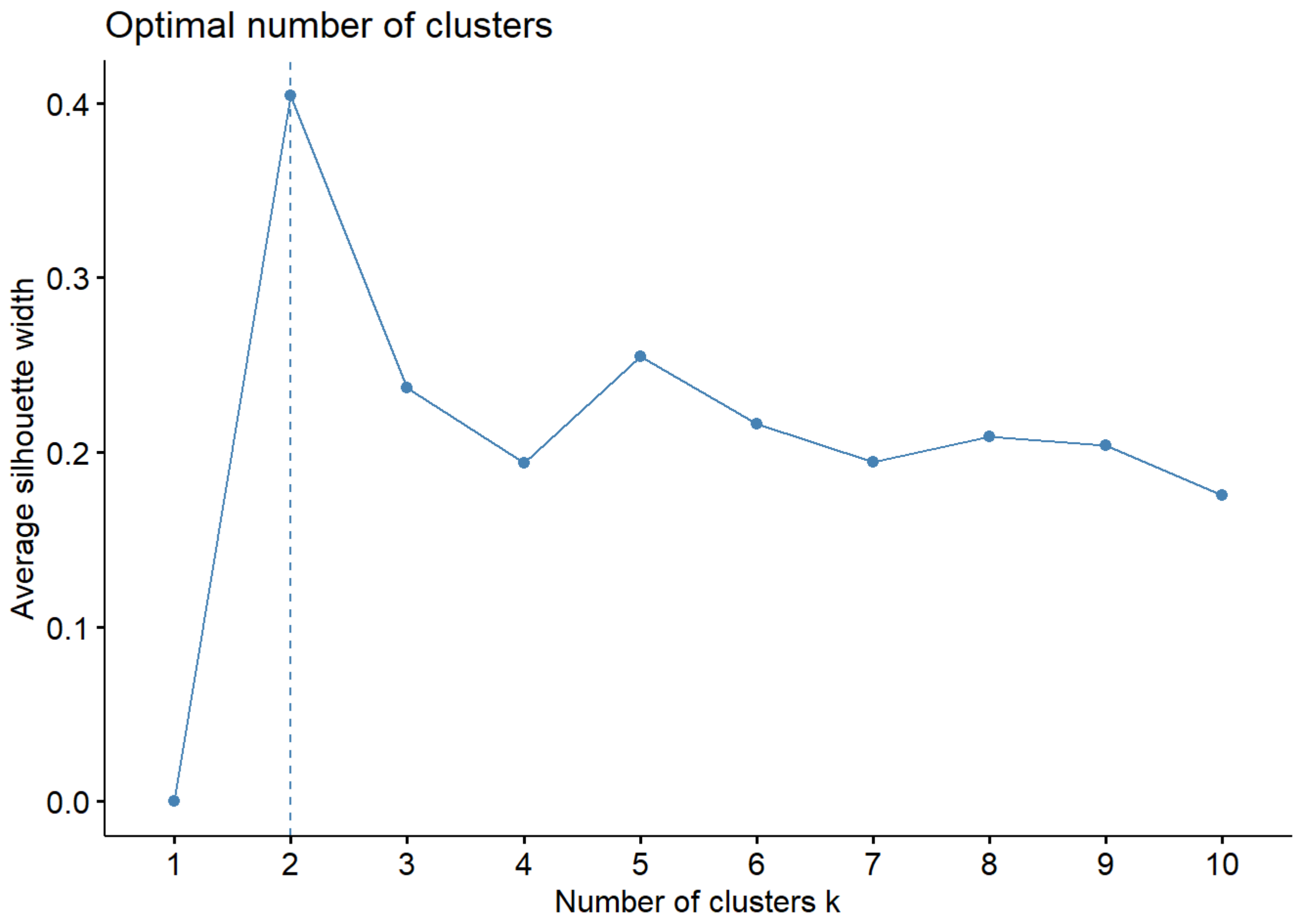
```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.3.2
```

```
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_nbclust(ispy_subdat, kmeans, method = "silhouette")
```

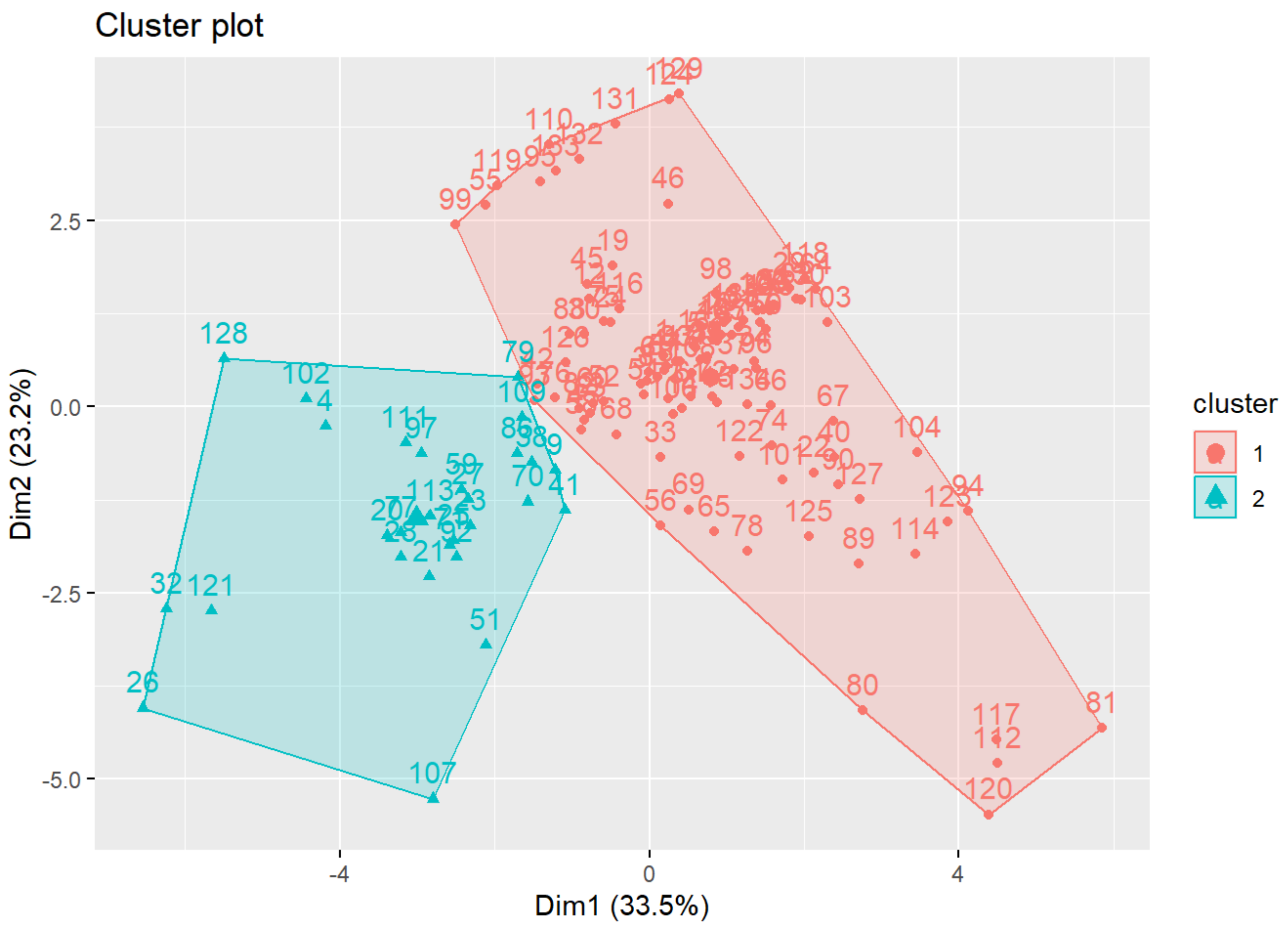


#

The optimal number of clusters is two.

13. (3 points) Refit k-means using the optimal number of clusters with 15 random initializations. Use the function `fviz_cluster()` to plot the clusters from K-means. Observations are represented by points in the plot, using principal components if $p > 2$. An ellipse is drawn around each cluster.

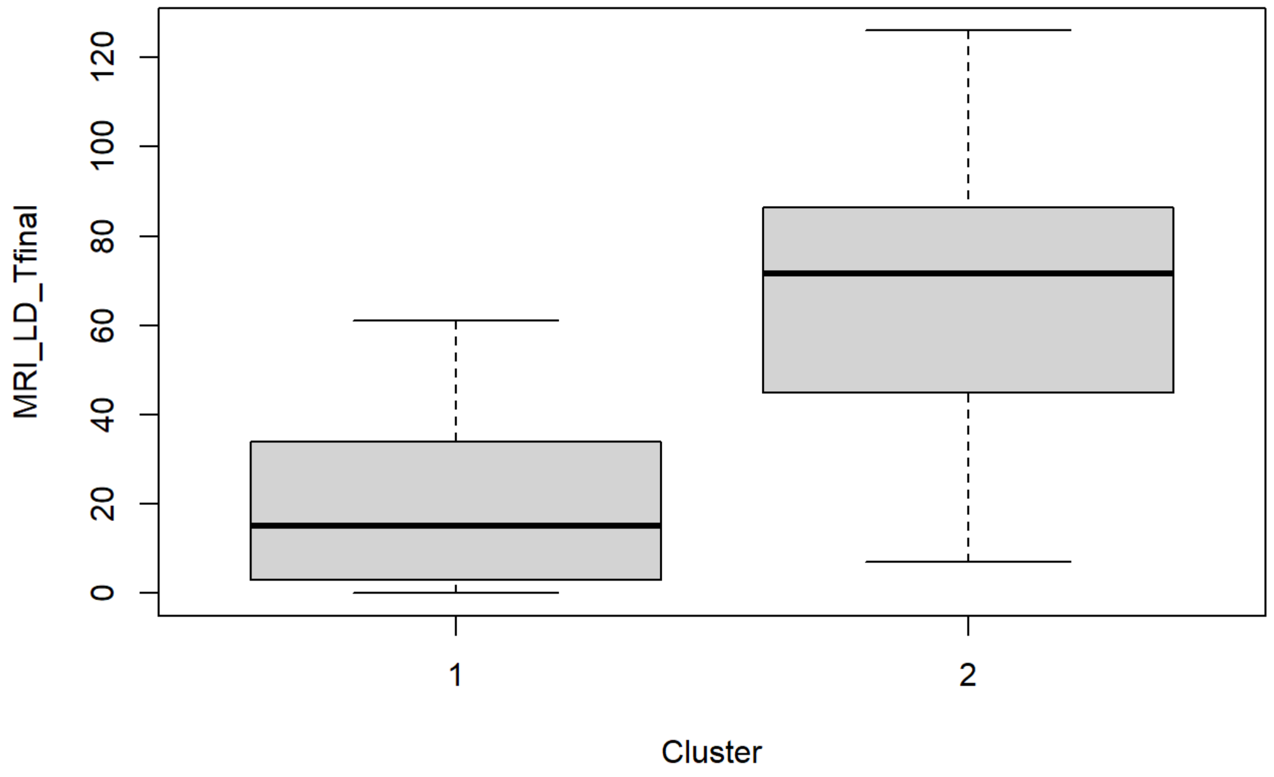
```
n_cluster <- 2
set.seed(42)
km <- kmeans(ispy_subdat, centers = n_cluster, nstart = 15)
fviz_cluster(km, data = ispy_subdat)
```



13. (1 point) Plot the distribution of `MRI_LD_Tfinal` for each cluster created by K-means. How do they differ across the clusters?

```
ispy_subdat$cluster <- km$cluster

boxplot(MRI_LD_Tfinal ~ cluster, data = ispy_subdat, xlab = "Cluster", ylab = "MRI_LD_Tfinal")
```



The cluster 1 is the subgroup of smaller MRI_LD_Tfinal and the cluster 2 is the bigger MRI_LD_Tfinal group.