Analytics Modeling HW2

Fall 2024

```
cc <- read.table("credit_card_data.txt",  stringsAsFactors = FALSE, header = FALSE)
iris <- read.table("iris.txt")
```

Question 3.1.a

Using the same dataset as the previous homework, use the kknn function to find a good classifier using cross validation.

The KKNN package has a couple built-in options for cross validation, including `cv.kknn` with the `kcv` parameter, which is what I ultimately went with for this problem.

Like with the previous homework, I used max_k as the square root of the number of rows (approximately 26) as the number of k values to test. Since the professor recommended 10 as a commonly used number of folds for k-Fold Cross Validation, I went with that for my `kcv` parameter.

While all of the models performed moderately well in terms of accuracy, model number 15 reached maximum accuracy of 85.77982%.

```
set.seed(42)                      # set seed for reproducibility of results

max_k <- ceiling(sqrt(nrow(cc)))  # calculate the max k value as the square root of the number of data points, rounded up
acc_cvknn <- vector()             # empty vector to store accuracy for each k

for (k in 1:max_k) {
  cv_knn <- cv.kknn(V11~.         # response variable
                  , cc
                  , kcv=10        # 10-folds
                  , k=k
                  , scale=TRUE)

  preds <- as.integer(cv_knn[[1]][,2] + 0.5) # kknn preds are continuous, so round to get preds to 0 or 1
  acc <- sum(preds == cc[,11]) / nrow(cc)    # calculate accuracy for each model
  acc_cvknn <- c(acc_cvknn, acc)             # store each model's accuracy in this vector

}
cat("Max accuracy achieved: ", max(acc_cvknn), "\n")
```

```
## Max accuracy achieved:  0.8577982
```

```
cat("Best performing k value: ", which.max(acc_cvknn))
```

```
## Best performing k value:  15
```

Question 3.1.b

Using the same dataset as the previous homework, use the ksvm or kknn function to find a good classifier by splitting the data into training, validation, and test data sets.

To remain consistent with the previous problem, I am using kknn here. We want the training set to be the largest, so the model has as much data as possible to learn from while leaving enough data to test and validate on. I decided to go with a 70/15/15 train/validate/test split. Since we're not just making a single model (we're trying out different values of k for K-Nearest Neighbors), we need a validation set to choose the correct model. Then the testing set can be used to evaluate the chosen model overall.

In this case, it looks like out of the 26 k values tried, k = 9 proved to have the highest accuracy during the training and validating portion of the process. Once k=9 was selected as the best option for modeling, I trained a new model with that parameter and used the testing set to do a final evaluation of the model. The final model's accuracy came out around 84%.

```
set.seed(42)                      # set seed for reproducibility of results
max_k <- ceiling(sqrt(nrow(cc)))  # calculate the max k value as the square root of the number of data points, rounded up
acc_tvt <- vector()               # empty vector to store accuracy for each k

# 70 / 15 / 15 split
train_size <- floor(0.70*nrow(cc)) # 70% of dataset for training = 457 rows

train_idx <- sample(x=nrow(cc)     # randomly grab the indices of 457 rows from dataset without replacement.
                    , size=train_size
                    , replace=FALSE)

rest <- cc[-train_idx,]              # grab the unused rows of data

validate_idx <- sample(x=nrow(rest)   # grab half of the unused indices to use for validation without replacement
                        , size=floor(nrow(rest)/2)
                        , replace=FALSE)

train_set <- cc[train_idx,]          # grab the rows associated with the training indices
validate_set <- rest[validate_idx,]  # grab the rows associated with the validation indices
test_set <- rest[-validate_idx,]     # grab the remaining rows for testing

# best k selection
for (k in 1:max_k) {
  knn_tvt <- kknn(formula=V11~.
                  , train=train_set      # train using the training set
                  , test=validate_set    # evaluate model with validation set
                  , kernel='optimal'
                  , k=k
                  , scale=TRUE)

  preds <- as.integer(fitted(knn_tvt)+0.5)
  acc_tvt[k] <- sum(preds == validate_set[,11]) / nrow(validate_set)
}
cat("Max accuracy achieved: ", max(acc_tvt), "\n")
```

```
## Max accuracy achieved:  0.8877551
```

```
cat("Best performing k value: ", which.max(acc_tvt), "\n")
```

```
## Best performing k value:  9
```

```
# model evaluation
knn <- kknn(formula=V11~.
            , train=train_set         # train using training set
            , test=test_set           # final evaluation with test set
            , kernel='optimal'
            , k=which.max(acc_tvt)    # choose the k that performed best during validation stage
            , scale=TRUE)

preds <- as.integer(fitted(knn)+0.5)
accuracy <- sum(preds == test_set[,11]) / nrow(test_set)
cat("Test set accuracy evaluation: ", accuracy, "\n")
```

```
## Test set accuracy evaluation:  0.8383838
```

Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you may use.

I work for a sports & casino gambling company as a data scientist. One problem I have used a clustering model for at my job is to analyze the days of the week in terms of basketball betting. For people who like to bet on NBA or NCAAB games, what knowledge can we gain by analyzing different basketball games and where they fall during the basketball season? Do their betting behaviors change depending based on the schedule of games?

The features we used for this simple clustering model were:

- the day of the week on which the games occur (Monday-Sunday)
- the time of day during which the games occur (Mornings, Afternoons, Evenings)

We tried a couple variations on the model, including using raw data, data where the weekdays and weekends were slightly weighted, and data where the weekdays and weekends were heavily weighted. The most effective number of clusters was k = 7 (for obvious reasons, since there are 7 days in a week) and k = 11. Certain games would be clustered together based on their similarities of betting behaviors due to the time the games occurred, even if the games occurred on different days.

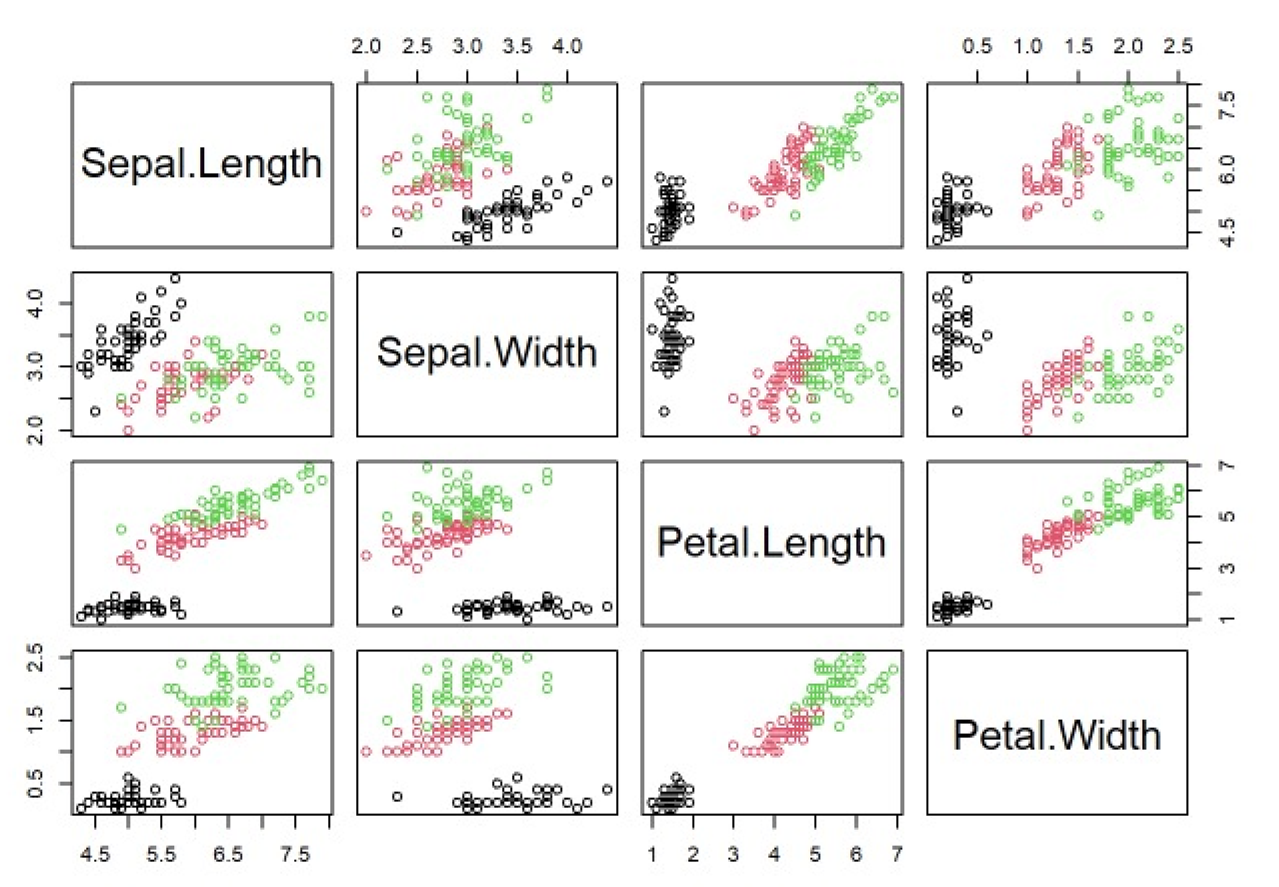
Question 4.2

Use the R function kmeans to cluster the points of the iris dataset as well as possible. Report the best combination of predictors, your suggested k value, and how well your best clustering type predicts flower type.

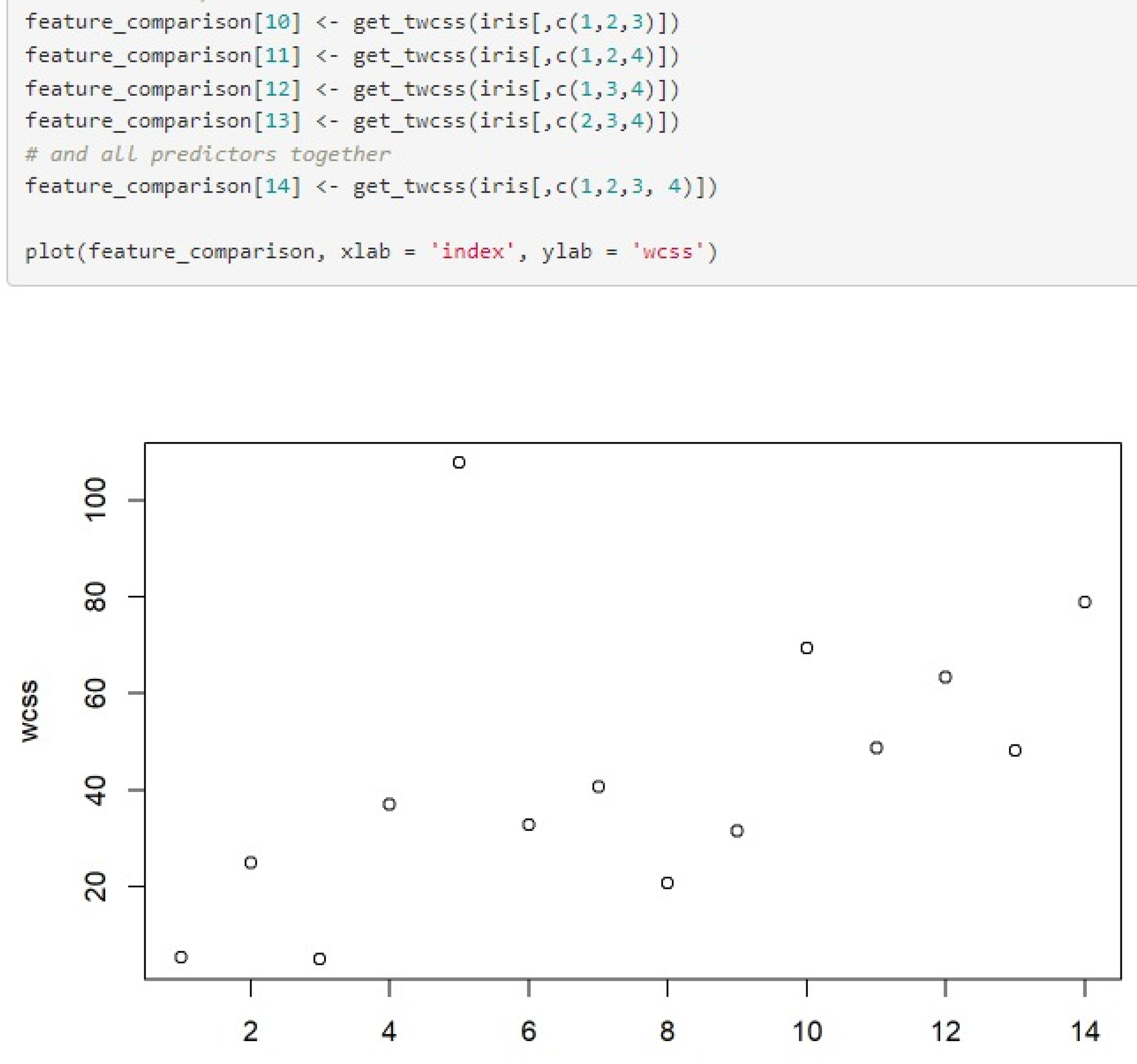
This dataset has four attributes (sepal length, sepal width, petal length, and petal width) that can predict the species (setosa, versicolor, virginica) of the flower. Using the Elbow method, it looks like a good value for k is 3 - which isn't surprising since there are three kinds of species in this dataset. There may be better performing numbers of clusters, but we want to find a good trade-off between model performance and computational intensity. We can see from the Elbow Plot that the Total Within-Cluster Sum of Squares metric decreases significantly until about k=3 and from that point on, there is smaller and smaller gain to be had from further increasing k.

After trying different kmeans models using various combinations of features, there seems to be quite a variety in metrics like the Total Within Cluster Sum of Squares. As more features are introduced, this error metric tends to grow (meaning there is more variation in them). There must be a point where the trade off between knowledge gained from the addition of features is worth the increase in error.

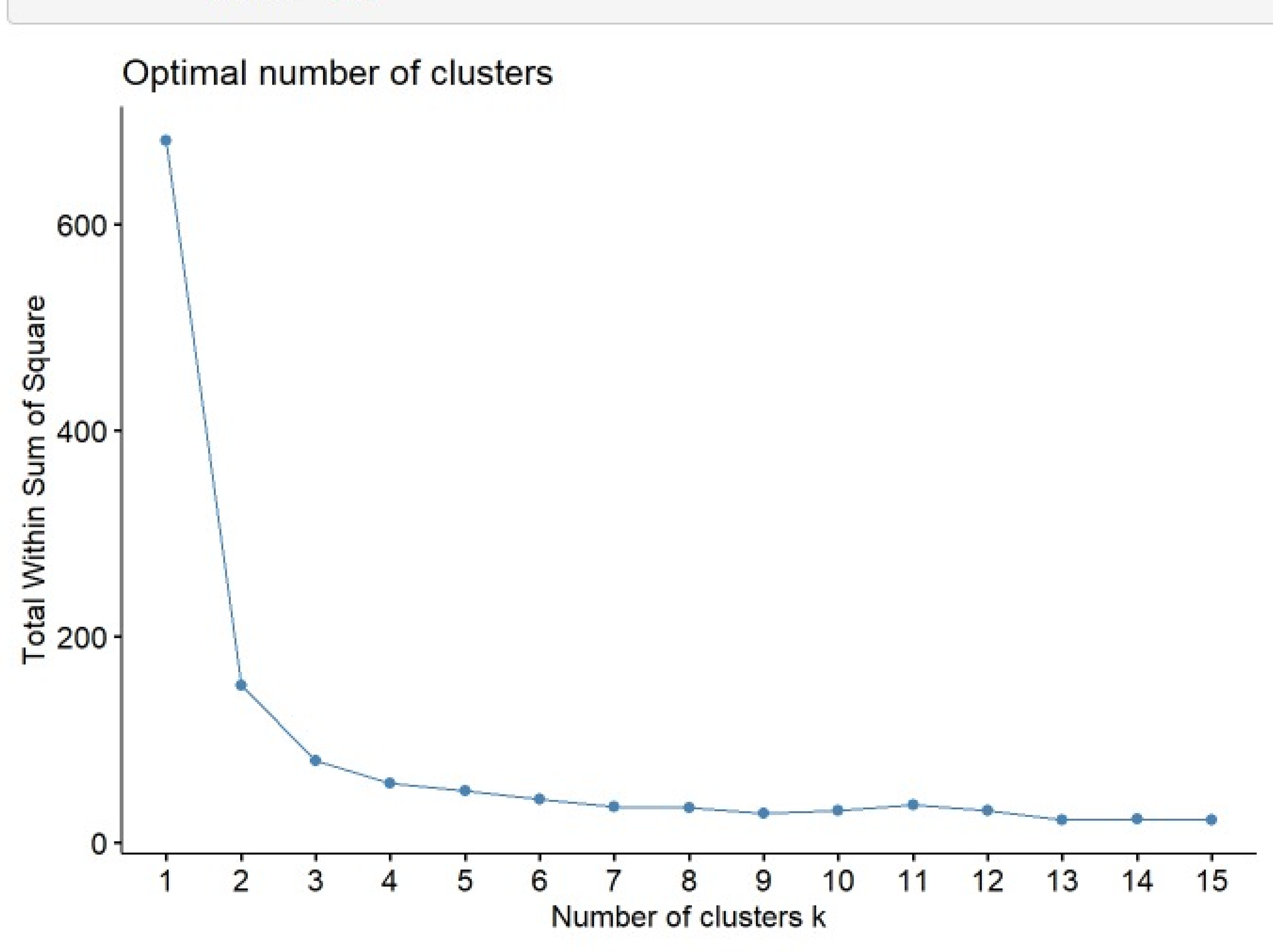
```
set.seed(42)               # set seed for reproducibility of results

iris$Species = as.factor(iris$Species) # change Species column from character to factor datatype

unlabeled_df <- iris[,1:4]   # get unlabeled data

# plot the data
pairs(iris[,1:4]
      , col = iris$Species)
```



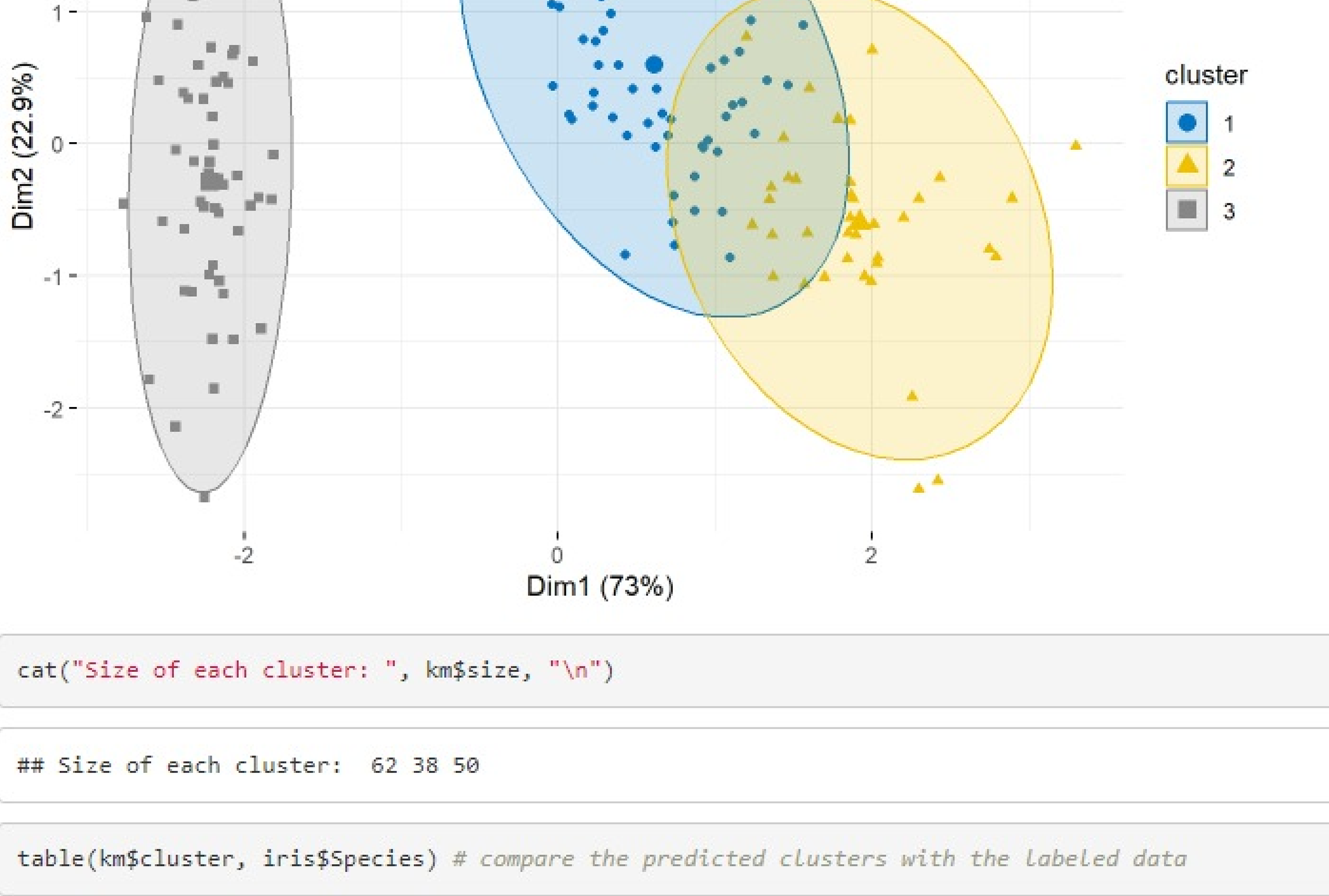

```
# how do different predictors perform?
get_twcss <- function(z) {
              kmeans(x=z
             , centers=3
             , iter.max=20)$tot.withinss # get the total within cluster sum of squares
}

feature_comparison <- vector()
# let's compare all the predictors by themselves
feature_comparison[0] <- get_twcss(iris[,1])
feature_comparison[1] <- get_twcss(iris[,2])
feature_comparison[2] <- get_twcss(iris[,3])
feature_comparison[3] <- get_twcss(iris[,4])
# and all the pairs of predictors
feature_comparison[4] <- get_twcss(iris[,c(1,2)])
feature_comparison[5] <- get_twcss(iris[,c(1,3)])
feature_comparison[6] <- get_twcss(iris[,c(1,4)])
feature_comparison[7] <- get_twcss(iris[,c(2,3)])
feature_comparison[8] <- get_twcss(iris[,c(2,4)])
feature_comparison[9] <- get_twcss(iris[,c(3,4)])
# and all 3-tuples
feature_comparison[10] <- get_twcss(iris[,c(1,2,3)])
feature_comparison[11] <- get_twcss(iris[,c(1,2,4)])
feature_comparison[12] <- get_twcss(iris[,c(1,3,4)])
feature_comparison[13] <- get_twcss(iris[,c(2,3,4)])
# and all predictors together
feature_comparison[14] <- get_twcss(iris[,c(1,2,3, 4)])

plot(feature_comparison, xlab = 'index', ylab = 'wcss')
```




```
# identify a good k value
fviz_nbclust(x = unlabeled_df
             , FUNcluster = kmeans
             , method = "wss"
             , k.max = 15)
```




```
# training a clustering model
km <- kmeans(x=unlabeled_df       # features
             , centers=3          # number of clusters
             , iter.max=20        # max number of iterations
             , nstart= 20         # number of random starting partitions
       )

# plot the clusters
fviz_cluster(object=km
             , data=unlabeled_df
             , ellipse.type = "norm"
             , palette = "jco"
             , ggtheme = theme_minimal()
             , geom = "point"
             )
```




```
cat("Size of each cluster: ", km$size, "\n")
```

```
## Size of each cluster:  62 38 50
```

```
table(km$cluster, iris$Species) # compare the predicted clusters with the labeled data
```

```
##
##     setosa versicolor virginica
##   1      0         48        14
##   2      0          2        36
##   3     50          0         0
```