Week2_HW_903431138

January 22, 2019

1 Question 3.1a

In this question I am using cross-validation with the k-neareset-neighbors model train.kknn() to identify a good classifier for the given credit card data.

In [1]:
```
# Reset environment
rm(list = ls())

# Load packages
suppressWarnings(library(dplyr))
suppressWarnings(library(kernlab))
suppressWarnings(library(kknn))
```

```
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
```

In this first step, I loaded the credit card data into a dataframe and set the seed to 1 to make sure my model output stays consistent every time I run the code. I tested 4 different kernels with train.knn() which is a cross-validated method of classifying and the output from the model suggesed that the triangular kernel is the most accurate. I executed the next few steps in my code using the triangular kernel as a result.

In [2]:
```
# Read data in
data <- tbl_df(read.table("credit_card_data-headers.txt", header = TRUE))

# Fixes randomness in model so you get same output each time
set.seed(1)
```

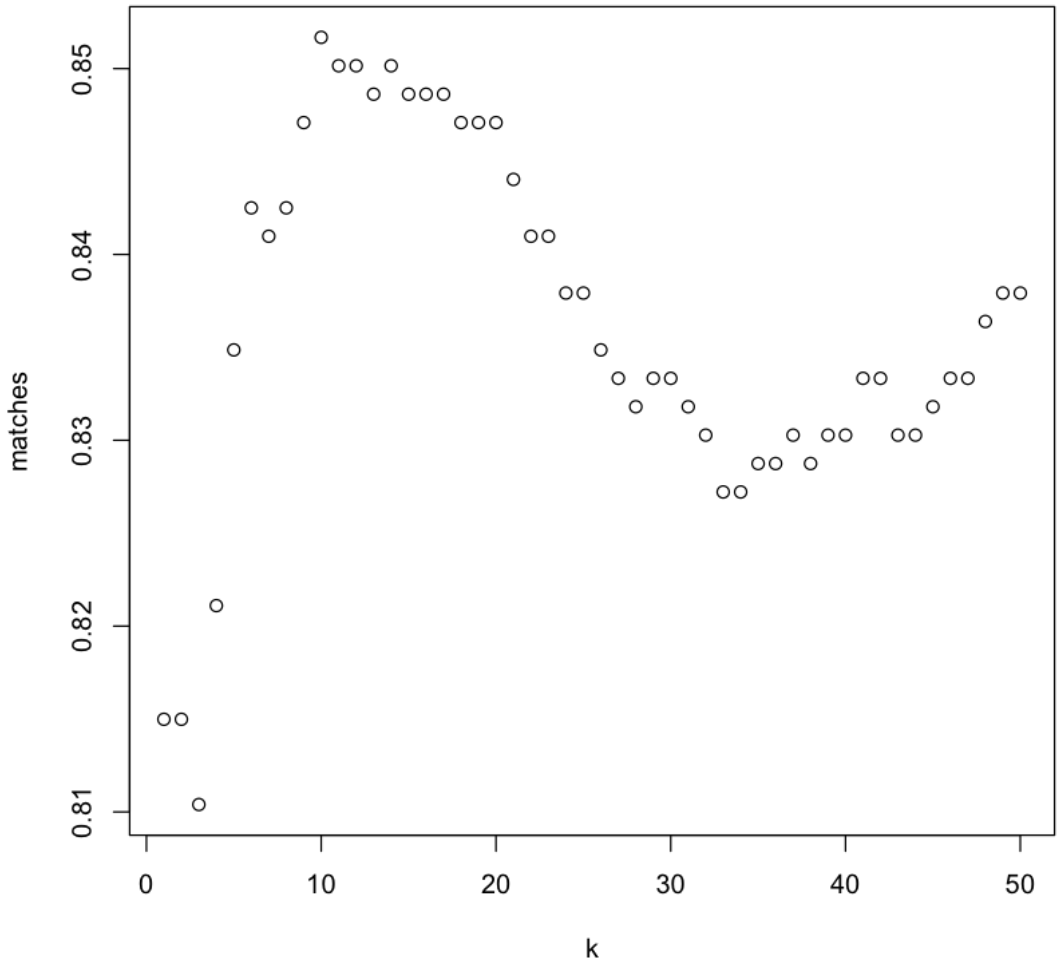
```
        # Test different kernels to get a recommendation
        model <- train.kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15,
                            data,
                            kmax = 50,
                            kernel = c("triangular", "rectangular", "epanechnikov", "optimal")
                            scale = TRUE)

        model
```

```
Call:
train.kknn(formula = R1 ~ A1 + A2 + A3 + A8 + A9 + A10 + A11 +     A12 + A14 + A15, data = data

Type of response variable: continuous
minimal mean absolute error: 0.1850153
Minimal mean squared error: 0.1074101
Best kernel: triangular
Best k: 49
```

After settling on using the "triangular" kernel, I ran train.kknn with a kmax of 50 which output 50 potential models. I then looped through each model to find which value of K produced the highest prediction accuracy and stored that in a dataframe called match_ratios. I then plotted the prediction accuracy for all 50 models which you can see below the next block of code.

```
In [3]: # Reset seed again
        set.seed(1)

        # Build set of cross-validated models with kmax = 50
        model <- train.kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15,
                            data,
                            kmax = 50,
                            kernel = "triangular",
                            scale = TRUE)

        # Create df for match ratios
        match_ratios = data.frame(k = integer(50),
                                  matches = integer(50))

        # Loop through k values to check accuracy of each model
        for (k_val in 1:50) {

          # populate k values into match_ratio df
          match_ratios$k[k_val] <- k_val


          # Compare predicition with true value
          fit_val <- fitted(model)[[k_val]][1:nrow(data)]
```

```
  actual_val <- data[, 11]

  # Add match ratio to dataframe for comparison
  match_ratios$matches[k_val] <- sum(round(fit_val) == data[,11]) / nrow(data)
}

plot(match_ratios)
```



In [4]:
```
# Find highest prediction accuracy
highest_accuracy <- max(match_ratios$matches)

# Find row with highest prediction accuracy
which(match_ratios$matches == highest_accuracy)
```

10

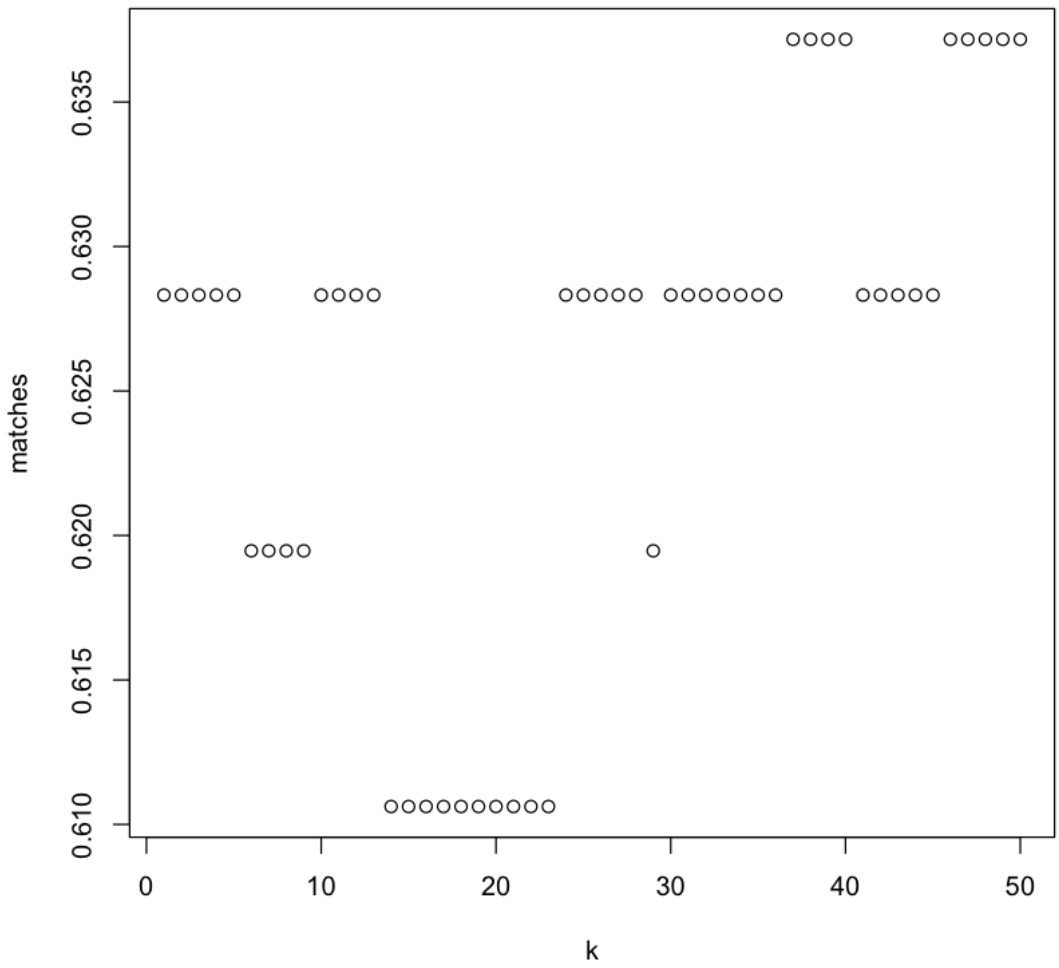
The optimal values of K that I identified was 10 so I will use that as my most accuracte k-value which is lower than we found in the previous hw. You can clearly see in the plot above that these values have a higher accuracy than the rest, and I confirmed that with the 2 lines of code above this text block.

2 Question 3.1b

In this question I am attempting to find a good classifier for the data like question 3.1a, however in this variation I am first splitting the data into training, validation, and test data sets. Splitting the data like this will help me to evaluate the actual prediction accuracy of the model produced. In the code block below, you can see that I split the data first and then I go through the same process as in question 3.1a where I loop through values of K to produce a dataframe containing the accuracy of each model. I finally plotted the data to see which model had the highest accuracy

```
In [5]: # Split dataset into training, validation, and testing sets
        split <- sample(1:3, size=nrow(data), prob=c(0.80,0.20,0.20), replace = TRUE)
        train <- data[split==1,]
        test <- data[split==2,]
        validate <- data[split==3,]

        # Create df for match ratios
        match_ratios = data.frame(k = integer(50),
                                  matches = integer(50))

        # Loop through k values
        for (k_val in 1:50) {

          # populate k values into match_ratio df
          match_ratios$k[k_val] <- k_val

          # Reset seed
          set.seed(1)

          # Create model using given k_value
          model <- kknn(R1~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15,
                        train,
                        validate,
                        k = k_val,
                        scale = TRUE)

            # Compare predicition with true value
            fit_val <- fitted.values(model)
            actual_val <- test[, 11]

            match_ratios$matches[k_val] <- sum(round(fit_val) == actual_val) / nrow(test)
        }
```

```
plot(match_ratios)
```




```
In [6]: # Find highest prediction accuracy
        highest_accuracy <- max(match_ratios$matches)

        # Find row with highest prediction accuracy
        which(match_ratios$matches == highest_accuracy)
```

1. 37 2. 38 3. 39 4. 40 5. 46 6. 47 7. 48 8. 49 9. 50

I identified 9 K values that produced the same prediction accuracy of about 64% so I am going with the lowest K value of 37. This K value is signficantly higher than in question 3.1a and produces a lower prediction accuracy, which can be expected given that this model is predicting values on a test set.

3 Question 4.1

3.0.1 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 might use.

A situation from everyday life that I think would be a good application of a clustering model is deciding whether a plant should be an indoor plant, or an outdoor plant in my home. Different plants have different requirements in terms of light, temperature, humidity, and water amonger other variables, and you can identify their needs based off of a few different factors which I will list below. These factors can identify the plants basic needs, and from their group them based on whether they would thrive inside a house or outside.

- **Leaf Color** - Plants with dark green leaves can surive with less light than bright green and can be categorized as indoor plants.
- **Leaf Thickness** - Plants with thicker leaves require less light and can survive in a house with lower light than a plant with thin leaves.
- **Plant Origin** - The part of the world that a plant comes from plays a role in it's temperature and humidity requirements. A plant from a rainforest will require higher temperatures and humidity than a plant from somewhere like Northern Europe. Plants that cannot survive cold weather will thus need to be inside during the winter.
- **Plant Ecosystem** - The actual ecosystem that a plant comes from plays a large role in their light requirements. An orchid for example is a vine plant that attaches itself to trees and sits below a canopy in a forest, so they are better suited to low-light and do well as indoor plants.

4 Question 4.2

In this question I am using the R function kmeans to cluster the points in the iris data set. My goal in the code below is to find the best combination of predictors and clusters that will produce the highest clustering accuracy against the response variable of species.

```
In [7]: # Load necessary libraries
        suppressWarnings(library(ggplot2))
        suppressWarnings(library(ggpubr))

        # Read iris data in
        iris <- tbl_df(read.table("iris.txt", header = TRUE))
        iris <- iris[,2:6] # drop index row

        # Check distribution of species
        table(iris$Species)
```

```
Attaching package: ggplot2

The following object is masked from package:kernlab:
```

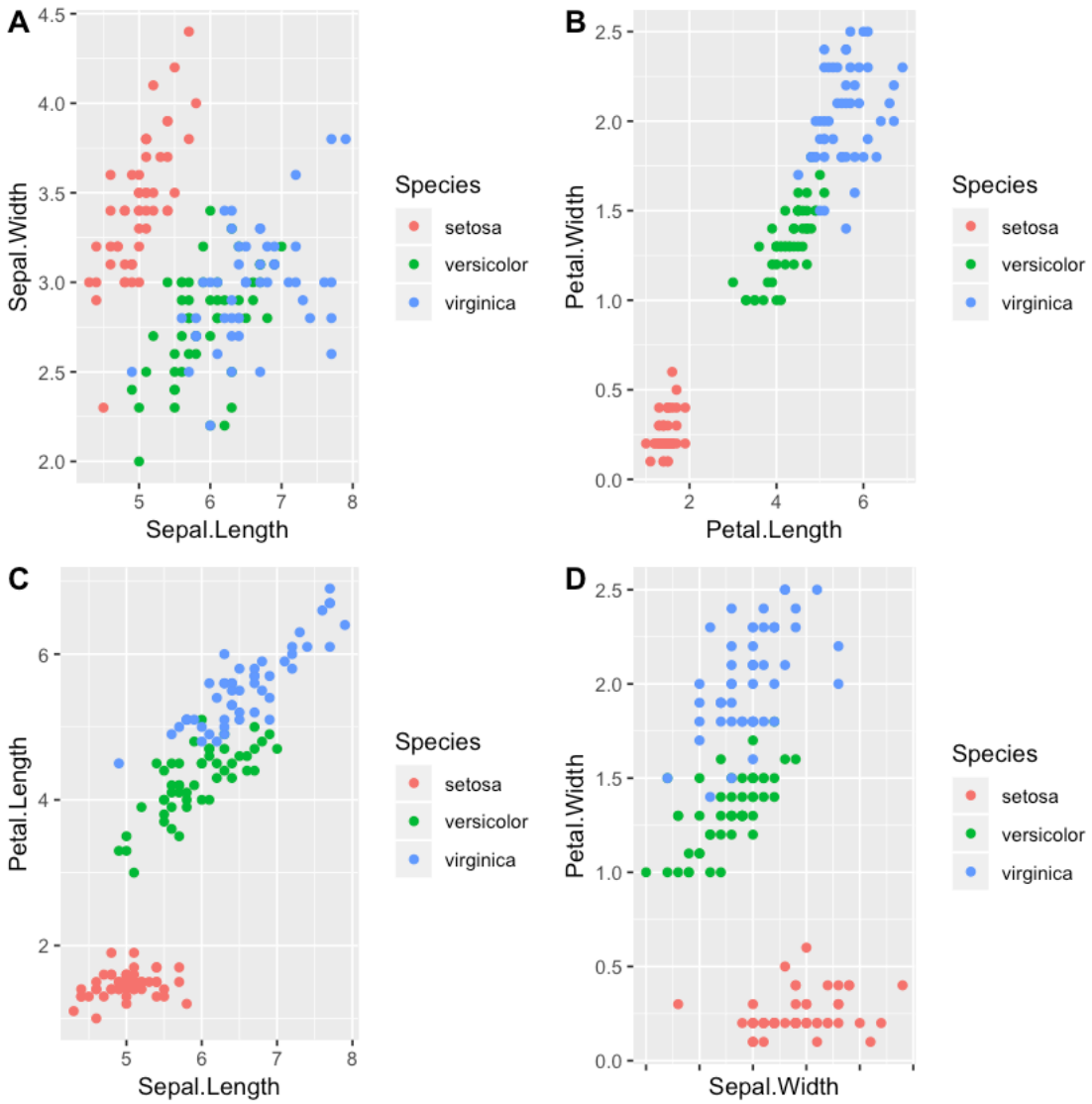
```
    alpha

Loading required package: magrittr



    setosa versicolor  virginica
        50         50         50
```

```
In [8]: # Check iris data set head
        head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

After getting a high level look at the data above I am goign to visualize the response variable species by different variables to see if there are some optimal clustering variables. On first pass, the species versicolor and virginica are pretty intertwined using variables like sepal width vs. sepal length. This is just to get a look at potential clusters by variable, however when actually executing k-means we won't know the response variable species. When looking at these 4 graphs, you can clearly see that there are distinct grouping with as few as 2 variables by species, which suggests that we may not have to use all of the predictors to get a strong model.

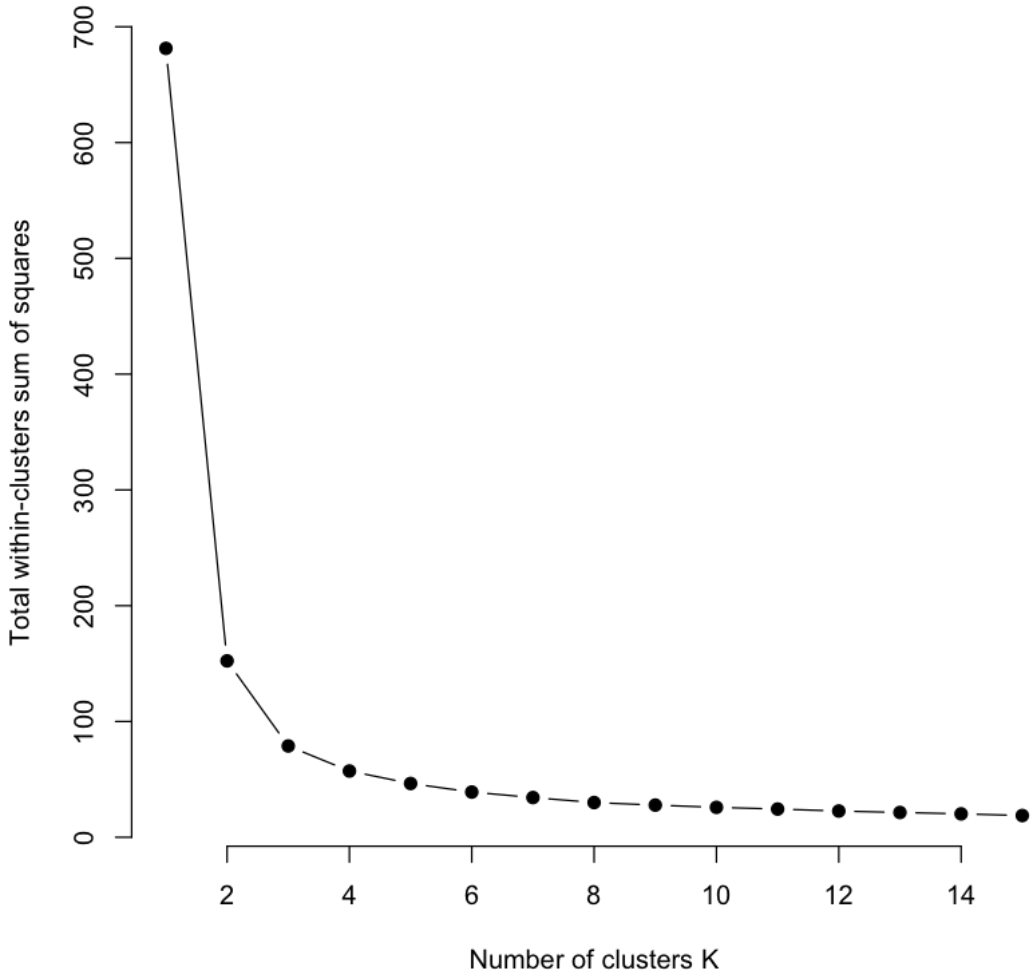
```
In [9]: # Plot 4 different graphs in one
        a <- ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point()
        b <- ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
        c <- ggplot(iris, aes(Sepal.Length, Petal.Length, color = Species)) + geom_point()
        d <- ggplot(iris, aes(Sepal.Width, Petal.Width, color = Species)) + geom_point()
        ggarrange(a, b, c, d + rremove("x.text"),
                  labels = c("A", "B", "C", "D"),
                  ncol = 2, nrow = 2)
```



After getting a look at the distribution of species clusters by using different predictor combinations, I then move on to test different kmeans models below. The first step I took is to test different K values from k = 2 to k = 15. I kept number of iter.max=15 to ensure the algorithm converges and nstart=50 to ensure that at leat 50 random sets are choosen. I visualized the accuracy of each model using an elbow graph. The elbow method looks at the percentage of variance explained as a function of the number of clusters where I choose a number of clusters so that adding another cluster doesn't give much better modeling accuracy. The number of clusters is chosen where the angle in the graph begins to flatten out. As you can see in the visualization below, the elbow occurs at a K value of 3.

```
In [13]: # Compute and plot wss for k = 2 to k = 15.
         k.max <- 15

         # Create different models and store tot.withinss values
```

```
wss <- sapply(1:k.max,
              function(k){kmeans(iris[,1:4], k, nstart=50,iter.max = 15)$tot.withinss

plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```



In [14]:
```
set.seed(1)

# The elbow occurs at k = 3. My base kmeans model as a result is below
kmm <- kmeans(iris[,1:4], centers = 3, nstart = 50, iter.max = 15)
```

```
# Check distribution by table to re-label clusters
table(kmm$cluster, iris$Species)

clusters <- tbl_df(as.data.frame(kmm$cluster)) %>%
  rename(cluster = `kmm$cluster`)

# Relabel clusters
clusters$cluster[clusters$cluster == 1] <- 'setosa'
clusters$cluster[clusters$cluster == 2] <- 'virginica'
clusters$cluster[clusters$cluster == 3] <- 'versicolor'

# Get model accuracy with all predictors
sum(clusters$cluster == iris$Species) / nrow(iris)
```

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

0.893333333333333

After confirming K value, I build a base model above using all predictor variables. I had to relabel the cluster output into each species for clarity when analyzing the data. The base kmeans classification accuracy with all predictors is 0.893. After this step, I went on to test different combinations of observations to see if any of them were stronger than all 4 observations together.

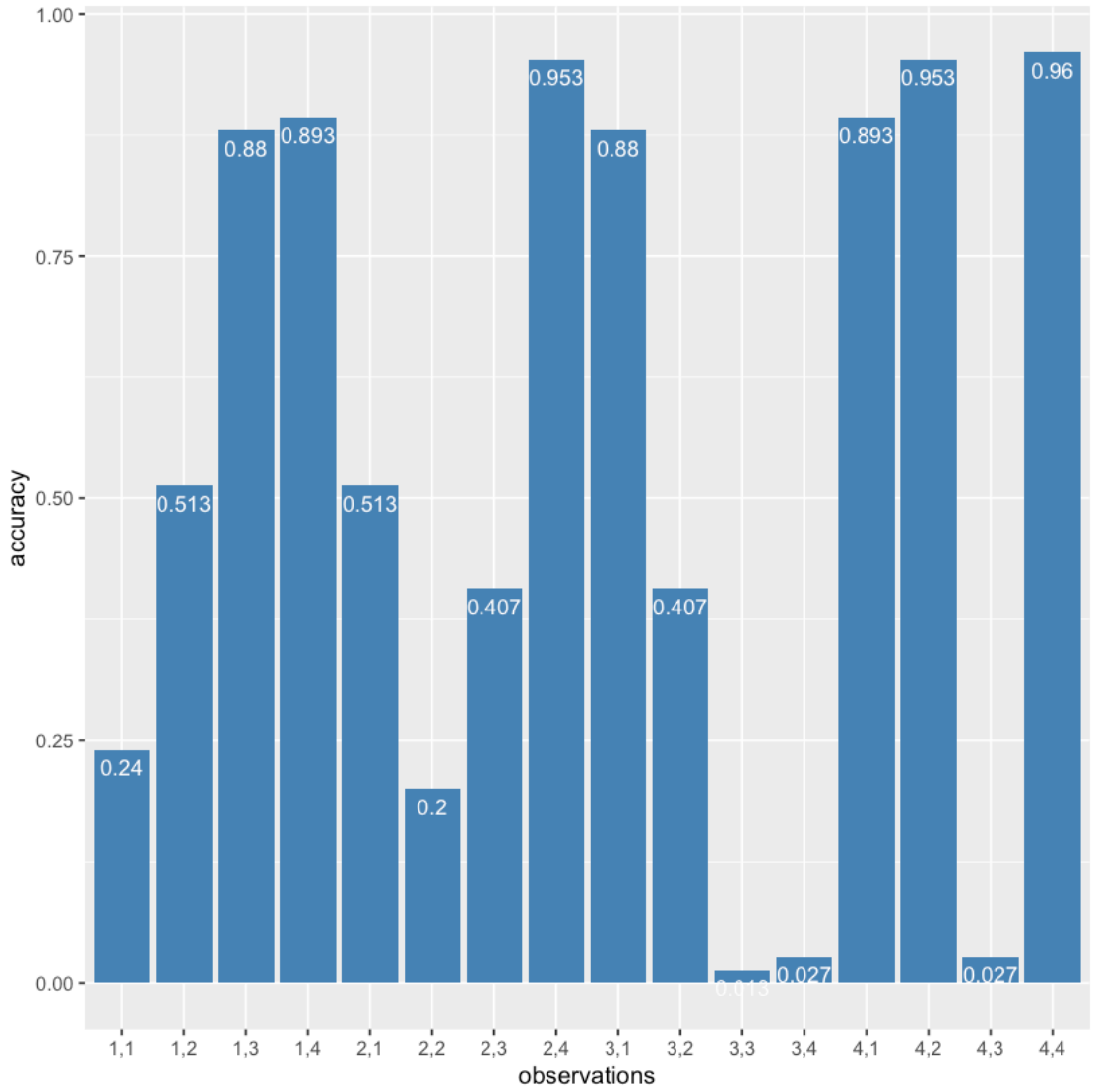
In [15]:
```
# Create dataframe for storing model classification accuracies
kmeans_accuracies <- data.frame(observations = character(16),
                                accuracy = integer(16),
                                stringsAsFactors = FALSE)
index <- 0

# Loop through all possible combinations and orders of predictors
for (i in 1:4) {
  for (j in 1:4) {
    set.seed(1)
    kmm <- kmeans(iris[,i:j], centers = 3, nstart = 50, iter.max = 15)

    clusters <- tbl_df(as.data.frame(kmm$cluster)) %>%
      rename(cluster = `kmm$cluster`)

    clusters$cluster[clusters$cluster == 1] <- 'setosa'
    clusters$cluster[clusters$cluster == 2] <- 'virginica'
    clusters$cluster[clusters$cluster == 3] <- 'versicolor'

    # Iterate count to populate rows in kmeans_accuracies df
```

```
      index <- index + 1

      kmeans_accuracies$observations[index] <- paste0(i,',',j)
      kmeans_accuracies$accuracy[index] <- sum(clusters$cluster == iris$Species) / nrow
    }
  }

  # Round accuracies for better visualization
  kmeans_accuracies$accuracy <- round(kmeans_accuracies$accuracy, 3)

  # Output acuracies for quick look at the data by predictor combination
  kmeans_accuracies
```

observations	accuracy
1,1	0.240
1,2	0.513
1,3	0.880
1,4	0.893
2,1	0.513
2,2	0.200
2,3	0.407
2,4	0.953
3,1	0.880
3,2	0.407
3,3	0.013
3,4	0.027
4,1	0.893
4,2	0.953
4,3	0.027
4,4	0.960

The highest accuracy when testing all combinations of observations in the Iris data set was actually just petal.width by itself at an accuracy of 0.96. The next highest accuracy which you can see in the visualization below was the combination of petal width, petal length, and sepal width. In the graph you can see this labeled in the bar 2, 4 as well as 4,2. These bars contain the same observations, just in reverse order. I included my final kmeans model below the visualization, you can see that out of the 150 datapoints, only 6 were labeled incorrectly using a K of 3 and only pedal.width.

```
In [16]: ggplot(kmeans_accuracies, aes(x=observations, y=accuracy)) +
           geom_bar(stat='identity',fill="steelblue") +
           geom_text(aes(label=accuracy), vjust=1.6, color="white", size=3.5)
```



In [17]:
```
# Final model choice below and table output
set.seed(1)
kmm <- kmeans(iris[,4:4], centers = 3, nstart = 50, iter.max = 15)
table(kmm$cluster, iris$Species)
```

```
   setosa versicolor virginica
1      50          0         0
2       0          2        46
3       0         48         4
```