FINAL PROJECT REPORT

MEDICAL INSURANCE EXPENSE

HA HAI VU

Data gathering and intergration:

For this project, I pick dataset *insurance.csv* from the book *"Machine Learning with R"* by Brett Lantz. I have discovered this data from GitHub, the link to the documentation is here.

This dataset includes 1,338 examples of beneficiaries currently enrolled in the insurance plan, with 7 features indicating characteristics of the patient as well as the total medical expenses charged to the plan for the calendar year. The features are:

- age: age of the primary beneficiary
- sex: insurance contractor gender, female, male
- bmi: body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight ($\frac{kg}{m^2}$) using the ratio of height to weight, ideally 18.5 to 24.9.
- children: Number of children covered by health insurance / Number of dependents
- smoker: this is yes/no depending on whether the insured regularly smokes tobacco.
- region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest
- charges: individual medical costs billed by health insurance

I have decided to use this dataset to predict insurance costs based on the provided content.

```
> setwd("D:/E - Working/@Master - Data Science/Study/7. DSC 441 - Fundamentals of Data Science/Week 10 - 03.07.2022/HW5")
> insurance = read.csv("insurance.csv")
> head(insurance)
  age    sex    bmi children smoker    region   charges
1  19 female 27.900        0    yes southwest 16884.924
2  18   male 33.770        1     no southeast  1725.552
3  28   male 33.000        3     no southeast  4449.462
4  33   male 22.705        0     no northwest 21984.471
5  32   male 28.880        0     no northwest  3866.855
6  31 female 25.740        0     no southeast  3756.622
```

Data exploration:

```
> str(insurance)
'data.frame':	1338 obs. of  7 variables:
 $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
 $ sex     : chr  "female" "male" "male" "male" ...
 $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
 $ children: int  0 1 3 0 0 0 1 3 2 0 ...
 $ smoker  : chr  "yes" "no" "no" "no" ...
 $ region  : chr  "southwest" "southeast" "southeast" "northwest" ...
 $ charges : num  16885 1726 4449 21984 3867 ...
```

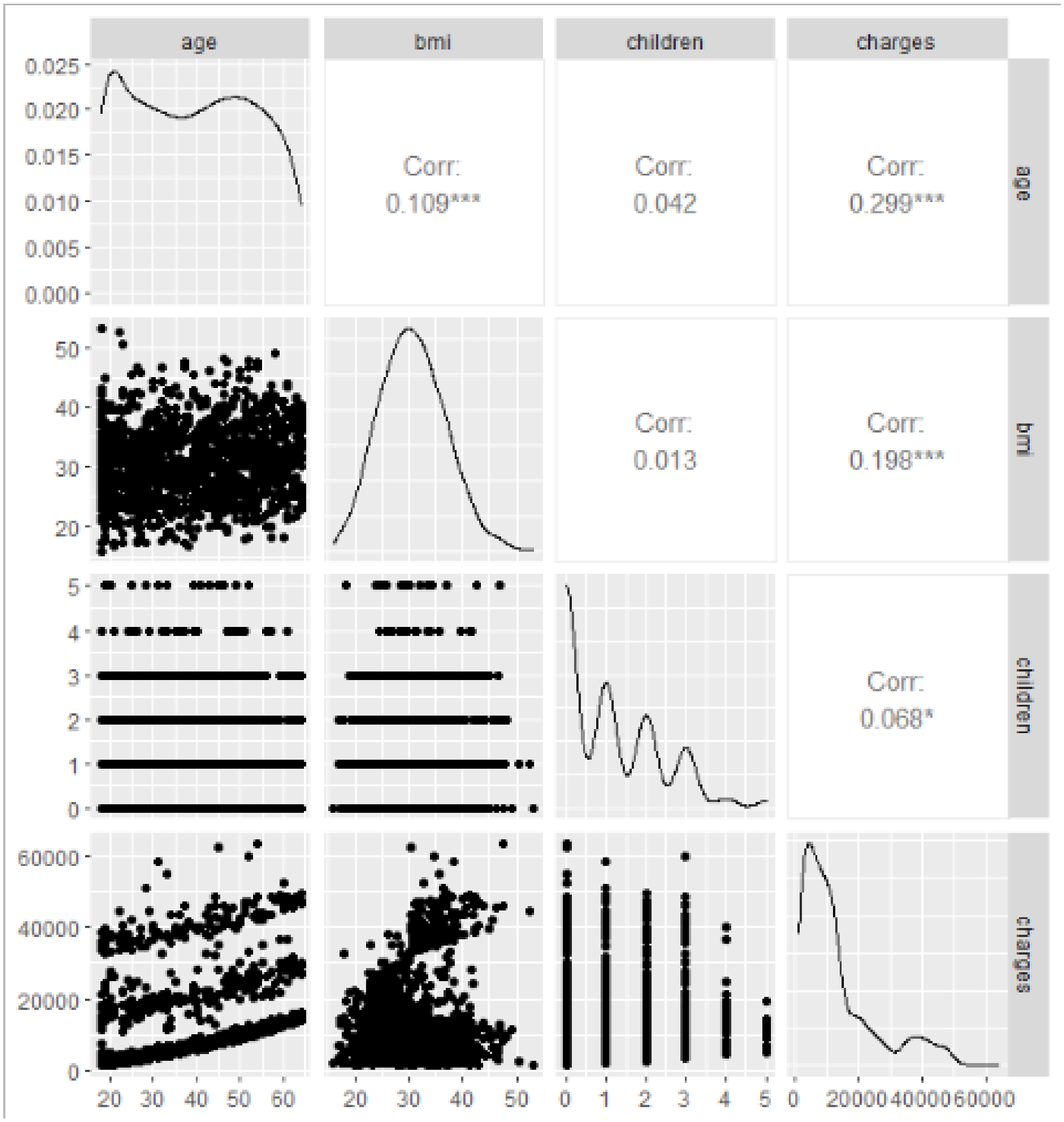
```
> summary(insurance)
      age            sex                 bmi           children        smoker             region             charges     
 Min.   :18.00   Length:1338        Min.   :15.96   Min.   :0.000   Length:1338        Length:1338        Min.   : 1122  
 1st Qu.:27.00   Class :character   1st Qu.:26.30   1st Qu.:0.000   Class :character   Class :character   1st Qu.: 4740  
 Median :39.00   Mode  :character   Median :30.40   Median :1.000   Mode  :character   Mode  :character   Median : 9382  
 Mean   :39.21                      Mean   :30.66   Mean   :1.095                                         Mean   :13270  
 3rd Qu.:51.00                      3rd Qu.:34.69   3rd Qu.:2.000                                         3rd Qu.:16640  
 Max.   :64.00                      Max.   :53.13   Max.   :5.000                                         Max.   :63770  
```

We can see that there are no missing values in this dataset. In this data, the types of variables included are categorical (sex, smoker, region) and numeric (age, bmi, children and charges).

Now, let's perform some more exploratory tasks and find existing relationships.

First, I'll create a scatterplot matrix of the numerical variables:

insurance %>% select(age, bmi, children, charges) %>% ggpairs()



We can see a few things worth from this graph: Firstly, while there are outliers, we can see graphically that charges generally increase with age. The Pearson correlation coefficient also shows a positive correlation between charges and age. Secondly, the trend between BMI and charges isn't that clear graphically, but we can see that there is definitely an increase in charges for some individuals after hitting the age of 30. The Pearson correlation coefficient also proves a somewhat positive correlation. This will be interesting to discover later based on sex and smoker status. Thirdly, we note the somewhat positive correlation number between age and BMI, although this is unclear graphically.

Now we view the number of women to men who are smokers to nonsmokers:

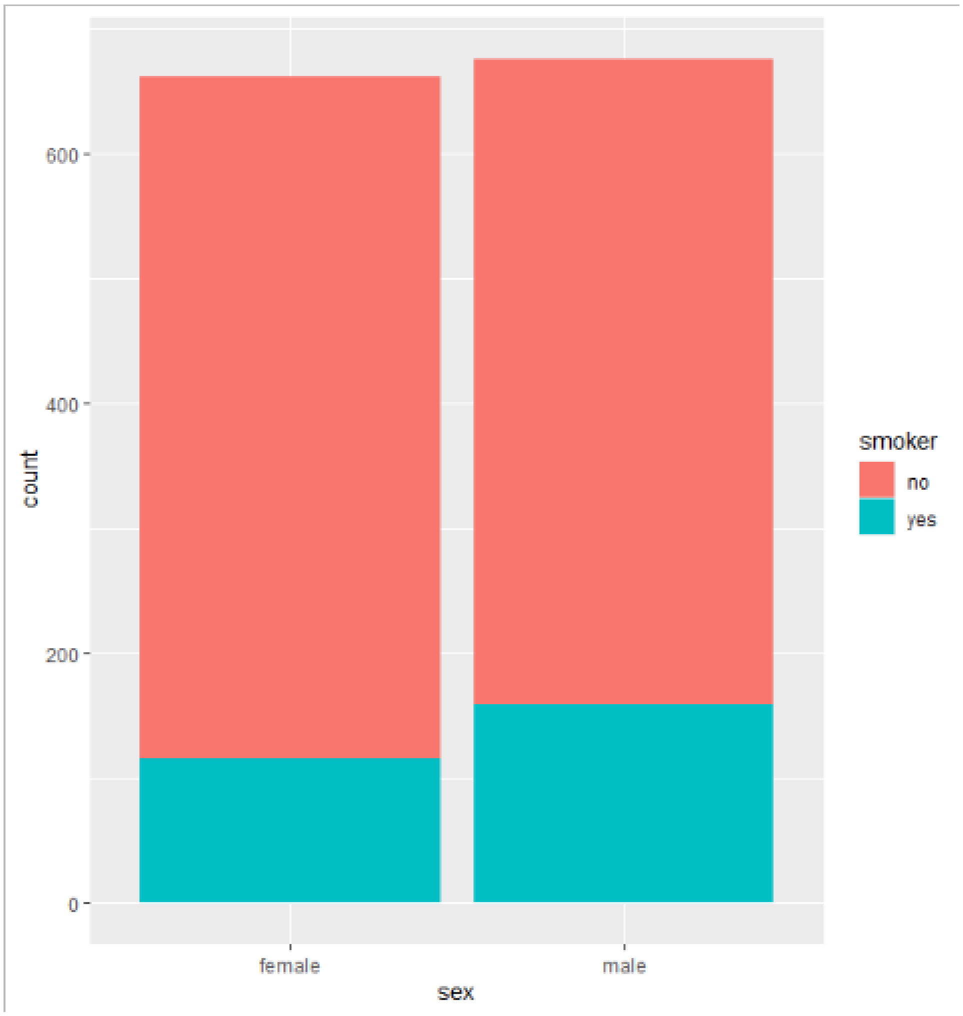
 # Convert to dataframe

df = as.data.frame(insurance)

 # Create ggplot object

p = ggplot(insurance, aes(x=sex, fill=smoker))

p + geom_bar(position="stack")

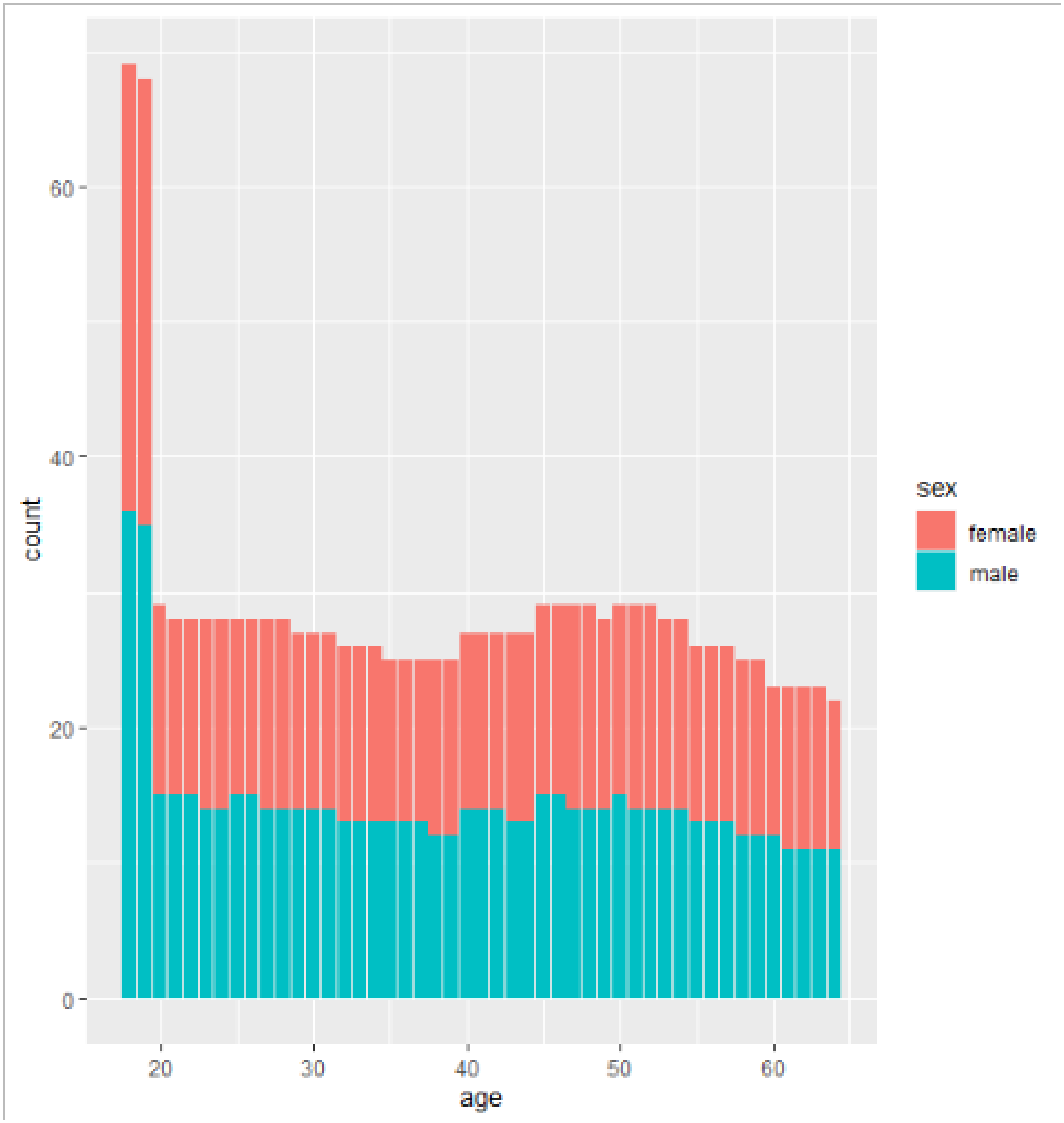


We can see here that, as expected, there are slightly more smokers amongst men than there are smokers amongst women. Another important thing to note is that there are almost equally as many men represented in the dataset as there are women.

Let's see the actual ages represented in this dataset, as well as sex per age, and try to see if it's balanced:

p = ggplot(df, aes(x=age, fill=sex))

p + geom_bar(position="stack")



We can see here that there is such a high representation of ages under 20. To eliminate bias towards such younger age groups, it may help to remove all input from ages under 20.

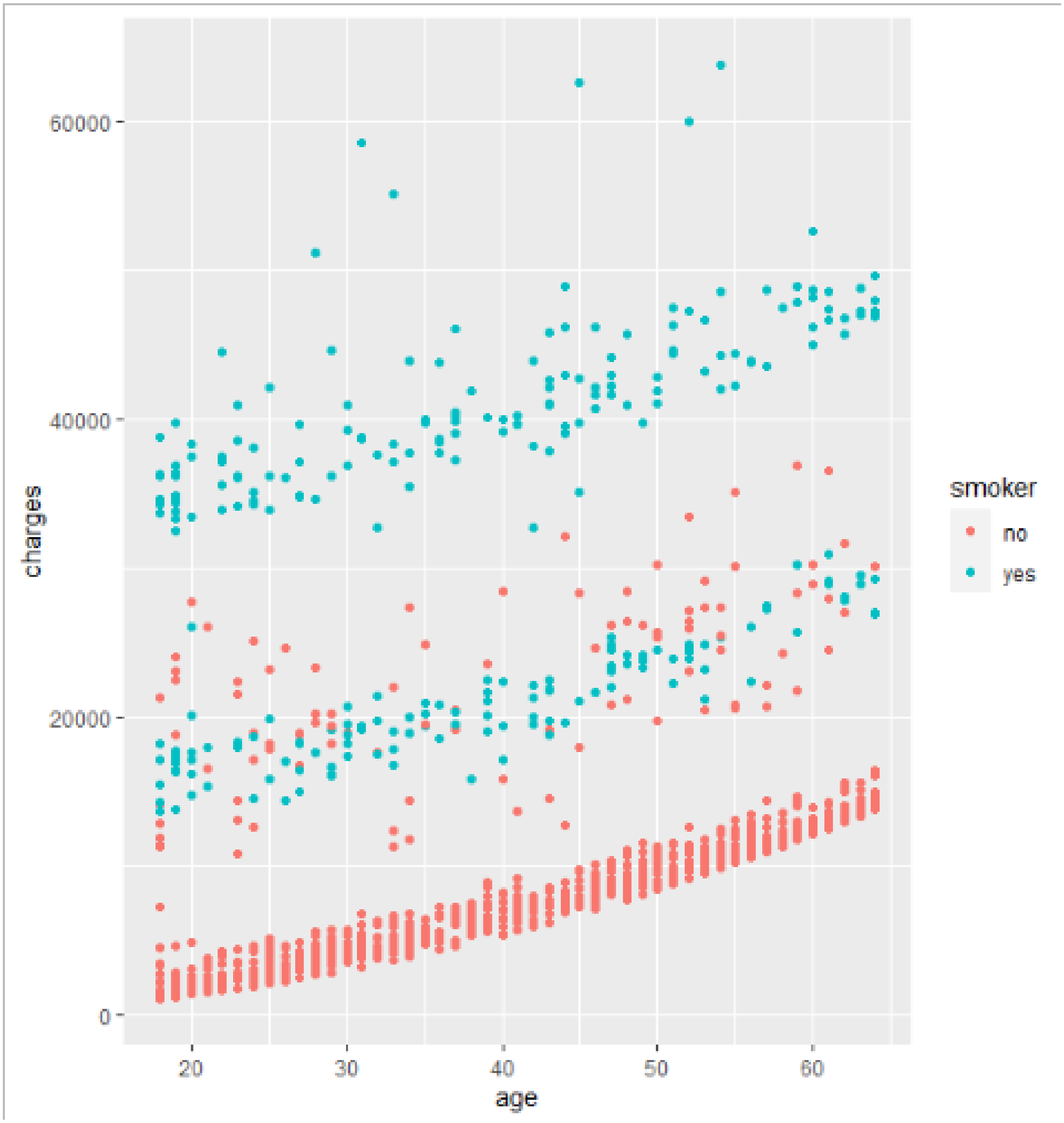
Let's quickly view the number of smokers here:

insurance %>% group_by(smoker) %>% summarise("count"=n())

```
> insurance %>% group_by(smoker) %>% summarise("count"=n())
# A tibble: 2 x 2
  smoker count
  <chr>  <int>
1 no      1064
2 yes      274
> 
```

We've got 274 smokers in this dataset. Now let's look at this graphically:

ggplot(insurance, aes(x=age, y=charges, color=smoker)) + geom_point()

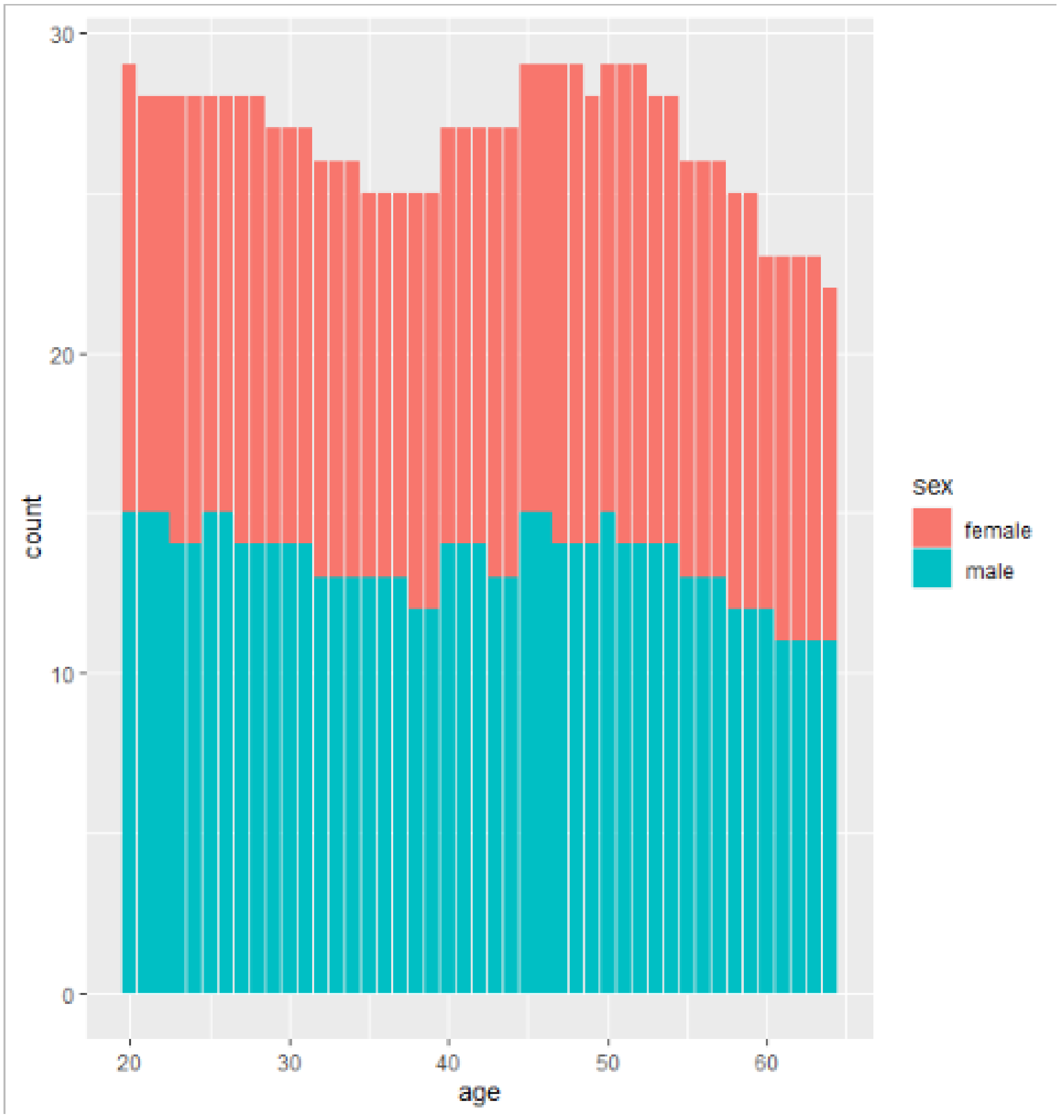


We can see that many of those smokers are the outliers here, in which they are charged much more for insurance than non-smokers.

Data Cleaning:

As we saw from the data exploration part above, this dataset seems to be mostly straightforward with no NA's or missing data. Although, we did notice that there may be a bias towards ages younger than 20, so we will not remove all data points of those younger than 20.

Now we exclude rows using subset function with condition of including ages greater than or equal to 20, and see how this looks graphically:



We can see that it now looks much more balanced.

Data Preprocessing:

Now, we will bin both age groups and charges which may be helpful when processing the data further. We will bin age groups into 5 different bins: "twenties", "thirties", "fourties", "fifties" and "sixtiesplus".

myinsurance <- myinsurance %>%

mutate(agegroup = cut(age, breaks=c(-Inf, 29, 39, 49, 59, Inf),labels=c("twenties", "thirties", "fourties", "fifties","sixtiesplus")))

head(myinsurance)

```
> head(myinsurance)
  age    sex    bmi children smoker    region   charges agegroup
3  28   male 33.000        3     no southeast  4449.462 twenties
4  33   male 22.705        0     no northwest 21984.471 thirties
5  32   male 28.880        0     no northwest  3866.855 thirties
6  31 female 25.740        0     no southeast  3756.622 thirties
7  46 female 33.440        1     no southeast  8240.590 fourties
8  37 female 27.740        3     no northwest  7281.506 thirties
```

Clustering:

Firstly we remove class labels and create dummy variables:

df = myinsurance

predictors <- df %>% select(-c(agegroup, region))

head(predictors)

```
> head(predictors)
  age    sex    bmi children smoker   charges
3  28   male 33.000        3     no  4449.462
4  33   male 22.705        0     no 21984.471
5  32   male 28.880        0     no  3866.855
6  31 female 25.740        0     no  3756.622
7  46 female 33.440        1     no  8240.590
8  37 female 27.740        3     no  7281.506
```

##create dummies

dummy <- dummyVars(charges ~ ., data = predictors)

dummies <- as.data.frame(predict(dummy, newdata = predictors))

##include charges

dummies$charges = myinsurance$charges

##rename predictors

predictors <- dummies

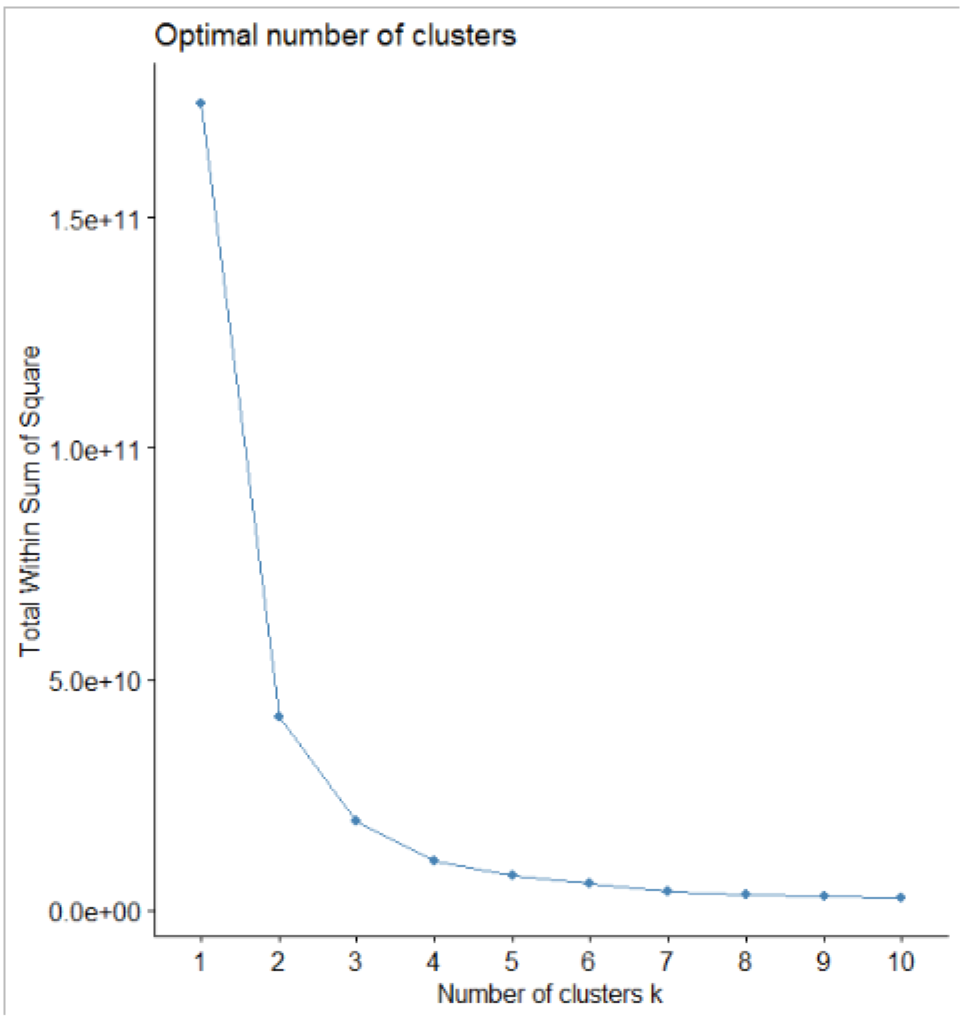
Now I will try K-means:

##set seed

set.seed(123)

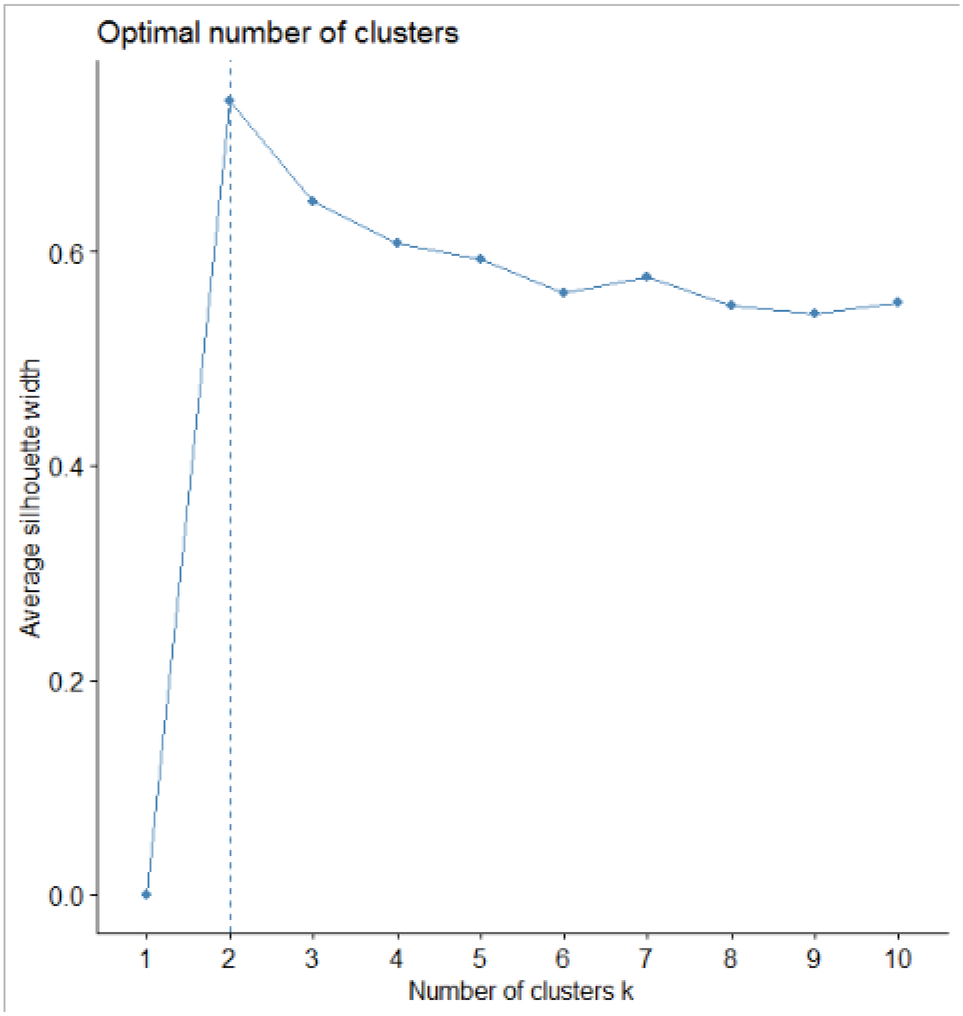
##find K; find the knee

fviz_nbclust(predictors, kmeans, method = "wss")



##use silhouette to find K

fviz_nbclust(predictors, kmeans, method = "silhouette")



The knee suggests a K of 4 but the silhoutte score suggests K = 2. Using K = 4 as suggestion by the plots we can fit our model using the k-means function.

 # Fit the data

fit <- kmeans(predictors, centers = 4, nstart = 25)

 # Display the kmeans object information

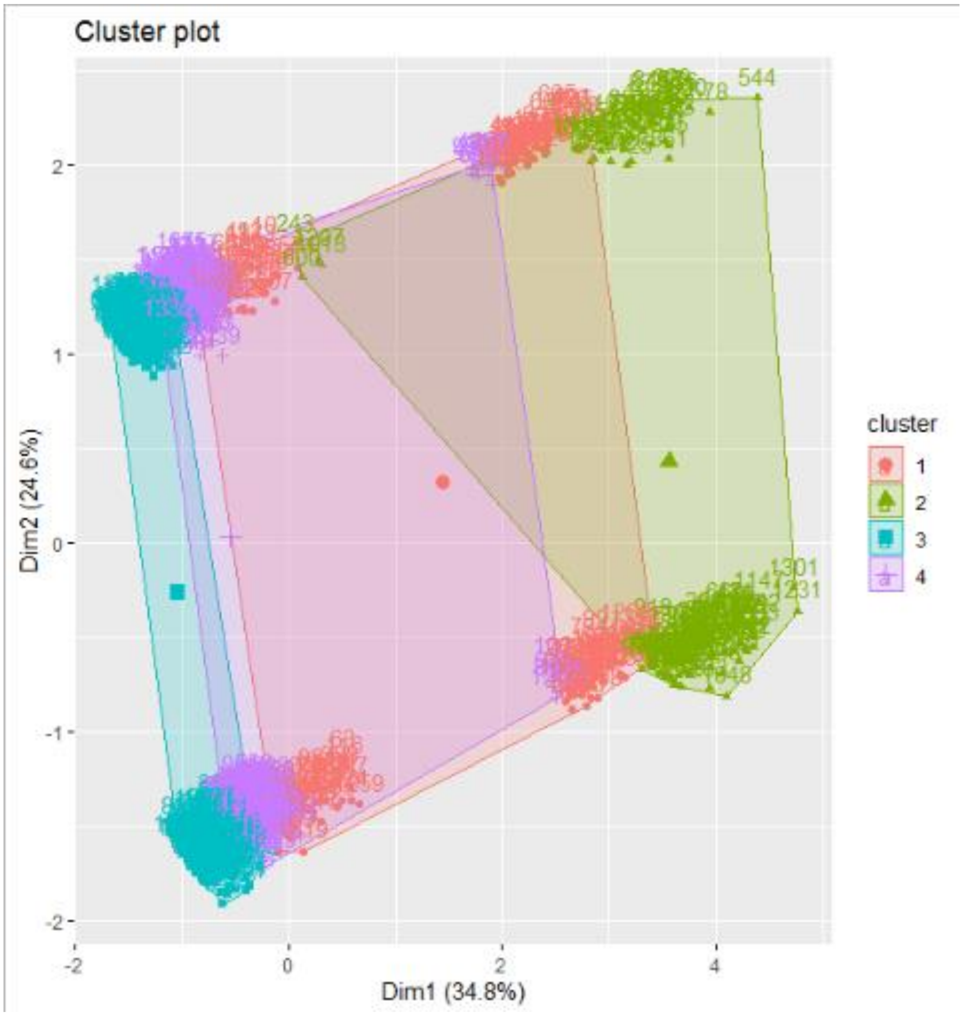
fit

```
K-means clustering with 4 clusters of sizes 164, 138, 481, 418

Cluster means:
       age sexfemale   sexmale      bmi children   smokerno  smokeryes   charges
1 42.78659 0.4756098 0.5243902 28.13838 1.250000 0.42682927 0.57317073 22850.673
2 41.94203 0.3695652 0.6304348 35.10152 1.231884 0.02898551 0.97101449 41963.079
3 31.94179 0.4989605 0.5010395 30.24236 1.143451 1.00000000 0.00000000  4803.633
4 52.04785 0.5430622 0.4569378 30.90012 1.148325 0.96172249 0.03827751 11375.730
```

Using the fviz_cluster function, we can visualize how the clusters are formed.

 # Display the cluster plot

fviz_cluster(fit, data = predictors)



For comparison we can generate our own PCA plot and color the points based on their charges.

Calculate PCA

pca = prcomp(predictors)

Save as dataframe

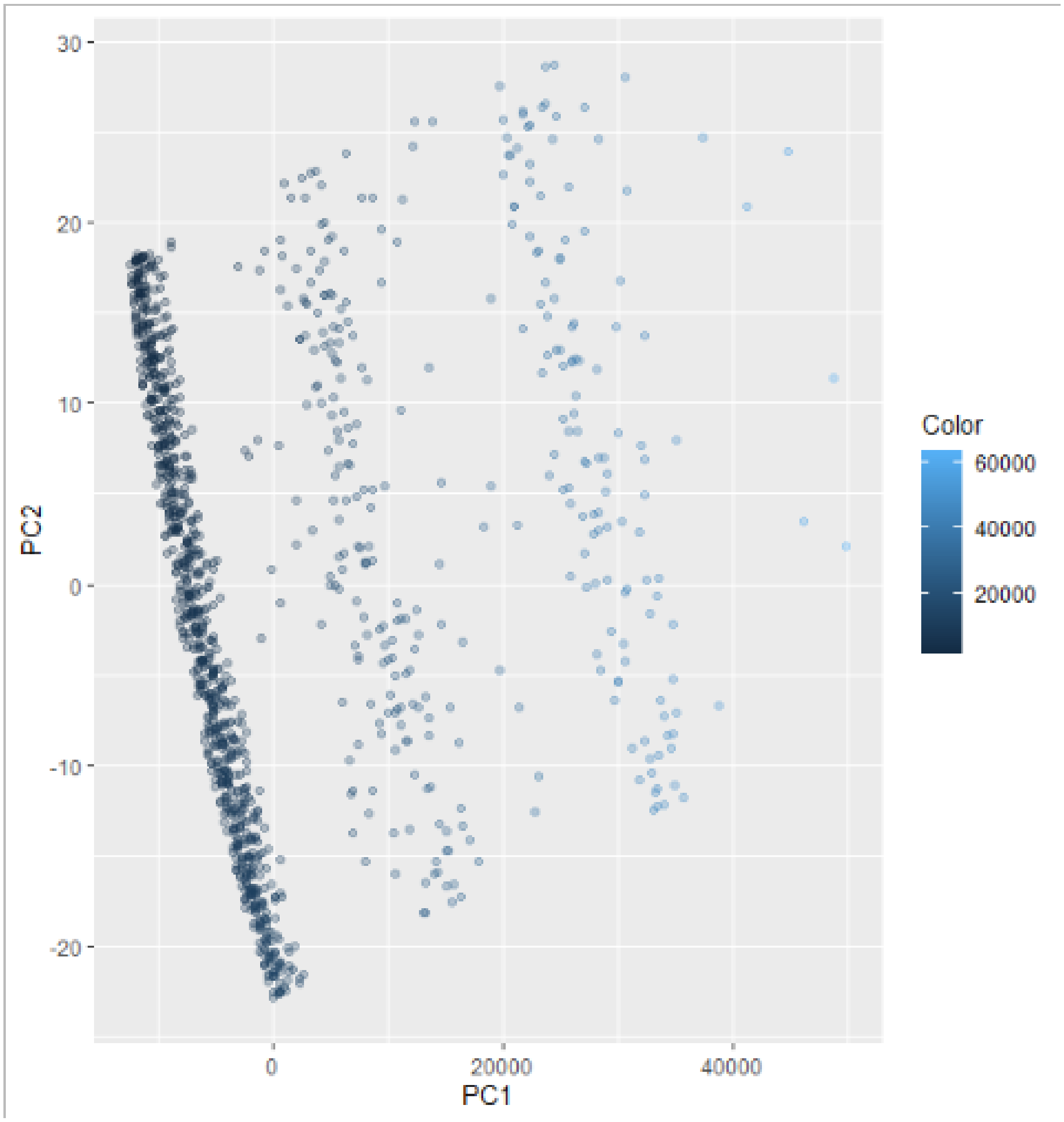
rotated_data = as.data.frame(pca$x)

Add original labels as a reference

rotated_data$Color <- df$charges

Plot and color by labels

ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(alpha= 0.3)



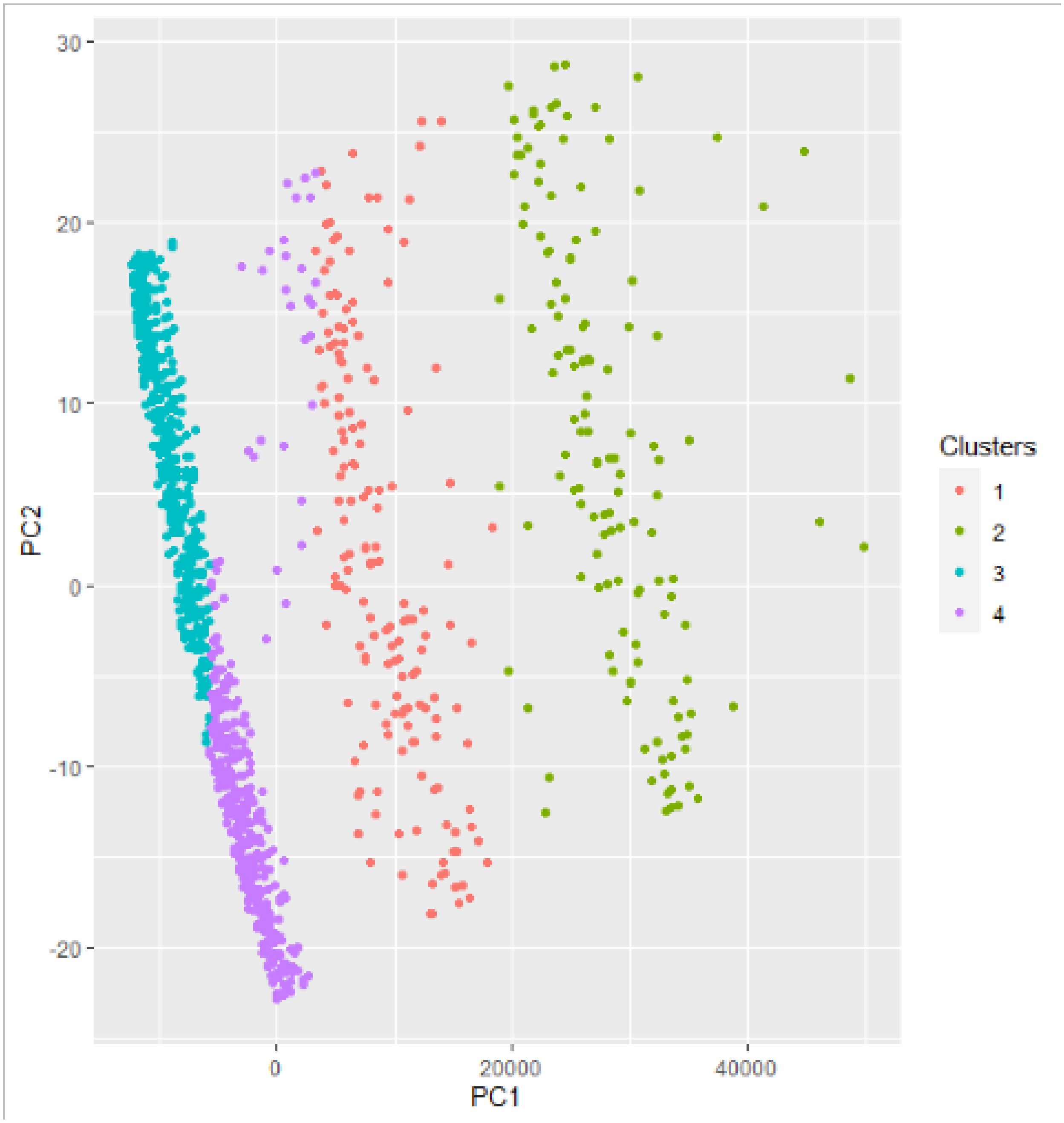
The cluster plot can also be done on ggplot based on the cluster result from the K-means algorithm:

Assign clusters as a new column

rotated_data$Clusters = as.factor(fit$cluster)

Plot and color by labels

ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Clusters)) + geom_point()



Classification

Now we use KNN to make predictions based on smokers:

set.seed(123)

ctrl = trainControl(method="cv", number = 10)

knnFit <- train(smoker ~ ., data = myinsurance, method = "knn", trControl = ctrl, preProcess = c("center","scale")) *knnFit*

```
k-Nearest Neighbors

1201 samples
   7 predictor
   2 classes: 'no', 'yes'

Pre-processing: centered (12), scaled (12)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1080, 1081, 1080, 1081, 1081, 1082, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.9075869  0.6734083
  7  0.9026076  0.6473205
  9  0.9018090  0.6354553

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

Fromm the output above, we can see that Accuracy and Kappa are reportedly high. Now, we'll attempt to control and find the best k value:

set.seed(123)

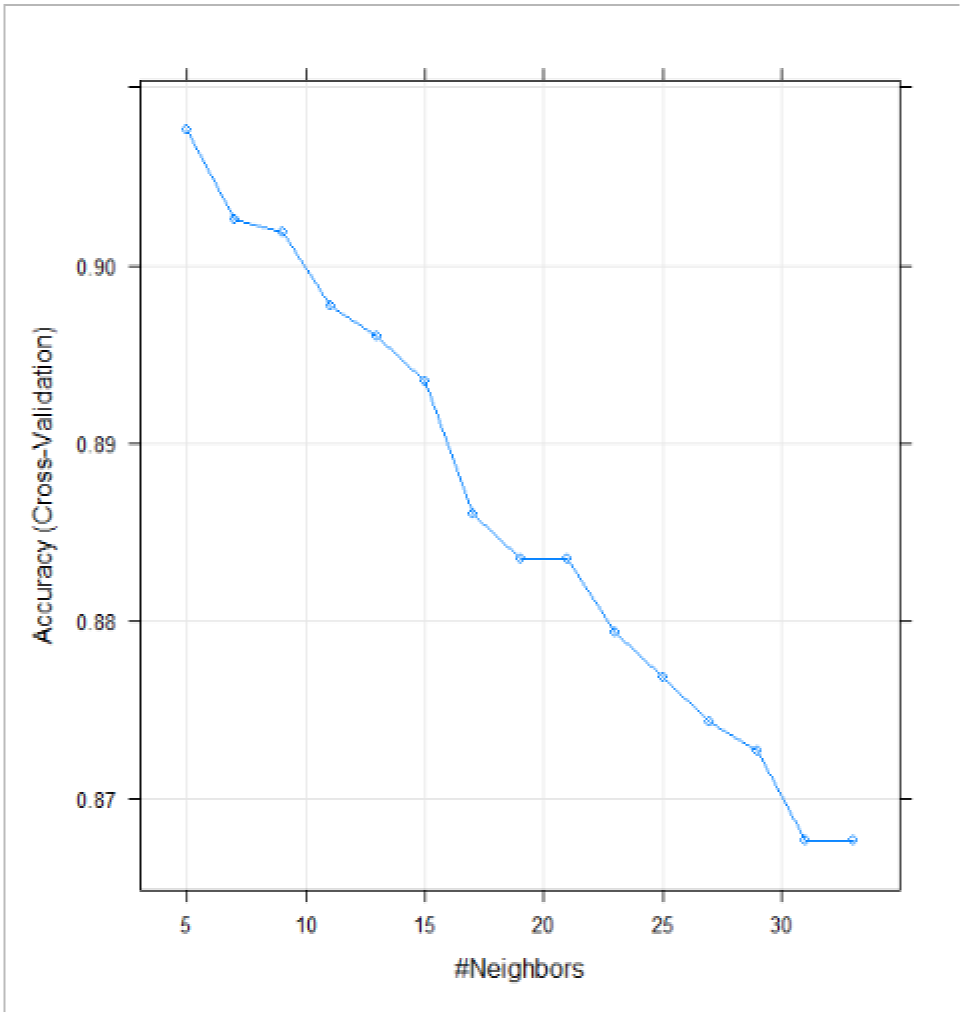
ctrl = trainControl(method="cv", number = 10)

knnFit <- train(smoker ~ ., data = myinsurance, method = "knn", trControl = ctrl, preProcess = c("center","scale"), tuneLength = 15)

knnFit

```
k-Nearest Neighbors

1201 samples
   7 predictor
   2 classes: 'no', 'yes'

Pre-processing: centered (12), scaled (12)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1080, 1081, 1080, 1081, 1081, 1082, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9075869  0.6734083
   7  0.9026076  0.6473205
   9  0.9018090  0.6354553
  11  0.8976699  0.6090430
  13  0.8960102  0.5968295
  15  0.8934962  0.5852356
  17  0.8860239  0.5465009
  19  0.8835237  0.5349892
  21  0.8835028  0.5346791
  23  0.8793431  0.5148305
  25  0.8768289  0.5018231
  27  0.8743288  0.4870325
  29  0.8726620  0.4771848
  31  0.8676758  0.4520441
  33  0.8676687  0.4511347

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

plot(knnFit)



Accuracy varies between 0.97 and 0.98, which is quite good. However, to avoid overfitting, it's best to choose a higher K number as chances of overfitting decrease. Therefore, a K value of 18 may be best.

Now, we'll use grid search to find the best value of K:

##setup a tuneGrid with the tuning parameters

tuneGrid <- expand.grid(kmax = 3:7, kernel = c("rectangular", "cos"), distance = 1:3)

tune and fit the model with 10-fold cross validation, standardization, and our specialized tune grid

kknn_fit <- train(smoker ~ ., data = myinsurance, method = 'kknn', trControl = ctrl, preProcess = c('center', 'scale'), tuneGrid = tuneGrid)

##Printing trained model provides report

kknn_fit

```
k-Nearest Neighbors

1201 samples
   7 predictor
   2 classes: 'no', 'yes'

Pre-processing: centered (12), scaled (12)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1082, 1081, 1081, 1082, 1080, 1082, ...
Resampling results across tuning parameters:

  kmax  kernel       distance  Accuracy   Kappa
  3     rectangular  1         0.9158848  0.7269340
  3     rectangular  2         0.9142249  0.7232849
  3     rectangular  3         0.9133778  0.7227479
  3     cos          1         0.9158848  0.7269340
  3     cos          2         0.9158501  0.7325850
  3     cos          3         0.9116972  0.7176086
  4     rectangular  1         0.9158848  0.7269340
  4     rectangular  2         0.9142249  0.7232849
  4     rectangular  3         0.9133778  0.7227479
  4     cos          1         0.9150445  0.7238570
  4     cos          2         0.9158501  0.7325850
  4     cos          3         0.9133778  0.7209788
  5     rectangular  1         0.9158848  0.7269340
  5     rectangular  2         0.9142249  0.7232849
  5     rectangular  3         0.9133778  0.7227479
  5     cos          1         0.9150445  0.7238570
  5     cos          2         0.9158501  0.7325850
  5     cos          3         0.9142112  0.7225564
  6     rectangular  1         0.9158848  0.7269340
  6     rectangular  2         0.9142249  0.7232849
  6     rectangular  3         0.9133778  0.7227479
  6     cos          1         0.9150445  0.7238570
  6     cos          2         0.9158501  0.7325850
  6     cos          3         0.9142112  0.7225564
  7     rectangular  1         0.9158848  0.7269340
  7     rectangular  2         0.9142249  0.7232849
  7     rectangular  3         0.9133778  0.7227479
  7     cos          1         0.9150445  0.7238570
  7     cos          2         0.9125167  0.7181108
  7     cos          3         0.9133778  0.7188768

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were kmax = 7, distance = 1 and kernel = rectangular.
```

Now we can apply the model based on the data above:

Predict

pred_knn <- predict(kknn_fit, myinsurance)

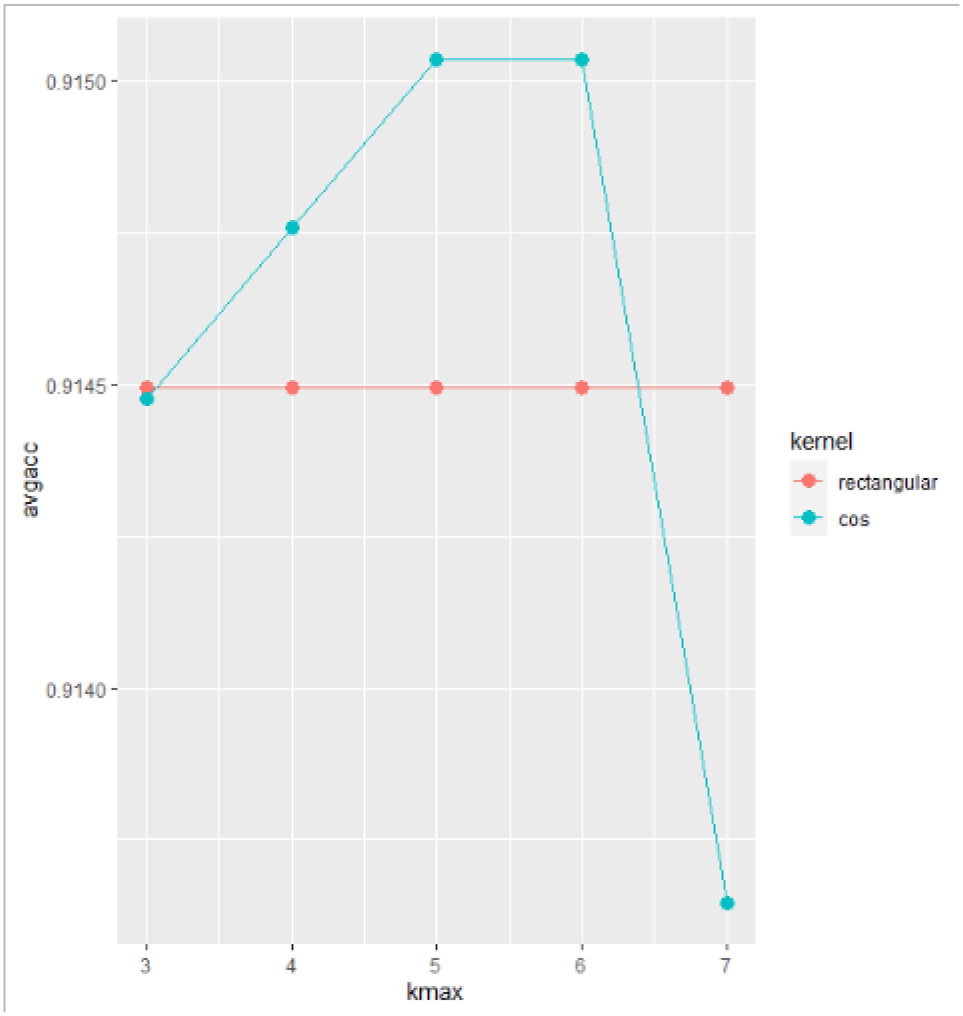
Generate confusion matrix

myinsurance$smoker = as.factor(myinsurance$smoker)

confusionMatrix(myinsurance$smoker, pred_knn)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  957   0
       yes   0 244

               Accuracy : 1
                 95% CI : (0.9969, 1)
    No Information Rate : 0.7968
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.7968
         Detection Rate : 0.7968
   Detection Prevalence : 0.7968
      Balanced Accuracy : 1.0000

       'Positive' Class : no
```

```
## Result

knn_results = kknn_fit$results

knn_results <- knn_results %>%

  group_by(kmax, kernel) %>%

  mutate(avgacc = mean(Accuracy))

ggplot(knn_results, aes(x=kmax, y=avgacc, color=kernel)) +

  geom_point(size=3) + geom_line()
```



Evaluation:

Using the better classifier kknn from previous part, we produce confusion matrix:

Generate confusion matrix

myinsurance$smoker = as.factor(myinsurance$smoker)

cm = confusionMatrix(myinsurance$smoker, pred_knn)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  957   0
       yes   0 244

               Accuracy : 1
                 95% CI : (0.9969, 1)
    No Information Rate : 0.7968
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.7968
         Detection Rate : 0.7968
   Detection Prevalence : 0.7968
      Balanced Accuracy : 1.0000

       'Positive' Class : no
```

	Actual Class		
Predict Class		Smoke = No	Smoke = Yes
	Smoke = No	957	0
	Smoke = Yes	0	244

Store the byClass object of confusion matrix as a dataframe

metrics <- as.data.frame(cm$byClass)

View the object

metrics

```
                    cm$byClass
Sensitivity           1.000000
Specificity           1.000000
Pos Pred Value        1.000000
Neg Pred Value        1.000000
Precision             1.000000
Recall                1.000000
F1                    1.000000
Prevalence            0.796836
Detection Rate        0.796836
Detection Prevalence  0.796836
Balanced Accuracy     1.000000
```

Now we calculate the precision and recall manually:

		Actual Class			
Predicted Class		Smoke = No	Smoke = Yes	Total	Recognition(%)
	Smoke = No	957	0	957	100%
	Smoke = Yes	0	244	244	100%
	Total	957	244	1201	
	Recognition(%)	100%	100%		

Finally we produce ROC plot:

Get class probabilities for KNN

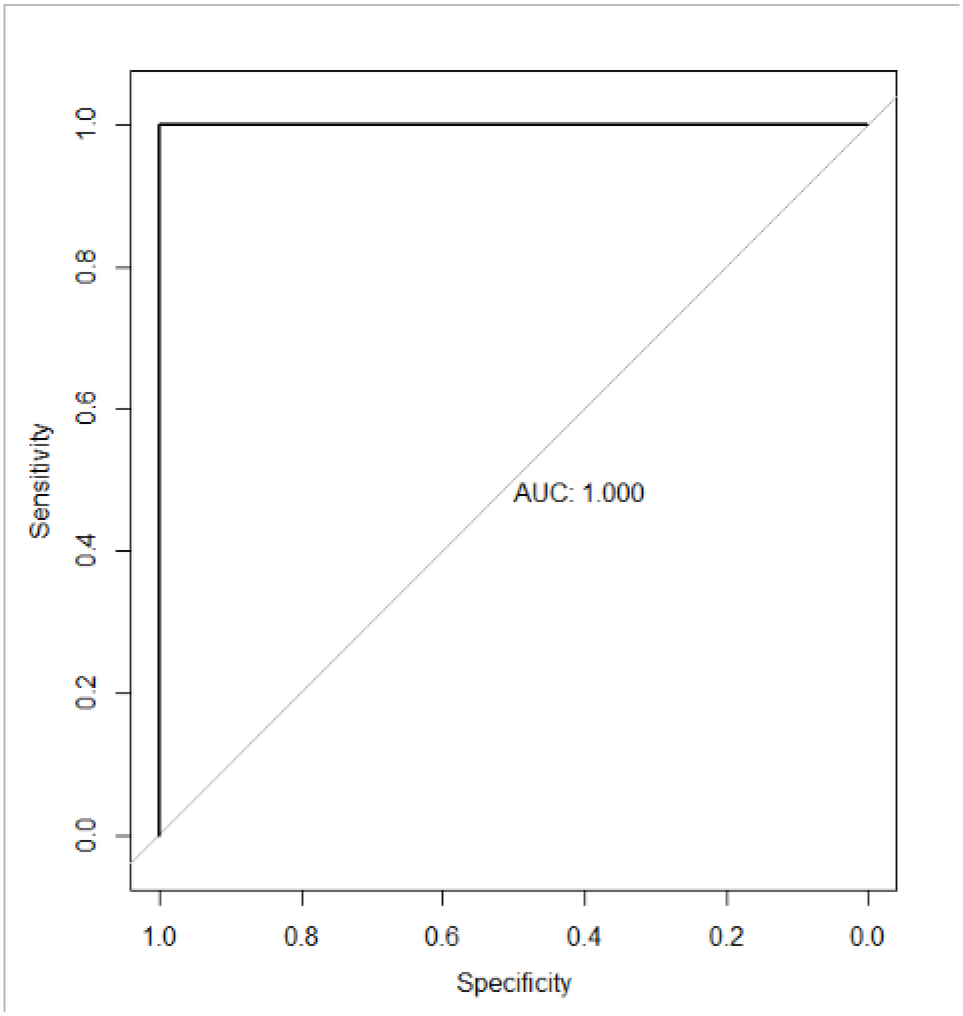
pred_prob <- predict(kknn_fit, myinsurance, type = "prob")

head(pred_prob)

And now we can create an ROC curve for our model.

roc_obj <- roc((myinsurance$smoker), pred_prob[,1])

plot(roc_obj, print.auc=TRUE)



We can see that these performance measures makes our classifier look the same as accuracy.

Report:

The insurance.csv dataset contains 1338 observations (rows) and 7 features (columns). The dataset contains 4 numerical features (age, bmi, children and expenses) and 3 nominal features (sex, smoker and region) that were converted into factors with numerical value designated for each level.

The purposes of this dataset are to look into different features to observe their relationship, from that plot a multiple linear regression based on several features of individual such as age, physical/family condition and location against their existing medical expense to be used for predicting future medical expenses of individuals that help medical insurance to make decision on charging the premium.

After exploring and cleaning data, using clustering and classification help us to predict and evaluate data. The model has shown good performance with 100% sensitivity. The hyperparameter tuning was done at the end to check the difference and have seen minor difference in the model performance. The ‘smoker’ variable is one of the variables which play the most important in the model that did not include the age variable and the model that included the age variable has shown that the ‘age’ variable was most important.

The confusion matrix has shown the various performance metrics for both knn and kknn classifier. The comparison reveals that Highest Sensitivity and Highest Accuracy both were seen in the kknn classifier.

The kknn classifier has great accuracy and sensitivity, and decreased possible confusion, it can be concluded that the exercise has ended with a better performing model in comparison to the models created. The model was successful in fulfilling the goals of the project. It will make a decision safe enough for the organizations to predict and organize their appointments as necessary. It would be interesting to see the results with a decision tree model or even clustering techniques in case a re-visit is planned with this dataset.