Homework 4

Elena Kharitonova

10/11/2020

1. Build a glm in R to classifier individuals as either Male or Female based on their weight and height. What is the accuracy of the model?

First the data needs to be loaded.

```
person = read.csv("500_Person_Gender_Height_Weight_Index.csv")
```

Next, we will choose 70% of the data to be used for training the model, 15% of the data for validating the model and tuning the parameters, and 15% of the data for testing the model. Upon inspection, the data in the table appears to be reported in a random order, so we will assign the first 350 entries to the training data set, the next 75 entries to the validation data set, and the last 75 entries to the testing data set.

```
set = c(rep("Train", 350), rep("Validate", 75), rep ("Test", 75))
person = person %>% mutate(set = set)
```

Next, we need to transform the gender variable into a set of 0's and 1's, so that we can use it in our model. Thus, female will take on the value of 1, and male will take on the value of female

```
female = person$Gender == "Female"
person = person %>% mutate(female = female)
```

Next, we build the generalized linear model using the training data set. The summary of the model is displayed below

```
model.glm <- glm(female ~ Height + Weight,
                 person %>% filter(set == "Train"),
                 family="binomial")
summary(model.glm)
```

```
##
## Call:
## glm(formula = female ~ Height + Weight, family = "binomial",
##     data = person %>% filter(set == "Train"))
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.230  -1.172  -1.119   1.179   1.234
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.538180   1.147679  -0.469    0.639
## Height       0.002230   0.006499   0.343    0.731
## Weight       0.001388   0.003366   0.412    0.680
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 485.19  on 349  degrees of freedom
## Residual deviance: 484.89  on 347  degrees of freedom
## AIC: 490.89
##
## Number of Fisher Scoring iterations: 3
```

In order to test the accuracy of our model, we first need to apply our model to the non-training data. Since we do not do any model selection, we will test the data on both the validating and testing data set.

```
test <- person %>% filter(set != "Train");
test$female.p <- predict(model.glm, test, type="response")
```

The accuracy of the model is displayed below. (It is the value on the left). This is compared to the accuracy simplest model possible, which guesses male for all of the data (this is the value on the right.

```
glm.accur = c(sum((test$female.p>0.5)==test$female)/nrow(test),sum(FALSE==test$female)/nrow(tes
t))
data.frame("Glm_Accuracy" = glm.accur[1], "Simplest_Model_Accuracy" = glm.accur[2])
```

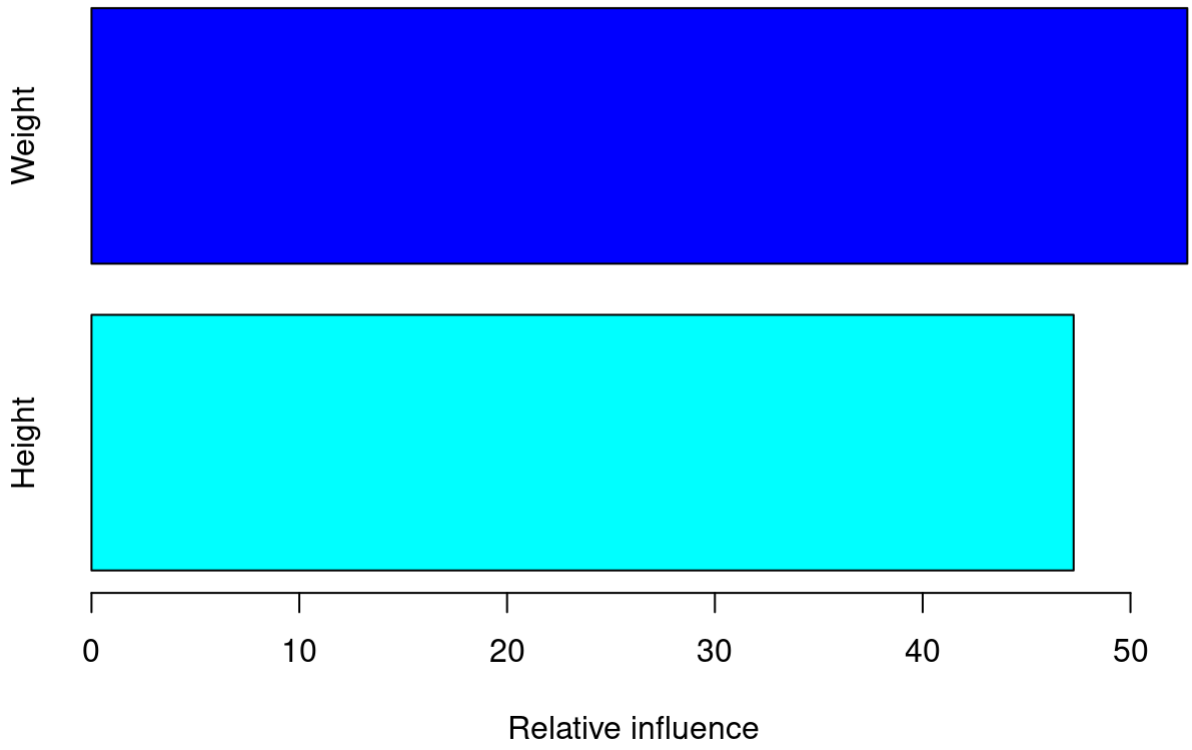
```
##   Glm_Accuracy Simplest_Model_Accuracy
## 1    0.4866667                   0.46
```

Thus our generalized linear model is only slightly superior to the simplest model possible. In fact, if we chose a model that always guessed female, it would have an accuracy of 54%, which would be more accurate than the generalized linear model created.

2. Use the ‘gbm’ package to train a similar model. Don’t worry about hyper parameter tuning for now. What is the accuracy of the model?

Since the data has already been prepared in step 1 (set has been assigned and gender has been changed to a value of 0's and 1's), this means that we are ready to create a gbm model. The summary of the model is displayed below

```
model.gbm <- gbm(female ~ Height + Weight,
                 person %>% filter(set == "Train"),
                 n.trees = 100,
                 interaction.depth = 2,
                 shrinkage=0.1, distribution = "bernoulli")
summary(model.gbm)
```




```
##           var  rel.inf
## Weight Weight 52.73511
## Height Height 47.26489
```

In order to test the accuracy of our model, we first need to apply our model to the non-training data. Since we do not do any model selection, we will test the data on both the validating and testing data set.

```
test$female.p1 <- predict(model.gbm, test, type="response")
```

```
## Using 100 trees...
```

The accuracy of the model is displayed below. (It is the value on the left). This is compared to the accuracy simplest model possible, which guesses male for all of the data (this is the value on the right.

```
gbm.accur = c(sum((test$female.p1>0.5)==test$female)/nrow(test),sum(FALSE==test$female)/nrow(tes
t))
data.frame("GBM_Accuracy" = gbm.accur[1], "Simplest_Model_Accuracy" = gbm.accur[2])
```

```
##   GBM_Accuracy Simplest_Model_Accuracy
## 1    0.5066667                   0.46
```

Our GBM model has a higher accuracy than the generalized linear model made in question 1 and the simplest model that always guesses male.

3. Filter the data set so that it contains only 50 Male examples. Create a new model for this data set. What is the F1 Score of the model?

First we need to make a data set that contains all the females, but only 50 males.

```
fem = filter(person, female == 1)
male = filter(person, female == 0)

## Choose 50 males at random
male = male[sample(nrow(male), 50), ]
data2 = rbind(fem, male)
```

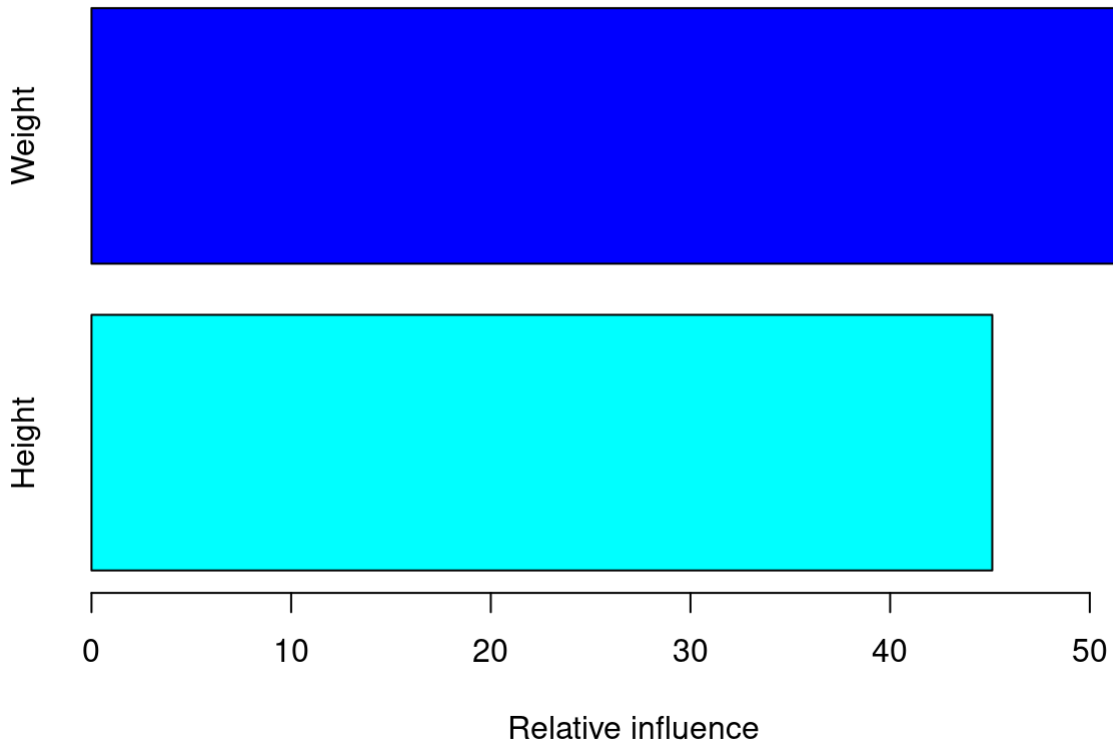
Next, we will choose 70% of the data to be used for training the model, 15% of the data for validating the model and tuning the parameters, and 15% of the data for testing the model. First we will randomize the order of the data in the data frame, then we will assign the first 213 entries to the training data set, the next 46 entries to the validation data set, and the last 46 entries to the testing data set.

```
set = c(rep("Train", 213), rep("Validate", 46), rep ("Test", 46))
## Randomize order of data frame
data2 = data2[sample(nrow(data2), 305), ]

data2 = data2 %>% mutate(set = set)
```

Since the gbm model was more accurate for the original data set, a gbm model will be used again on this data set. The summary of the model is displayed below.

```
model2.gbm <- gbm(female ~ Height + Weight,
                  data2 %>% filter(set == "Train"),
                  n.trees = 100,
                  interaction.depth = 2,
                  shrinkage=0.1, distribution = "bernoulli")
summary(model2.gbm)
```




```
##           var  rel.inf
## Weight Weight 54.88415
## Height Height 45.11585
```

In order to calculate the F1 score, we first need to apply our model to the validation data.

```
validate <- data2 %>% filter(set == "Validate");
pred <- predict(model2.gbm, validate, type="response");
```

```
## Using 100 trees...
```

```
pred0 = pred > 0.5
```

Now the F1-score is obtained. Let tp be the true positive rate, let fn be the false negative rate, and let fp be the false positive rate.

$$F_1 = \frac{tp}{tp + \frac{1}{2}(fp + fn)}$$

```
tp = mean(pred0[validate$female])
fp = mean(pred0[!validate$female])
fn = mean(1 - pred0[!validate$female])

f1 = tp/(tp+0.5*(fp+fn))
f1
```

```
## [1] 0.6666667
```

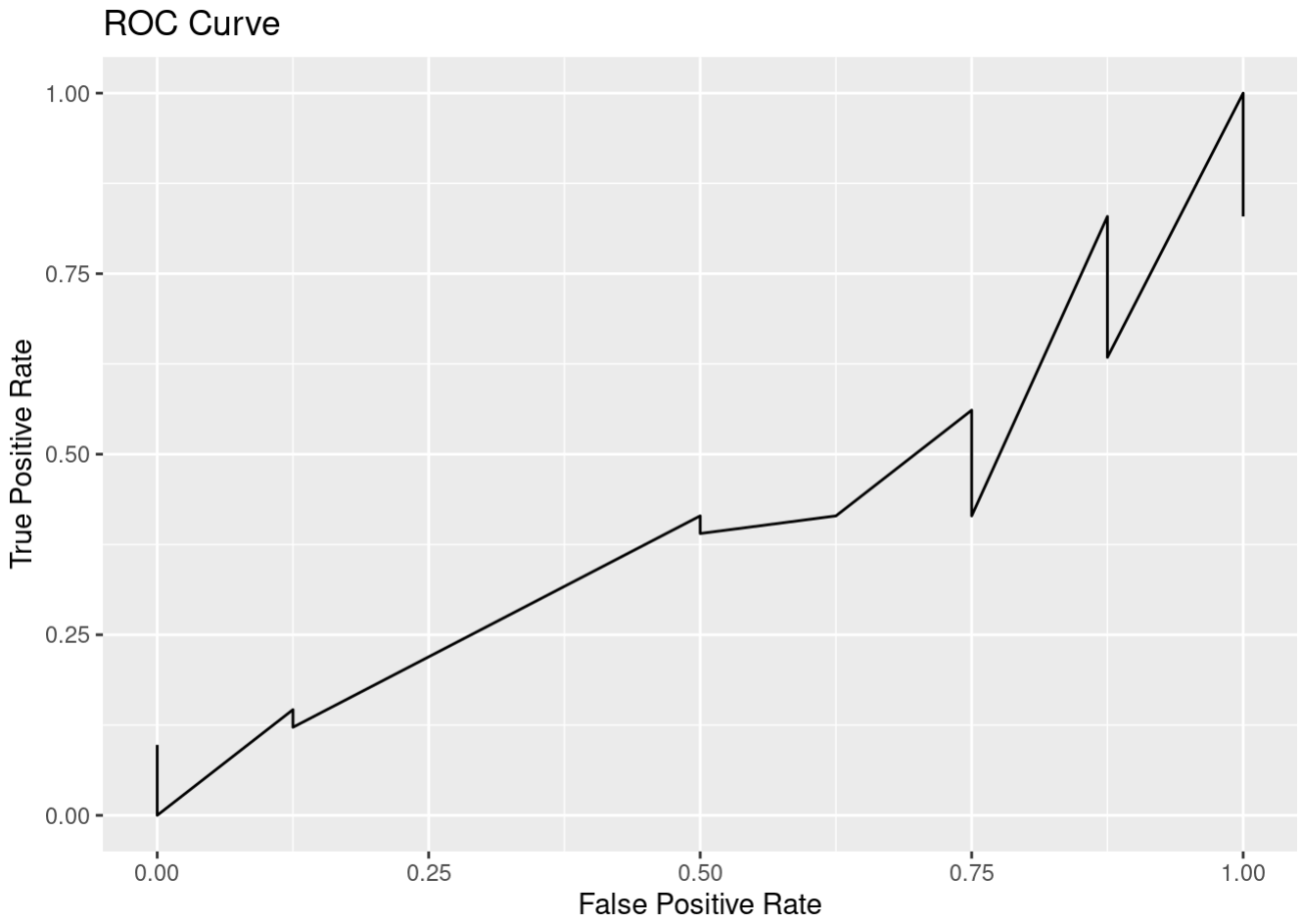
So the f1 score is displayed above. The value makes sense as since there are so few males, our model guesses female for nearly all the subjects.

Therefore the true positive and false positive rates are close to 1 and the true negative and false negative rates are close to 0, which is why the F1 score is 0.667 or 2/3.

4. For the model in the previous example plot an ROC curve. What does this ROC curve mean?

```
roc <- do.call(rbind, Map(function(threshold){
    p <- pred > threshold;
    tp <- sum(p[validate$female])/sum(validate$female);
    fp <- sum(p[!validate$female])/sum(!validate$female);
    tibble(threshold=threshold,
           tp=tp,
           fp=fp)
},seq(100)/100))

ggplot(roc, aes(fp,tp)) + geom_line() + xlim(0,1) + ylim(0,1) +
    labs(title="ROC Curve",x="False Positive Rate",y="True Positive Rate")
```



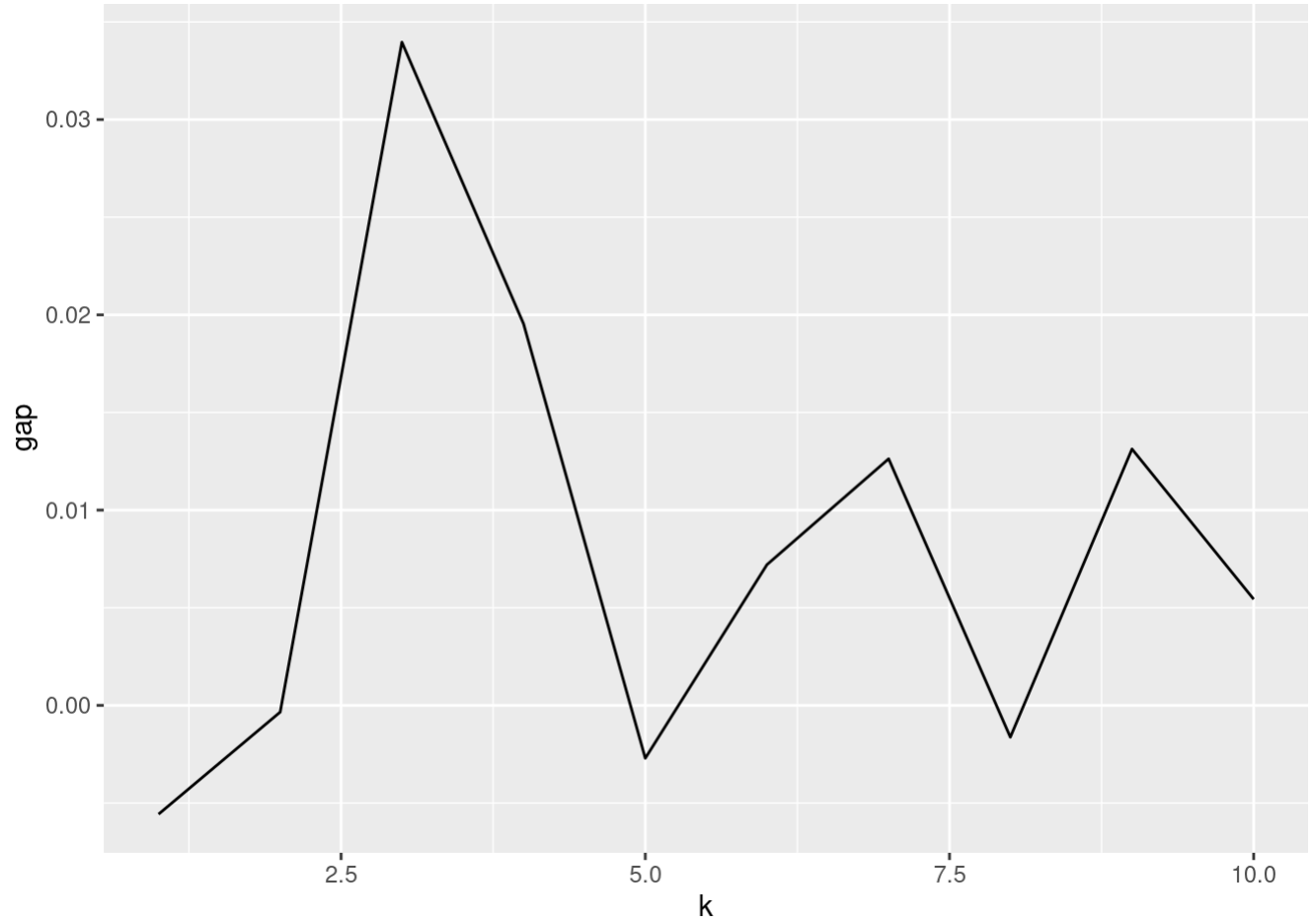
This ROC curve generated is primarily linear as opposed to a tilted concave down curve. This tells us that the area under the curve is close to 0.5. This is not an optimal model as when there is more area under the curve it means our model has better performance. When the area is 0.5, this means the model cannot distinguish between males and females. This is not surprising as our data set is not only strongly biased towards females, but was also generated by randomly assigning height and weight, so there is no true difference between males and females in the data set. Therefore it makes sense that the model we generated is similar to guessing gender at random.

5. Using K-Means, cluster the same data set. Can you identify the clusters with the known labels? Provide an interpretation of this result.

This analysis is done using the original data set. First, we determine what the optimal number of clusters is using a gap plot

```
results <- clusGap(person %>%
                   select(Height, Weight, female),
                   kmeans,
                   K.max = 10,
                   B = 500);
```

```
ggplot(results$Tab %>% as_tibble() %>%
       mutate(k=seq(nrow(.))), aes(k,gap)) + geom_line();
```



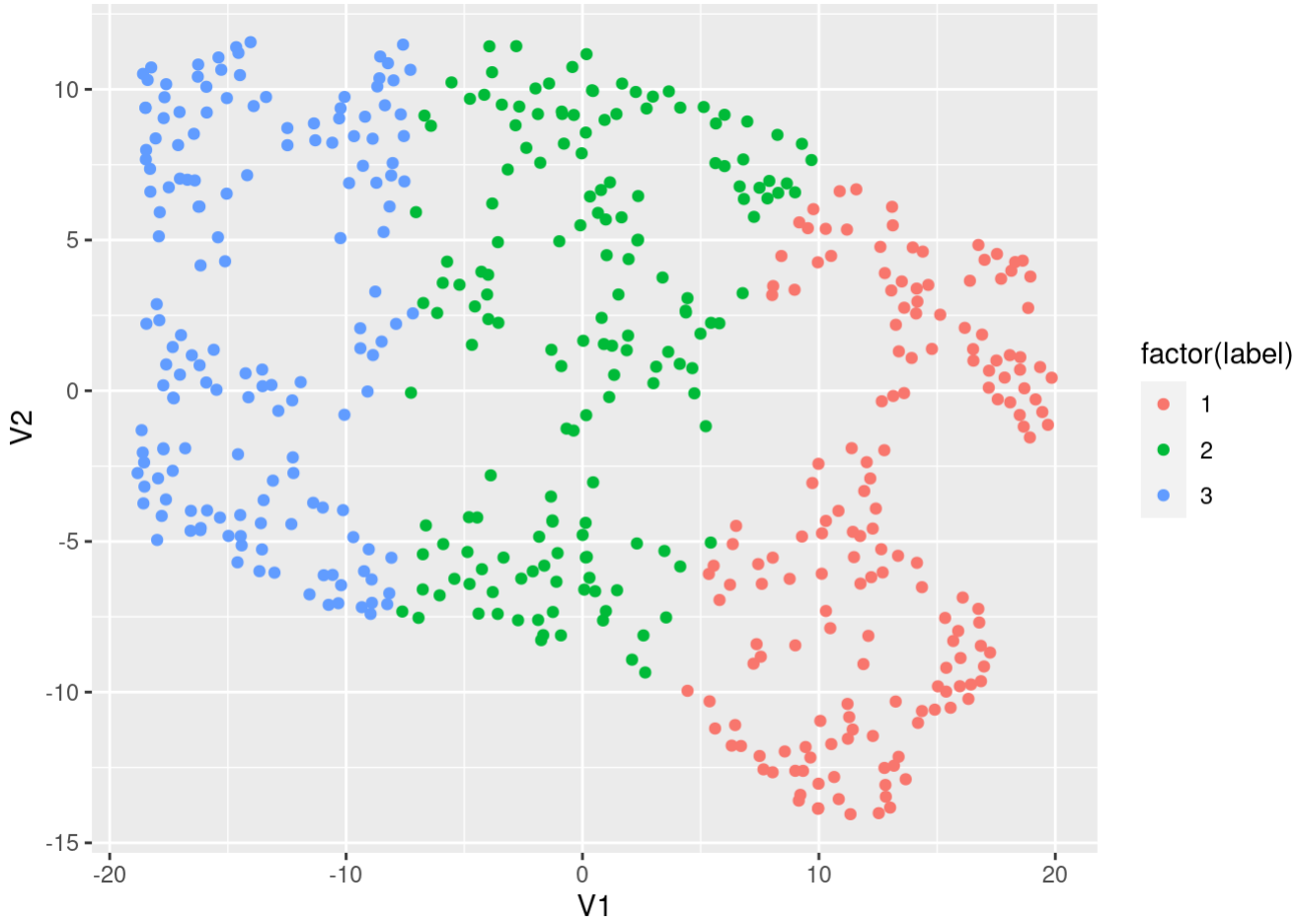
The peak occurs at k = 3, so three clusters will be used.

```
cc<-kmeans(person[,c(2,3,6)],3)

data3 = person[,c(2,3,6)]
xy = cc$cluster
xz = duplicated(data3)
xy = xy[!xz]

fit <- Rtsne(person[!duplicated(person[,c(2,3,6)]),c(2,3,6)], dims = 2)

ggplot(fit$Y %>% as.data.frame() %>% as_tibble() %>% mutate(label=xy),aes(V1,V2)) +
    geom_point(aes(color=factor(label)))
```



The centers are

```
cc$centers
```

```
##     Height   Weight    female
## 1 168.8214 143.2976 0.5119048
## 2 172.6707 105.6287 0.5029940
## 3 168.3273  68.4000 0.5151515
```

The weights differ a lot more than the heights between the clusters. This means that one cluster is for those that are a lighter weight (label 3). Another cluster is of those that are average weight (label 2) and the last cluster is those that are of a heavier height (label 1). Almost all labels have a 50% female ratio, which means that the gender cannot be identified by clustering. This is not surprising as the weight and height distributions were generated randomly, so there should be no significant difference between the genders.