

COVID Project

Xinxin Luo

4/18/2020

Load the data and Explore the data

```
data <- read.csv("Book2.csv") # load data
str(data) # see all columns type and values
```

```
## 'data.frame': 3397 obs. of 24 variables:
## $ id : int 1 2 3 4 5 6 7 8 9 10 ...
## $ case_in_country : int NA NA NA NA NA NA NA NA NA NA NA ...
## $ reporting.date : Factor w/ 57 levels "1/13/2020","1/15/2020",...: 4 4 5 5 5 5 5 5 5 5 ...
## $ X : logi NA NA NA NA NA NA NA ...
## $ summary : Factor w/ 2103 levels "confirmed COVID-19 pneumonia patient No.11 in Tianjin: femal
e, 55, symptom onset on 01/23/2020, hospitalized on"| __truncated__,...: 67 65 51 1977 1978 52 68 2019 2020 2023
...
## $ location : Factor w/ 241 levels "", "Aargau", "Afghanistan",...: 189 186 239 207 207 38 191 23 23
23 ...
## $ country : Factor w/ 39 levels "", "Afghanistan",...: 10 10 10 10 10 10 10 10 10 10 ...
## $ gender : Factor w/ 4 levels "", "20", "female",...: 4 3 4 3 4 3 4 4 4 4 ...
## $ age : num 66 56 46 60 58 44 34 37 39 56 ...
## $ symptom_onset : Factor w/ 74 levels "", "1/10/2020",...: 23 7 26 NA NA 7 3 6 31 8 ...
## $ If_onset_approximated : int 0 0 0 NA NA 0 0 0 0 0 ...
## $ hosp_visit_date : Factor w/ 71 levels "", "1/1/2020",...: 4 8 10 12 7 NA NA 13 7 13 ...
## $ international_traveler : int NA NA NA NA NA NA NA NA NA NA NA ...
## $ domestic_traveler : int NA NA NA NA NA NA NA NA NA NA NA ...
## $ exposure_start : Factor w/ 47 levels "", "1/10/2020",...: 29 NA NA NA NA NA NA 2 21 24 ...
## $ exposure_end : Factor w/ 64 levels "", "1/10/2020",...: 26 4 23 NA NA NA NA 3 26 8 ...
## $ traveler : int NA NA NA NA NA NA NA NA NA NA NA ...
## $ visiting.Wuhan : int 1 0 0 1 0 0 0 1 1 1 ...
## $ from.Wuhan : int 0 1 1 0 0 1 1 0 0 0 ...
## $ death : Factor w/ 25 levels "", "0", "1", "2/1/2020",...: 2 2 2 2 2 2 2 2 2 2 ...
```

```
## $ recovered      : Factor w/ 37 levels "", "0", "1", "1/15/2020", ...: 2 2 2 2 2 2 2 2 2 2 ...
## $ symptom        : Factor w/ 138 levels "", "back pain, fever, runny nose", ...: 1 1 1 1 1 1 1 1 1 1 ...
## $ source         : Factor w/ 110 levels "", "1Point3Acres", ...: 76 68 36 108 108 17 109 10 10 10 ...
## $ link           : Factor w/ 534 levels "", "http://behdasht.gov.ir/news/%DA%A9%D8%B1%D9%88%D9%86%D8%A7
+%D9%88%DB%8C%D8%B1%D9%88%D8%B3/199807/%D8%AF%D8%B1+%|" __truncated__, ...: 6 513 13 43 43 7 44 5 5 5 ...
```

```
data[data==""] <- NA #treat all blank values as missing elements of the data matrix
```

Find correlation between age, and incubation (derived)

```
data$incubation <- as.numeric(
  as.Date(as.character(data$symptom_onset), format = "%m/%d/%Y") - as.Date(as.character(data$exposure_start), format = "%m/%d/%Y")
) # derive variable incubation = symptom_onset date - exposure_start date
```

Normalized the numeric variables

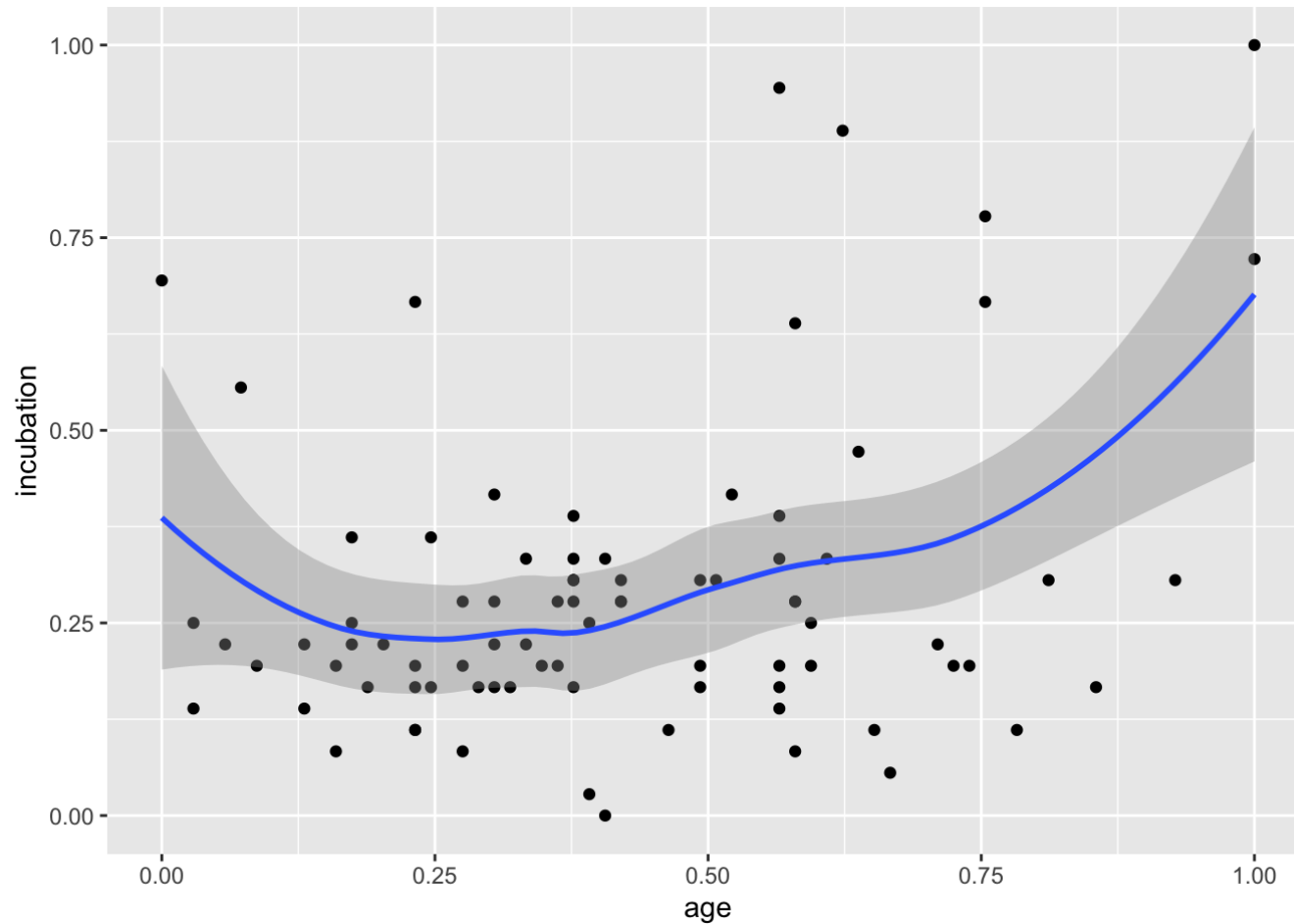
```
normalize <- function(x){
  return((x-min(x))/(max(x)-min(x))) } # normalize function created
data1 <- data[,c(9,25)] #age and incubation selected
data1 <- na.omit(data1)
data1_n <- as.data.frame(lapply(data1, normalize))
summary(data1_n) # check if it is normalized
```

```
##      age      incubation
## Min.   :0.0000   Min.    :0.0000
## 1st Qu.:0.2464   1st Qu.:0.1667
## Median :0.3768   Median :0.2222
## Mean   :0.4222   Mean    :0.2898
## 3rd Qu.:0.5797   3rd Qu.:0.3333
## Max.   :1.0000   Max.    :1.0000
```

Find correlation between age, and incubation (derived)

```
library(ggplot2)
ggplot(data=data1_n) +
  geom_point(mapping = aes(x=age, y=incubation)) +
  geom_smooth(mapping = aes(x=age, y=incubation))# no obvious correlation between age and incubation period.
```

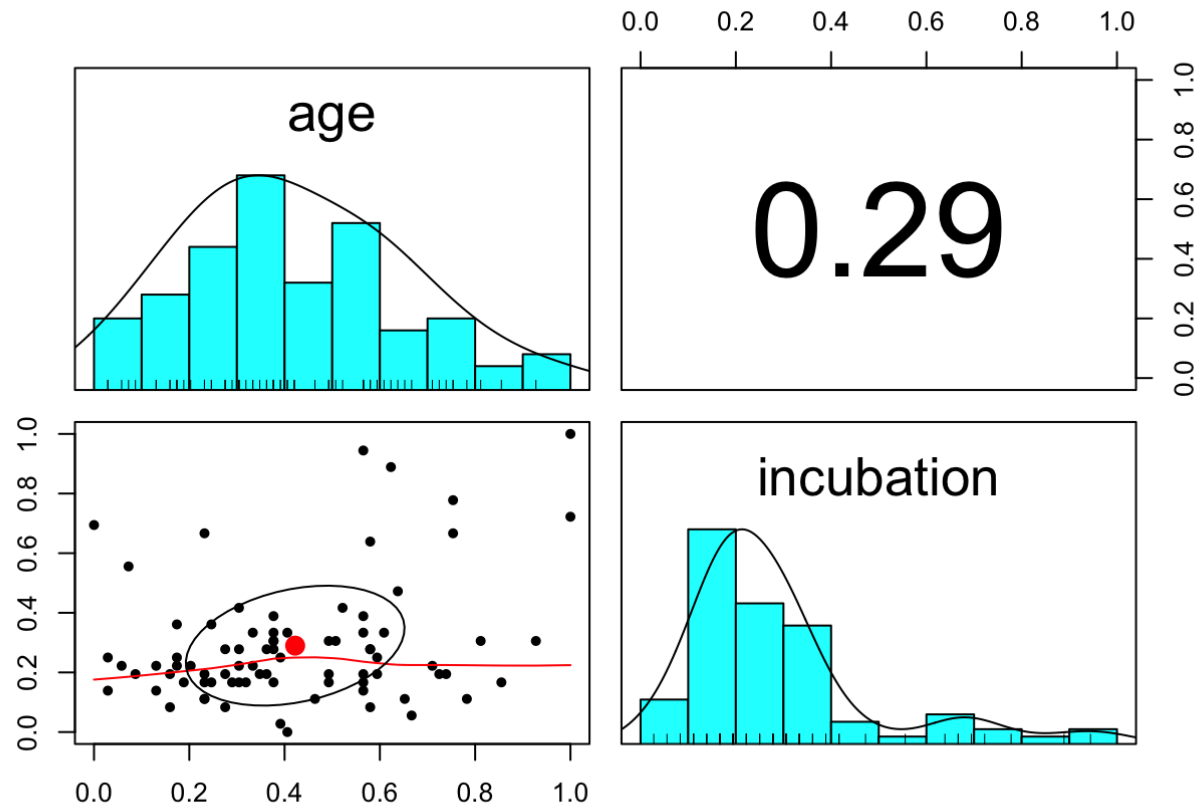
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



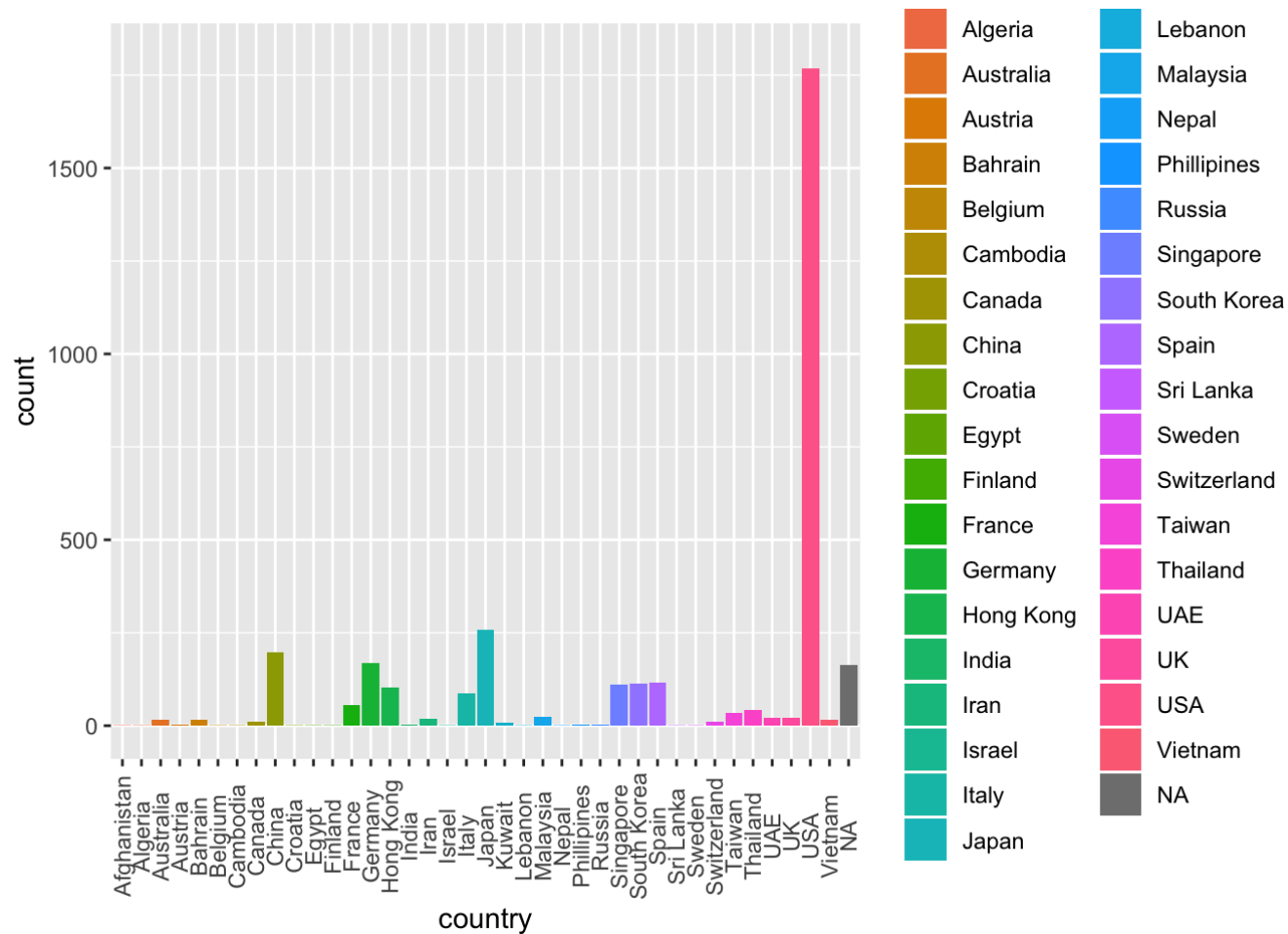
```
cor(data1_n$age,data1_n$incubation,use="complete.obs", method = "pearson") # 0.3 indicates 1 weak positive linear relationships between age and incubation period.
```

```
## [1] 0.2917508
```

```
psych::pairs.panels(data1_n[,c(1,2)]) # correlation visualization between age and incubation.
```



```
ggplot(data, aes(x=country, fill=country)) +
  ylim(0,1800) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



Predict the probability of death using different prediction models: MODEL 1: multiple regression model - lm from stats package

```
library(data.table)
```

```

data$death<- as.numeric(fifelse(data$death==0, 0, 1, na=NA)) # changing death to numeric variable

library(stringr)

for (i in 1:nrow(data)){
  data$n_symptom[i] <- (str_count(data$symptom[i], ',')+1)
} # assuming more number of symptoms there are, more severe the problem is.

data$gender1 <- ifelse(data$gender!="female"&
                      data$gender!="male",
                      NA,data$gender);

for (i in 1:nrow(data)){
  data$gender1[i] <- (data$gender1[i]-3)
}; # 0 is female; 1 is male

lm <-lm(death~case_in_country+country+gender1+age+incubation+n_symptom,data) # creating multiple linear regression model

summary(lm); # model r-squared value is around 15%, which is quite small; only 15% of variances are explained by the model and 5% of standard deviation are explained.

```

```

##
## Call:
## lm(formula = death ~ case_in_country + country + gender1 + age +
##      incubation + n_symptom, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.16615 -0.05438 -0.01293  0.00372  0.82968
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.1949904   0.4215324  -0.463   0.649
## case_in_country -0.0007051   0.0010175  -0.693   0.497
## countryHong Kong  0.2858215   0.3267511   0.875   0.393
## countryJapan    0.2148380   0.3363094   0.639   0.531

```

```
## countryMalaysia      0.2153029  0.4041220  0.533  0.601
## countrySingapore     0.1812980  0.3774848  0.480  0.637
## countrySouth Korea   0.2322287  0.4234381  0.548  0.590
## countryTaiwan         0.1601536  0.3479455  0.460  0.651
## countryUSA            0.1413528  0.3856444  0.367  0.718
## countryVietnam        0.1800077  0.3510145  0.513  0.614
## gender1               0.0765412  0.0853975  0.896  0.382
## age                   -0.0015201  0.0026254 -0.579  0.570
## incubation            0.0004375  0.0062654  0.070  0.945
## n_symptom             0.0335138  0.0725135  0.462  0.649
##
## Residual standard error: 0.2147 on 18 degrees of freedom
## (3365 observations deleted due to missingness)
## Multiple R-squared:  0.1436, Adjusted R-squared:  -0.475
## F-statistic: 0.2321 on 13 and 18 DF,  p-value: 0.9949
```

```
test <- data.frame(case_in_country=210, country="USA",gender1=0, age=70, n_symptom=6, incubation=4) # make a prediction with a novel patient

test$death.p <- predict(lm,test)
test$death.p
```

```
## [1] -0.1052773
```

```
# the likelihood of patient death is close to 0.
```

MODEL 2 K-Means Clustering (to discover any unknown subtypes of patients who have being diagnosed with COVID)

```
# removing unnecessary columns
interests<- data[,c(2,9,25,26)]
# The kmeans() function requires a data frame containing only numeric data and a parameter specifying the desired number of clusters. We will choose only 4 features that represents a patient with COVID-19.

# removing non-existing values
interests <- na.omit(interests)
```

```
# apply as.integer to all vectors
data.interest_int <- as.data.frame(lapply(interests,as.integer))
summary(data.interest_int)
```

```
## case_in_country      age      incubation      n_symptom
## Min.   : 1.00   Min.   :16.00   Min.   : -1.00   Min.   :1.000
## 1st Qu.:10.75   1st Qu.:35.00   1st Qu.: 5.75   1st Qu.:1.000
## Median :17.00   Median :42.50   Median : 8.50   Median :2.000
## Mean   :36.81   Mean   :47.16   Mean   :10.44   Mean   :1.781
## 3rd Qu.:52.25   3rd Qu.:55.50   3rd Qu.:12.00   3rd Qu.:2.000
## Max.   :169.00   Max.   :85.00   Max.   :34.00   Max.   :5.000
```

```
# apply z-score standardization
data_int_z <- as.data.frame(lapply(data.interest_int,scale))

# Set the seed, so results are reproducible
set.seed(1234)

# create index for randomly selecting 75% rows of the original data set
index <- sample.int(n = nrow(interests), size = floor(.75*nrow(interests)), replace = F)

# create training dataset, which accounts for 75% of the data set
data.train_z <- data.interest_int[index, ]
# create testing dataset, which accounts for 25% of the data set
data.test_z  <- data.interest_int[-index, ]

# set specific starting point for k-means algorithm
RNGversion("3.5.2")
```

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```



```
# building clustering model
patient_clusters <- kmeans(data.interest_int,3) # 3 for 3 clusters
```

Evaluating Model 2- Check machine learning performance evaluation for k-means clustering

```
# check size of all 3 groups
patient_clusters$size
```

```
## [1]  8  2 22
```

```
# examine cluster centroids
patient_clusters$centers
```

```
##   case_in_country    age incubation n_symptom
## 1      70.75000 51.50000  14.750000  1.750000
## 2     158.50000 55.00000   6.500000  1.500000
## 3      13.40909 44.86364   9.227273  1.818182
```

Comment: we can see that group 2 and 3 are mostly alders, and group 1 is younger people. For elders has less incubation period reported than young adults, compared in group 1. For patients from countrys that have more cases, the incubation period is much shorter as well. It seems like the numbers of cases encouraged patients to seek for medical attention and increased health care awareness; this could explain why the incubation period is shorter in countries with more cases.

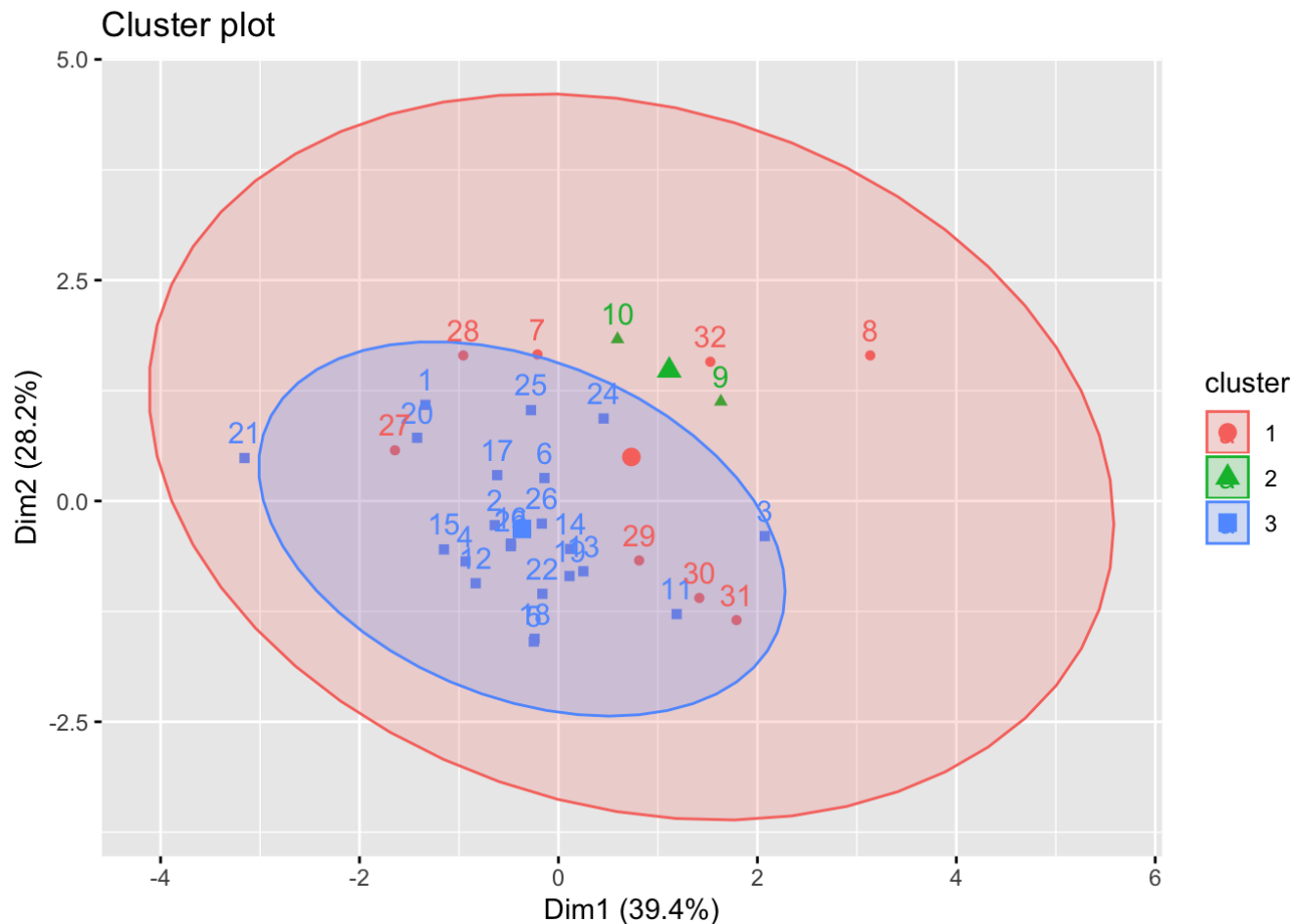
Visualize the clusters

```
# load necessary package
library(factoextra)
```

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

```
# function fviz_cluster can be used to visualize the 3 clusters created previously.  
fviz_cluster(patient_clusters,data.interest_int,ellipse.type = "norm")
```

```
## Too few points to calculate an ellipse
```



Comment: Given the graph, we would have a clear depiction of 3 groups of patients related to COVID-19. Based on these profiles, the doctor could anticipate the days of symptoms onset after the patient is exposed to the high hazardous factors and how long symptoms would.

```
# shortcoming of the model: after missing values being omitted, there are too little data points available for a good evaluation in group 2.
```

Model 3 - kNN algorithm

```
# load necessary packages  
library(caret)
```

```
## Loading required package: lattice
```

```
library(ISLR)
```

```
# removing unnecessary columns
```

```
interests<- data[,c(2,9,19,20,26,27)] # includes columns: case_in_country, age, from.wuhan, death, n_symptom, and gender1.
```

```
# removing non-existing values
```

```
interests <- na.omit(interests)
```

```
# normalize data
```

```
data_n <- as.data.frame(lapply(interests,normalize))
```

```
summary(data_n)# check if it is normalized
```

```
## case_in_country      age      from.Wuhan      death  
## Min.   :0.00000  Min.   :0.0000  Min.   :0.00000  Min.   :0.00000  
## 1st Qu.:0.07843  1st Qu.:0.4505  1st Qu.:0.00000  1st Qu.:0.00000  
## Median :0.32941  Median :0.5604  Median :0.00000  Median :0.00000  
## Mean   :0.37894  Mean   :0.5503  Mean   :0.08182  Mean   :0.02424  
## 3rd Qu.:0.65784  3rd Qu.:0.6703  3rd Qu.:0.00000  3rd Qu.:0.00000  
## Max.   :1.00000  Max.   :1.0000  Max.   :1.00000  Max.   :1.00000  
## n_symptom      gender1  
## Min.   :0.0000  Min.   :0.0000  
## 1st Qu.:0.0000  1st Qu.:0.0000  
## Median :0.2000  Median :1.0000  
## Mean   :0.2103  Mean   :0.5697
```

```
## 3rd Qu.:0.4000 3rd Qu.:1.0000
## Max. :1.0000 Max. :1.0000
```

```
# set seed
set.seed(12378)

# create index for randomly selecting 75% rows of the original data set
index <- sample.int(n = nrow(data_n), size = floor(.75*nrow(data_n)), replace = F)

# create training dataset, which accounts for 75% of the data set
data.train <- data_n[index, ]
# create testing dataset, which accounts for 25% of the data set
data.test <- data_n[-index, ]

# column bind labels to the normalized test and train dataset
data_train_label <- as.data.frame(data.train$death)
data_test_label <- as.data.frame(data.test$death)
names(data_train_label) <- "death"
names(data_test_label) <- "death"
data_train1 <- cbind(data_train_label,data.train)
data_test1 <- cbind(data_test_label,data.test)
```

```
# set up control object which controls the computational nuances of the train function.
ctrl <- trainControl(method="repeatedcv", repeats = 3) #Cross-Validated (10 fold, repeated 3 times)

# training the model
knnFit <- train(death~., data=data_train1,method="knn",trControl=ctrl,preProcess=c("center","scale"),tuneLength=2
)
```

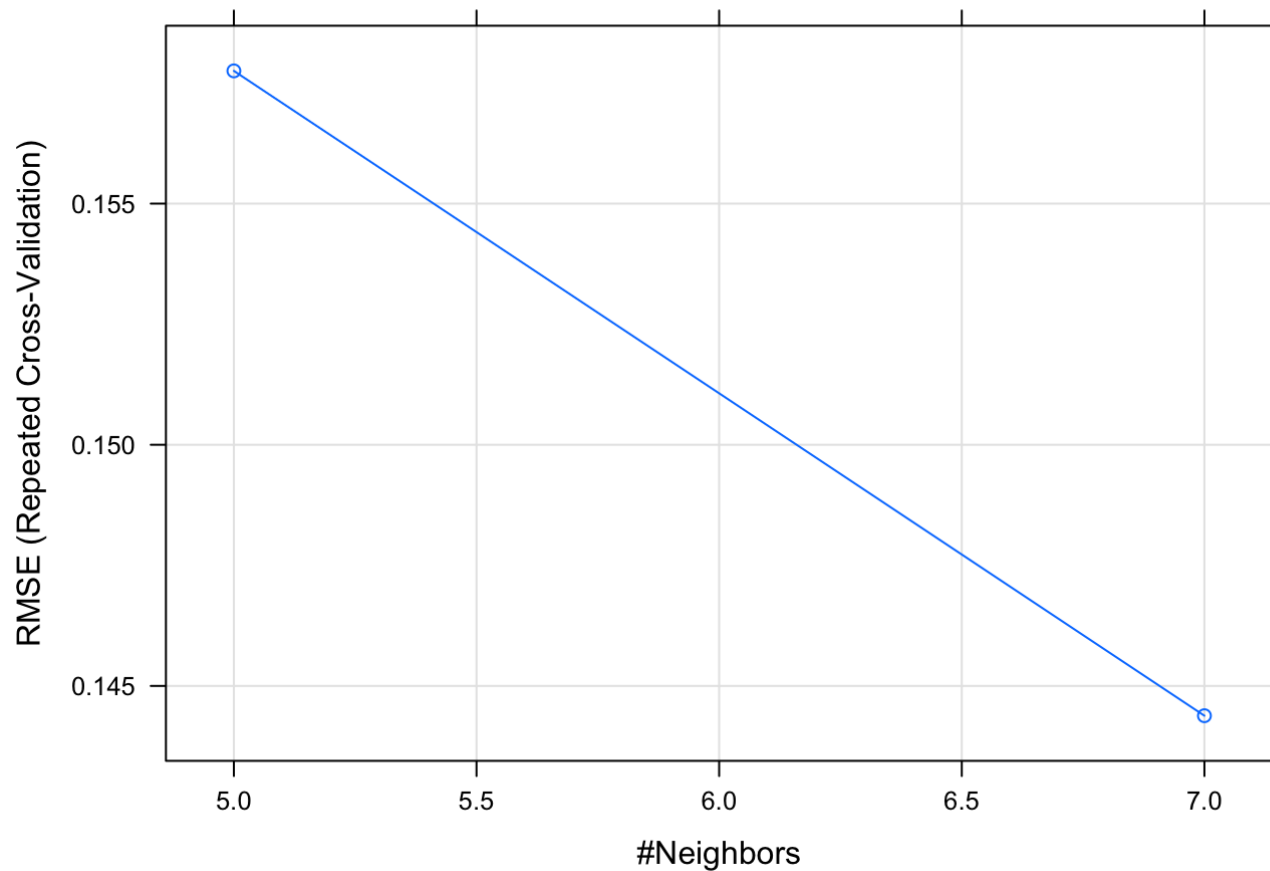
```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :  
## There were missing values in resampled performance measures.
```

```
# output of the knn fit  
knnFit
```

```
## k-Nearest Neighbors  
##  
## 247 samples  
## 5 predictor  
##  
## Pre-processing: centered (5), scaled (5)  
## Resampling: Cross-Validated (10 fold, repeated 3 times)  
## Summary of sample sizes: 223, 223, 222, 222, 223, 222, ...  
## Resampling results across tuning parameters:  
##  
## k  RMSE      Rsquared    MAE  
## 5  0.1577519  0.01190423  0.05233148  
## 7  0.1443826  0.04507728  0.04845139  
##  
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 7.
```

```
# plotting yields number of neighbours vs accuracy (based on repeated cross validation)  
plot(knnFit)
```



```
# predict using testing data
knnPredict <- predict(knnFit,newdata=data_test1)
knnPredict <- as.factor(ifelse(knnPredict>0,1,0))
knnPredict
```

```
## [1] 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [39] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1 0
## [77] 0 0 0 0 1 0 0
## Levels: 0 1
```

```
#data_test1$death <- as.factor(data_test1$death)
data_test1$death
```

[illegible]

```
death_label <- as.factor(data_test1$death)
# use confusion matrix to calculate accuracy for caret algorithm
confusionMatrix(knnPredict,death_label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0    1
##           0 71   1
##           1 11   0
##
##
##           Accuracy : 0.8554
##           95% CI : (0.7611, 0.923)
##           No Information Rate : 0.988
##           P-Value [Acc > NIR] : 1.000000
##
##
##           Kappa : -0.0226
##
##
##           Mcnemar's Test P-Value : 0.009375
##
##
##           Sensitivity : 0.8659
##           Specificity : 0.0000
##           Pos Pred Value : 0.9861
##           Neg Pred Value : 0.0000
##           Prevalence : 0.9880
##           Detection Rate : 0.8554
##           Detection Prevalence : 0.8675
```

```
##      Balanced Accuracy : 0.4329
##
##      'Positive' Class : 0
##
```

```
mean(knnPredict==death_label)
```

```
## [1] 0.8554217
```

```
# comment: the accuracy for caret is 86%.
```

Comparison of models

```
# According to evaluations for each model, the one with the most accurate data is k-NN algorithm, which has an accuracy of 86%.
```