Homework Assignment 3

Data Science and Machine Learning 1 - CEU 2018

Kristof Menyhert

2018-02-11

```
library(data.table)
library(datasets)
library(MASS)
library(ISLR)
library(caret)
library(ggplot2)
library(NbClust)
library(factoextra)

theme_set(theme_bw()) #globally set ggplot theme to black & white
```

1. PCA for supervised learning

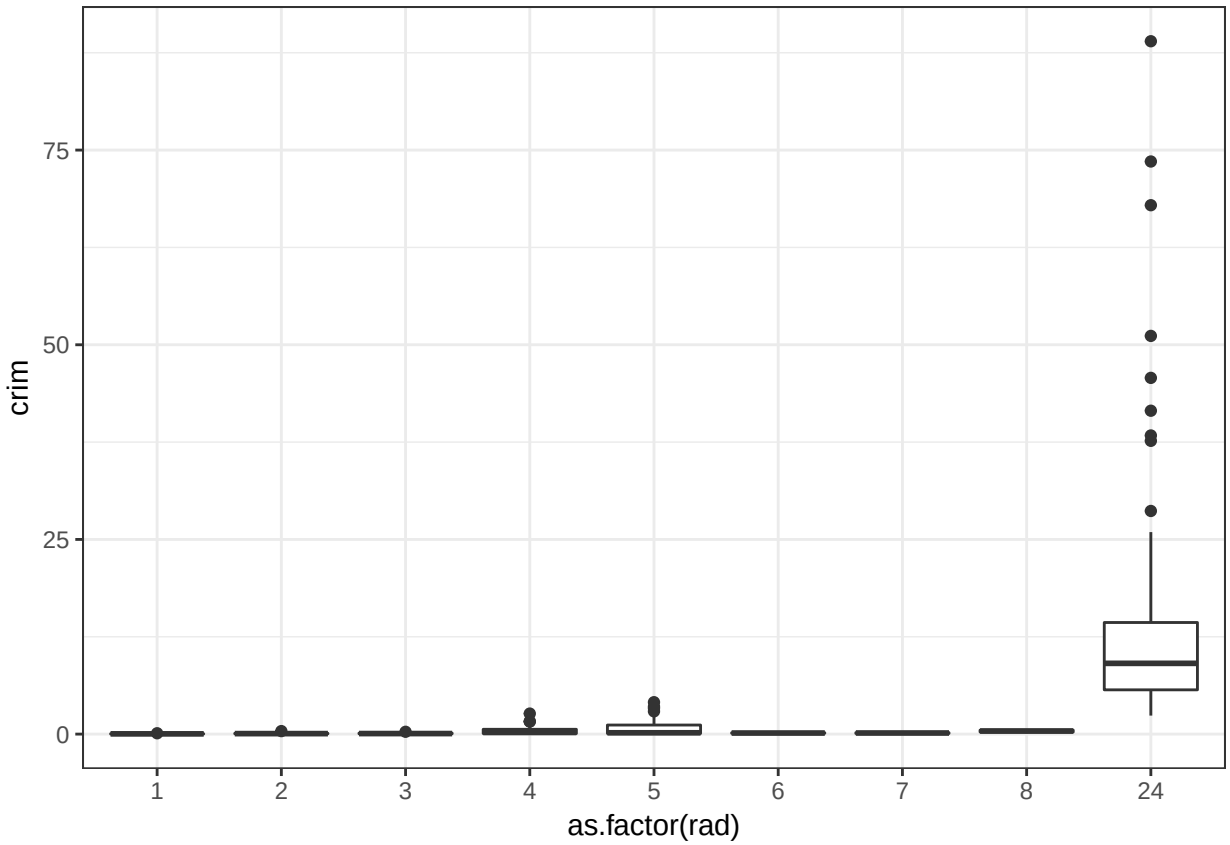
In this problem you are going to analyze the Boston dataset from the MASS package (read more about the data here). The goal will be to predict the variable crim which is the crime rate.

```
data <- data.table(Boston)
```

a) Do a short exploration of data and find possible predictors of the target variable.

VARIABLE: RAD - index of accessibility to radial highways:

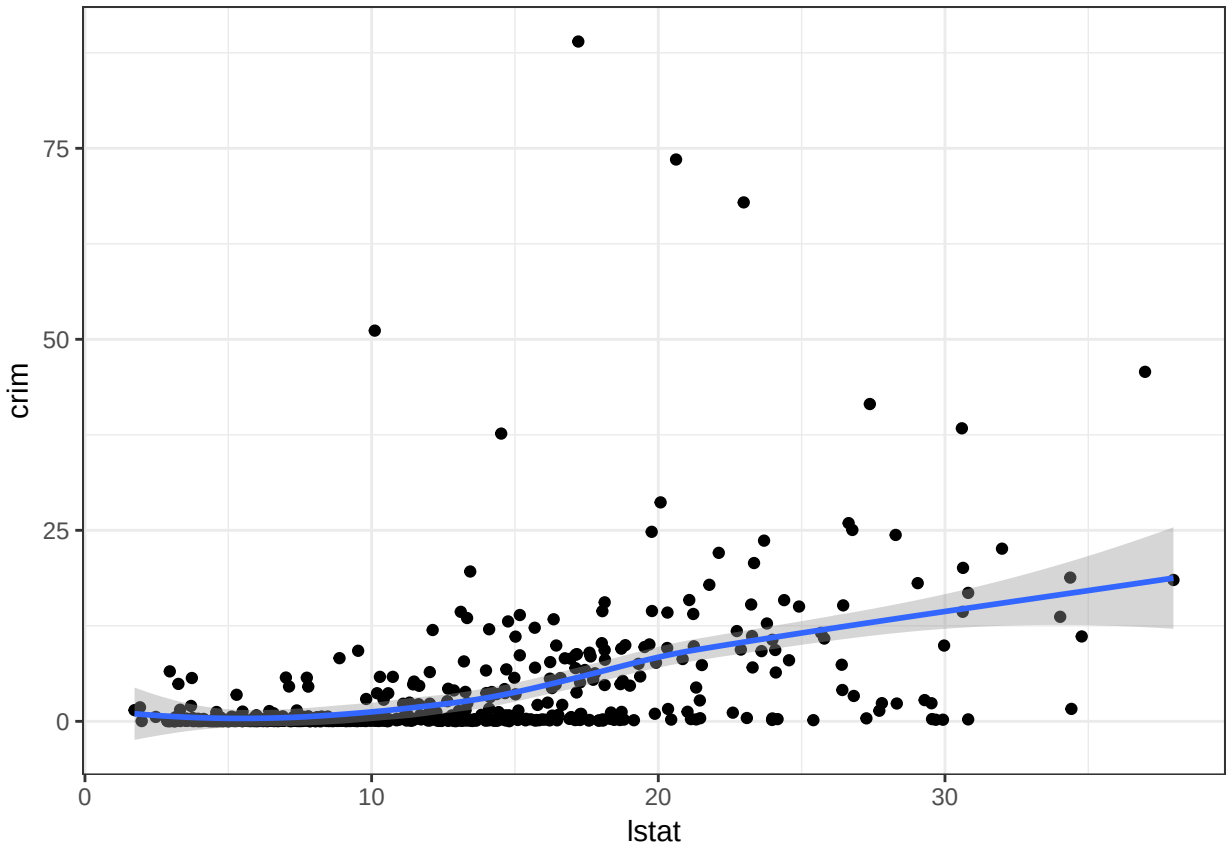
```
ggplot(data, aes(x=as.factor(rad), y=crim)) +
  geom_boxplot()
```



It seems like there is an extreme value, namely rad 24 which is contributing to the per capita crime rate.

VARIABLE: LSTAT - % lower status of the population:

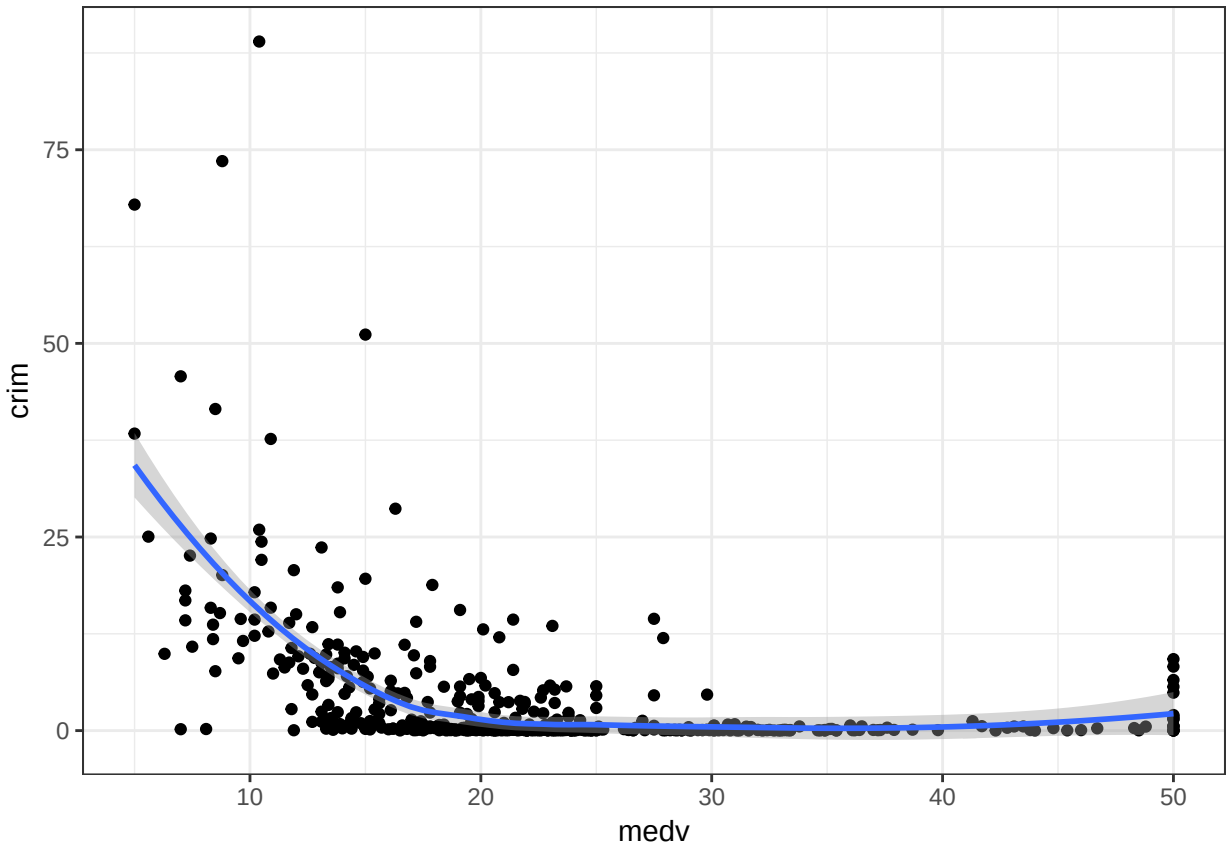
```
ggplot(data, aes(x=lstat, y=crim)) +
  geom_point() +
  geom_smooth()
```



Seems like the higher the ratio of the lower status of the population in a given area indicating higher crime rate per capita on average.

VARIABLE: MEDV - Median value of owner-occupied homes in $1000's:

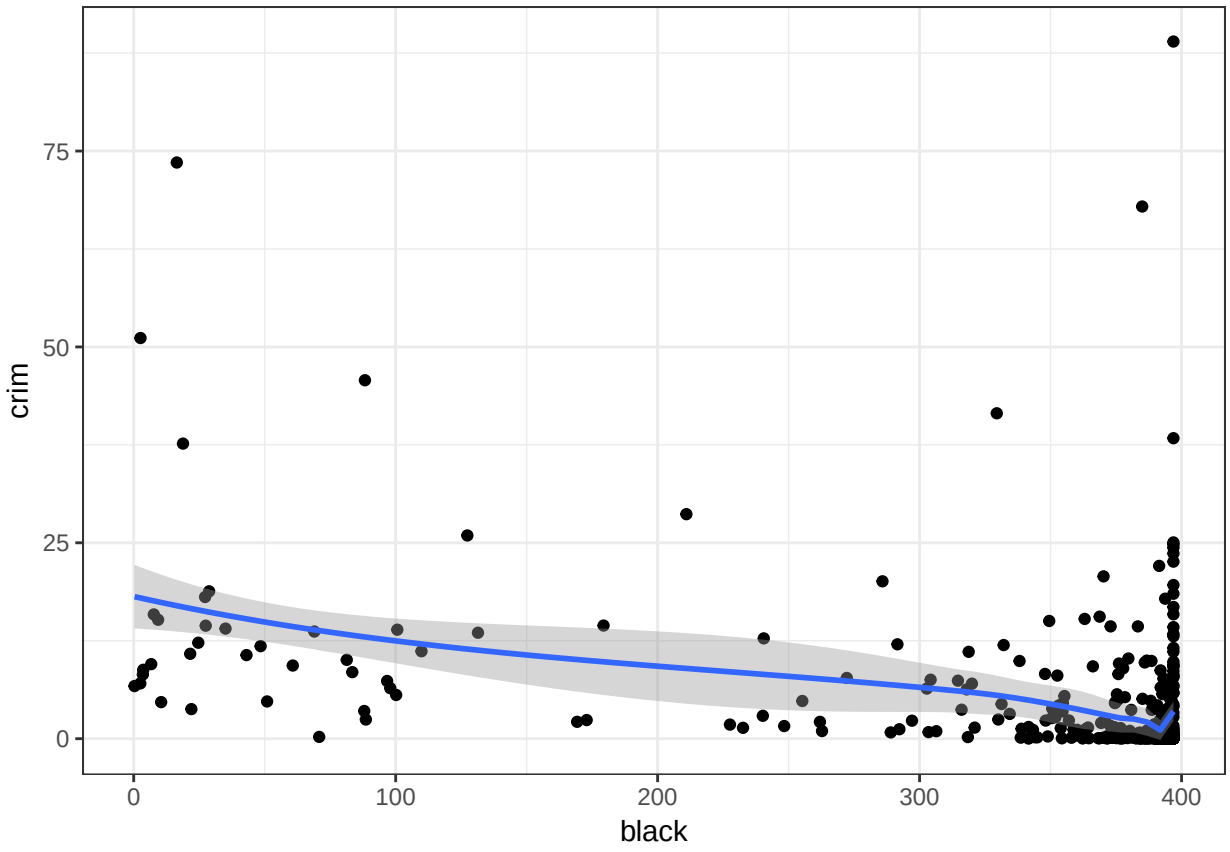
```
ggplot(data, aes(x=medv, y=crim)) +
  geom_point() +
  geom_smooth()
```



Seems like under 20.000$s crime rate per capita is negatively correlated with this variable.

VARIABLE: B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town:

```
ggplot(data, aes(x=black, y=crim)) +
  geom_point() +
  geom_smooth()
```



b) Create a training and a test set of 50%.

Cutting the data into two parts:

```
set.seed(1234)
cut <- createDataPartition(y = data$crim, times = 1, p = 0.5, list = FALSE)

data_train <- data[cut, ]

data_test <- data[-cut, ]

# check the cut
length(data$crim) == (length(data_train$crim) + length(data_test$crim))
```

```
## [1] TRUE
```

c) Use a linear regression to predict crim and use 10-fold cross validation to assess the predictive power.

LM model with all variables:

```
set.seed(1234)
lm_model <- train(crim ~ .,
                  method = "lm",
                  data = data_train,
                  trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale"))
```

```
lm_model
```

```
## Linear Regression
##
## 254 samples
##  13 predictor
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 228, 230, 230, 227, 229, 228, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   6.048258  0.6321251  3.211443
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

d) Try to improve the model by using PCA for dimensionality reduction. Center and scale your variables and use pcr to conduct a search for the optimal number of principal components. Does PCA improve the fit over the simple linear model?

```
tune_grid <- data.frame(ncomp = 1:10)
set.seed(1234)
pcr_fit <- train(crim ~ . ,
                 data = data_train,
                 method = "pcr",
                 trControl = trainControl(method = "cv", number = 10),
                 tuneGrid = tune_grid,
                 preProcess = c("center", "scale"))
pcr_fit
```

```
## Principal Component Analysis
##
## 254 samples
##  13 predictor
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 228, 230, 230, 227, 229, 228, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    1     6.597052  0.5007794  3.933333
##    2     6.594269  0.5023370  3.993859
##    3     6.248335  0.6076207  3.272706
##    4     6.226802  0.6160569  3.217555
##    5     6.245386  0.6134530  3.232879
##    6     6.185813  0.6095310  3.294459
##    7     6.201866  0.6040667  3.348272
##    8     6.067988  0.6289923  3.217254
##    9     6.123378  0.6225883  3.229351
##   10     6.127400  0.6218771  3.177955
##
```

```
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 8.
```

The best model using PCA is almost as good as without using it.

e) Use penalized linear models for the same task. Make sure to include lasso (alpha = 0) to your tune grid. How does the best model compare to that found in d)? Would pre-processing via PCA help this model? (add pca to preProcess). Why do you think the answer can be expected?

```
# lasso model
tune_grid <- expand.grid("alpha" = c(0), # lasso, allows to be 0 ridge just shring to near 0
                         "lambda" = c(0.3, 0.1, 0.01, 0.001, 0.0001))

set.seed(1234)
lasso_fit <- train(crim ~ .,
                   data = data_train,
                   method = "glmnet",
                   preProcess = c("center", "scale", "pca"), # to normailize
                   tuneGrid = tune_grid,
                   trControl = trainControl(method = "cv",
                                            number = 10,
                                            preProcOptions = list(thresh=0.65)))

lasso_fit
```

```
## glmnet
##
## 254 samples
##  13 predictor
##
## Pre-processing: centered (13), scaled (13), principal component
##  signal extraction (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 228, 230, 230, 227, 229, 228, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE      Rsquared   MAE
##   1e-04   6.205868  0.6076207  3.188249
##   1e-03   6.205868  0.6076207  3.188249
##   1e-02   6.205868  0.6076207  3.188249
##   1e-01   6.205868  0.6076207  3.188249
##   3e-01   6.205868  0.6076207  3.188249
##
## Tuning parameter 'alpha' was held constant at a value of 0
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda = 0.3.
```

I tried out several threshold and managed to get almost the same RMSE than without using lasso, the RMSE difference is small with using the threshold = 0.65.

But actually I got a little bit worst result for the best model with PCA.

Inspect how many columns are regured for the 0.65 threshold:

```
pre_process <- preProcess(data_train, method = c("center", "scale", "pca"), thresh = 0.65)
pre_process
```

```
## Created from 254 samples and 14 variables
##
## Pre-processing:
##   - centered (14)
##   - ignored (0)
##   - principal component signal extraction (14)
##   - scaled (14)
##
## PCA needed 3 components to capture 65 percent of the variance
```

PCA needed only 3 variables to capture 65% of the variance. So we managed to catch almost the same variance with only 3 columns.

f) Evaluate your preferred model on the test set.

I am using the lasso model with PCA for predicting the crims variable.

```
data_test$model_prediction <- predict.train(lasso_fit, newdata = data_test)
```

Calculate RMSE:

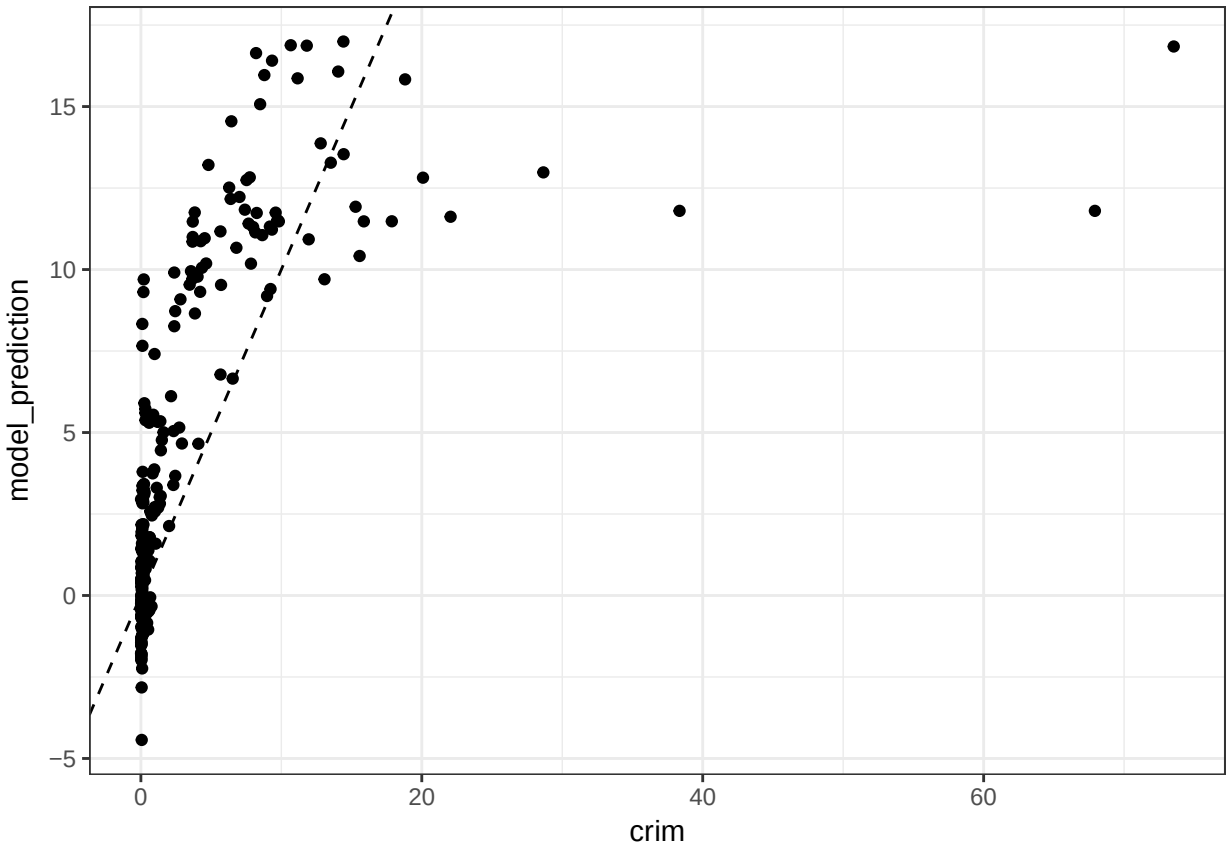
```
RMSE <- function(x, true_x) sqrt(mean((x - true_x)^2))

RMSE(data_test$model_prediction, data_test$crim)
```

```
## [1] 6.320611
```

RMSE is almost the same on the training set compered to the test set. Therefore I consider my model a well fitted one.

See the predicted vs. actual crim on ggplot:

```
ggplot(data_test, aes(x=crim, y=model_prediction)) +
  geom_point() +
  geom_abline(slope = 1, linetype = 2)
```



We get negative values in the predicted crim column which can't be the case. Negative crime rates can't exist.

One additional thing what we can do is to try to predict the log(crim). So I am doing that in the following lines:

```
set.seed(1234)
lasso_fit_log <- train(log(crim) ~ .,
                   data = data_train,
                   method = "glmnet",
                   preProcess = c("center", "scale", "pca"), # to normailize
                   tuneGrid = tune_grid,
                   trControl = trainControl(method = "cv",
                                            number = 10,
                                            preProcOptions = list(thresh=0.65)))

lasso_fit_log
```

```
## glmnet
##
## 254 samples
##  13 predictor
##
## Pre-processing: centered (13), scaled (13), principal component
##  signal extraction (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 228, 230, 230, 227, 229, 228, ...
## Resampling results across tuning parameters:
```

```
##
##   lambda  RMSE       Rsquared   MAE
##   1e-04   0.9001288  0.8424319  0.7425852
##   1e-03   0.9001288  0.8424319  0.7425852
##   1e-02   0.9001288  0.8424319  0.7425852
##   1e-01   0.9001288  0.8424319  0.7425852
##   3e-01   0.9174617  0.8424319  0.7597248
##
## Tuning parameter 'alpha' was held constant at a value of 0
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda = 0.1.
```

Predict to a column:

```
# on the test data:

data_test$model_prediction_log <- predict.train(lasso_fit_log, newdata = data_test)
data_test[, log_crim := log(crim)] # create log crim to evaluate and to plot
```

Evaluate the final model where I predicted the log values:

Define RMSE:

```
RMSE <- function(x, true_x) sqrt(mean((x - true_x)^2))
```

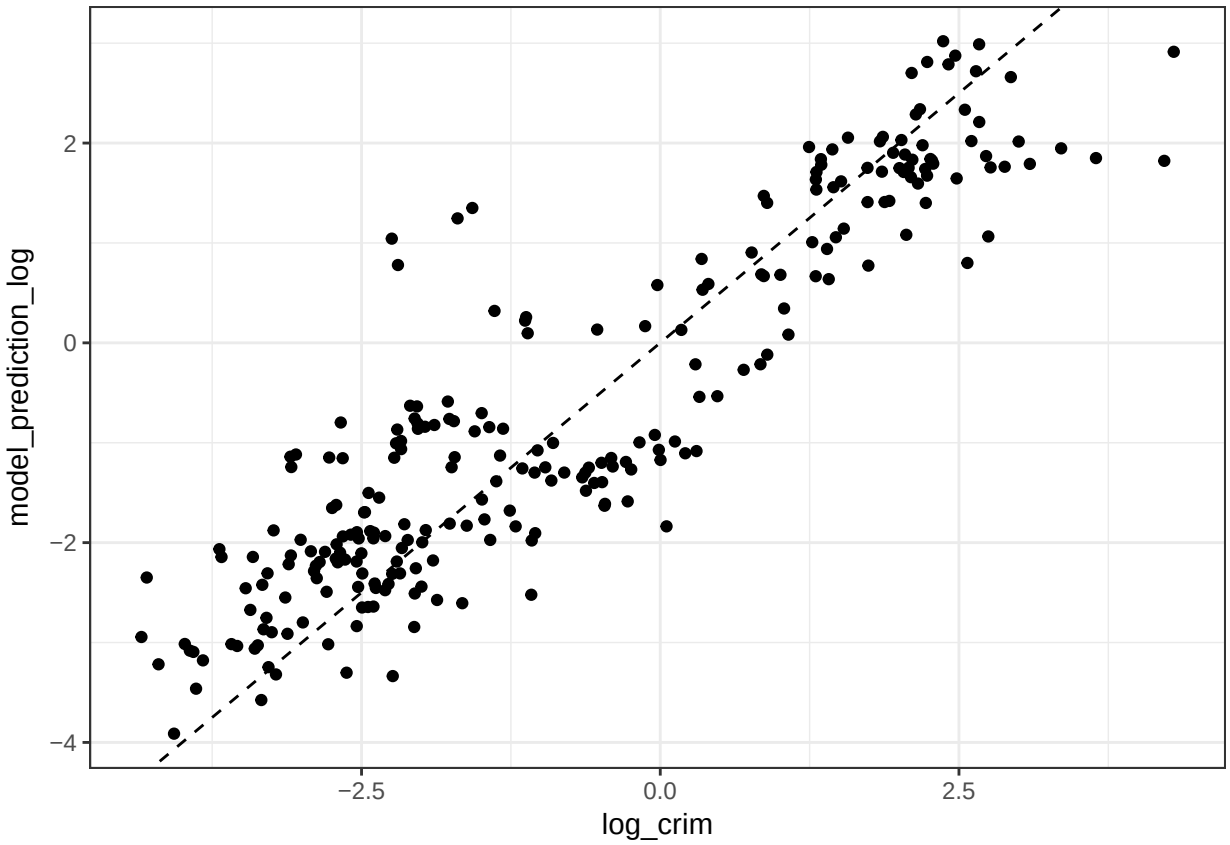
On the test set:

```
RMSE(data_test$model_prediction_log, data_test$log_crim)
```

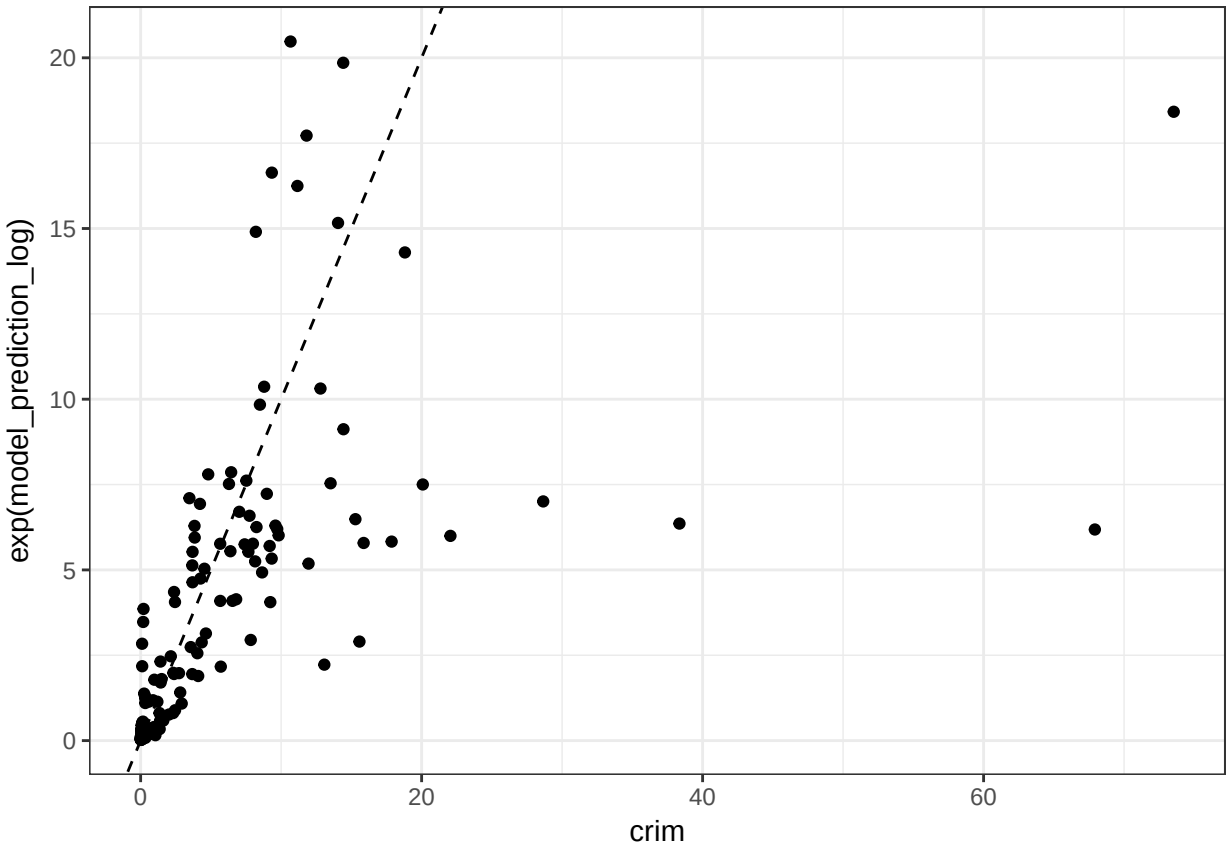
```
## [1] 0.9027618
```

See prediction on a plot:

```
ggplot(data_test, aes(x=log_crim, y=model_prediction_log)) +
  geom_point() +
  geom_abline(slope = 1, linetype = 2)
```




```
#real values:
ggplot(data_test, aes(x=crim, y=exp(model_prediction_log))) +
  geom_point() +
  geom_abline(slope = 1, linetype = 2)
```



I think the log model is a better one (also Rsquared is suggesting that).

Clustering on the USArrests dataset

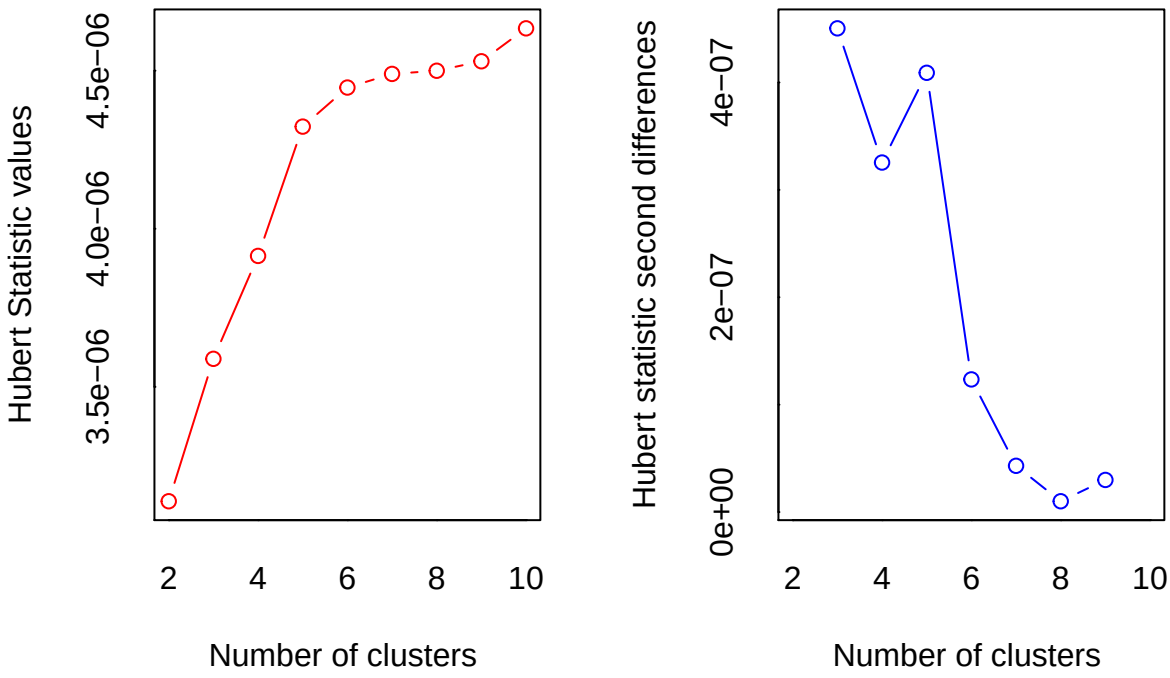
In this problem use the USArrests dataset we used in class. Your task is to apply clustering then make sense of the clusters using the principal components.

Load the data:

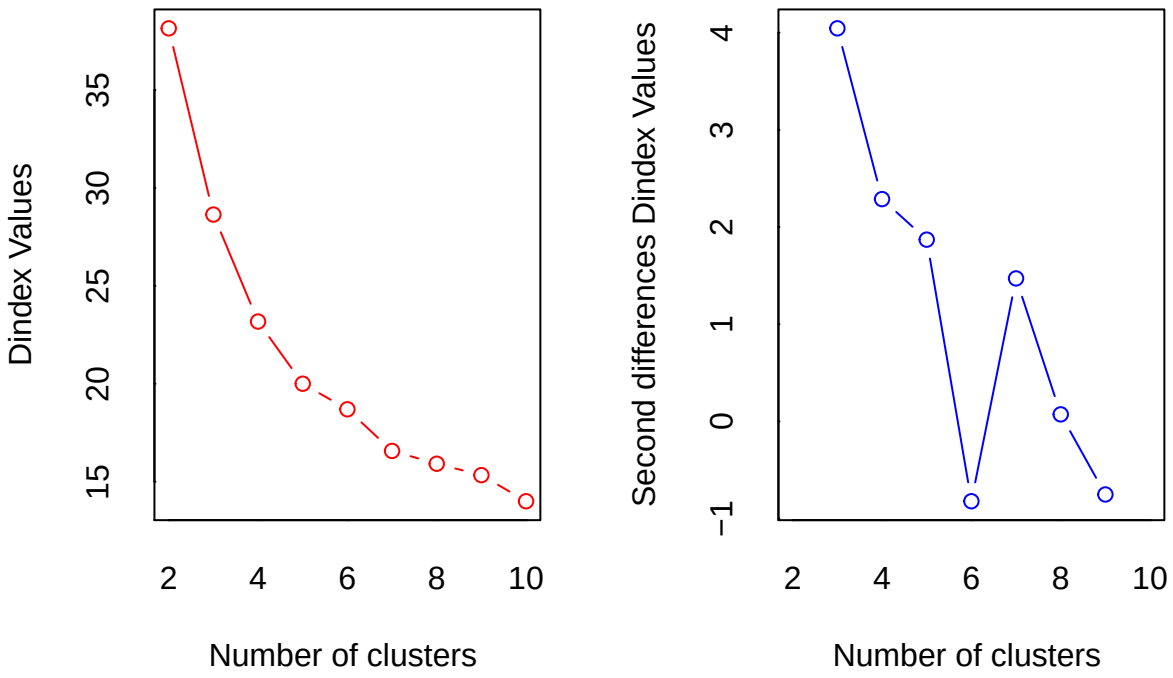
```
data <- data.table(USArrests)
```

a) Determine the optimal number of clusters as indicated by NbClust heuristics.

```
nb <- NbClust(data, method = "kmeans", min.nc = 2, max.nc = 10, index = "all")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

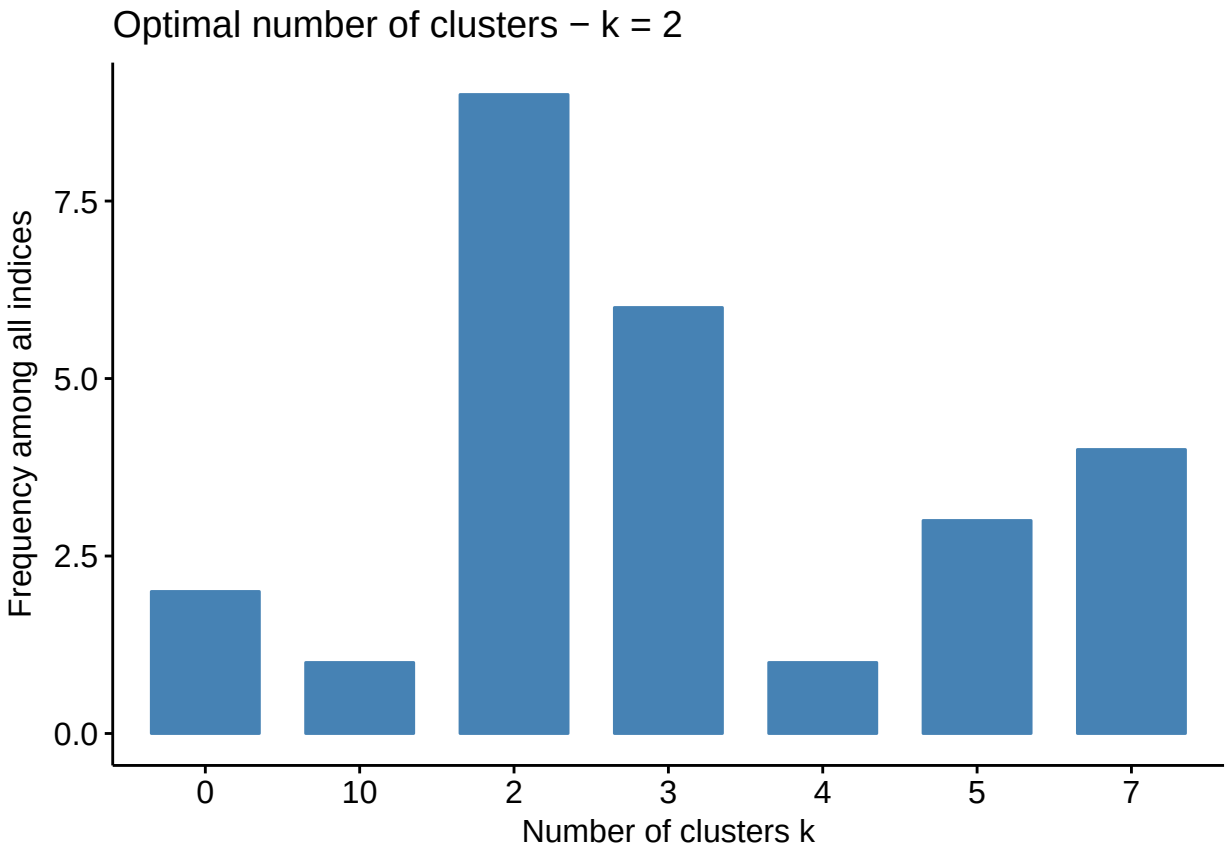



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 9 proposed 2 as the best number of clusters
## * 6 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 3 proposed 5 as the best number of clusters
## * 4 proposed 7 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
fviz_nbclust(nb)
```

```
## Among all indices:
## ==================
## * 2 proposed  0 as the best number of clusters
## * 9 proposed  2 as the best number of clusters
```

```
## * 6 proposed  3 as the best number of clusters
## * 1 proposed  4 as the best number of clusters
## * 3 proposed  5 as the best number of clusters
## * 4 proposed  7 as the best number of clusters
## * 1 proposed  10 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```



Optimal number of clusters based on the method showed above - based on the majority rule - are 2 clusters.

b) Use the k-means method to cluster states using the number of clusters found in a) and anything else that you think that makes sense. Plot observations colored by clusters in the space of urban population and another (crime-related) variable. (See example code from class, use factor(km$cluster) to create a vector of class labels).

```
km <- kmeans(data, centers = 2)
km
```

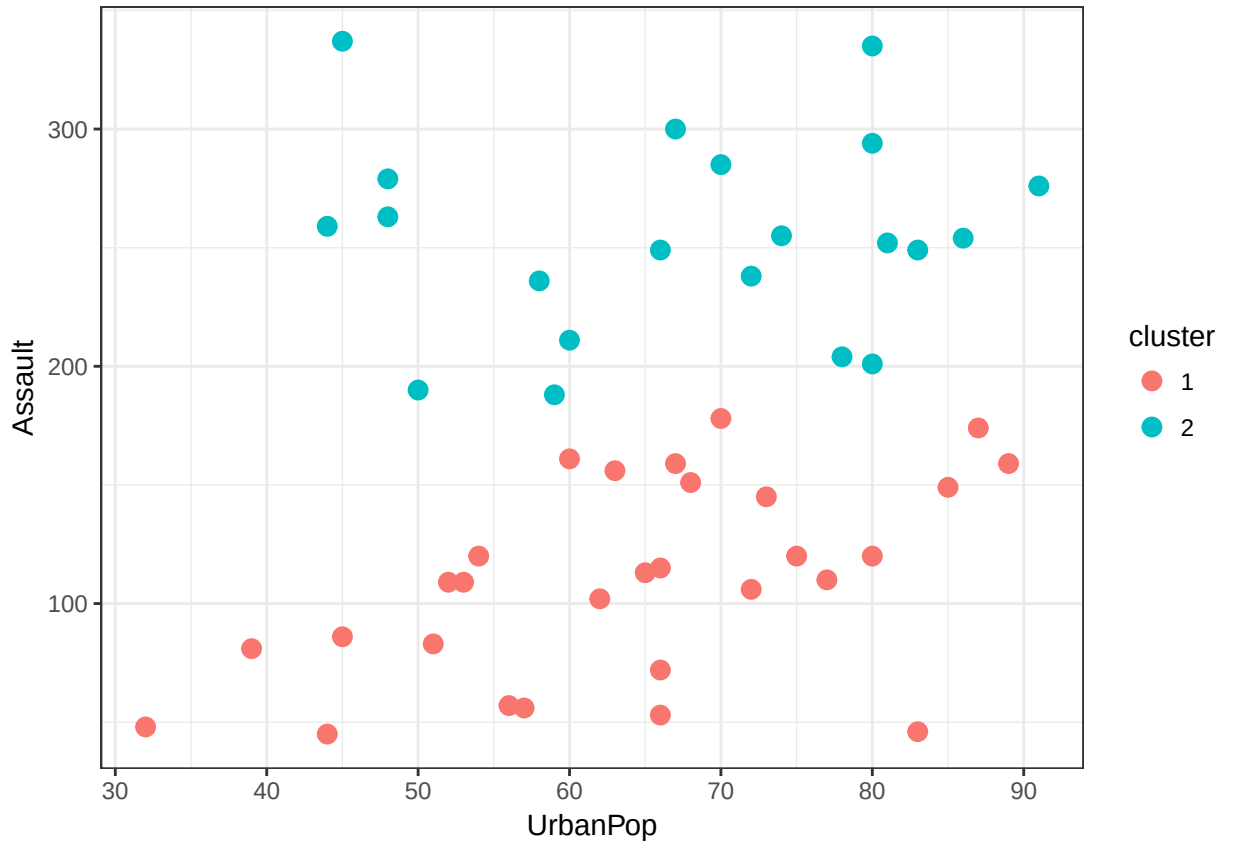
```
## K-means clustering with 2 clusters of sizes 29, 21
##
## Cluster means:
##      Murder  Assault UrbanPop     Rape
## 1  4.841379 109.7586 64.03448 16.24828
## 2 11.857143 255.0000 67.61905 28.11429
##
```

```
## Clustering vector:
##  [1] 2 2 2 2 2 2 1 2 2 2 1 1 2 1 1 1 1 2 1 2 1 2 1 2 1 1 1 2 1 1 2 2 2 1 1
## [36] 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 54762.30 41636.73
##  (between_SS / total_SS =  72.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
data_clustered <- cbind(data, data.table("cluster" = factor(km$cluster)))

head(data_clustered)
```

```
##    Murder Assault UrbanPop Rape cluster
## 1:   13.2     236       58 21.2       2
## 2:   10.0     263       48 44.5       2
## 3:    8.1     294       80 31.0       2
## 4:    8.8     190       50 19.5       2
## 5:    9.0     276       91 40.6       2
## 6:    7.9     204       78 38.7       2
```

```
ggplot(data_clustered, aes(x = UrbanPop, y = Assault, color = cluster)) +
  geom_point(size = 3)
```



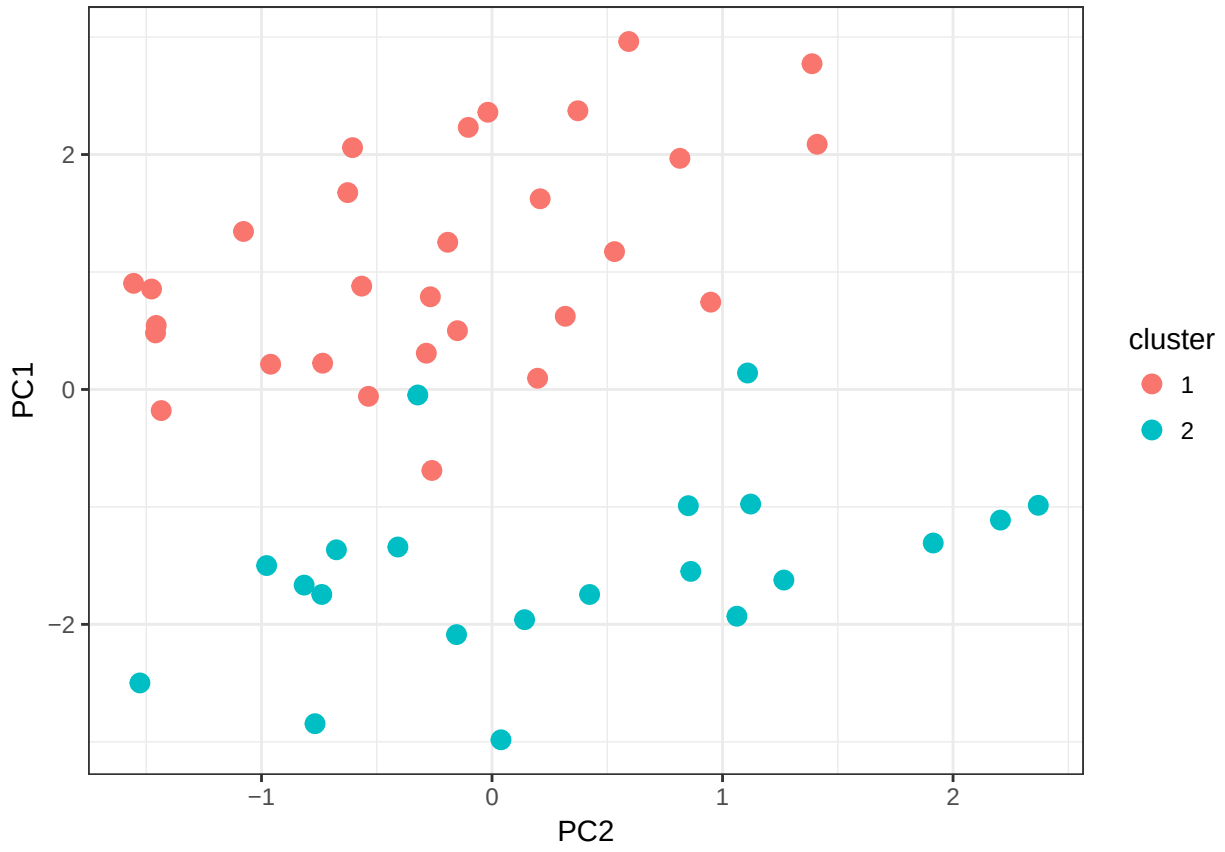
c) Perform PCA and get the first two principal component coordinates for all observations by

```
pca_result <- prcomp(data, scale. = TRUE)
first_two_pc <- data.table(pca_result$x[, 1:2])
```

Plot clusters in the coordinate system defined by the first two principal components. How do clusters relate to these?

Using the same clustering as above but plot the PC1 and the PC2:

```
data_clustered2 <- cbind(data_clustered, first_two_pc)

ggplot(data_clustered2, aes(x=PC2, y=PC1, color = cluster)) + geom_point(size = 3)
```



We can see that this method also divided the observations almost just as above just by looking at it with our eyes. Looks like we can conclude something based on the PC1 and PC2 columns if we want to divide our observations into two parts.

3) PCA of high-dimensional data

In this exercise you will perform PCA on 40 observations of 1000 variables. This is very different from what you are used to: there are much more variables than observations! These are measurements of genes of tissues of healthy and diseased patients: the first 20 observations are coming from healthy and the others from diseased patients.

```
data <- fread("C:/Users/Chronos/OneDrive - Central European University/R/machine_learning1/hw3/gene_data

data[, is_diseased := factor(is_diseased)]
dim(data)
```

```
## [1]   40 1001
```

```
tail(names(data))
```

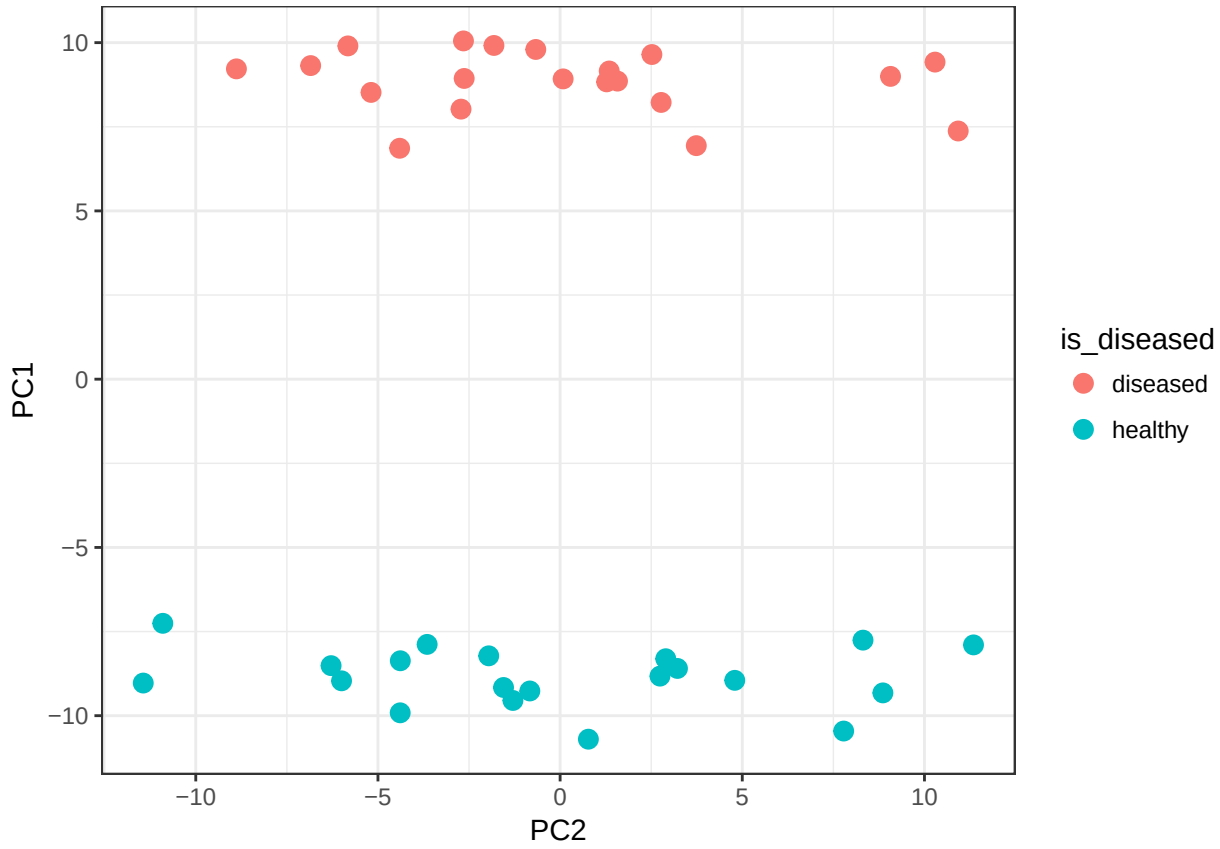
```
## [1] "measure_995" "measure_996" "measure_997" "measure_998" "measure_999"
## [6] "is_diseased"
```

a) Perform PCA on this data with scaling features.

```
data_features <- copy(data)
data_features[, is_diseased := NULL]
```

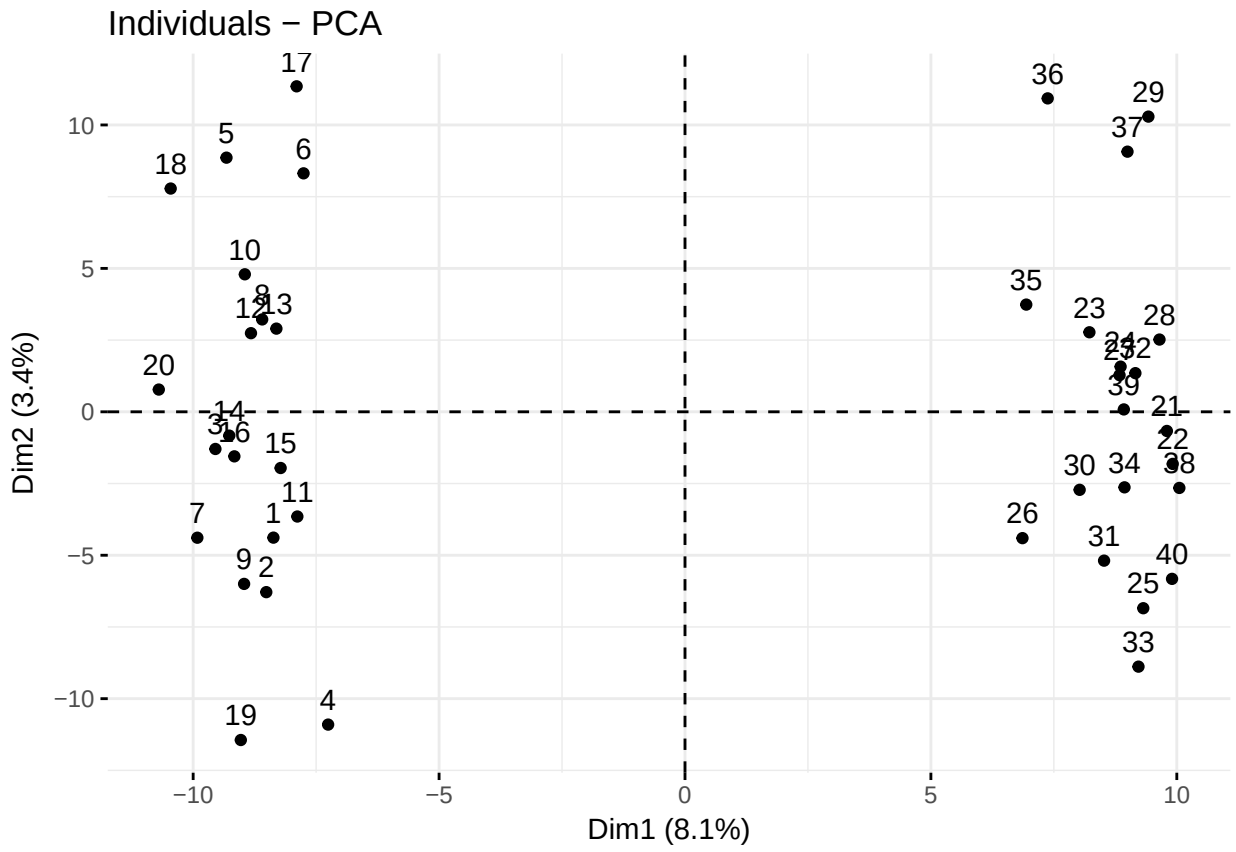
```r
pca_result <- prcomp(data_features, scale. = TRUE)
first_two_pc <- data.table(pca_result$x[, 1:2])

data_new <- cbind(data, first_two_pc)
```

```r
ggplot(data_new, aes(x=PC2, y=PC1, color = is_diseased)) + geom_point(size = 3)
```




```r
fviz_pca_ind(pca_result, axes = c(1, 2), geom = c("point", "text"), scale = 0)
```

```
## Warning: Ignoring unknown parameters: scale
```



We can clearly tell that two separate groups have formed. And in one part there are the diseased and in the other there are who are healthy.

```
table <- data.table(pca_result$rotation, keep.rownames = T)
```

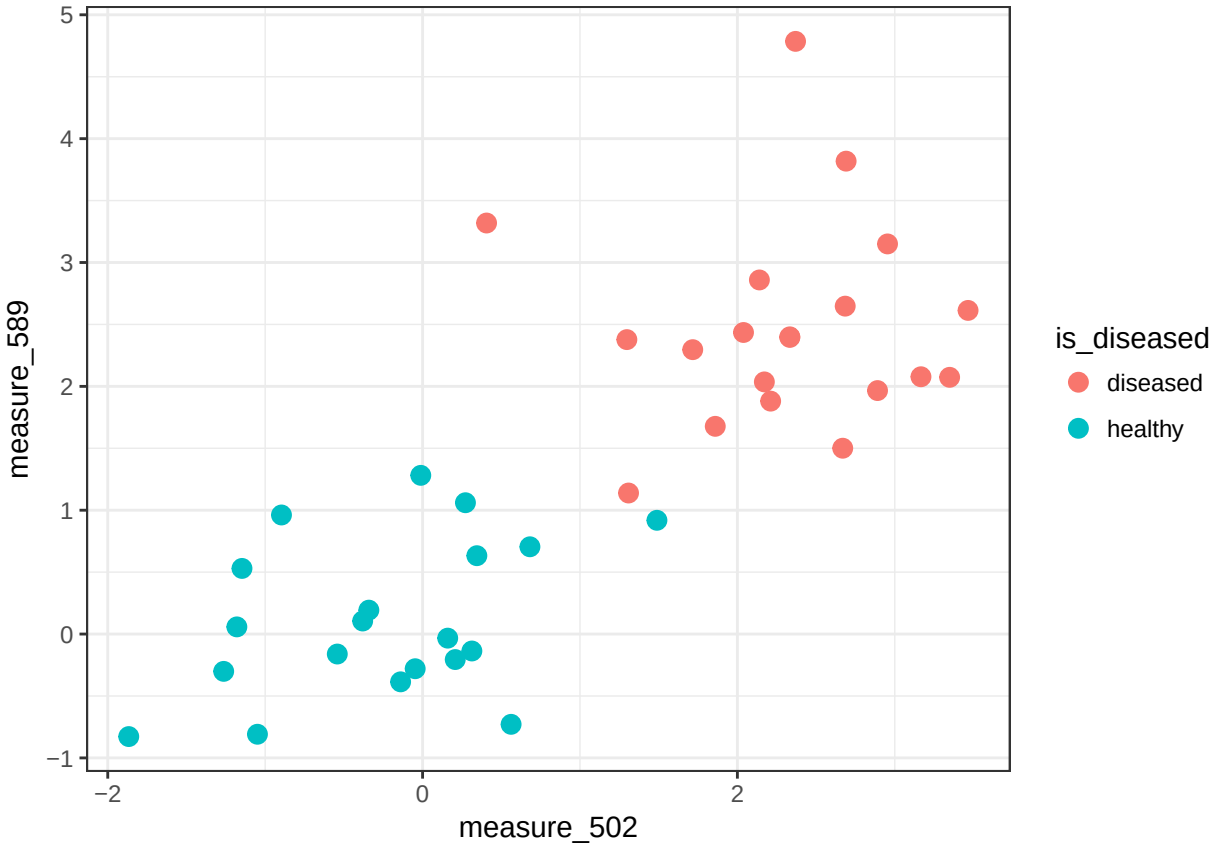
```
two_max_value <- sort(abs(table$PC1), decreasing = TRUE)[1:2]
table[PC1 %in% two_max_value, c(rn, PC1)]
```

```
## [1] "measure_502"        "measure_589"        "0.094850437514589"
## [4] "0.0944976592728264"
```

“measure_502” and “measure_589” have the highest values.

Plot these two column from the original data set:

```
ggplot(data_new, aes(measure_502, measure_589, color = is_diseased)) +
  geom_point(size = 3)
```



Seems like this two variable are (strongly) correlated with each other. And We can clearly see that we can make a separation based on them. Healthy people are in the bottom left corner and diseased are in the top right corner.

BTW: this PCA method is amazing.