covid19_solution.R

user

2020-03-18

```
################################################################################
## Course: Machine Learning for Economists and Business Analysts
## Topic: Self-Study - Exercise 1 - COVID19
################################################################################
#rm(list = ls())

#getwd()
#setwd("")

# Load data
covid19 <- read.csv("covid19.csv")
countries <- read.csv("countries.csv")

################################################################################
# Task 1
################################################################################

deaths <- (countries$deaths > 0)
table(countries$continent, deaths)
```

```
##            deaths
##             FALSE TRUE
##   Africa       28    4
##   America      24    8
##   Asia         26   11
##   Australia     2    3
##   Europe       23   22
```

```
################################################################################
# Task 2
################################################################################

# OLS model without intercept
ols <- lm(deaths ~ confirmed - 1, data = countries)
# Offical authorities report a fatality rate of 0.005
# Probably several countries have measurement error in the number of infections
summary(ols)
```

```
##
## Call:
## lm(formula = deaths ~ confirmed - 1, data = countries)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -296.49   -4.35   -0.73   -0.09  951.79
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
```

```
## confirmed 0.043110   0.001029   41.88   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 91.35 on 150 degrees of freedom
## Multiple R-squared:  0.9212, Adjusted R-squared:  0.9207
## F-statistic:  1754 on 1 and 150 DF,  p-value: < 2.2e-16
```

```
# Fitted values
fit <- predict(ols)

print(paste0("Deaths per 1000 Infected: ",round(1000*ols$coefficients, digits=0)))
```

```
## [1] "Deaths per 1000 Infected: 43"
```

```
pred_swiss <- round(fit[countries$countryregion== "Switzerland"], digits = 1)
actual_swiss <- countries$deaths[countries$countryregion== "Switzerland"]
fatality_rate_swiss <-  round(countries$deaths[countries$countryregion== "Switzerland"]
                          /countries$confirmed[countries$countryregion== "Switzerland"],
                          digits = 5)

print(paste0("Predicted Deaths Switzerland: ", pred_swiss))
```

```
## [1] "Predicted Deaths Switzerland: 94.8"
```

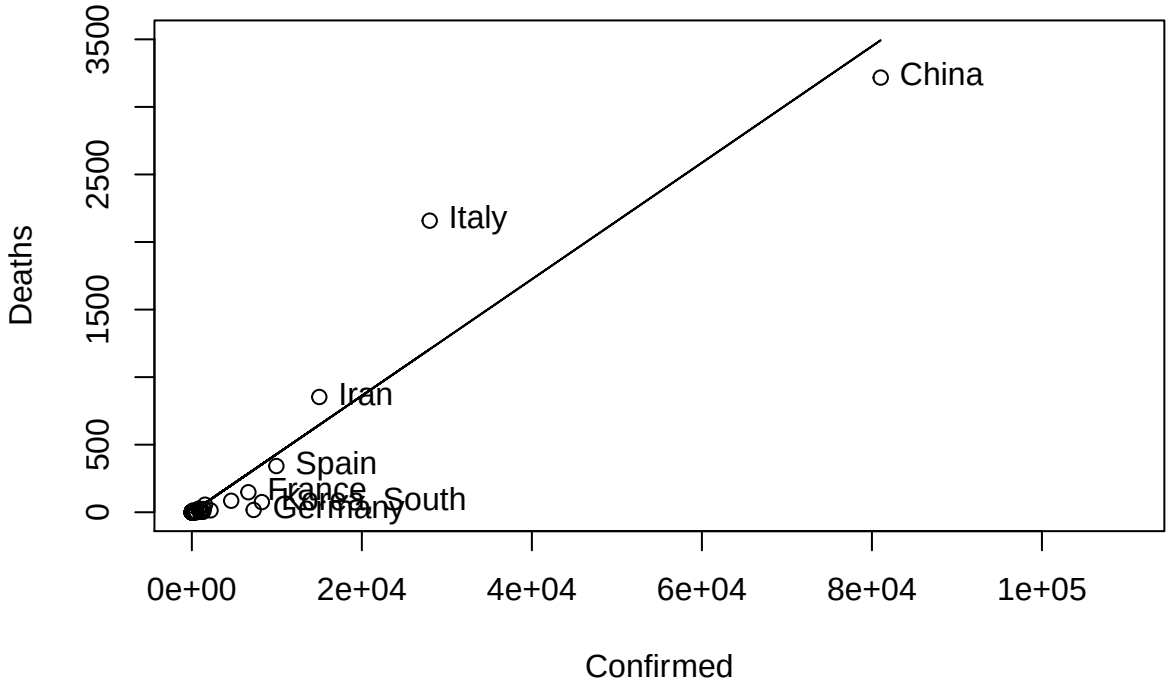
```
print(paste0("Actual Deaths Switzerland: ", actual_swiss))
```

```
## [1] "Actual Deaths Switzerland: 14"
```

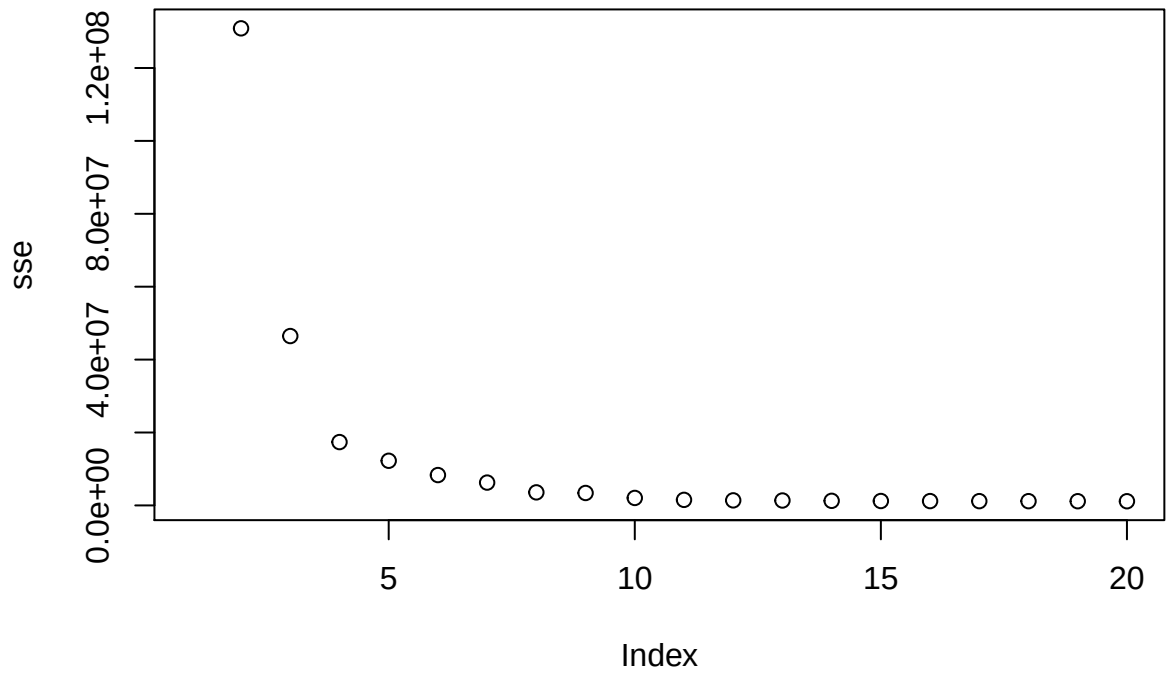
```
print(paste0("Fatality Rate Switzerland: ", fatality_rate_swiss))
```

```
## [1] "Fatality Rate Switzerland: 0.00636"
```

```
##############################################################################
# Task 3
##############################################################################

# Generate rank variable with infections
# The country with the most infections has rank 1
rank <- nrow(countries) - rank(countries$confirmed) + 1

# Scatterplot Infections and Deaths
plot(countries$confirmed, countries$deaths, xlab = 'Confirmed', ylab = 'Deaths',
    text(countries$confirmed[rank<=7], countries$deaths[rank<=7],
    labels = countries$countryregion[rank<=7], pos = 4), xlim= c(0,110000),
    ylim = c(0,3500))
lines(countries$confirmed,fit)
```




```
##############################################################################
# Task 4
##############################################################################
set.seed(123456789)

# How many clusters are optimal?
sse <- c()
sse[1] <- Inf

for (ind_cl in c(2:20)) {

  set.seed(3)
  km.out = kmeans(covid19, ind_cl, iter.max = 100, nstart = 100)
  sse[ind_cl] = km.out$tot.withinss

}

plot(sse)
```




```
# Optimum 3-4 clusters

set.seed(123456789)
# K-means clustering with 3 clusters
km.out = kmeans(covid19, 3, iter.max = 100, nstart = 100)

# Tabulate party vs cluster
table(rank[rank<= 20], km.out$cluster[rank<= 20])
```

```
##
##      1 2 3
##   1  0 0 1
##   2  1 0 0
##   3  1 0 0
##   4  1 0 0
##   5  0 1 0
##   6  1 0 0
##   7  1 0 0
##   8  0 1 0
##   9  0 1 0
##   10 0 1 0
##   11 0 1 0
##   12 0 1 0
##   13 0 1 0
##   14 0 1 0
##   15 0 1 0
```

```
##   16 0 1 0
##   17 0 1 0
##   18 0 1 0
##   19 0 1 0
##   20 0 1 0
```

```r
# South Korea is Outlier

# Report cluster means of confirmed infections and deaths
aggregate(x = countries$confirmed, by = list(km.out$cluster), FUN = mean)
```

```
##   Group.1          x
## 1       1 13367.0000
## 2       2   227.4483
## 3       3 81033.0000
```

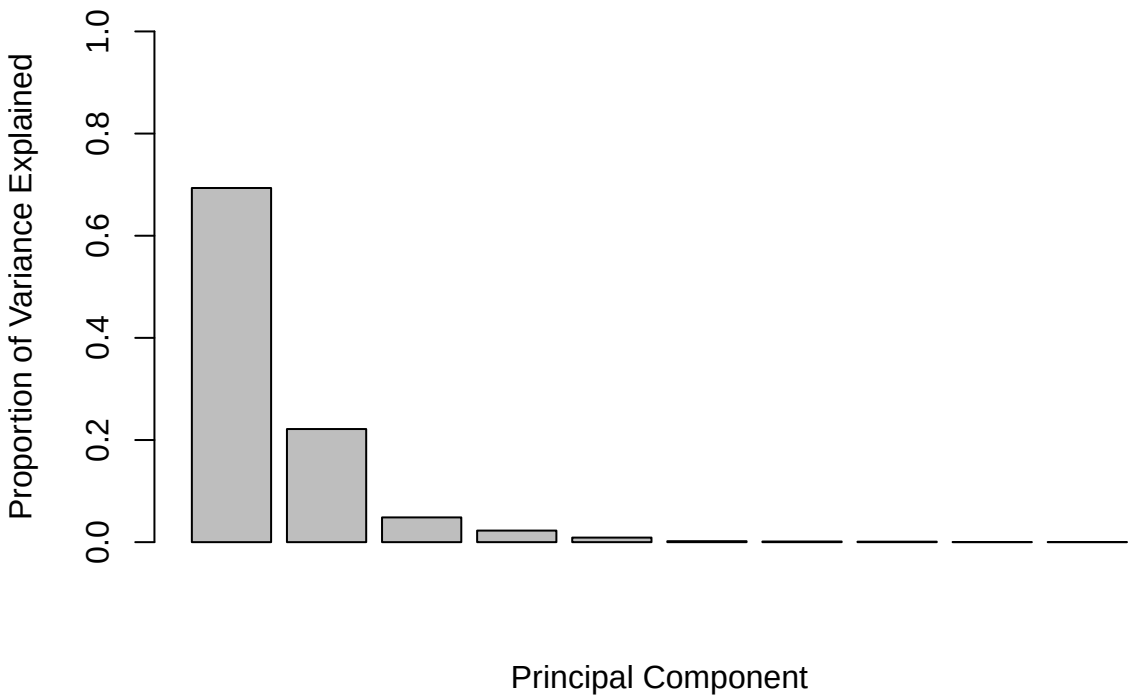
```r
aggregate(x = countries$deaths, by = list(km.out$cluster), FUN = mean)
```

```
##   Group.1           x
## 1       1  703.600000
## 2       2    2.648276
## 3       3 3217.000000
```

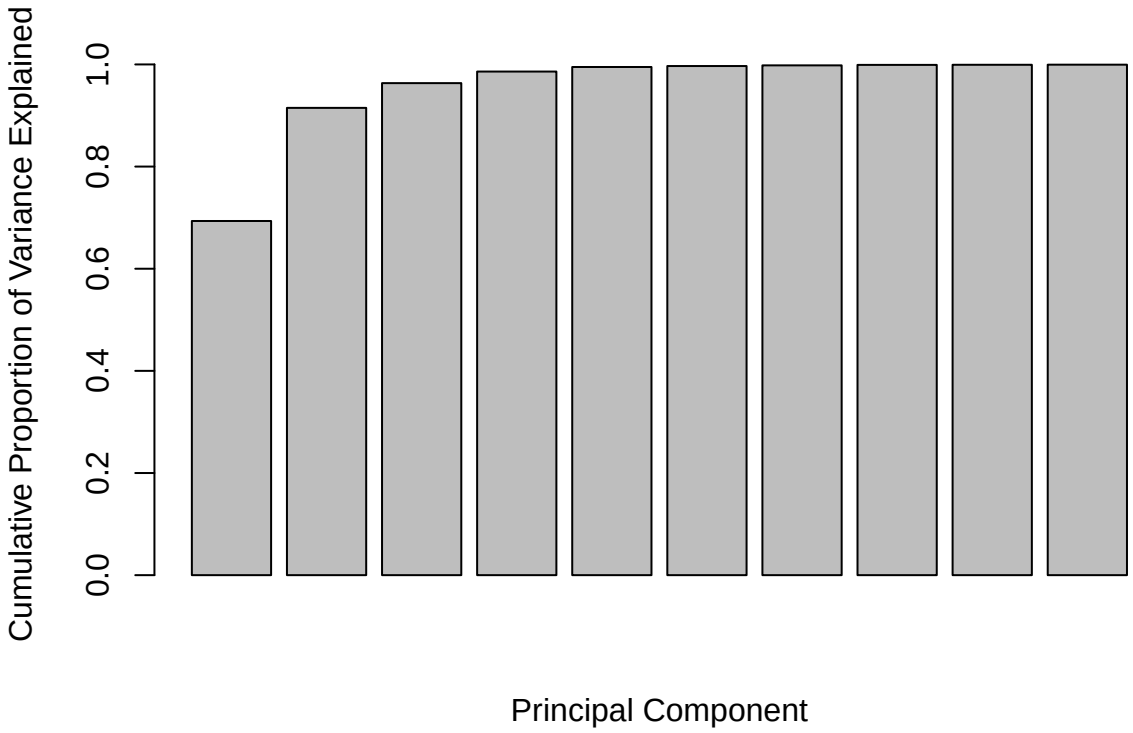
```r
################################################################################
# Task 5
################################################################################

# PCA
pr.out = prcomp(covid19, center = TRUE, scale = TRUE)

# Number of principal components
dim(pr.out$rotation)[2]
```

```
## [1] 108
```

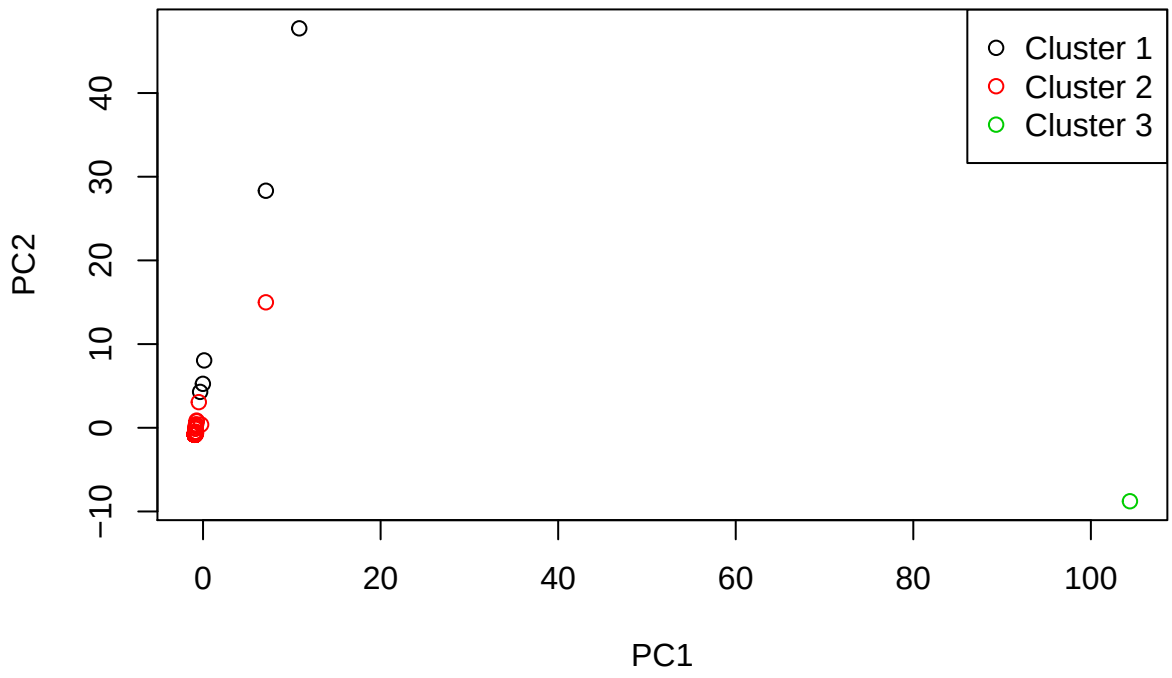
```r
# How many principal components are optimal?

# variance explained by each component
pr.var = pr.out$sdev^2

# Proportion of variance explained
pve=pr.var/sum(pr.var)

# Plot the first 10 PC
barplot(pve[1:10], xlab=" Principal Component ",
        ylab=" Proportion of Variance Explained ", ylim=c(0,1))
```




```
# Cummulative plot
barplot(cumsum(pve[1:10]), xlab=" Principal Component ",
        ylab ="Cumulative Proportion of Variance Explained ", ylim=c(0,1))
```




```
##############################################################################
# Task 6
##############################################################################

# Plot the first two principal components on the 3 clusters
plot(pr.out$x[,1], pr.out$x[,2], xlab = "PC1", ylab = "PC2", col = km.out$cluster)
legend('topright', legend = c("Cluster 1", "Cluster 2", "Cluster 3"), col = 1:5,  pch = 1)
```




```
##########################################################################################
# Task 7
##########################################################################################

# Look at the largest (absolute) loadings of the first two principal components
loadings <- pr.out$rotation
loadings[order(abs(loadings[,1]), decreasing=TRUE)[1:10],2]
```

```
## death_feb_25 death_feb_26 death_feb_24 death_feb_29 death_feb_28
## -0.016012059 -0.015795450 -0.022422342  0.019047300  0.002391352
##  conf_feb_22 death_feb_19 death_feb_22 death_feb_20  conf_jan_31
## -0.016361223 -0.028342136 -0.028790778 -0.029639847 -0.029654803
```

```
loadings[order(abs(loadings[,2]), decreasing=TRUE)[1:10],2]
```

```
##   conf_mar_8   conf_mar_7  conf_mar_11   conf_mar_9 death_mar_14
##    0.1987739    0.1966286    0.1934100    0.1928498    0.1913216
## death_mar_16   conf_mar_5  conf_mar_14 death_mar_15 death_mar_11
##    0.1895955    0.1894628    0.1883832    0.1866532    0.1863755
```