Project IV (PageRank and Anomaly Detection)

Quaye E, George

Due: 10/19/2020

Contents

1 Problem 1: PageRank

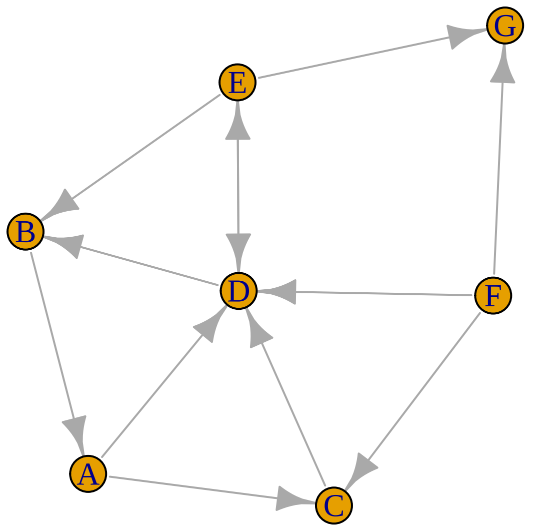
Based on the links in Figure 1, obtain the link matrix L and then accordingly compute the PageRank score for each webpage. Provide a barplot of the PageRank score. Which pages come to the top-3 list?

```
# The Link Matrix Link
L <- matrix(c(0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0,
1, 0, 1, 0, 1, 1, 0,
0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 0), nrow = 7, ncol = 7, byrow = F)
colnames(L)<-c("A","B","C","D","E","F","G")
row.names(L)<-c("A","B","C","D","E","F","G")

L
```

```
##   A B C D E F G
## A 0 0 1 1 0 0 0
## B 1 0 0 0 0 0 0
## C 0 0 0 1 0 0 0
## D 0 1 0 0 1 0 0
## E 0 1 0 1 0 0 1
## F 0 0 1 1 0 0 1
## G 0 0 0 0 0 0 0
```

The matrix Link is made up of a 7×7 square matrix.

```
#Plot of the Link Matrix

library(igraph)
graph <- graph_from_adjacency_matrix(L)
par(mfrow=c(1,1), mar=rep(4,4))
plot(graph)
```



```r
#plot(graph, layout=layout_with_fr, vertex.size=20,
#    vertex.label.dist=2, vertex.color="lightblue", edge.arrow.size=0.5)
```

The plot above is to confirmed my link matrix L from the question.

```r
pagerank <- function(G, method='eigen',d=.85,niter=100){
  cvec <- apply(G,2,sum)
  cvec[cvec==0] <- 1
  n <- nrow(G)
  delta <- (1-d)/n
  A <- matrix(delta,nrow(G),ncol(G))
  for (i in 1:n)   A[i,] <- A[i,] + d*G[i,]/cvec
#  print(A)
  if (method=='power'){
    x <- rep(1,n)
    for (i in 1:niter) x <- A%*%x
  } else {
    x <- Re(eigen(A)$vector[,1])
  }
  x/sum(x)
}
```

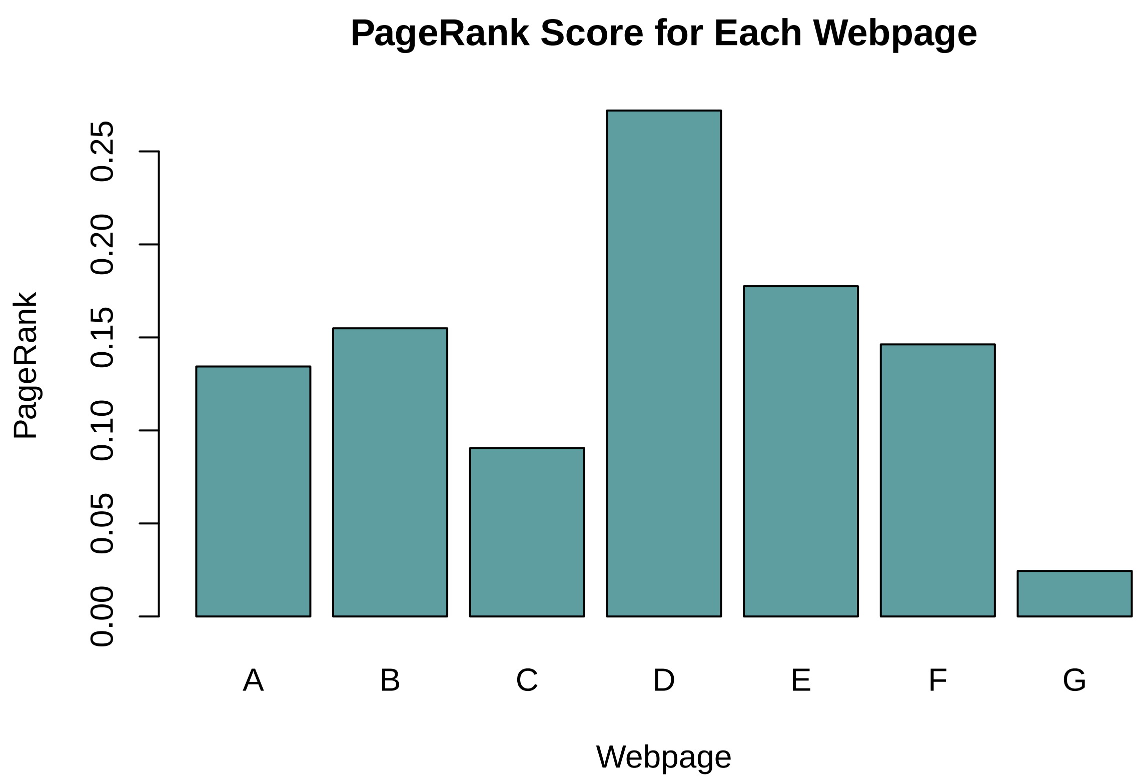
```r
#Computing the PageRank score for each webpage
pg <- pagerank(L, method='power')
sum(pg)
```

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

```r
pg <- data.frame("WebPage"= c("A","B","C","D","E","F","G"), "PageRank"= pg)
pg
```

```
##   WebPage   PageRank
## 1       A 0.13438041
## 2       B 0.15491010
## 3       C 0.09047138
## 4       D 0.27197905
## 5       E 0.17753139
## 6       F 0.14625695
## 7       G 0.02447073
```

```r
#Barplot of the PageRank score.
barplot(pg$PageRank, names= pg$WebPage, col="cadetblue", xlab="Webpage", ylab="PageRank", main="PageRan
```



The Bar plot above shows the PageRank scores, It is observed that Web page D has the highest PageRank score and Web page G has the least PageRank score. Implying that page D holds lot of information to access and more visits compared to that of page G.

```
#The top-3 list PageRank score
Top3 <- pg[ order(pg$PageRank, decreasing = TRUE), ]
head(Top3, n=3)
```

```
##   WebPage  PageRank
## 4       D 0.2719790
## 5       E 0.1775314
## 2       B 0.1549101
```

The top 3 list pages based on the page rank scores are Webpages D, E and B with 27.2%, 17.8% and 15.5% respectively from the outputs above.

2 Problem 2: Anomaly Detection

We consider the HTP (high tech part) data available from R Package ICSOutlier. This data set contains the results of $p = 88$ numerical tests for $n = 902$ high-tech parts. Based on these results the producer considered all parts functional and all of them were sold. However two parts, 581 and 619, showed defects in use and were returned to the manufacturer. These two observations can thus be considered as outliers and the objective is to detect them by re-examining the test data.

(a) Bring in the data with the following R code:

```
library("ICSOutlier")
data(HTP)
dat <- HTP; dim(dat); head(dat)
```

```
## [1] 902  88
```

```
##              V.1           V.2           V.3           V.4           V.5
## 1  2.726436e-04  3.237405e-06  3.040539e-04  3.258806e-06  4.395332e-06
## 2 -1.268664e-04  5.179741e-05 -1.861661e-04  5.263881e-05  5.112533e-05
## 3  3.536364e-05  4.897405e-06  6.137388e-05  4.918806e-06  5.045332e-06
## 4 -3.900464e-04  1.663741e-05 -5.111861e-04  1.716881e-05  1.759533e-05
## 5 -5.985264e-04  5.097405e-06 -6.724361e-04  4.098806e-06  4.725332e-06
## 6 -7.890464e-04 -2.546259e-05 -8.462061e-04 -2.575119e-05 -2.593467e-05
##              V.6           V.7           V.8           V.9          V.10
## 1  1.949953e-04  2.604694e-06  1.314538e-04  0.0008319774  3.191322e-04
## 2 -5.665467e-05  5.025469e-05 -5.258623e-05 -0.0007294226 -3.276878e-04
## 3  3.327533e-05  4.074694e-06 -8.619623e-05 -0.0004575226  7.338218e-05
## 4 -2.185047e-04  1.458469e-05 -1.113562e-04 -0.0007819226 -5.975478e-04
## 5 -4.391347e-04  2.524694e-06 -1.264562e-04  0.0001531774 -6.203649e-04
## 6 -5.641747e-04 -2.741531e-05 -1.601762e-04  0.0003164774 -6.633731e-04
##             V.11          V.12          V.13          V.14          V.15
## 1  3.912975e-06  3.215580e-04  3.688435e-06  7.498488e-05  3.067430e-04
## 2  6.597298e-05 -2.260220e-04  5.214843e-05 -2.545119e-06 -3.322770e-04
## 3  2.742975e-06  8.162802e-05  5.148435e-06 -4.699512e-05  7.170302e-05
## 4  2.087298e-05 -5.750220e-04  1.698843e-05 -4.527512e-05 -5.864270e-04
## 5  5.672975e-06 -6.888920e-04  4.728435e-06 -5.844512e-05 -5.975832e-04
## 6 -2.855702e-05 -8.379420e-04 -2.593157e-05 -8.992512e-05 -6.341554e-04
##             V.16          V.17          V.18          V.19          V.20
## 1  3.287189e-06  2.915543e-04  1.292947e-04  5.546208e-06  2.690776e-06
## 2  4.909719e-05 -1.405857e-04 -1.832527e-05 -2.073792e-06  5.259078e-05
## 3  5.667189e-06  3.306428e-05  4.924473e-05  1.829621e-05  6.100776e-06
## 4  1.688719e-05 -4.314957e-04 -9.228527e-05 -1.120379e-05  1.691078e-05
## 5  5.037189e-06 -6.263457e-04 -2.904753e-04  4.762084e-07  5.270776e-06
## 6 -2.521281e-05 -8.188057e-04 -3.283753e-04  4.762084e-07 -2.569922e-05
##             V.21          V.22          V.23          V.24          V.25
## 1 -1.258796e-04  3.158345e-06  3.241261e-06 -4.091042e-05  2.604191e-04
## 2 -2.581296e-04  5.121835e-05  5.282126e-05  5.039958e-05 -9.240094e-05
## 3 -5.466396e-04  4.618345e-06  5.311261e-06  1.064958e-05 -1.139509e-04
## 4  6.859042e-05  1.737835e-05  1.715126e-05  3.969958e-05 -2.237309e-04
## 5 -3.685996e-04  5.938345e-06  4.791261e-06  3.829579e-06 -2.563409e-04
## 6 -3.250596e-04 -2.523165e-05 -2.525874e-05  1.757958e-05 -2.928209e-04
##             V.26          V.27          V.28          V.29          V.30
## 1  4.302839e-06 -1.622775e-05 -3.077118e-05  4.277639e-06  3.279396e-04
## 2  5.032284e-05 -3.373775e-05  1.339888e-04  4.978764e-05 -2.512904e-04
## 3  5.262839e-06  1.716225e-05  5.477882e-05  1.457639e-06  7.862959e-05
## 4  1.699284e-05 -3.617749e-06  1.028688e-04  1.430764e-05 -5.951304e-04
## 5  5.552839e-06  1.152523e-04  1.697688e-04  2.667639e-06 -6.847504e-04
## 6 -2.571716e-05  4.992251e-06  4.920882e-05 -3.125236e-05 -8.075404e-04
##             V.31         V.32          V.33          V.34          V.35
## 1 -6.030747e-07 0.0001997316  2.901919e-04  2.950068e-06  9.566039e-05
## 2  1.876385e-06 0.0015270316 -1.487181e-04  5.336007e-05 -9.959606e-06
## 3  8.165053e-07 0.0003932316  4.526186e-05  4.830068e-06  4.692039e-05
## 4  7.084353e-07 0.0002697316 -4.490381e-04  1.666007e-05 -5.015961e-05
## 5 -3.982147e-07 0.0009875316 -6.384581e-04  5.320068e-06 -2.397096e-04
## 6  4.684353e-07 0.0001635316 -8.303081e-04 -2.645993e-05 -2.389896e-04
##            V.36          V.37          V.38          V.39          V.40
## 1 -1.092576e-04 -2.045656e-05  2.309891e-04  3.163167e-04  5.269756e-05
## 2 -2.481976e-04  3.506344e-05 -8.244094e-05 -2.151833e-04 -6.306244e-05
## 3 -3.570176e-04  2.174344e-05  3.489906e-05  7.352674e-05  2.175610e-07
## 4 -5.331756e-05  2.096344e-05 -2.838009e-04 -5.567033e-04 -2.136244e-05
## 5 -2.556276e-04  9.913437e-06 -5.061909e-04 -6.835733e-04  7.071756e-05
## 6 -2.761976e-04  6.434368e-07 -6.748509e-04 -8.461033e-04  1.537561e-06
##            V.41          V.42          V.43          V.44          V.45
## 1  3.247248e-04  0.0002232093  4.880209e-06  0.0003201276  3.003677e-06
## 2 -2.621252e-04 -0.0001921907  5.232021e-05 -0.0001927824  5.197368e-05
## 3  8.469475e-05 -0.0003442907  4.400209e-06  0.0000677476  4.163677e-06
## 4 -6.017852e-04 -0.0002545907  1.695021e-05 -0.0005321124  1.660368e-05
## 5 -6.787152e-04  0.0003723093  5.310209e-06 -0.0006819024  3.843677e-06
## 6 -7.937471e-04  0.0002165093 -2.554979e-05 -0.0008634424 -2.630632e-05
##            V.46          V.47          V.48          V.49          V.50
## 1  0.0006315873  2.111365e-04  4.352026e-07  1.594326e-04 -6.199151e-05
## 2 -0.0009373127 -6.793345e-05  9.058426e-07 -3.317744e-05  3.087849e-05
## 3 -0.0009843127  2.850655e-05  3.516726e-07  4.349256e-05  9.531849e-05
## 4 -0.0008205127 -2.531135e-04 -6.437395e-09 -1.461174e-04  5.563849e-05
## 5 -0.0001046127 -4.731935e-04  3.893726e-07 -3.537474e-04  3.750185e-04
## 6  0.0002708873 -6.194835e-04  7.549726e-07 -4.385374e-04  8.896849e-05
##            V.51          V.52          V.53          V.54          V.55
## 1  3.343624e-04  4.261938e-04  3.041318e-04 -4.887982e-06  2.822660e-06
## 2 -1.225376e-04  8.544378e-05 -1.625982e-04  4.583202e-05  5.271266e-05
## 3  1.996242e-05 -5.385622e-05  4.706179e-05  5.122202e-05  5.302660e-06
## 4 -2.811376e-04 -1.897262e-04 -4.811382e-04  7.822018e-06  1.724266e-05
## 5  3.775624e-04 -4.276462e-04 -6.568382e-04  2.335202e-05  4.982660e-06
## 6  2.430624e-04 -4.580262e-04 -8.505282e-04  1.902018e-06 -2.669734e-05
##            V.56          V.57          V.58          V.59          V.60
## 1 -2.674375e-05  3.852197e-04  3.020560e-04  3.211581e-04  3.162044e-05
## 2  1.999625e-05 -3.127029e-05 -3.207240e-04 -2.664319e-04  3.974044e-05
## 3  1.216625e-05 -7.964029e-05  7.486604e-05  9.366806e-05 -1.807956e-05
## 4  1.749625e-05 -2.833003e-04 -5.875340e-04 -6.118219e-04 -5.759557e-06
## 5 -1.987375e-05 -4.108703e-04 -6.017588e-04 -6.804719e-04 -2.085956e-05
## 6  6.536253e-06 -4.664803e-04 -6.421109e-04 -7.871287e-04 -5.376956e-05
##            V.61          V.62          V.63          V.64          V.65
## 1  2.551865e-04  2.523415e-06 -1.751972e-05  3.235729e-04  3.323603e-06
## 2 -9.046352e-05  5.313341e-05  3.014028e-05 -3.158071e-04  5.209360e-05
## 3  2.789648e-05  5.213415e-06  4.020277e-06  8.353288e-05  5.603603e-06
## 4 -3.202635e-04  1.755341e-05  2.527028e-05 -6.100771e-04  1.621360e-05
## 5 -5.447535e-04  4.483415e-06  1.413028e-05 -6.389371e-04  4.873603e-06
## 6 -7.210835e-04 -2.617659e-05  1.310277e-06 -6.994114e-04 -2.598640e-05
##            V.66          V.67          V.68          V.69          V.70
## 1 -4.169452e-05  3.178148e-06  1.838077e-04  3.722320e-06  2.483639e-06
## 2  2.312548e-05  5.204815e-05 -3.980233e-05  5.760232e-05  5.217364e-05
## 3 -4.754523e-06  4.538148e-06  3.911767e-05  3.152320e-06  6.093639e-06
## 4  1.930548e-05  1.657815e-05 -1.792423e-04  1.722232e-05  1.721364e-05
## 5  8.995477e-06  5.448148e-06 -4.043023e-04  3.742320e-06  5.063639e-06
## 6  1.086548e-05 -2.572185e-05 -5.126923e-04 -2.721768e-05 -2.590636e-05
##            V.71          V.72          V.73          V.74          V.75
## 1  3.586079e-05  3.679196e-06  1.176750e-05  3.259493e-04  1.457141e-04
## 2 -7.087921e-05  5.337920e-05  4.259750e-05 -2.990307e-04 -2.348594e-05
## 3  3.596079e-05  4.839196e-06 -3.382502e-06  9.824927e-05  4.660406e-05
## 4 -2.051921e-05  1.769920e-05  6.917498e-06 -6.217807e-04 -1.204559e-04
## 5 -1.808992e-04  5.639196e-06 -5.542502e-06 -6.662907e-04 -3.239259e-04
## 6 -1.325792e-04 -2.603080e-05 -3.558250e-05 -7.413748e-04 -3.859359e-04
##            V.76          V.77          V.78          V.79          V.80
## 1  7.455461e-05 -0.0001527421  1.181130e-04  0.0009664207  0.0000613574
## 2 -9.185393e-06 -0.0002001521 -1.337699e-05  0.0007458207 -0.0000301626
## 3  4.406461e-05 -0.0006433421  4.530301e-05 -0.0002043793  0.0000420874
## 4 -3.179539e-05  0.0001423779 -7.149699e-05 -0.0005676793 -0.0000234526
## 5 -2.072654e-04 -0.0004419521 -2.734770e-04  0.0004198207 -0.0001997426
## 6 -1.850954e-04 -0.0003172321 -3.002370e-04  0.0003945207 -0.0001587826
##            V.81          V.82          V.83          V.84          V.85
## 1  3.140559e-04  3.304995e-04  6.671859e-05  3.385299e-06  2.476920e-04
## 2 -3.100041e-04 -2.801905e-04 -6.161141e-05  5.195530e-05 -1.079280e-04
## 3  7.483592e-05  8.852954e-05  7.458592e-06  4.645299e-06  3.029203e-05
## 4 -5.928241e-04 -6.221205e-04 -5.221408e-06  1.709530e-05 -3.499080e-04
## 5 -6.312941e-04 -6.785005e-04  8.011859e-05  5.455299e-06 -5.641680e-04
## 6 -6.998402e-04 -7.788191e-04 -1.716141e-05 -2.591470e-05 -7.471280e-04
##            V.86          V.87          V.88
## 1  2.129363e-06 -2.691822e-05  3.447164e-04
## 2  6.336936e-05  1.345918e-04  1.488464e-04
## 3  3.199363e-06  5.587178e-05 -7.975364e-05
## 4  1.980936e-05  1.015818e-04 -6.473636e-06
## 5  4.809363e-06  1.737518e-04 -2.378336e-04
## 6 -2.830064e-05  4.943178e-05 -1.816936e-04
```

```r
outliers.true <- c(581, 619)
```

The data has 88 columns with 6 rows.

(b) First obtain robust estimates of the mean vector $\hat{\mu}$ and the VCOV matrix $\hat{\sum}$ of the data with MCD with a breakdown point of your choice.

```r
library(robustbase)
# Obtain MCD estimates with a breakdown point of 15%
fit.robust <- covMcd(dat, cor = FALSE, alpha = 0.85)
```

A breakdown point of 15% is used.

```r
# Robust estimates of the mean vector for 15 variables:
Mean_vector <- fit.robust$center
Mean_vector[1:15]
```

```
##           V.1           V.2           V.3           V.4           V.5
##  1.035526e-05 -6.499236e-07  1.641517e-06 -6.597203e-07 -6.234638e-07
##           V.6           V.7           V.8           V.9          V.10
##  2.081152e-05 -6.929527e-07  5.969568e-06 -7.714012e-05 -1.874791e-05
##          V.11          V.12          V.13          V.14          V.15
## -8.217945e-07 -4.474919e-06 -6.329331e-07  1.655186e-06 -2.034320e-05
```

Given above is the first 15 $\hat{\mu}$ for the robust estimates.

```r
# Robust estimates of the VCOV matrix
Cov_matrix <- fit.robust$cov
Cov_matrix[1:15]
```

```
##  [1]  1.520255e-07 -4.390573e-09  1.766173e-07 -4.344921e-09 -4.342740e-09
##  [6]  1.040584e-07 -3.842857e-09  4.738963e-08  1.380648e-07  1.831486e-07
## [11] -5.058166e-09  1.868722e-07 -4.346865e-09  2.168496e-08  1.797745e-07
```

Given above is the first 15 $\hat{\sum}$ for the robust estimates.

Computing the robust Mahalanobis distance of each observation with respect to the MCD estimates and plot them. You may add a threshold based on the $\chi^2(p)$ distribution and highlight the two defective parts.

```r
#Robust squared Mahalanobis distance
Mahalanobis_Dist <- mahalanobis(dat,Mean_vector,Cov_matrix)
head(Mahalanobis_Dist)
```

```
## [1]  97.06014 140.17806  82.75311  73.83783  76.46966  75.19781
```

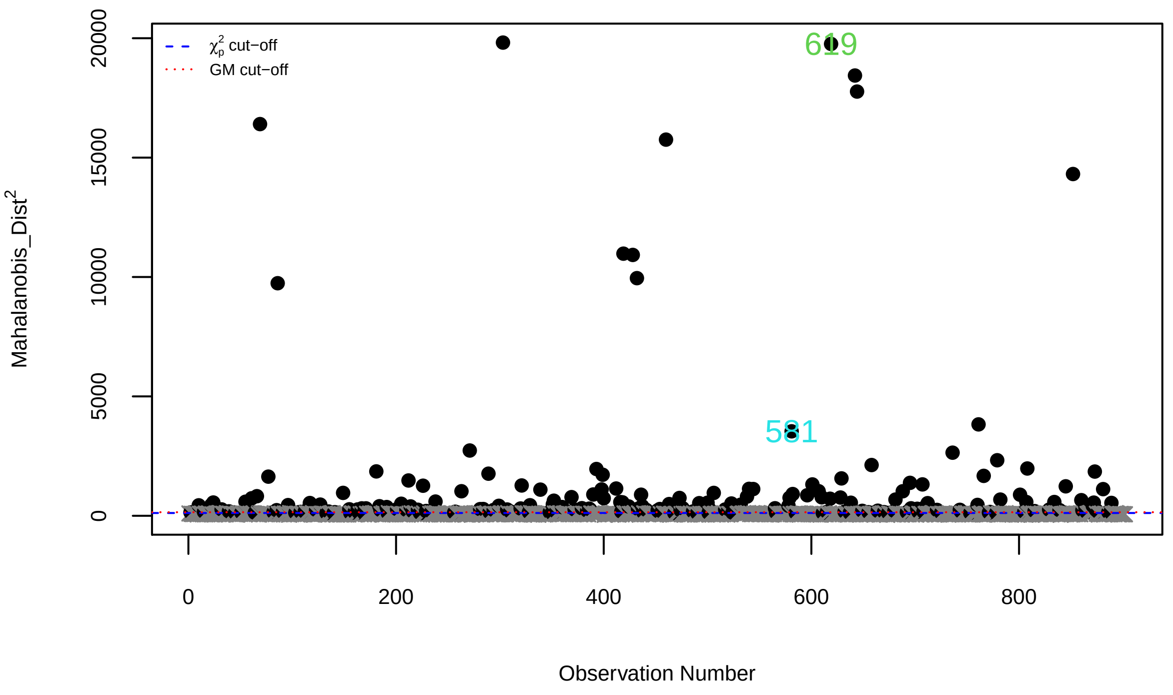
```r
# Cut-off based on the chi-square distribution
cutoff.chi.sq <- qchisq(0.975, df = ncol(dat)); cutoff.chi.sq
```

```
## [1] 115.8414
```

```r
# Another Cut-off Suggested by Green and Martin (2014)
library("CerioliOutlierDetection")
n <- nrow(dat); p <- ncol(dat)
cutoff.GM <- hr05CutoffMvnormal(n.obs = n, p.dim=p, mcd.alpha = 0.75,
    signif.alpha = 0.025, method = "GM14",
    use.consistency.correction = TRUE)$cutoff.asy
cutoff.GM
```

```
## [1] 149.9075
```

```r
# PLOT THE RESULTS
colPoints <- ifelse(Mahalanobis_Dist >= min(c(cutoff.chi.sq, cutoff.GM)), 1, grey(0.5))
pchPoints <- ifelse(Mahalanobis_Dist >= min(c(cutoff.chi.sq, cutoff.GM)), 16, 4)

plot(seq_along(Mahalanobis_Dist), Mahalanobis_Dist, pch = pchPoints, col = colPoints,
    ylim=c(0, max(Mahalanobis_Dist, cutoff.chi.sq, cutoff.GM) + 2), cex.axis = 0.7, cex.lab = 0.7,
    ylab = expression(Mahalanobis_Dist**2), xlab = "Observation Number")

abline(h = c(cutoff.chi.sq, cutoff.GM), lty = c("dashed", "dotted"), col=c("blue", "red"))

legend("topleft", lty = c("dashed", "dotted"), cex = 0.5, ncol = 1, bty = "n",
legend = c(expression(paste(chi[p]**2, " cut-off")), "GM cut-off"), col=c("blue", "red"))
text(619, Mahalanobis_Dist[619], labels=619, col=619)
text(581, Mahalanobis_Dist[581], labels=581, col=581)
```



The threshold used are based on the chi-square distribution and another suggested by Green and Martin (2017). From the plot above we observe that the two defective parts are in the top list of potential outliers (ie 619, 581).

(c) Apply isolation forest (iForest), local outlier factor (LOF), and, optionally, one-class SVM for the same task. Choose the involved parameters appropriately based on your own judgment. Plot the results and compare. Comment on the similarities and differences of their results. In particular, pay attention to whether the two defective parts are deemed anomalies by each method.

```
#Isolation Forest

#install.packages("IsolationForest", repos="http://R-Forge.R-project.org")

library(IsolationForest)
```

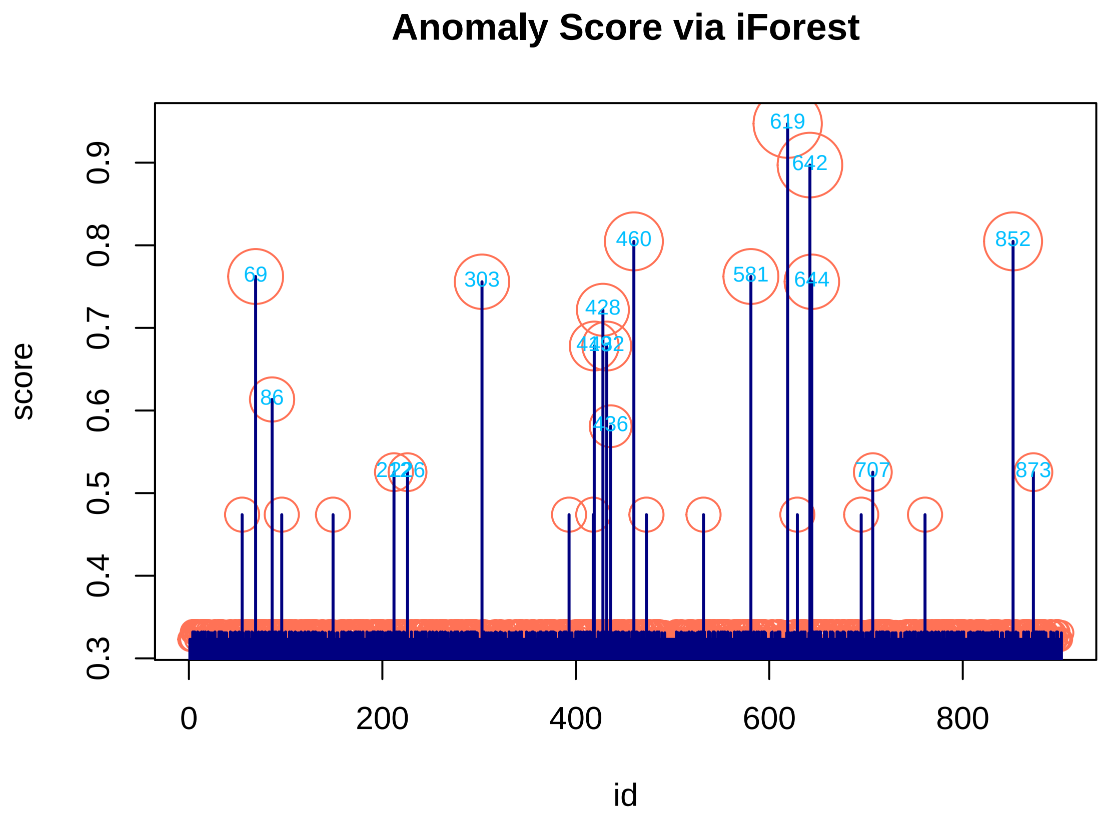
```
## IsolationForest 0.0-26
```

```
#Building isolation trees
Tree1 <- IsolationTrees(dat, rFactor=0)

#Evaluate anomaly score
anomaly_score <- AnomalyScore(dat,Tree1);

# show anomaly score
Ascore <- anomaly_score$outF;

# PLOT OF THE SCORES
par(mfrow=c(1,1), mar=rep(4,4))
plot(x=1:length(Ascore), Ascore, type="p", pch=1,
    main="Anomaly Score via iForest",
        xlab="id", ylab="score", cex=Ascore*5, col="coral1")
```

```r
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
    lty=1, lwd=1.5, col="navyblue")
apply(data.frame(id=1:length(Ascore), score=Ascore), 1, FUN=add.seg)
```

```
## NULL
```

```r
eps <- 0.98
id.outliers <- which(Ascore > quantile(Ascore, eps))
text(id.outliers, Ascore[id.outliers]+0.003, label=id.outliers,
    col="deepskyblue1", cex=0.7)
```



With probability of 0.98, it is observe from the plot that the two defective parts 581 and 619 are included in the top list of potential or expected outliers.

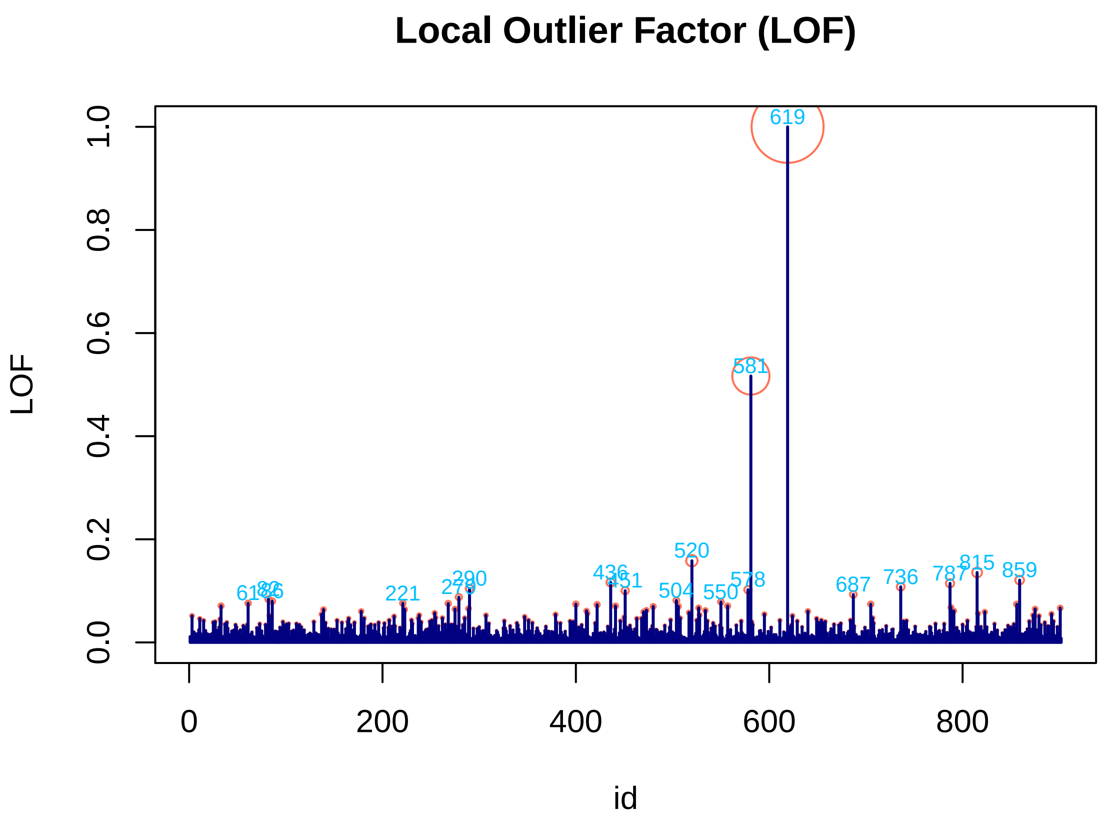
```r
# LOF - Local Outlier Factor

library(Rlof)
outlier.scores <- lof(dat, k=5);
which(outlier.scores > quantile(outlier.scores, 0.95))
```

```
##  [1]  33  61  82  86 139 178 221 223 268 275 279 289 290 400 411 422 436 441 451
## [20] 470 473 480 504 506 517 520 527 534 550 557 578 581 619 640 687 705 736 787
## [39] 788 791 815 823 856 859 875 901
```

```r
# PLOT OF THE LOF SCORES
score <- scale(outlier.scores, center = min(outlier.scores),
    scale = max(outlier.scores)-min(outlier.scores)) # NORMALIZED TO RANGE[0,1]
par(mfrow=c(1,1), mar=rep(4,4))
plot(x=1:length(score), score, type="p", pch=1,
    main="Local Outlier Factor (LOF)",
        xlab="id", ylab="LOF", cex=score*5, col="coral1")
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
    lty=1, lwd=1.5, col="navyblue")
apply(data.frame(id=1:length(score), score=score), 1, FUN=add.seg)
```

```
## NULL
```

```r
eps <- 0.98
id.outliers <- which(outlier.scores > quantile(outlier.scores, eps))
text(id.outliers, score[id.outliers]+0.02, label=id.outliers,
    col="deepskyblue1", cex=0.7)
```



The LOF finds anomalous data points by measuring the local deviation of the data point with respect to its neighbors. Given this problem, a size of the neighborhood ($k = 5$) is used. From the graph, we see that the two defective parts 581 and 619 are indeed anomalies since they're perfectly detected.

Comparison:

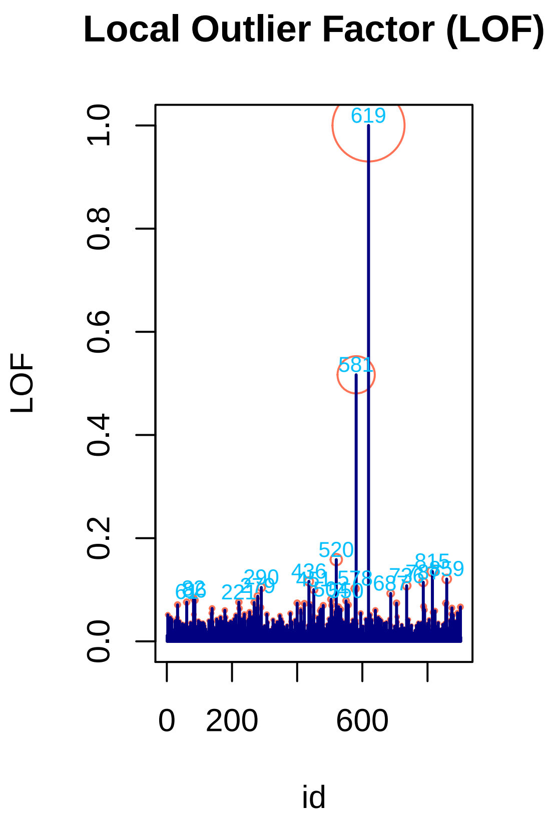
```r
par(mfrow=c(1,2), mar=rep(4,4))
# LOC
plot(x=1:length(score), score, type="p", pch=1,
    main="Local Outlier Factor (LOF)",
        xlab="id", ylab="LOF", cex=score*5, col="coral1")
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
    lty=1, lwd=1.5, col="navyblue")
apply(data.frame(id=1:length(score), score=score), 1, FUN=add.seg)
```

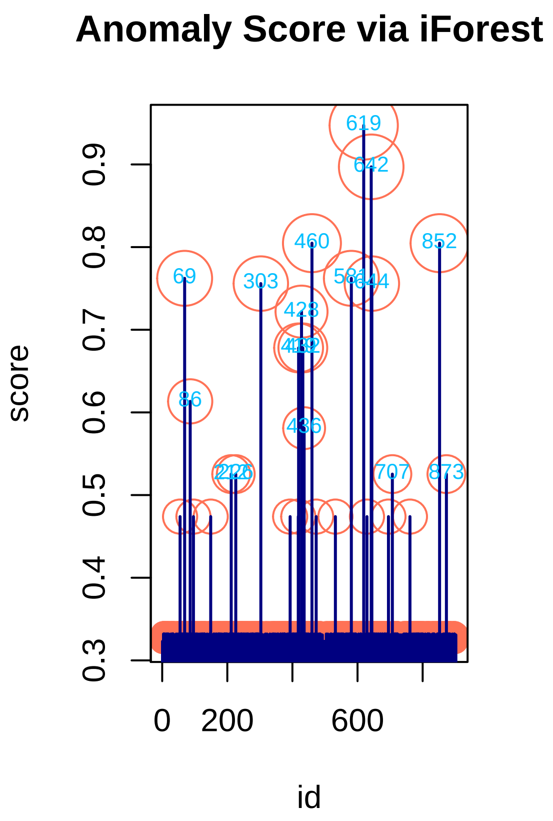
```
## NULL
```

```r
eps <- 0.98
id.outliers <- which(outlier.scores > quantile(outlier.scores, eps))
text(id.outliers, score[id.outliers]+0.02, label=id.outliers,
    col="deepskyblue1", cex=0.7)


# iForest
plot(x=1:length(Ascore), Ascore, type="p", pch=1,
    main="Anomaly Score via iForest",
        xlab="id", ylab="score", cex=Ascore*5, col="coral1")
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
    lty=1, lwd=1.5, col="navyblue")
apply(data.frame(id=1:length(Ascore), score=Ascore), 1, FUN=add.seg)
```

```
## NULL
```

```r
eps <- 0.98
id.outliers <- which(Ascore > quantile(Ascore, eps))
text(id.outliers, Ascore[id.outliers]+0.003, label=id.outliers,
    col="deepskyblue1", cex=0.7)
```





Comparing the LOF and iForest methodologies for detecting potential outliers, It is observed from both plots that the parts 581 and 619 are indeed anomalies. Since they were among the list of potential or expected outliers for the two methods. The LOF was able to adequately detect the defective parts compared to the iForest methodology as evident in the plot above. From the LOF plot the “correct outliers” are clearly seperated from list of other potential outliers, this is not the case for iForest. Thus we conclude that in this situation or scenario, the anomaly detection using iForest is weaker compared to LOF .