Project-III

Md Al Masum Bhuiyan

October 22, 2018

Contents

1 Problem 1 - Page Rank

I have obtained a connectivity matrix L from Figure-1 (links among several webpages) and computed the Pagerank score for each page.

```
# defining matrix columnwise
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)
(L = t(L)) #taking transpose, since page j points to page i
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    0    1    0    0    0    0    0
## [2,]    0    0    0    1    1    0    0
## [3,]    1    0    0    0    0    1    0
## [4,]    1    0    1    0    1    1    0
## [5,]    0    0    0    1    0    0    0
## [6,]    0    0    0    0    0    0    0
## [7,]    0    0    0    0    1    1    0
```

Here, I did not just count the number of linking webpages, *i.e.*, did not want to treat all linking webpages equally. Instead of this, I weight the links from different webpages, and considered the webpages that link to i, and have high PageRank scores themselves, as more weight.

In the code below, the parameter d is a damping factor which I set 0.85 (between 0 and 1). The Pagerank score vector pg corresponds to the eigenvector of a particular matrix A corresponding to eigenvalue 1.

```
pagerank <- function(L, method='eigen', d=.85, niter=100){
  cvec <- apply(L,2,sum) # COMPUTING COLUMN SUMS
  cvec[cvec==0] <- 1 # nodes with degree 0 will cause problems if we divide by 0.
  n <- nrow(L)
  delta <- (1-d)/n
  A <- matrix(delta, nrow(L), ncol(L))
  for (i in 1:n)
    A[i,] <- A[i,] + d*L[i,]/cvec
  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)
}
```

So the PageRank scores have been calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the webpages. The scores form a probability distribution over web pages, so the sum of all web pages, PageRank scores will be 1.

```
pg <- pagerank(L, method = 'eigen') # Use Eigen method
sum(pg)
```

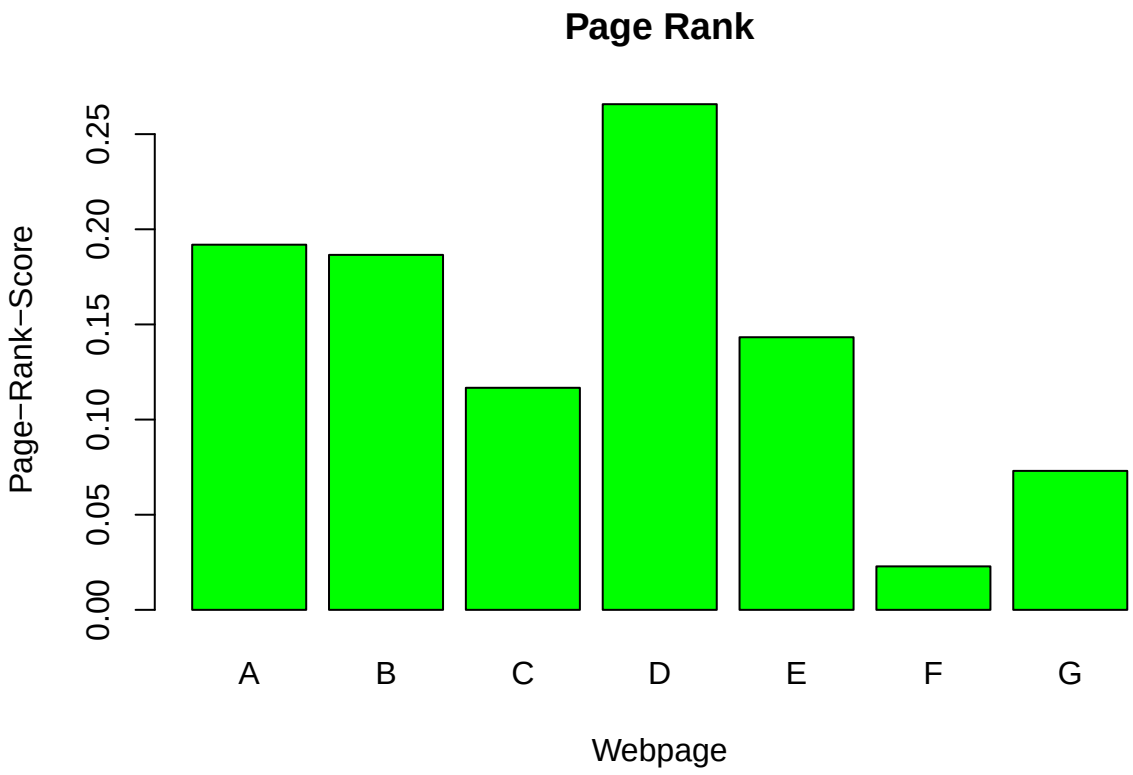
```
## [1] 1
```

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

```
##   Names Rank_Score
## 1     A 0.19189501
## 2     B 0.18653569
## 3     C 0.11670092
## 4     D 0.26573770
## 5     E 0.14325934
## 6     F 0.02284669
## 7     G 0.07302466
```

I provided the barplot for pagerank of webpages as follows:

```
barplot(pg$Rank_Score, names = pg$Names, xlab = "Webpage", ylab =
          "Page-Rank-Score",col="green", main = "Page Rank")
```



The list of top-3 scores is as follows:

```
top <- pg[order(-pg$Rank_Score), ]
head(top, n=3)
```

```
##   Names Rank_Score
## 4     D  0.2657377
## 1     A  0.1918950
## 2     B  0.1865357
```

2 Problem 2 (Anomaly Detection)

We consider the HTP (high tech part) data available from R Package ICSOutlier.

2.1 Problem 2(a)

The HTP data set contains 902 high-tech parts designed for consumer products characterized by 88 tests. These tests are performed to ensure a high quality of the production. All these 902 parts were considered functional and have been sold. However, the two parts 581 and 619 showed defects in use and were returned to the manufacturer by the customer. Therefore these two can be considered as outliers.

```
#install.packages("ICSOutlier")
library("ICSOutlier")
data(HTP)
dat <- HTP;
dim(dat);
```

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

```
head(dat)
```

```
##             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
```

```
n <- nrow(dat)
p <- ncol(dat)
outliers.true <- c(581, 619)
```

2.2 Problem 2(b)

I now use a distance method - Minimum Covariance Determinant (MCD) for anomaly outlier detection with multivariate data. The squared Mahalanobis distance from a center of the data is used to rank data. In the presence of outliers, both mean and sample variance-covariance matrix can be severely affected. Here the objective is to seek m out of n points that yields the ellipsoid with minimum volume, or equivalently, to seek m out of n points whose classical sample covariance matrix has the lowest determinant.

```
library(mvoutlier)
library(robustbase)
# To detect potential outliers I obtain MCD estimates with a breakdown point of 2.5%
fit.robust <- covMcd(dat, cor = FALSE, alpha = 0.975)
mean1 <- fit.robust$center
head(mean1)
```

```
##           V.1           V.2           V.3           V.4           V.5
## -4.237127e-06 -1.746298e-06 -9.233458e-06 -1.736818e-06 -1.704013e-06
##           V.6
##  8.024468e-07
```

```
vcov <- fit.robust$cov
head(vcov, 3)
```

```
##               V.1           V.2           V.3           V.4           V.5
## V.1  1.453915e-07 -3.726662e-09  1.671378e-07 -3.690369e-09 -3.685719e-09
## V.2 -3.726662e-09  9.293819e-10 -4.653561e-09  9.231512e-10  9.179876e-10
## V.3  1.671378e-07 -4.653561e-09  1.941633e-07 -4.608297e-09 -4.601648e-09
##               V.6           V.7           V.8           V.9          V.10
## V.1  1.025971e-07 -3.204452e-09  4.534386e-08  1.150770e-07  1.716472e-07
```

```
## V.2 -2.160018e-09  9.080790e-10 -6.295271e-10 -1.058590e-08 -5.474373e-09
## V.3  1.156520e-07 -4.032449e-09  5.332728e-08  1.408883e-07  2.033703e-07
##              V.11          V.12          V.13         V.14          V.15
## V.1 -4.271234e-09  1.760666e-07 -3.682719e-09 2.105975e-08  1.684636e-07
## V.2  1.080333e-09 -5.122272e-09  9.256226e-10 9.368513e-11 -5.455690e-09
## V.3 -5.345656e-09  2.058137e-07 -4.601706e-09 2.473818e-08  2.000066e-07
##              V.16          V.17          V.18          V.19          V.20
## V.1 -3.635124e-09  1.531412e-07  6.646641e-08 -1.603978e-09 -3.723077e-09
## V.2  9.111830e-10 -4.029195e-09 -1.002505e-09  5.527745e-11  9.279722e-10
## V.3 -4.539346e-09  1.766223e-07  7.322651e-08 -1.883681e-09 -4.647581e-09
##              V.21          V.22         V.23          V.24          V.25
## V.1  2.461825e-08 -3.677981e-09 -3.70214e-09 -4.350142e-09  8.409219e-08
## V.2 -1.312063e-09  9.148381e-10  9.24837e-10  4.439885e-10 -1.652020e-09
## V.3  2.942246e-08 -4.590185e-09 -4.62260e-09 -5.302776e-09  9.845181e-08
##              V.26          V.27          V.28          V.29          V.30
## V.1 -3.632804e-09 -4.852707e-09 -1.756410e-08 -2.470912e-09  1.773899e-07
## V.2  9.144292e-10  2.995288e-10  1.932977e-09  8.878656e-10 -5.262661e-09
## V.3 -4.539771e-09 -5.826187e-09 -2.151591e-08 -3.162034e-09  2.079580e-07
##              V.31          V.32          V.33          V.34          V.35
## V.1 -1.918461e-10 -9.882795e-08  1.565330e-07 -3.753467e-09  5.239185e-08
## V.2  2.034946e-11  2.293082e-08 -4.168133e-09  9.361350e-10 -6.087789e-10
## V.3 -2.337972e-10 -1.241170e-07  1.807966e-07 -4.687299e-09  5.705649e-08
##              V.36          V.37          V.38          V.39          V.40
## V.1  2.298668e-08 -3.354992e-09  1.205958e-07  1.737818e-07 -1.534497e-10
## V.2 -1.270103e-09  2.494965e-10 -2.776156e-09 -4.988661e-09 -2.175009e-11
## V.3  2.760866e-08 -3.982979e-09  1.371174e-07  2.027556e-07 -1.177023e-11
##              V.41          V.42          V.43          V.44          V.45
## V.1  1.776983e-07  2.377362e-08 -3.693374e-09  1.711073e-07 -3.484511e-09
## V.2 -5.327043e-09 -3.991705e-09  9.219102e-10 -4.814666e-09  9.157483e-10
## V.3  2.086425e-07  2.857078e-08 -4.612190e-09  1.990503e-07 -4.364978e-09
##              V.46          V.47          V.48          V.49          V.50
## V.1  1.068333e-07  1.118435e-07 -1.927120e-10  8.231249e-08 -2.098896e-08
## V.2 -1.057821e-08 -2.475332e-09  2.236536e-11 -1.486359e-09  1.021397e-09
## V.3  1.312487e-07  1.266512e-07 -2.365157e-10  9.168636e-08 -2.597994e-08
##              V.51          V.52          V.53          V.54          V.55
## V.1  4.138006e-08  1.142014e-07  1.623434e-07 -5.833626e-09 -3.671648e-09
## V.2 -4.218967e-09 -1.544182e-09 -4.407146e-09  4.406771e-10  9.243186e-10
## V.3  4.990852e-08  1.295302e-07  1.879373e-07 -6.936611e-09 -4.587403e-09
##              V.56          V.57          V.58          V.59         V.60
## V.1 -1.609479e-09  1.186765e-07  1.683240e-07  1.785509e-07 6.055215e-09
## V.2  1.149690e-10 -2.219111e-09 -5.402747e-09 -5.372274e-09 6.979156e-10
## V.3 -1.928282e-09  1.373204e-07  1.996883e-07  2.098715e-07 6.928942e-09
##              V.61          V.62          V.63          V.64          V.65
## V.1  1.302511e-07 -3.759649e-09 -3.030377e-09  1.751807e-07 -3.729624e-09
## V.2 -3.132812e-09  9.404251e-10  2.249937e-10 -5.522317e-09  9.305679e-10
## V.3  1.487006e-07 -4.695548e-09 -3.632547e-09  2.071943e-07 -4.658116e-09
##              V.66          V.67          V.68          V.69          V.70
## V.1 -2.312227e-09 -3.743964e-09  9.380604e-08 -3.889388e-09 -3.722783e-09
```

```
## V.2  9.942531e-11  9.321262e-10 -1.858068e-09  1.004404e-09  9.227289e-10
## V.3 -2.700913e-09 -4.674950e-09  1.052442e-07 -4.870111e-09 -4.646057e-09
##              V.71          V.72          V.73          V.74          V.75
## V.1  3.317173e-08 -3.779282e-09  1.659226e-10  1.790661e-07  7.466845e-08
## V.2 -2.456240e-10  9.407743e-10  7.829497e-10 -5.565562e-09 -1.229931e-09
## V.3  3.577903e-08 -4.719002e-09 -2.356207e-11  2.113317e-07  8.275384e-08
##              V.76          V.77          V.78          V.79          V.80
## V.1  4.431938e-08  2.535115e-08  6.127061e-08 -2.403093e-08  3.976298e-08
## V.2 -4.050580e-10 -1.380630e-09 -8.549280e-10  1.707292e-08 -3.380114e-10
## V.3  4.788688e-08  3.032111e-08  6.727552e-08 -3.153245e-08  4.291778e-08
##              V.81          V.82          V.83          V.84          V.85
## V.1  1.725481e-07  1.803511e-07 -2.598666e-10 -3.707804e-09  1.359798e-07
## V.2 -5.385142e-09 -5.501030e-09 -3.958187e-11  9.250239e-10 -3.358525e-09
## V.3  2.038024e-07  2.122558e-07 -1.353101e-10 -4.629063e-09  1.556680e-07
##              V.86          V.87         V.88
## V.1 -4.093431e-09 -1.760415e-08 7.034681e-08
## V.2  1.041498e-09  1.943747e-09 1.630103e-10
## V.3 -5.124032e-09 -2.156338e-08 7.625068e-08
```

In this case, I obtain the estimation of mean ($\vec{\mu}$) and VCOV matrix ($\vec{\sum}$) with a breakdown point $\alpha = 0.975$, providing the squared Mahalanobis distance (RD) for each points in data.

```
# Robust (Squared) Mahalanobis distance with MCD results
RD <- mahalanobis(dat, mean1, vcov)
head(RD)
```

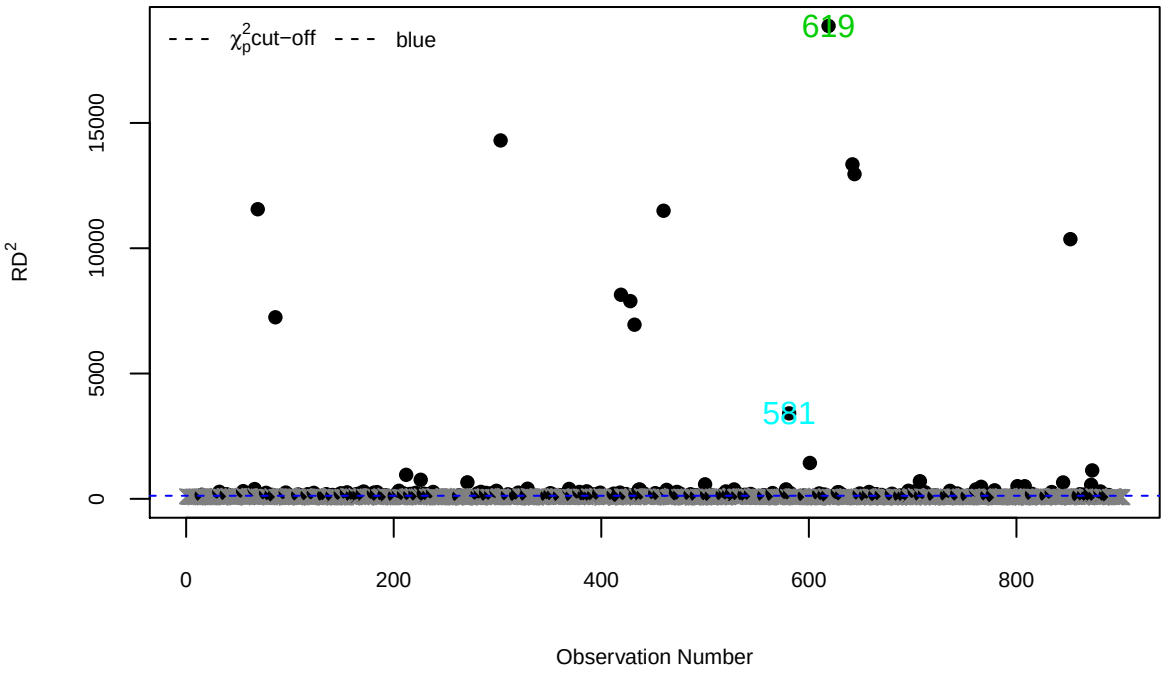
```
## [1] 99.91964 85.42118 83.22738 76.69094 79.98647 74.61408
```

Now I plot the result using a cut-off based on the chi-square distribution.

```
# Cut-off based on the chi-square distribution
(cutoff.chi.sq <- qchisq(0.99, df = ncol(dat)))
```

```
## [1] 121.7671
```

```
# Plot the RESULTS
colPoints <- ifelse(RD >= cutoff.chi.sq, 1, grey(0.5))
pchPoints <- ifelse(RD >= cutoff.chi.sq, 16, 4)
plot(seq_along(RD), RD, pch = pchPoints, col = colPoints,
     ylim=c(0, max(RD, cutoff.chi.sq) + 2), cex.axis = 0.7, cex.lab = 0.7,
     ylab = expression(RD**2), xlab = "Observation Number")
abline(h = cutoff.chi.sq, lty = "dashed", col="blue")
legend("topleft", lty = "dashed", cex = 0.7, ncol = 2, bty = "n",
legend = expression(paste(chi[p]**2, "cut-off"), col="blue"))
text(619, RD[619], labels=619, col=619)
text(581, RD[581], labels=581, col=581)
```



Here, I see almost 13 observations clearly stand out using the cutoff value, and the outlier 581, 619 are present their as well. So the two defective parts in the top list are potential outliers.

2.3 Problem 2(c)

I now use isolation forest algorithm to score the observation and to detect anomaly. Since isolation forest is an unsupervised learning algorithm, it will only use feature data. I therefore first train a model of Isolation Forest and then evaluate the anomaly score using that model. Here the "IsolationTrees " function builds random binary trees and provide a data structure that represent an Isolation Forest model. The rfactor is a randomisation factor, range from 0 to 1, where 0 for fully deterministic, 1 for fully random. Here I use fully deterministic to detect potential outlier.

```
library(IsolationForest)
```

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

```
# train a model of Isolation Forest
istr <- IsolationTrees(dat, rFactor=0)
```

The Anomalyscore function passes each sample in data through forest to obtain expected path length. It then uses path length and parameter in forest to determine anomaly score.

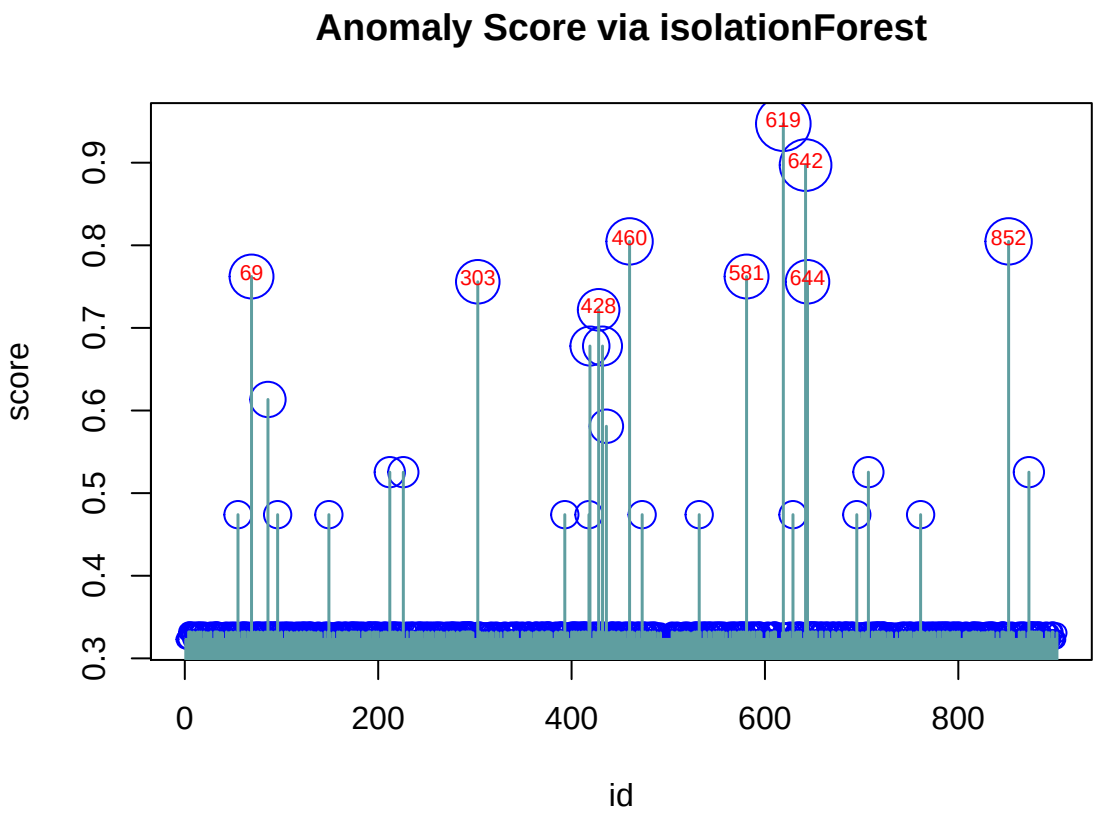
```
#evaluate anomaly score
eas <- AnomalyScore(dat, istr)
# show anomaly score
score_iso <- eas$outF # anomaly scores of samples in data
```

Here I plot the scores. The size of the circle of outliers represnets the potentiality. The bigger the circle, the more potential it is.

```
par(mfrow=c(1,1), mar=rep(4,4))
plot(x=1:length(score_iso), score_iso, type="p", pch=1,
     main="Anomaly Score via isolationForest",
     xlab="id", ylab="score", cex=score_iso*4, col="blue")
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
                                lty=1, lwd=1.5, col="cadetblue")
apply(data.frame(id=1:length(score_iso), score=score_iso), 1, FUN=add.seg)
```

```
## NULL
```

```
eps <- 0.99
id.outliers <- which(score_iso > quantile(score_iso, eps))
text(id.outliers, score_iso[id.outliers]+0.005, label=id.outliers,
     col="red", cex=0.7)
```



2.4 Local Outlier Factor (LOF) Method

The LOF measure essentially captures a local outlier whose local density is relatively low comparing to the local densities of its K-distance nearest neighborhood (KNN). It assigns a degree of being an outlier to each observation. It is local in that the degree depends on how isolated the observation is with respect to the surrounding neighborhood. The lower the local reachability density of x ($x_i \in$ data), and the higher the local reachability density of the KNN of x, the higher LOF. I chose the parameter k, size of the neighborhood, as 7.

```
library(Rlof)
outlier.scores <- lof(dat, k=7)
which(outlier.scores > quantile(outlier.scores, 0.95))
```

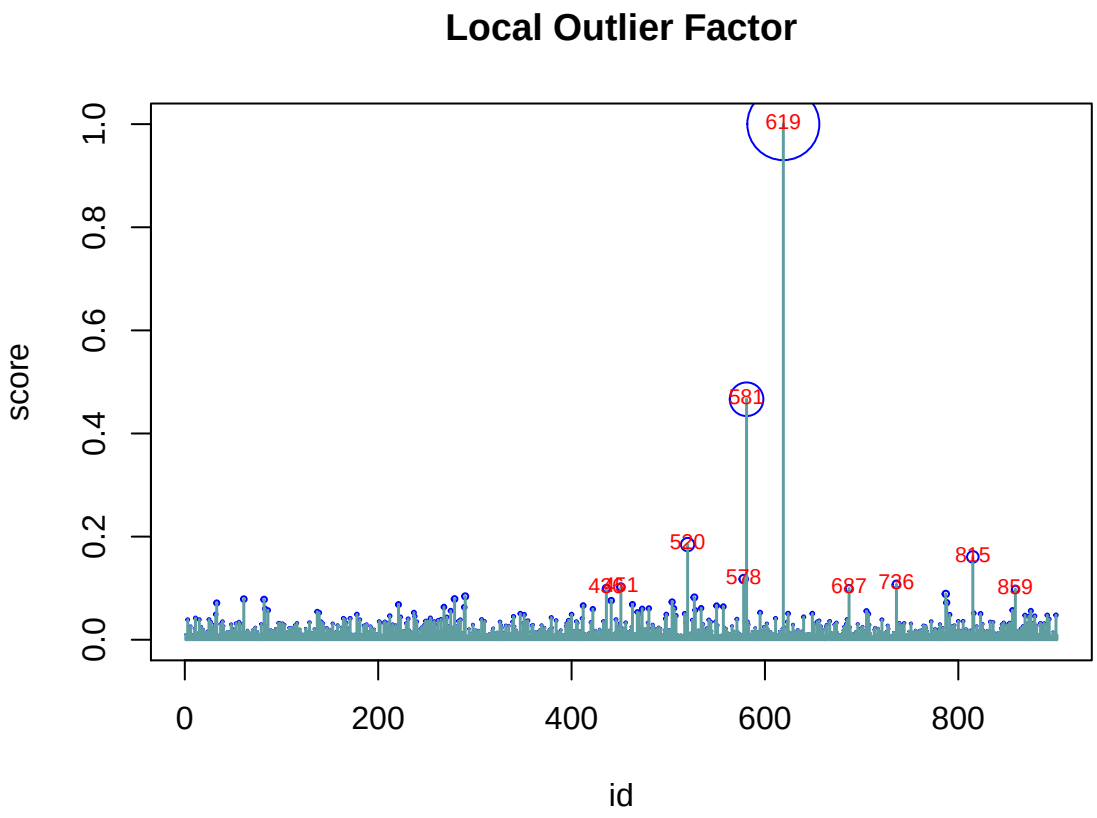
```
##  [1]  33  61  82  83  86 137 139 221 237 268 275 279 289 290 412 422 436
## [18] 441 451 463 468 470 473 480 504 506 520 527 528 534 550 557 578 581
## [35] 595 619 687 705 736 787 788 815 816 856 859 875
```

Here I plot the scores. The size of the circle of outliers represents the potentiality. The bigger the circle, the more potential it is.

```
# PLOT OF THE LOF SCORES
score_lof <- 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_lof), score_lof, type="p", pch=1,
     main="Local Outlier Factor",
     xlab="id", ylab="score", cex=score_lof*5, col="blue")
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
                                lty=1, lwd=1.5, col="cadetblue")
apply(data.frame(id=1:length(score_lof), score=score_lof), 1, FUN=add.seg)
```

```
## NULL
```

```
eps <- 0.99
id.outliers <- which(score_lof > quantile(score_lof, eps))
text(id.outliers, score_lof[id.outliers]+0.005, label=id.outliers,
     col="red", cex=0.7)
```



2.4.1 Comparison

Now I discuss the comparison between two methods: Isolation Forest and LOF to detect anomalies.

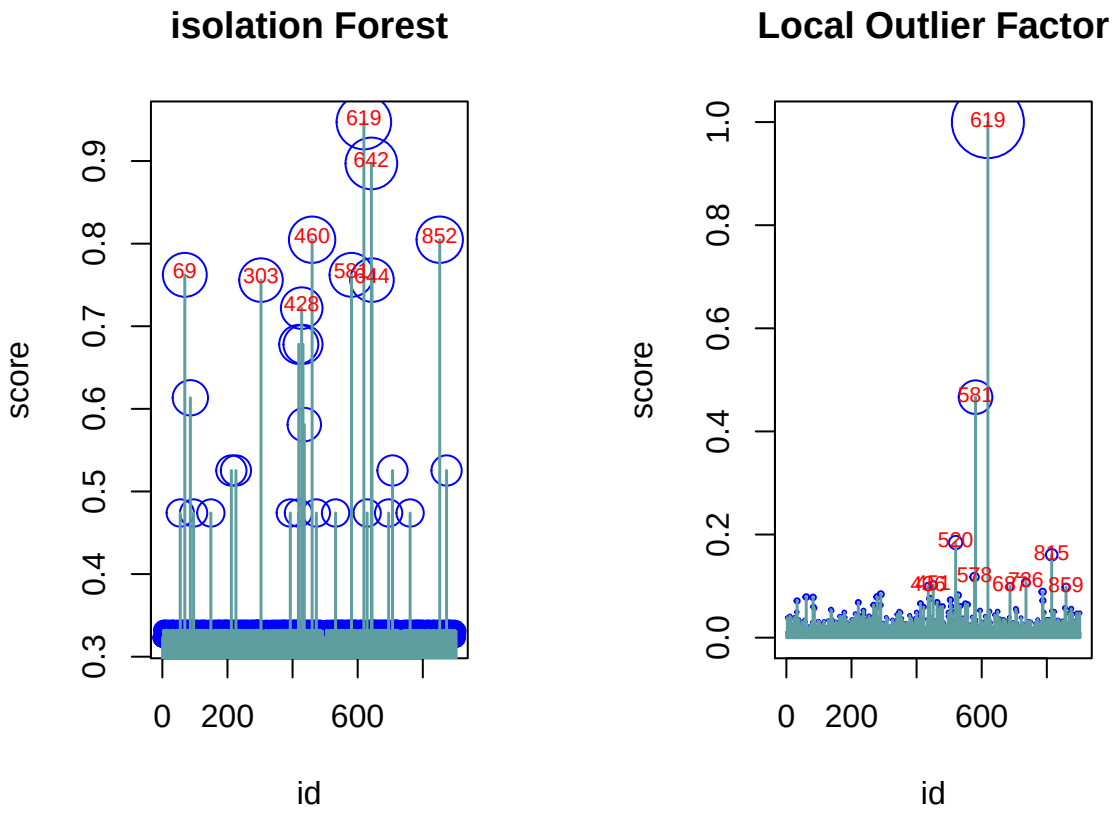
```
par(mfrow=c(1,2), mar=rep(4,4))
plot(x=1:length(score_iso), score_iso, type="p", pch=1,
     main="isolation Forest",
     xlab="id", ylab="score", cex=score_iso*4, col="blue")
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
                                lty=1, lwd=1.5, col="cadetblue")
apply(data.frame(id=1:length(score_iso), score=score_iso), 1, FUN=add.seg)
```

```
## NULL
```

```
eps <- 0.99
id.outliers <- which(score_iso > quantile(score_iso, eps))
text(id.outliers, score_iso[id.outliers]+0.005, label=id.outliers,
     col="red", cex=0.7)
#=====================================================
plot(x=1:length(score_lof), score_lof, type="p", pch=1,
     main="Local Outlier Factor",
     xlab="id", ylab="score", cex=score_lof*5, col="blue")
add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],
                                lty=1, lwd=1.5, col="cadetblue")
apply(data.frame(id=1:length(score_lof), score=score_lof), 1, FUN=add.seg)
```

```
## NULL
```

```
eps <- 0.99
id.outliers <- which(score_lof > quantile(score_lof, eps))
text(id.outliers, score_lof[id.outliers]+0.005, label=id.outliers,
     col="red", cex=0.7)
```



Remarks:

In this high tech data, we already know that the observation 581 and 619 are defectives. The similarity of these two methods are that both of them can detect the potential anomalies. In the plot we see, both of them detect the defectives (619 and 581) very well.

On the other hand, there are some dissimilarities as well. The circle of 518 and 619 is much bigger than the other observations in the case of LOF method, where the Isolation Forest method does not that type of case. The defective parts are far away from the other outliers in the case of LOF method. It proves that the LOF can easily capture the potential outliers. For example, the anomlay 581 is easily deemable in LOF method (more isolated), but not in Isolation Forest method.

So I can say that LOF is more reliable than Isolation Forest in this case.