Project Three

Ruben Solis [*]

Due: 23 October 2018

Contents

1 PageRank

We first begin by translating the information contained in the graph for this problem into a link matrix

```
L <- matrix(c(0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
              0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,
              0, 0, 0, 0, 0, 0, 0), ncol=7, byrow=TRUE)
L
```

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

Next, we need to compute the PageRank values so that we can rank the webpages.

```
pagerank <- function(G, method = 'eigen', d = .85, niter=100){
  cvec <- apply(G, 2, sum) # Computing Columns Sums
  cvec[cvec==0] <- 1 # nodes with indegree 0 will  cause problems
  n <- nrow(G)
```

[*]rsolis5@miners.utep.edu

```
  delta <- (1-d)/n
  A <- matrix(delta, nrow(G), ncol(G))
  for (i in 1:n)  A[i,] <- A[i,] + d*G[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)
}
pg <- pagerank(L)
pg
```

```
## [1] 0.13438041 0.15491010 0.09047138 0.27197905 0.17753139 0.14625695
## [7] 0.02447073
```

Finally, we plot the PageRank given to each webpage in a barplot.

```
barplot(pg, main="PageRank", ylab="PageRank", xlab="Webpage",
        names.arg=LETTERS[1:7])
```

From both the `R` output and the barplot in Figure 1, we see that the top webpages are D, E, and B, respectively.

2 Anomaly Detection

Before we can use the required anomaly detection techniques, we need to bring the data into `R`.

```
suppressMessages(library(ICSOutlier))
data(HTP)
dat <- HTP
known.outliers <- c(581, 619)
```

2.1 MCD

The first anomaly detection technique that we will use MCD. We obtain the robust estimates of the mean vector and variance-covariance matrix with the `covMcd()` function available in the `robustbase` package.

```
library(robustbase)
fit.robust <- covMcd(dat, cor = FALSE, alpha = 0.75)
```

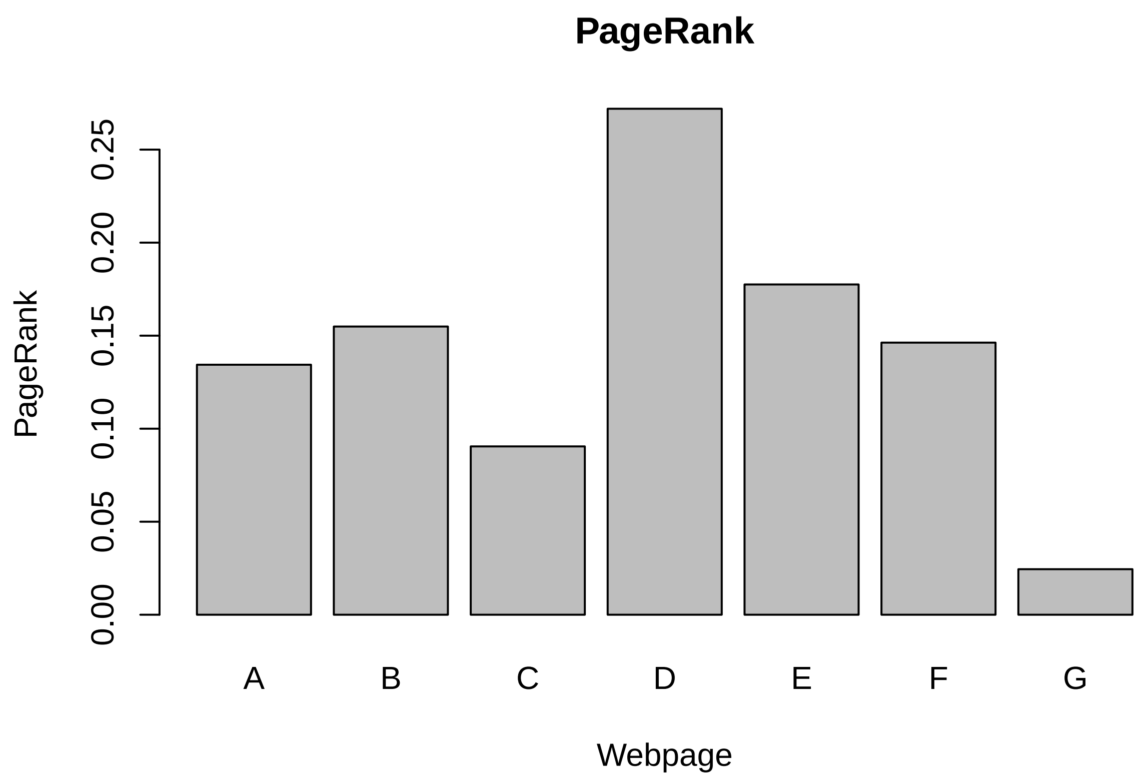

Figure 1: Barplot of the PageRank results from the link matrix.

Next, we compute the robust Mahalanobis distance of each observation with resect to the MCD estimates and plot them with the following code.

```
library(CerioliOutlierDetection)
RD <- mahalanobis(dat, fit.robust$center, fit.robust$cov)
cutoff.chi.sq <- qchisq(0.975, df = ncol(dat))
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
colPoints <- ifelse(RD >= min(c(cutoff.chi.sq, cutoff.GM)), "orange", "grey")
pchPoints <- ifelse(RD >= min(c(cutoff.chi.sq, cutoff.GM)), 16, 4)
labs <- rep(NA, n)
labs[581] <- "581"
labs[619] <- "619"
plot(seq_along(RD), RD, pch = pchPoints, col = colPoints,
     ylim=c(0, max(RD, cutoff.chi.sq, cutoff.GM) + 2), cex.axis = 0.7,
     cex.lab = 0.7, ylab = expression(RD**2), xlab = "Observation Number",
     main = "Minimum Covariance Determinate (MCD)")
abline(h = c(cutoff.chi.sq, cutoff.GM), lty = c("dashed", "dotted"),
       col=c("blue", "red"))
legend("topleft", lty = c("dashed", "dotted"), cex = 0.7, ncol = 2, bty = "n",
       legend = c(expression(paste(chi[p]**2, "cut-off")), "GM cut-off"),
       col = c("blue","red"))
text(RD, labels=labs)
```

We see in Figure 2 that when we use both the χ^2 and GM cutoff, MCD is able to identify our known outliers as outliers.

2.2 iForest

The second anomaly detection technique that we use is isolation forest. We construct 1000 trees and we then plot the outliers with the code below.

```
library(isofor)
fit.isoforest <- iForest(dat, nt = 1000, phi=256)
pred <- predict(fit.isoforest, newdata = dat)
score <- scale(pred, center = min(pred), scale = max(pred)-min(pred))
par(mfrow=c(1,1), mar=rep(4,4))
plot(x=1:length(score), score, type="p", pch=19,
     main="Isolated Forest", xlab="id", ylab="Anomaly score", cex=score*3,
     col="orange")
eps <- 0.99
id.outliers <- which(score > quantile(score, eps))
```

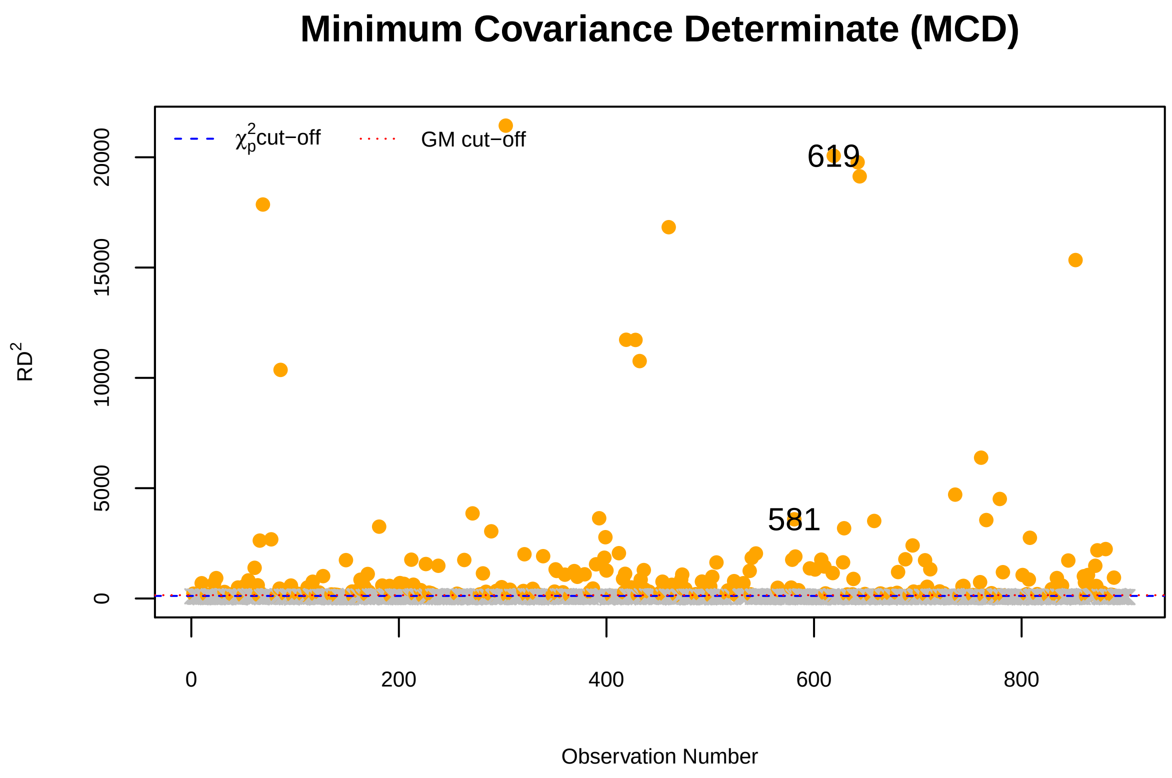

Figure 2: Outlier plot using MCD.

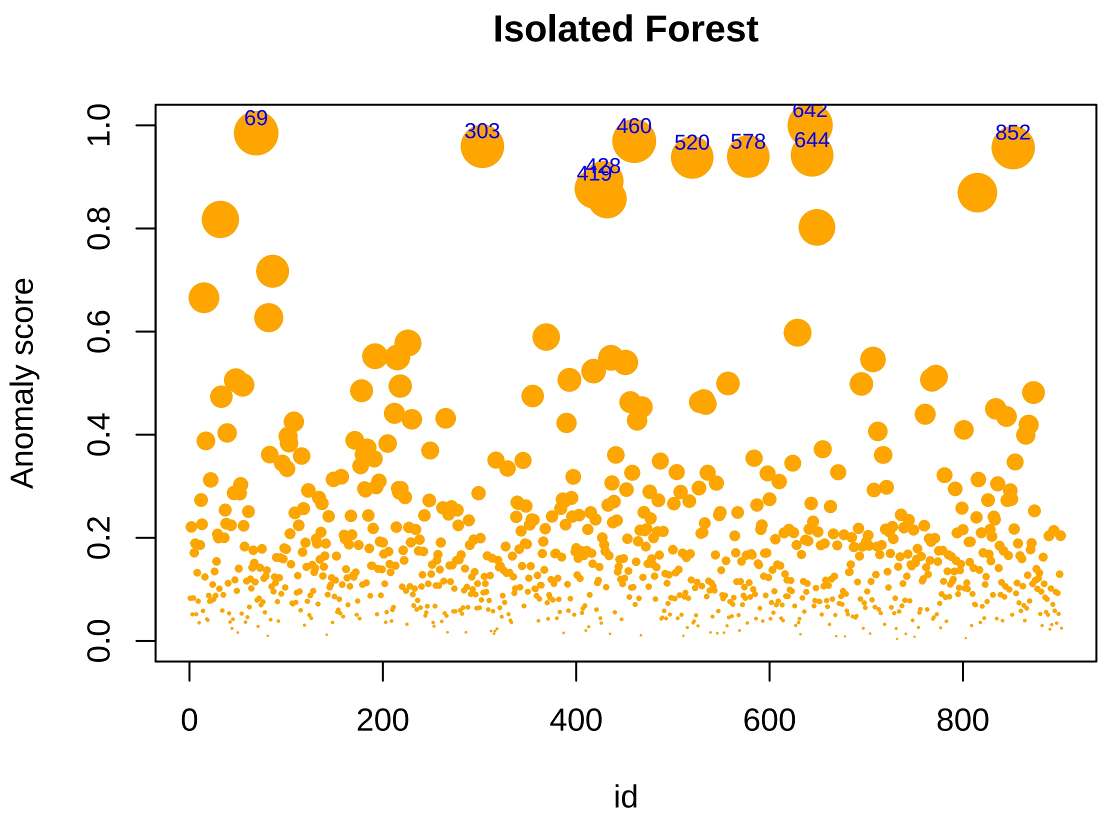

Figure 3: Outlier plot using iForest

```r
text(id.outliers, score[id.outliers]+0.03, label=id.outliers,
     col="blue", cex=0.7)
```

In Figure 3 we that iForest has identified several outliers. Rather than check the id numbers in the plot individually, we can use the `is.element()` function in `R` as shown below.

```r
is.element(known.outliers, id.outliers)
```

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

From the above output we see that iForest was unable to identify our known outliers.

2.3 LOF

The final anomaly detection technique that we use is local outlier factor. We then plot the outliers with the code below.

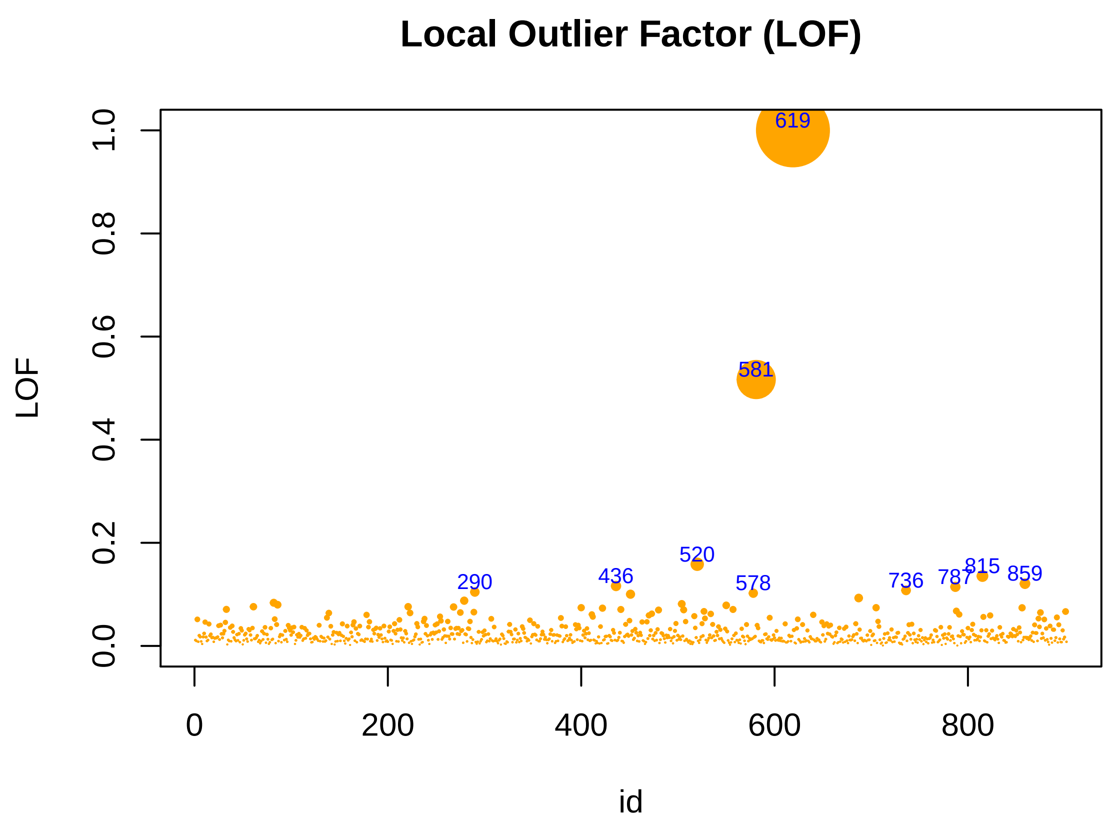

Figure 4: Outlier plot using LOF

```
suppressMessages(library(Rlof))
outlier.scores <- lof(dat, k=5)
score <- scale(outlier.scores, center = min(outlier.scores),
    scale = max(outlier.scores)-min(outlier.scores))
par(mfrow=c(1,1), mar=rep(4,4))
plot(x=1:length(score), score, type="p", pch=19,
     main="Local Outlier Factor (LOF)",
     xlab="id", ylab="LOF", cex=score*5, col="orange")
eps <- 0.99
id.outliers <- which(outlier.scores > quantile(outlier.scores, eps))
text(id.outliers, score[id.outliers]+0.02, label=id.outliers,
     col="blue", cex=0.7)
```

In Figure 4, we see that LOF has identified several outliers. Rather than check the id numbers in the plot individually, we can use the `is.element()` function in R, as shown below.

```
is.element(known.outliers, id.outliers)
```

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

From the above output we see that LOF was able to identify our known outliers.

3 Conclusion

Only two of the methods were able to identify our known outliers: LOF and MCD. iForest was unable to identify any of our known outliers.