Data Mining Assignment 9

Xuan Han
han.xua@husky.neu

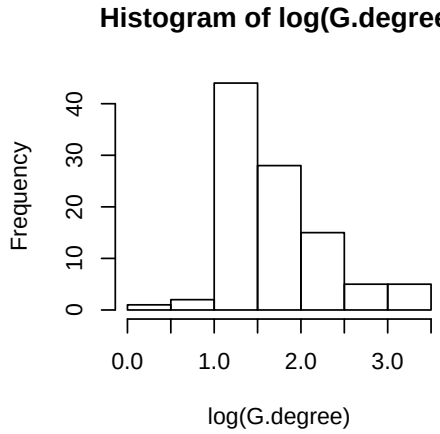
December 8, 2015

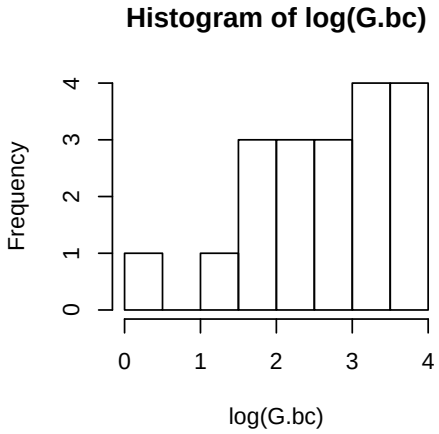
```
> library(igraph)
> edge.table <- read.table("yeast_broad.tsv", header = FALSE, stringsAsFactors = F)
> load("yeast_names.Rdata")
> G <- graph.data.frame(edge.table, vertices = vertex.table, directed = TRUE)
```

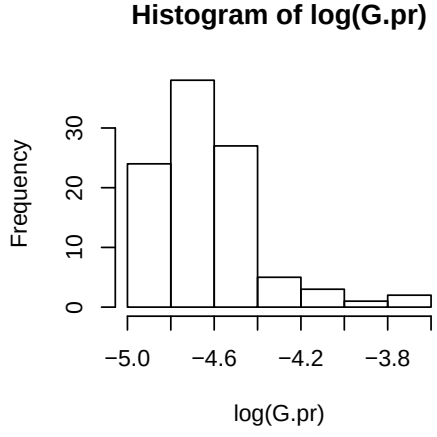
1: Yeast

(a) Degree BC PR

```
> G.degree = degree(G)
> G.bc = betweenness(G)
> G.pr = page.rank(G)$vector
> par(mfrow = c(2,2))
> hist(log(G.degree))
> hist(log(G.bc))
> hist(log(G.pr))
```

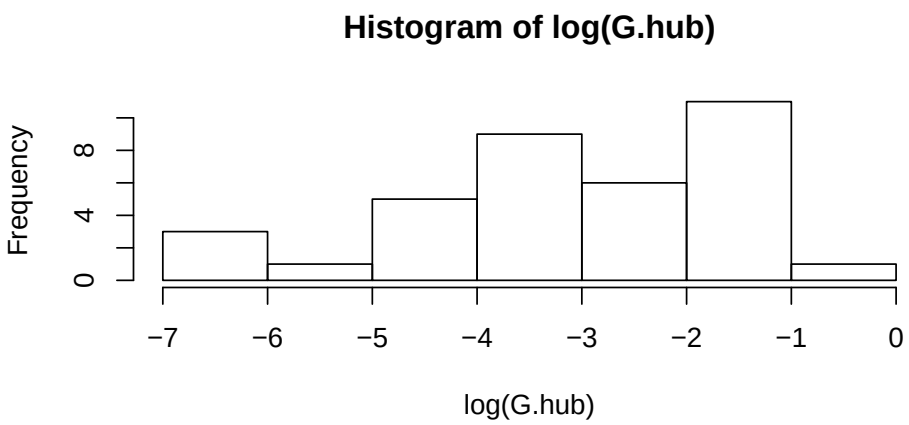


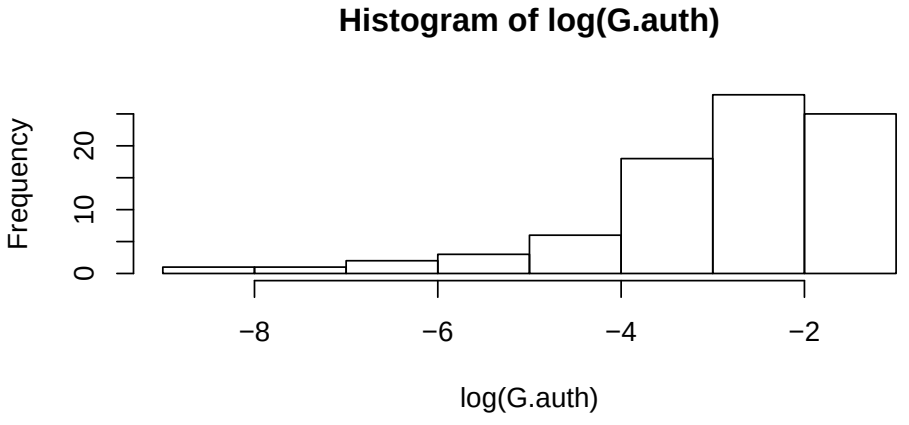




(b)HITS

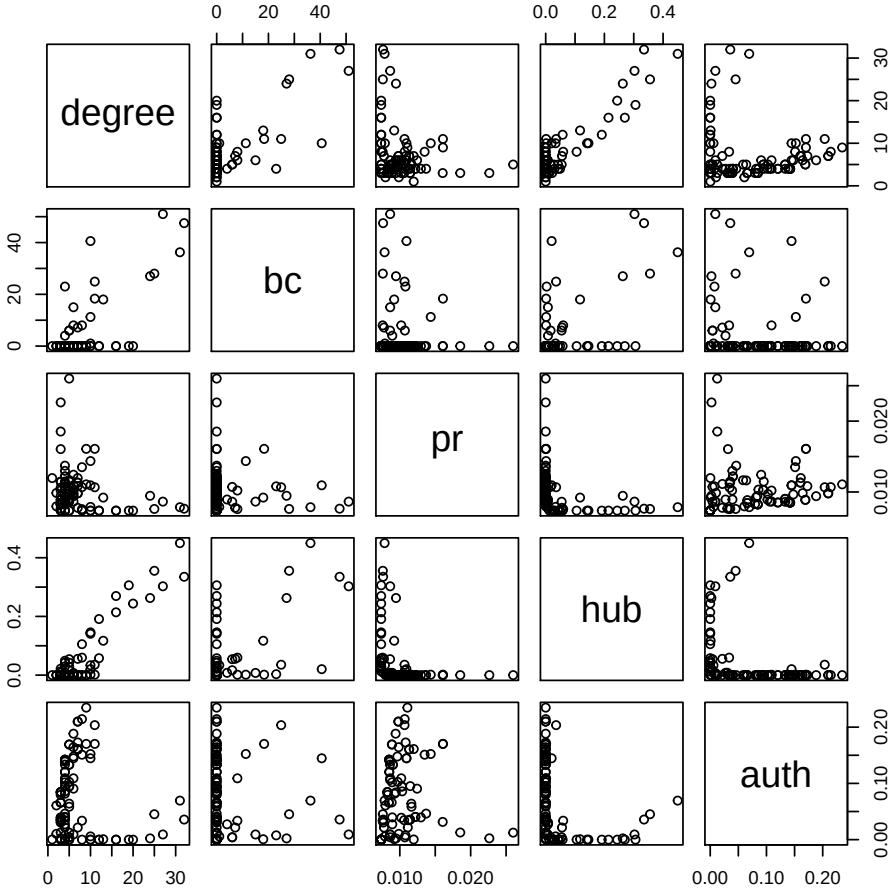
```
> m = as.matrix(get.adjacency(G))
> hits = function(A, round){
+     hub = rep(1, dim(A)[1])
+     auth = rep(1, dim(A)[1])
+
+     while(round > 0) {
+         auth = t(A) %*% hub
+         auth = auth / sqrt(sum(auth ^ 2))
+         hub = A %*% auth
+         hub = hub / sqrt(sum(hub ^ 2))
+         round = round - 1
+     }
+     result <- list(hub=hub,auth=auth)
+     return(result)
+ }
> G.hub = hits(m, 20)$hub
> G.auth = hits(m, 20)$auth
> par(mfrow = c(2, 1))
> hist(log(G.hub))
> hist(log(G.auth))
```





(c) Compare

```
> feature = data.frame(degree = G.degree, bc = G.bc, pr = G.pr, hub = G.hub, auth = G.auth)
> pairs(feature)
```

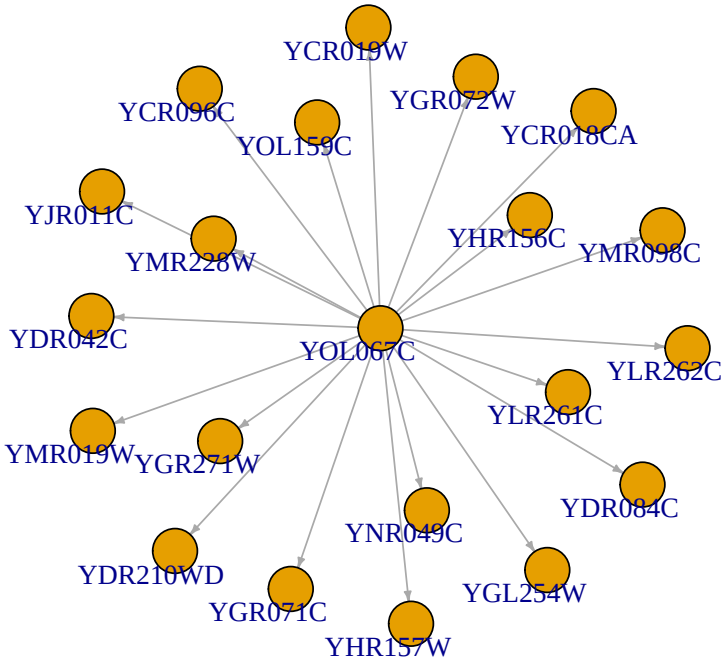



```
> gap = abs(rank(G.auth) - rank(G.degree))
> rank(gap)
```

```
 YCL066W  YCL067C  YCR040W  YCR096C  YCR097W  YDR081C  YDR421W  YDR463W
     1.5      3.0     11.0     16.0     71.5      9.0     49.5     94.0
 YFL021W  YGL035C  YGL209W  YGL254W  YGR044C  YHR006W  YIR017C  YJL056C
    87.0     88.0     95.5      4.5     21.0     57.0     59.0     95.5
 YJL089W  YJL110C  YJR147W  YKL020C  YKR034W  YLR013W  YLR098C  YML051W
    91.0     79.0     49.5     92.5     86.0     99.0     92.5     35.0
 YML099C  YML113W  YMR019W  YMR042W  YMR280C  YNL103W  YOL067C  YOL089C
    73.0     97.5      6.5     49.5     67.0      9.0    100.0     97.5
 YOR032C  YPL038W  YPL248C  YPR199C  YKL178C  YPL187W  YDR103W  YCR039C
    82.0     82.0     90.0     89.0     49.5     14.5     14.5     22.5
 YDR042C  YDR317W  YCL065W YIL002WA YCR018CA  YCR019W  YDR078C  YDR079W
    12.5      9.0     30.0     22.5     25.5     25.5     65.5     65.5
 YDR084C  YGL001C  YGR071C  YGR072W  YHR156C  YHR157W  YJL085W  YJR011C
    34.0     39.0     32.5     32.5     63.5     63.5     68.5     31.0
 YLR261C  YLR262C  YMR228W  YNR049C  YOL126C  YOL159C  YDR040C YDR210WD
    82.0     82.0     82.0     47.0     52.0     58.0     24.0      4.5
 YDR520C  YDR522C  YGR271W  YLR023C  YMR098C  YMR258C  YBR019C  YBR020W
    85.0     12.5     71.5     54.0     60.0     36.5     19.5     19.5
 YDL149W  YDL151C  YDR009W  YLR377C  YOR140W  YCR018C  YCR041W  YDR545W
    44.5     44.5     54.0      6.5     36.5     28.0     28.0     54.0
 YER189W  YGR084C  YNL336W  YNL337W  YNL339C  YER190W YMR086CA  YMR087W
    38.0     40.5     56.0     44.5     44.5     40.5     17.5     17.5
 YNL117W  YBL109W  YBL111C  YBL112C  YBL113C  YBR166C  YDR543C  YDR544C
    28.0     75.5     75.5     75.5     75.5     78.0     70.0     61.5
```

```
 YGR296W  YHR091C  YBL074C  YPL177C
    68.5     61.5      1.5     42.0
```

```
> nei.nodes.names = V(G)[nei('YOL067C')]$name
> sub.nodes.names = c(nei.nodes.names, 'YOL067C')
> sub.nodes = V(G)[sub.nodes.names]
> G.sub = induced.subgraph(G, sub.nodes)
> lay.out <- layout.auto(G.sub)
> plot.igraph(G.sub,
+             layout = lay.out,
+             vertex.label.dist = -.5,
+             edge.arrow.size = .3,
+             main = "Subgraph")
```



1. Let's use degree and authrity score as the two metrics. We find that node 'YOL067C' has biggest gap of the two metrics.

2. After plotting the subgraph, It's clear why this happen: this node has 20 out degrees and 0 in degree. So although it has big degree, it has low authority score, which is 0.

2: Running time

(a)

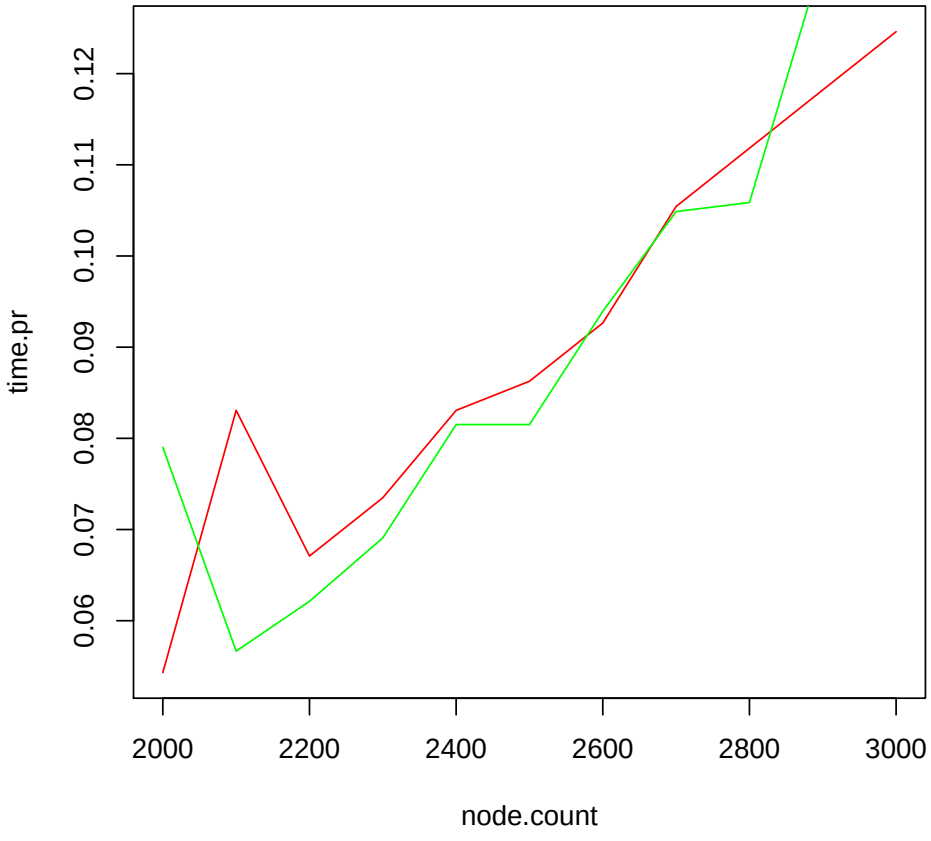
```
> node.count = seq(from = 2000, to = 3000, by = 100)
> time.pr = rep(0, length(node.count))
> time.hits = rep(0, length(node.count))
> for (i in 1:length(node.count)){
+     n = node.count[i]
+     g <- sample_gnp(n, 1 / 20, directed = TRUE)
+     time.pr[i] = system.time(page.rank(g))
+     m = as.matrix(get.adjacency(g))
+     time.hits[i] = system.time(hits(m, 2))
+ }
> time.pr

 [1] 0.017 0.026 0.021 0.023 0.026 0.027 0.029 0.033 0.035 0.037 0.039

> time.hits

 [1] 0.159 0.114 0.125 0.139 0.164 0.164 0.189 0.211 0.213 0.267 0.267

> time.pr = time.pr / sum(time.pr)
> time.hits = time.hits / sum(time.hits)
> plot(node.count, time.pr, col = 'red', type = 'l')
> lines(node.count, time.hits, col = 'green', type = 'l')
```



(b)

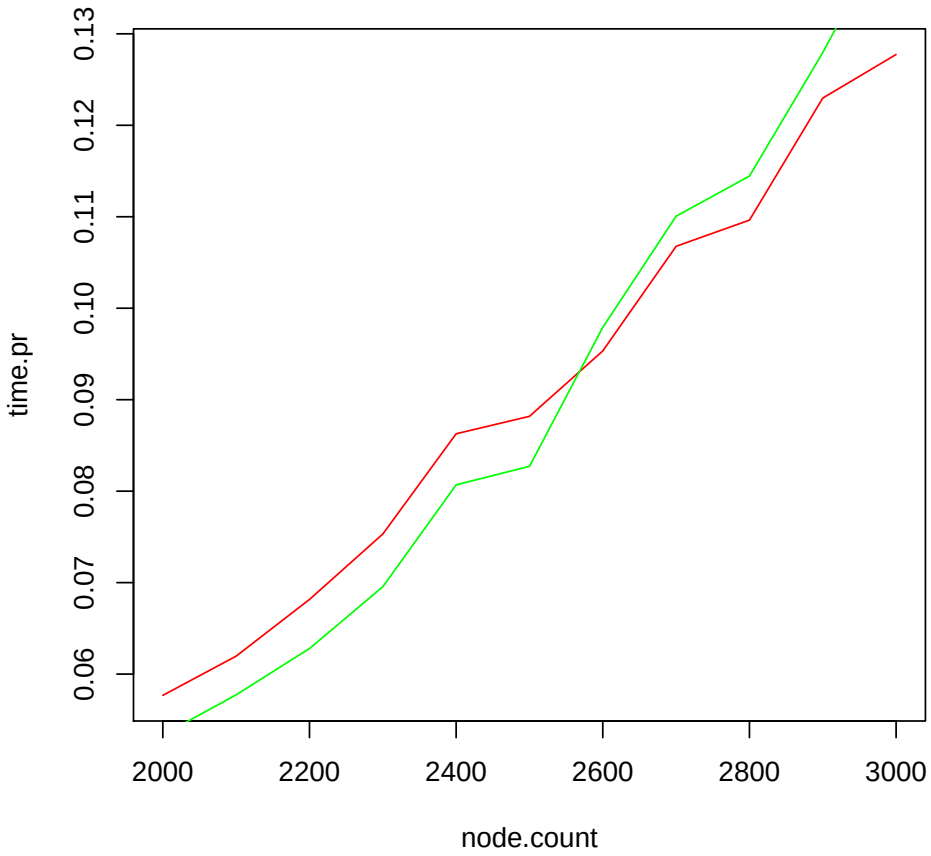
```
> for (i in 1:length(node.count)){
+     n = node.count[i]
+     g <- sample_gnp(n, 1 / 3, directed = TRUE)
+     time.pr[i] = system.time(page.rank(g))
+     m = as.matrix(get.adjacency(g))
+     time.hits[i] = system.time(hits(m, 3))
+ }
> time.pr

 [1] 0.121 0.130 0.143 0.158 0.181 0.185 0.200 0.224 0.230 0.258 0.268

> time.hits

 [1] 0.158 0.171 0.186 0.206 0.239 0.245 0.290 0.326 0.339 0.379 0.423

> time.pr = time.pr / sum(time.pr)
> time.hits = time.hits / sum(time.hits)
> plot(node.count, time.pr, col = 'red', type = 'l')
> lines(node.count, time.hits, col = 'green', type = 'l')
```



1. More nodes results in a increase in time to compute both PageRank and HITS.
2. Higher density result in a increase in time to compute both PageRank and HITS.
3. HITS needs more time than PageRank, which is 10 times, since it needs operation on matrix.
4. I have to scale them inorder to present in the same plot.