HW9 Solution CS6220-Data Mining

```r
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

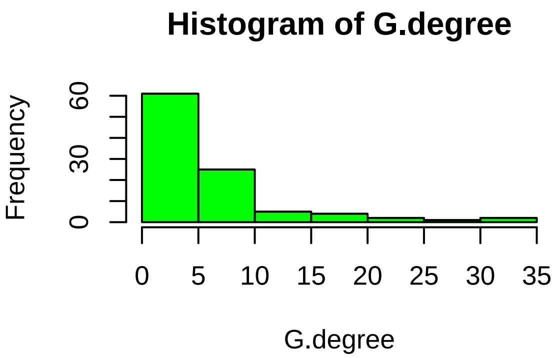
Problem 1

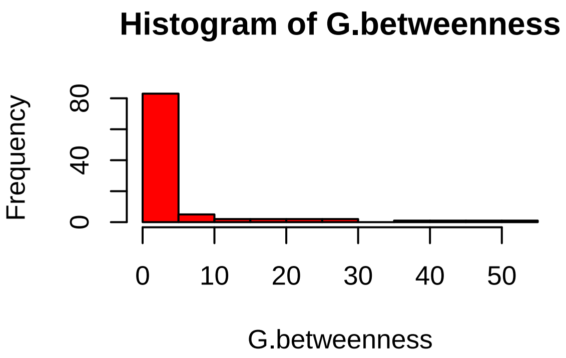
(a)

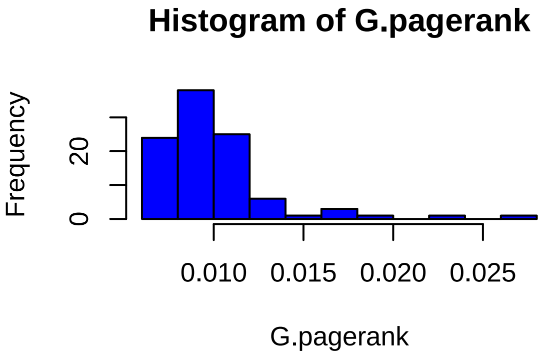
```r
#Load the data
setwd('/Users/ovitek/Dropbox/Olga/Teaching/CS6220/Fall15/Homeworks/Hw9')
mytable <- read.table("yeast_broad.tsv", header = FALSE, stringsAsFactors = FALSE)
load("yeast_names.Rdata")
ls()
```

```
## [1] "mytable"      "vertex.table"
```

```r
G <- graph.data.frame(mytable, vertices = vertex.table, directed = TRUE)

G.degree = degree(G)
G.betweenness = betweenness(G)
G.pagerank = page.rank(G)$vector

par(mfrow = c(2,2))
hist(G.degree, breaks = 10, col = "green" )
hist(G.betweenness, col = "red" )
hist(G.pagerank, col = "blue" )
```

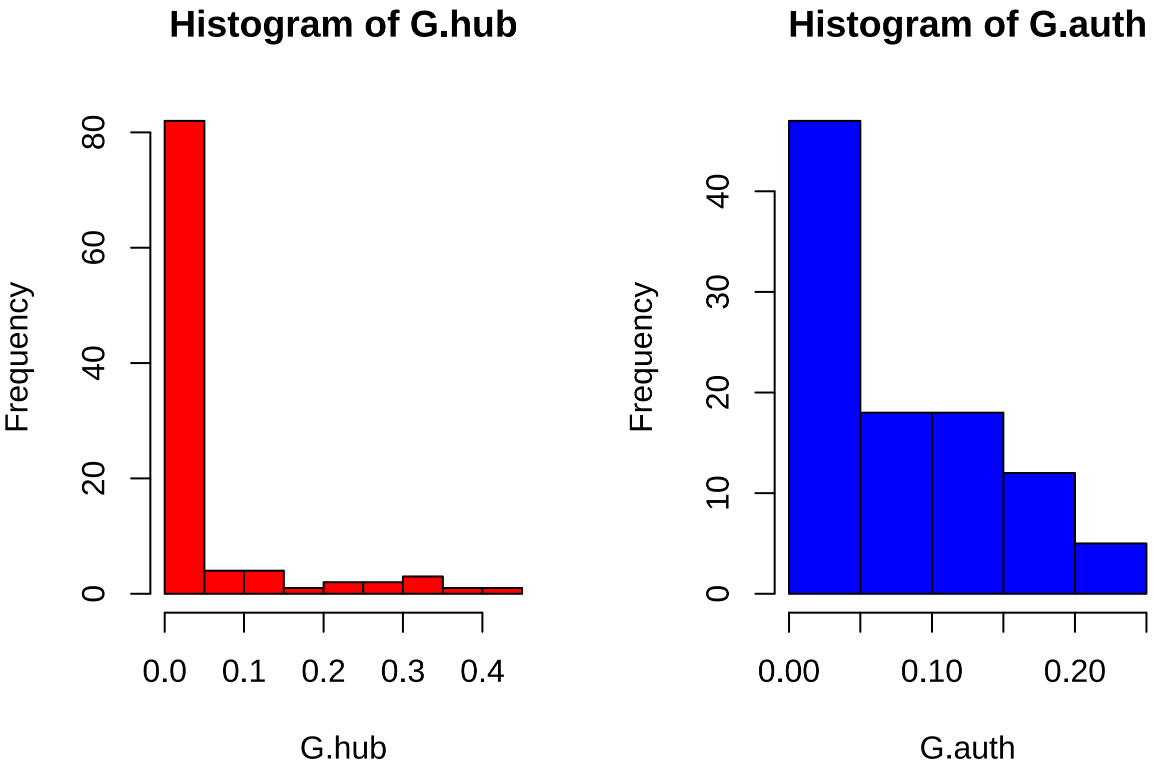






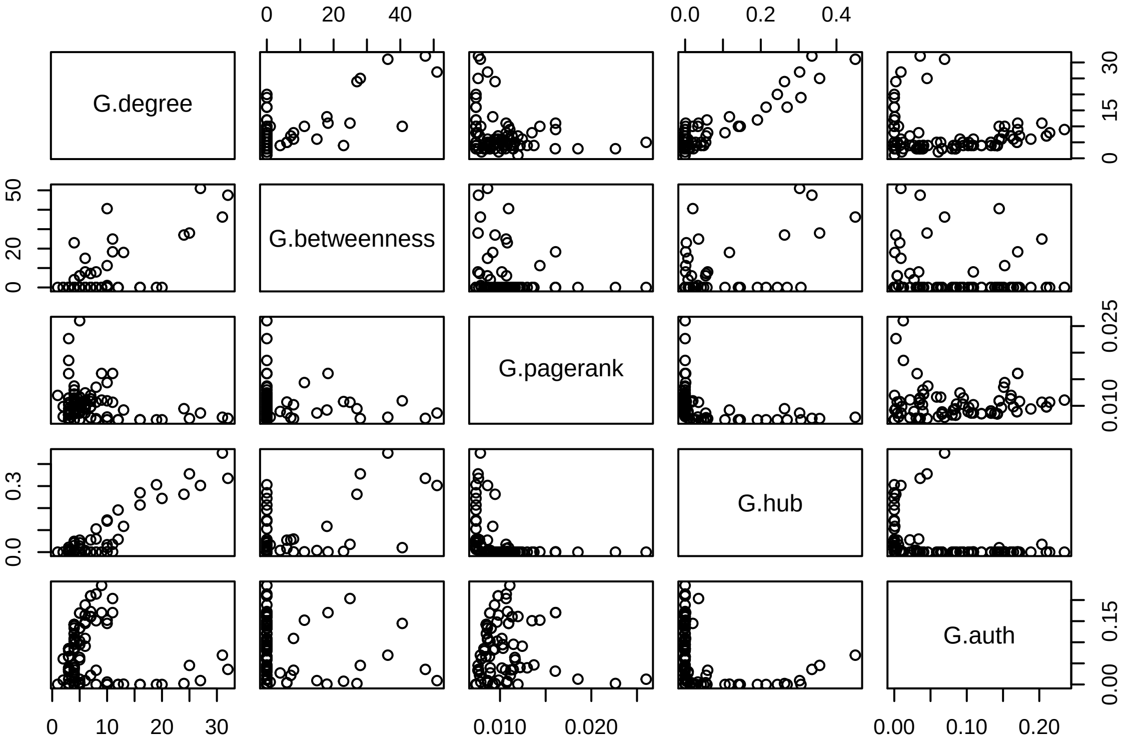
(b)

```
# Implementation of the HITS algorithm
#-----------------------------------
hitsFunc = function(A, times) {
  h = rep(1, dim(A)[1])
  a = rep(1, dim(A)[1])

  for (i in 1:times) {
    a <- t(A) %*% h
    a <- a / sqrt(sum(a ^ 2))
    h <- A %*% a
    h <- h / sqrt(sum(h ^ 2))
  }
  return(list(hub = h, auth = a))
}

myMatrix <- as.matrix(get.adjacency(G))
hits = hitsFunc(myMatrix, 20)
G.hub = hits$hub
G.auth = hits$auth

par(mfrow = c(1,2))
hist(G.hub, col = "red")
hist(G.auth, col = "blue")
```

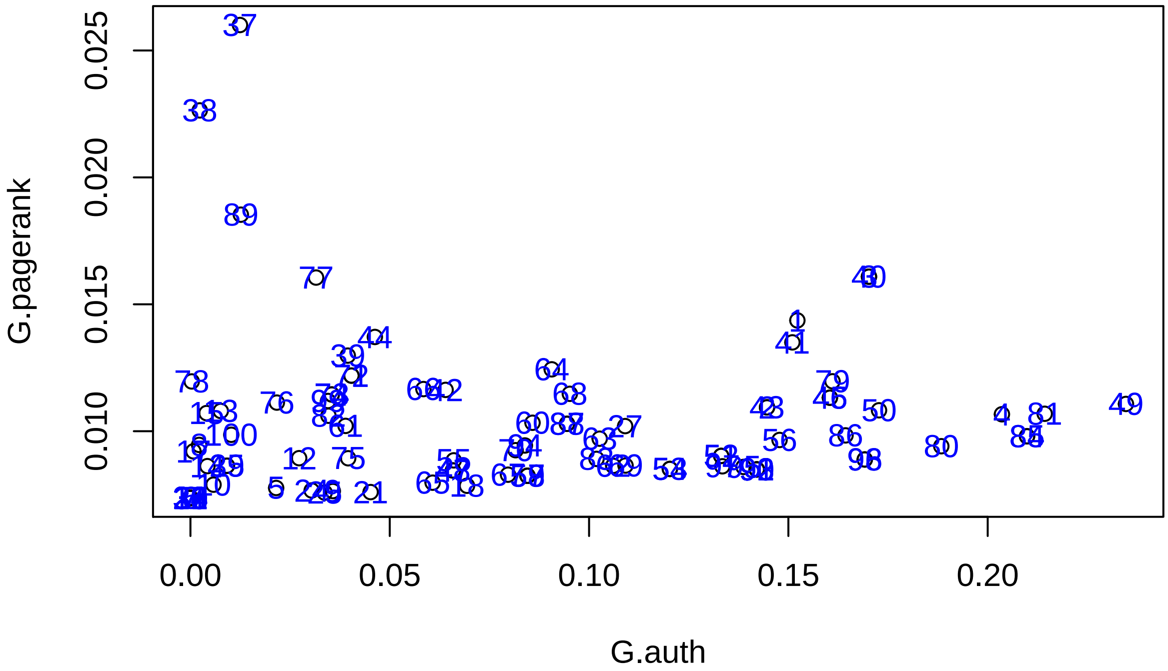


(c)

```r
pairs( data.frame(G.degree, G.betweenness, G.pagerank, G.hub, G.auth) )
```



```
plot (G.auth, G.pagerank)
text(G.auth, G.pagerank, col = "blue", cex=1)
```

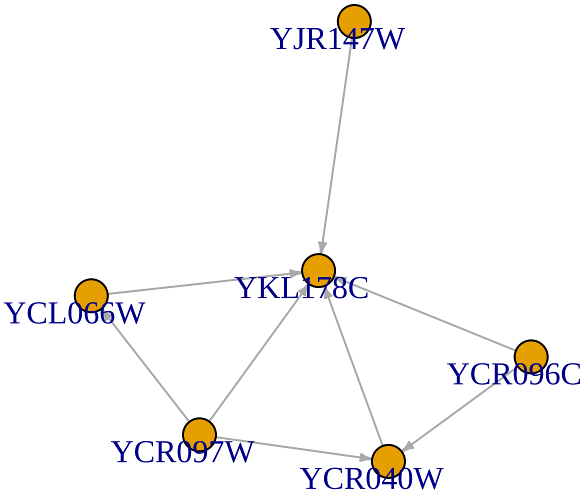


Node 37 and 38 seems to have low authority but high page rank.

```
n.name <- V(G)[37]$name
n.name
```

```
## [1] "YKL178C"
```

```
names = V(G)[nei(n.name)]$name
sub = c(names, n.name)
subG = induced.subgraph(G, V(G)[sub])

plot.igraph(subG, layout = layout.auto(subG), vertex.label.dist = -.5, edge.arrow.size = .3)
```



Problem 2

(a)

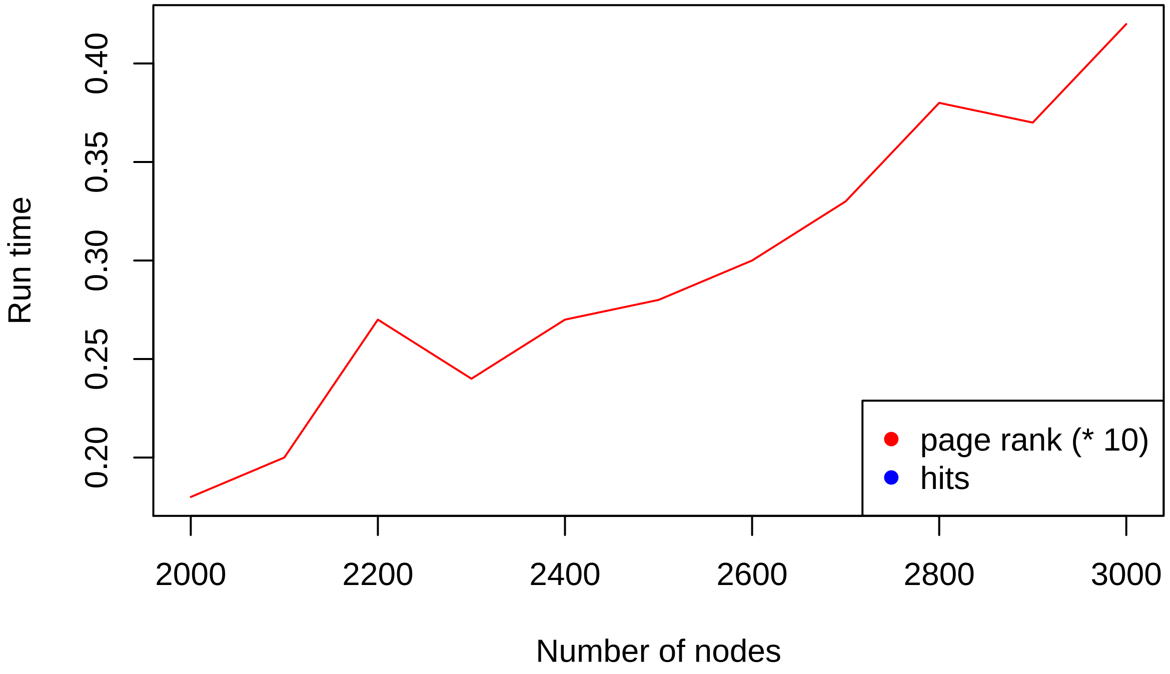
```
p = 1/20
numnode = seq(from = 2000, to = 3000, by = 100)
len = length(numnode)
prtime = rep(0, len)
hitstime  = rep(0, len)

for (i in 1:len) {
  g <- sample_gnp(numnode[i], 1/20, directed=TRUE)
  prtime[i] = system.time(page.rank(g))
  mat = as.matrix(get.adjacency(g))
  hitstime[i] = system.time(hitsFunc(mat, 20))
 }
#
prtime
```

```
##  [1] 0.018 0.020 0.027 0.024 0.027 0.028 0.030 0.033 0.038 0.037 0.042
```

```
hitstime
```

```
##  [1] 1.397 1.236 1.402 1.644 1.641 2.101 2.273 2.479 2.528 3.034 3.399
```

```
plot (numnode, 10 * prtime, col = "red", type = 'l', ylab = "Run time", xlab = "Number of nodes")
lines (numnode, hitstime, col = "blue", type = 'l')
legend("bottomright", pch=16, col=c("red", "blue"), c("page rank (* 10)", "hits"))
```



(b)

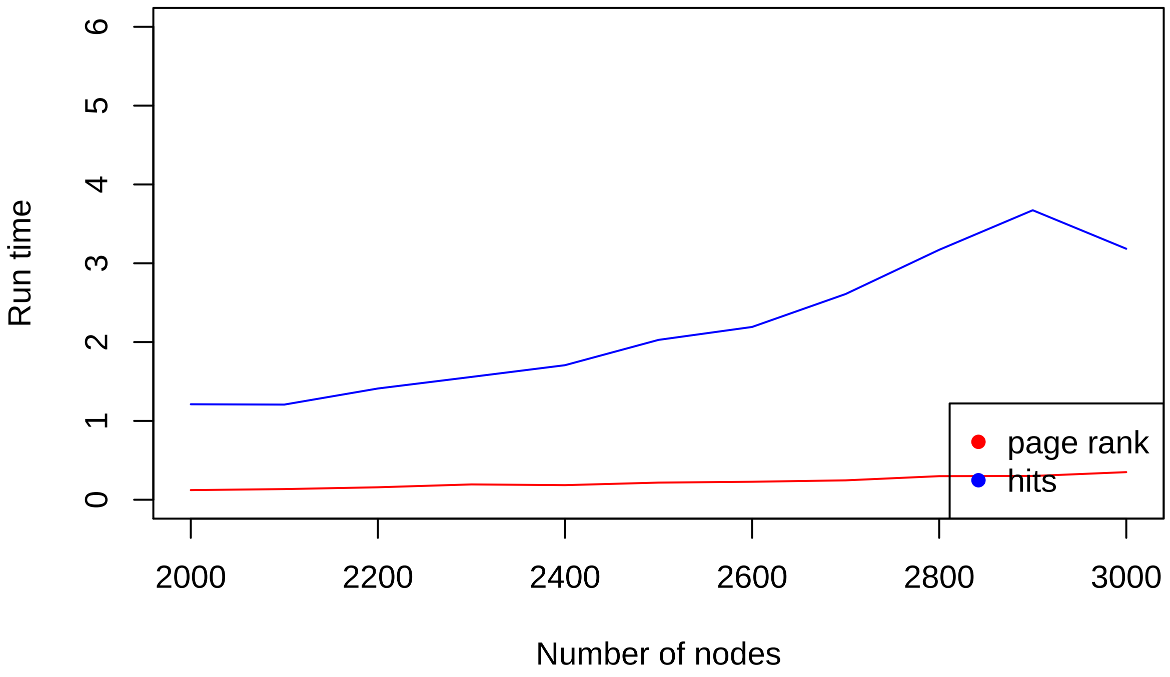
```
for (i in 1:len) {
  g <- sample_gnp(numnode[i], 1/3, directed=TRUE)
  prtime[i] = system.time(page.rank(g))
  mat = as.matrix(get.adjacency(g))
  hitstime[i] = system.time(hitsFunc(mat, 20))
 }
#
prtime
```

```
##  [1] 0.122 0.135 0.158 0.194 0.185 0.217 0.228 0.246 0.299 0.301 0.350
```

```
hitstime
```

```
##  [1] 1.211 1.207 1.411 1.558 1.707 2.028 2.192 2.610 3.171 3.673 3.184
```

```
plot (numnode, prtime, col = "red", type = 'l', ylim = c(0,6), ylab = "Run time", xlab = "Number of nod
lines (numnode, hitstime, col = "blue", type = 'l')
legend("bottomright", pch=16, col=c("red", "blue"), c("page rank", "hits"))
```



(c)

Increasing the edge probability means increasing the number of edges. The run time of the pagerank algorithm increases as either the number of nodes or edges increase. In the other hand, while the run time of the HITS algorithm increases with the number of nodes, it doesn't change when the number of edges increases.

The pattern is due to our inefficient implementation of the HITS algorithm. Matrix multiplications are very expensive and do not scale well.