social network

eric

11/09/2019

```
head(product)
```

```
##   id                                                          title
## 1  1                          Patterns of Preaching: A Sermon Sampler
## 2  2                                   Candlemas: Feast of Flames
## 3  3                  World War II Allied Fighter Planes Trading Cards
## 4  4      Life Application Bible Commentary: 1 and 2 Timothy and Titus
## 5  5                   Prayers That Avail Much for Business: Executive
## 6  6 How the Other Half Lives: Studies Among the Tenements of New York
##   group salesrank review_cnt downloads rating
## 1  Book    396585          2         2    5.0
## 2  Book    168596         12        12    4.5
## 3  Book   1270652          1         1    5.0
## 4  Book    631289          1         1    4.0
## 5  Book    455160          0         0    0.0
## 6  Book    188784         17        17    4.0
```

```
head(copurchase)
```

```
##   Source Target
## 1      1      2
## 2      1      4
## 3      1      5
## 4      1     15
## 5      2     11
## 6      2     12
```

Delete products that are not books from “products” and “copurchase” files. Note: In social network analysis, it important to define the boundary of your work; in other words, the boundary of the network.

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
product_book<-filter(product, group=="Book" &
                       product$salesrank<=150000 &
                       product$salesrank !=-1)
copurchase_book<-filter(copurchase, copurchase$Source %in% product_book$id &
                         copurchase$Target %in% product_book$id)
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
g <- graph.data.frame(copurchase_book, directed = T)

in_degree <- degree(g, mode = 'in')
head(in_degree)
```

```
##  12  74  77  79 117 120
##   5   1   3   0   9   3
```

```
out_degree <- degree(g, mode = 'out')
head(out_degree)
```

```
##  12  74  77  79 117 120
##   1   1   1   1   1   1
```

```
all_degree <- degree(g, mode = 'all')
max(all_degree)
```

```
## [1] 53
```

We ran the degree function to determine how many degrees are connected to the nodes. Then, we asked R to spit out the maximum number of in and out degrees for all the nodes, to which we got 53. From there, we needed to figure out which of the nodes had 53 degrees or in other words, are connected to this particular focal product. We found out that node 4429 and node 33 both have 53 degrees.

```
all_degree[all_degree==53]
```

```
## 4429   33
##   53   53
```

As for our subcomponent, we decided to use the node 33.

```
sub <- subcomponent(g, "33",'all')
sub
```

```
## + 904/20684 vertices, named, from c985450:
##   [1] 33     224    577    626    2558   3909   4068   8396   8715   9487
##  [11] 10787  17360  18508  21577  27795  28536  32485  39846  58399  59222
##  [21] 60366  64566  65985  68557  72753  98134  124680 144104 147832 156667
##  [31] 158642 178420 182671 184545 187824 188486 195144 195240 198014 208329
##  [41] 209650 212464 215934 216861 219348 224446 225689 236814 245203 245307
##  [51] 245321 245894 258438 261899 193    66465  7406   94702  3861   5355
##  [61] 7325   14950  18771  26080  26373  28759  29073  38188  38486  44035
##  [71] 72034  110901 122108 143888 237935 242813 3032   5018   5821   8732
##  [81] 9299   38040  42344  59791  108623 231027 5388   79074  10365  27932
##  [91] 33868  14623  7754   26786  9355   53729  86997  77404  101859 36718
## + ... omitted several vertices
```

```
graph <- induced_subgraph(g, sub)
graph
```

```
## IGRAPH b62ecdf DN-- 904 1173 --
## + attr: name (v/c)
## + edges from b62ecdf (vertex names):
##  [1] 77  ->422  130 ->78   148 ->302  187 ->321  187 ->322  187 ->78
##  [7] 193 ->224  224 ->193  224 ->33   321 ->187  321 ->322  321 ->78
## [13] 322 ->187  322 ->321  322 ->78   422 ->77   422 ->1644 556 ->78
## [19] 577 ->33   626 ->33   724 ->302  1051->302  1644->422  1644->5293
## [25] 1817->976  1822->193  1822->724  1851->78   1971->193  2071->3155
## [31] 2210->2279 2210->2285 2279->2210 2279->2326 2285->2330 2326->193
## [37] 2326->2210 2330->2343 2330->2345 2332->4140 2343->2285 2343->2330
## [43] 2423->5410 2470->556  2501->3588 2505->2501 2558->33   2572->4184
## + ... omitted several edges
```

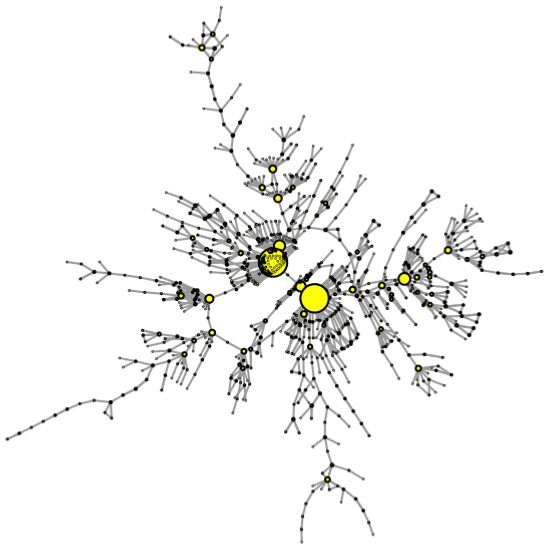
```
V(graph)
```

```
## + 904/904 vertices, named, from b62ecdf:
##   [1] 77     130    148    187    193    224    321    322    422    556
##  [11] 577    626    724    1051   1644   1817   1822   1851   1971   2071
##  [21] 2210   2279   2285   2326   2330   2332   2343   2423   2470   2501
##  [31] 2505   2558   2572   2657   2658   2806   2807   2959   3032   3119
##  [41] 3191   3217   3306   3427   3588   3670   3737   3861   3909   4002
##  [51] 4014   4038   4068   4099   4140   4174   4184   4185   4222   4223
##  [61] 4345   4429   4977   4993   4994   5018   5059   5163   5164   5293
##  [71] 5355   5388   5623   5638   5639   5655   5670   5821   5851   5875
##  [81] 6012   6014   6058   6059   6392   6411   6445   6546   6711   6713
##  [91] 6807   6808   6817   6942   7196   7198   7222   7233   7325   7376
## + ... omitted several vertices
```

```
E(graph)
```

```
## + 1173/1173 edges from b62ecdf (vertex names):
##  [1] 77  ->422  130 ->78   148 ->302  187 ->321  187 ->322  187 ->78
##  [7] 193 ->224  224 ->193  224 ->33   321 ->187  321 ->322  321 ->78
## [13] 322 ->187  322 ->321  322 ->78   422 ->77   422 ->1644 556 ->78
## [19] 577 ->33   626 ->33   724 ->302  1051->302  1644->422  1644->5293
## [25] 1817->976  1822->193  1822->724  1851->78   1971->193  2071->3155
## [31] 2210->2279 2210->2285 2279->2210 2279->2326 2285->2330 2326->193
## [37] 2326->2210 2330->2343 2330->2345 2332->4140 2343->2285 2343->2330
## [43] 2423->5410 2470->556  2501->3588 2505->2501 2558->33   2572->4184
## [49] 2572->4185 2657->2658 2658->77   2806->2807 2807->302  2959->1673
## [55] 3032->2558 3119->976  3191->2279 3217->4319 3306->2071 3306->4345
## + ... omitted several edges
```

```
V(graph)$label <- V(graph)$name
V(graph)$degree <- degree(graph)
plot(graph,
     vertex.color='yellow',
     vertex.size= V(graph)$degree*0.2,
     edge.arrow.size=0.01,
     vertex.label.cex=0.01,
     layout=layout.kamada.kawai)
```



```
diameter(graph, directed = T, weights = NA)
```

```
## [1] 9
```

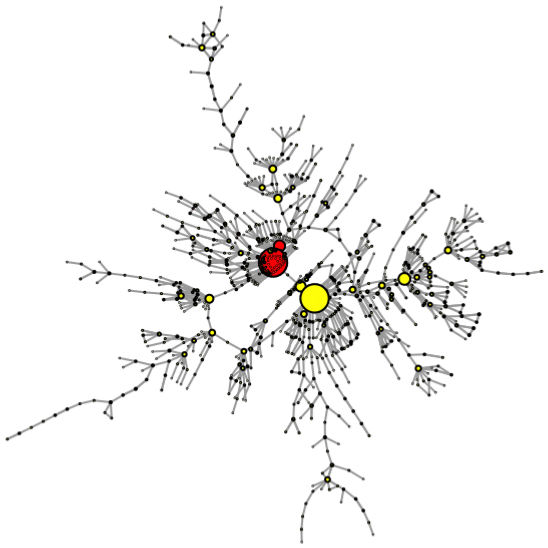
```
d <- get_diameter(graph, weights = NULL)
d
```

```
## + 10/904 vertices, named, from b62ecdf:
##  [1] 37895 27936 21584 10889 11080 14111 4429  2501  3588  6676
```

Diameter is the longest distance between two vertices, and we found the diameter to be 9. In the graph, the 10 red nodes are the vertices that on the longest path, and they are 37895, 27936, 21584, 10889, 11080, 14111, 4429, 2501, 3588, 6676.

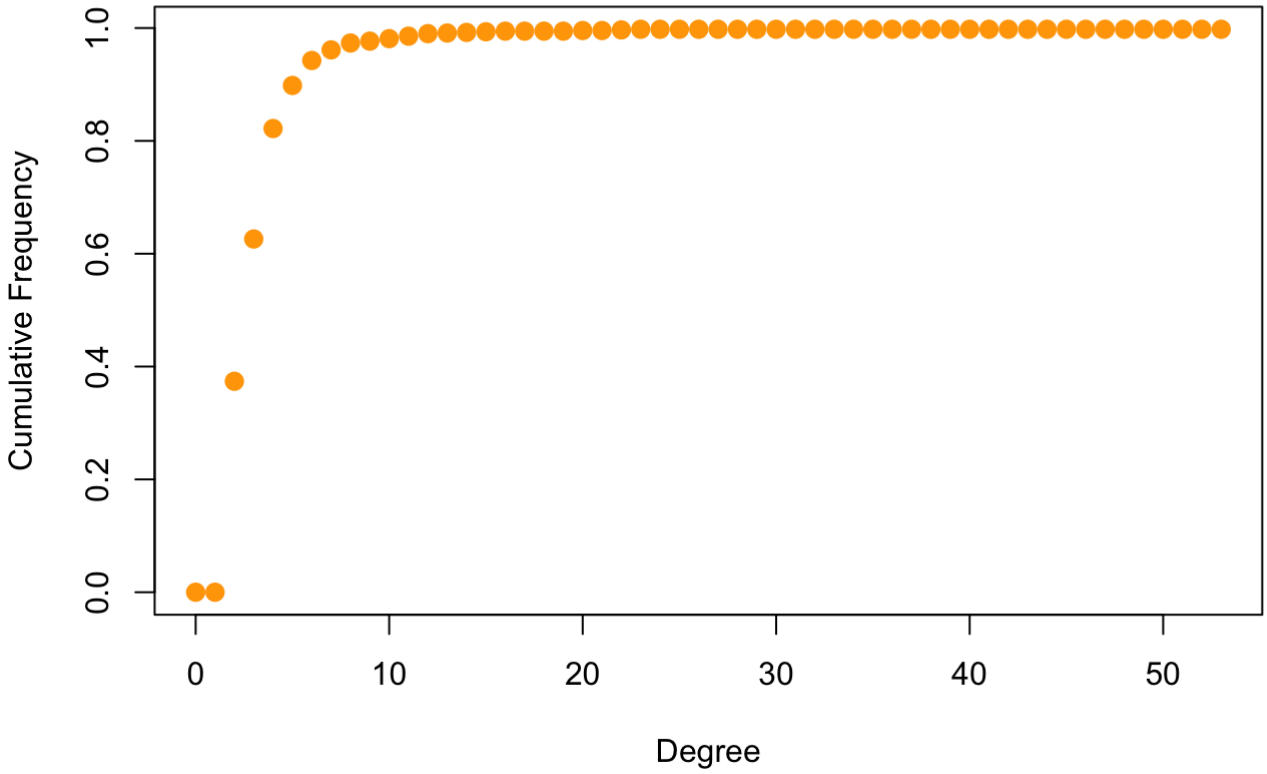
```
V(graph)$color<-"yellow"
V(graph)$color[d]<-"red"

plot(graph,
     vertex.color=V(graph)$color,
     vertex.size= V(graph)$degree*0.2,
     edge.arrow.size=0.01,
     vertex.label.cex=0.01,
     layout=layout.kamada.kawai)
```



The graph demonstrates 904 vertices. These 904 vertices are the book ids that connected to the book whose id = 33, directly and indirectly. Size of the vertices represents the number of vertices that connected to a vertice; the bigger of the vertice, the more vertices link to it. The distance between each vertice represents how strong the vertices connect to each other; the longer the ties, the weaker the relationship. Therefore, some vertices look like clusters in the middle with short edges, which means these books have strong connections. Some vertices are nodes on the edges, which means weaker connections.

```
#degree_distribution
dd_all <- degree_distribution(graph, cumulative=T, mode="all")
plot( x=0:max(all_degree), y=1-dd_all, pch=19, cex=1.2, col="orange",
      xlab="Degree", ylab="Cumulative Frequency")
```



Degree means the number of ties. In our degree distribution graph, degree increases at a decrease rate. Density is the proportion of present edges from all possible edges in the network. The density of our graph is 5.250029e-05, which is small; therefore, the networking is pretty dense. Centrality counts the number of links held by each node and points at individuals that can quickly connect with the wider network. Centrality here calculates the centrality of all the 904 nodes, and our results vary from 0 to 53, and 53 is the highest centrality. centrality based on distance to other nodes, and it calculates the shortest paths Closeness is the Betweenness is the centrality based on a broker position connecting others. between all nodes, then assigns each node a score based on its sum of shortest paths and is useful for finding the individuals who are best placed to influence the entire network most quickly

```
#density
edge_density(graph, loops=F)
```

```
## [1] 0.001436951
```

```
#centrality
centr_degree(graph)
```

```
## $res
##   [1]  4  2  2  6 11  4  5  6  4  5  3 19  2  2  5  1  2  1  1  3  4  5  3
##  [24]  3  5  1  3 11  2 21  5 11  2  1  2  1  4  1  2  1  1  3  2  1  3  2
##  [47]  1  6  3  3  6  3  2  4  2  3  2  5  2  4  2 53  2  7  3  1  3  4  5
##  [70]  4  1  3  3  2  3  1  4  3  1  9  7  1  4  5  2  1  1  2  2  2  9  1
##  [93]  2  2  3  3  3  4  6  2  2  9  5  6  5  2  1  2  3  3  2  9  3  3  1
## [116]  2  3  3  2  5  7  2  5  3  4  2  5  3  3  3  2  1  3  1  1  6  1  5
## [139]  3  3  2  4  1  1 10  1  3  1  5  2  1  2  2  2  1  2  2  4  4  3  3
## [162]  5  2  2  1  3  2  1  4  1  5  3  3  3  1  1  2  2  1  4  1  1  2  5
## [185]  3  2  3  1  3  4  2  3  1  4  7  1  1  2  1  1  2  1  2  4  3  2  5
## [208]  2  4  7  1  2  1  2  2  6  3  1  3  1  3  1  4  4  3  3  3  3  2  1
## [231]  1  3  1  6  3  2  1  1  2  4  3  1  2  4  2  6  1  6  7  3  3  1  1
## [254]  2  1  3  2  3  1  4  2  1  1  1  3  5  2  2  3  1  2  1  3  2  1  4
## [277]  2  2  5  1  4  4  1  1  1  2  3  1  7  2  1  1  3  3  2  3  1  4  4
## [300]  1  1  1  1  6  1  2  3  2  1  2  5  2  1  1  2  1  1  3  5  2  4  3
## [323]  1  2  3  2  1  3  1  2  3  2  1  1  2  1  2  1  2  2  3  3  3  7  2
## [346]  2  5  1  1  1  6  2  3  2  1  4  2  4  5  4  2  2  1  4  4  1  1  1
## [369]  5  2  2  3  3  6  1  2  7  1  3  1  1  3  4  1  3  3  3  2  1  2  3
## [392]  2  3  3  1  1  1  1  3  1  1 10  3  3  3  1  6  2  1  1  1  1  2  3
## [415]  3  1  1  3  3  2  1  5  1  1  2  3  1  5  1  2  2  3  1  3  2  3  1
## [438]  1  3  3  3  2  1  3  2  2  1  1  1  1  1  2  2  1  5  1  3  2  2  2
## [461]  5  2  1  4  2  2  3  2  1  1  1  4  1  2  1  1  1  2  4  1  3  1  4
## [484]  3  3  5  2  3  1  2  6  2  3  1  2  2  3  1  5  7  1  2  1  1  4  4
## [507]  1  2  2  2  3  1  2  1  1  2  3  1  1  1  4  4  3  1  2  1  1  3  2
## [530]  1  2  1  1  2  2  2  1  1  1  1  2  2  3  1  1  2  2  1  1  1  3  3
## [553]  1  1  1  1  1  2  1  1  1  2  2  2  1  1  2  4  3  7  4  1  2  3  1
## [576]  1  2  5  1  4  1  1  1  1  1  5  4  1  1  2  1  1  1  1  3  3  3  1
## [599]  2  2  4  1  1  2  1  1  3  4  1  3  3  2  3  4  2  1  4  1  1  3  1
## [622]  2  1  1  3  5  1  3  1  2  2  2  2  1  3  2  3  3  3  3  2  1  1  2
## [645]  3  3  1  2  2  1  4  1  1  1  2  2  4  2  5  1  1  2  1  2  1  2  2
## [668]  3  3  2  3  1  6  3  1  4  3  1  1  1  1  1  2  3  2  1  1  1  1  1
## [691]  2  2  3  1  3  2  3  1  1  1  1  3  1  3  1  3  3  1  1  1  1  1  3
## [714]  2  4  3  3  1  1  1  2  1  1  1  3  3  2  1  2  1  1  1  1  2  1  1
## [737]  3  1  3  3  2  1  1  1  2  2  2  2  1  1  3  2  1  1  3  1  3  2  2
## [760]  2  1  2  2  3  1  2  2  1  1  3  2  1  1  1  1  2  2  3  2  1  1  1
## [783]  1  1  2  3  1  2  1  2  3  1  1  1  1  1  1  2  1  1  1  1  2  1  1
## [806]  5  1  1  4  2  2  4  3  4  3  2  1  3  1  3  3  2  2  4  3  2 11 22
## [829] 53 10 12  2  2  5  1  2  2 15  2 10  3  8  3 14  3  1  2  4  1  8  2
## [852]  3  1  1 13  1  5  4  1  8  2  3  2  1  1  1  1  1  2  1  1  1  4  1
## [875]  1  1  1  2  2  1  4  1  4  2  2  1  1  1  1  4  2  3  1  1  1  1  1
## [898]  1  2  1  1  1  1  1
##
## $centralization
## [1] 0.02794058
##
## $theoretical_max
## [1] 1630818
```

```
closeness<-closeness(graph, mode='all', weights=NA)
head(closeness)
```

```
##           77          130          148          187          193
## 9.045681e-05 1.077935e-04 1.009897e-04 1.076774e-04 1.414227e-04
##          224
## 1.496110e-04
```

```
betweenness<-betweenness(graph, directed='T', weights=NA)
head(betweenness)
```

```
##  77 130 148 187 193 224
##  12   1   2   2  40  31
```

```
#hub/authority scores
hub_score<-hub.score(graph)$vector
head(hub_score)
```

```
##           77          130          148          187          193
## 2.239872e-16 5.531568e-04 3.592652e-05 5.989914e-04 3.880532e-16
##          224
## 9.836847e-01
```

```
authority_score<-authority.score(graph)$vector
head(authority_score)
```

```
##           77          130          148          187          193
## 4.449831e-17 2.473186e-17 2.567663e-17 2.431071e-05 2.317514e-02
##          224
## 5.135327e-17
```

```
product$id<-as.vector(product$id)
sub_id<-as_ids(sub)
product_sub<-product[product$id %in% sub_id,]
head(product_sub)
```

```
##      id
## 33   33
## 77   77
## 78   78
## 130 130
## 148 148
## 187 187
##
title
## 33                                                                  Double Jeopardy (T*Witche
s, 6)
## 77                                                                       Water Touching
Stone
## 78                                                    The Ebony Cookbook: A Date With a
Dish
## 130 The O'Reilly Factor: The Good, the Bad, and the Completely Ridiculous in American
Life
## 148                                                                                  Fir
ebird
## 187                        Words for Smart Test Takers (Academic Test Preparation Se
ries)
##     group salesrank review_cnt downloads rating
## 33   Book     97166          4         4    5.0
## 77   Book     27012         11        11    4.5
## 78   Book    140480          3         3    4.5
## 130  Book     29460        375       375    3.5
## 148  Book     77008         42        42    4.0
## 187  Book     17104          4         4    5.0
```

```
mean<-copurchase_book %>%
  group_by(Target) %>%
  inner_join(product_sub,by=c('Source'='id'))%>%
  summarise(nghb_mn_rating=mean(rating),
            nghb_mn_salesrank=mean(salesrank),
            nghb_mn_review_cnt=mean(review_cnt))
head(mean)
```

```
## # A tibble: 6 x 4
##   Target nghb_mn_rating nghb_mn_salesrank nghb_mn_review_cnt
##    <int>          <dbl>             <dbl>              <dbl>
## 1     33           4.10            82153.              21.1
## 2     77           4.67            41744                4
## 3     78           4.5             73179              158.
## 4    130           4.5             19415                6
## 5    148           0               46701                0
## 6    187           4.5            133547.               3.67
```

Include the variables (taking logs where necessary) created in Parts 2-6 above into the “products” information and fit a Poisson regression to predict salesrank of all the books in this subcomponent using products’ own information and their neighbor’s information. Provide an interpretation of your results. Note: Lower salesrank

means higher sales. Data points in the network are related.The performance of one node is influenced by the performance of its neighbors. Also, it's not necessary that all variables matter

```
#convert all igraph lists to data frames
#shift index col to the right and rename col. accordingly
in_degree1 <- as.data.frame(in_degree)
in_degree1 <- cbind(newColName = rownames(in_degree1), in_degree1)
rownames(in_degree1) <- 1:nrow(in_degree1)
colnames(in_degree1)[1] = "Nodes"

out_degree1 <- as.data.frame(out_degree)
out_degree1 <- cbind(newColName = rownames(out_degree1), out_degree1)
rownames(out_degree1) <- 1:nrow(out_degree1)
colnames(out_degree1) <- c("Nodes", "out_degree")

closeness1 <- as.data.frame(closeness)
closeness1 <- cbind(newColName = rownames(closeness1), closeness1)
rownames(closeness1) <- 1:nrow(closeness1)
colnames(closeness1) <- c("Nodes", "closeness")

betweenness1 <- as.data.frame(betweenness)
betweenness1 <- cbind(newColName = rownames(betweenness1), betweenness1)
rownames(betweenness1) <- 1:nrow(betweenness1)
colnames(betweenness1) <- c("Nodes", "betweenness")

hub_score1 <- as.data.frame(hub_score)
hub_score1 <- cbind(newColName = rownames(hub_score1), hub_score1)
rownames(hub_score1) <- 1:nrow(hub_score1)
colnames(hub_score1) <- c("Nodes", "hub_score")

authority_score1 <- as.data.frame(authority_score)
authority_score1 <- cbind(newColName = rownames(authority_score1), authority_score1)
rownames(authority_score1) <- 1:nrow(authority_score1)
colnames(authority_score1) <- c("Nodes", "authority_score")
```

```
#combine data frames into one data frame by nodes
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shared
object '/Library/Frameworks/R.framework/Resources/modules//R_X11.so':
##   dlopen(/Library/Frameworks/R.framework/Resources/modules//R_X11.so, 6): Library not
loaded: /opt/X11/lib/libSM.6.dylib
##   Referenced from: /Library/Frameworks/R.framework/Resources/modules//R_X11.so
##   Reason: image not found
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/
## Resources/library/tcltk/libs//tcltk.so'' had status 1
```

```
## Could not load tcltk.  Will use slower R code instead.
```

```
## Loading required package: RSQLite
```

```
poisson_data <- sqldf("SELECT mean.Target, hub_score, betweenness, authority_score,
closeness, in_degree, out_degree, nghb_mn_rating, nghb_mn_salesrank, nghb_mn_review_cnt,
product.review_cnt, product.downloads, product.rating, product.salesrank
                      FROM mean, product, hub_score1, betweenness1, authority_score1, cl
oseness1, in_degree1, out_degree1
                      WHERE mean.Target = betweenness1.Nodes
                      and mean.Target = authority_score1.Nodes
                      and mean.Target = closeness1.Nodes
                      and mean.Target = in_degree1.Nodes
                      and mean.Target = out_degree1.Nodes
                      and mean.Target = hub_score1.Nodes
                      and mean.Target = product.id")
head(poisson_data)
```

```
##   Target    hub_score betweenness authority_score    closeness in_degree
## 1     77 2.239872e-16          12    4.449831e-17 9.045681e-05         3
## 2    130 5.531568e-04           1    2.473186e-17 1.077935e-04         1
## 3    148 3.592652e-05           2    2.567663e-17 1.009897e-04         1
## 4    187 5.989914e-04           2    2.431071e-05 1.076774e-04         3
## 5    193 3.880532e-16          40    2.317514e-02 1.414227e-04        10
## 6    224 9.836847e-01          31    5.135327e-17 1.496110e-04         2
##   out_degree nghb_mn_rating nghb_mn_salesrank nghb_mn_review_cnt
## 1          1       4.666667           41744.0           4.000000
## 2          1       4.500000           19415.0           6.000000
## 3          1       0.000000           46701.0           0.000000
## 4          3       4.500000          133546.7           3.666667
## 5          1       4.050000           59470.6          75.700000
## 6          2       3.250000           79068.0         167.500000
##   review_cnt downloads rating salesrank
## 1         11        11    4.5     27012
## 2        375       375    3.5     29460
## 3         42        42    4.0     77008
## 4          4         4    5.0     17104
## 5        261       260    3.0     10350
## 6          1         1    5.0    138623
```

```
#run poisson regression
summary(salesrating_prediction<- glm(salesrank ~ review_cnt + downloads + rating + hub_s
core + betweenness +
                                       authority_score + closeness + in_degree + out_deg
ree +
                                       nghb_mn_rating + nghb_mn_salesrank + nghb_mn_revi
ew_cnt,, family="poisson",
                                     data=poisson_data))
```

```
##
## Call:
## glm(formula = salesrank ~ review_cnt + downloads + rating + hub_score +
##     betweenness + authority_score + closeness + in_degree + out_degree +
##     nghb_mn_rating + nghb_mn_salesrank + nghb_mn_review_cnt,
##     family = "poisson", data = poisson_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -363.25  -160.45    -7.61   122.01   519.58
##
## Coefficients:
##                      Estimate Std. Error   z value Pr(>|z|)
## (Intercept)         1.119e+01  1.108e-03 10096.697   <2e-16 ***
## review_cnt         -2.868e-02  1.877e-04  -152.749   <2e-16 ***
## downloads           2.457e-02  1.879e-04   130.759   <2e-16 ***
## rating             -7.061e-03  1.098e-04   -64.314   <2e-16 ***
## hub_score           2.452e-01  8.593e-04   285.400   <2e-16 ***
## betweenness        -7.349e-04  1.111e-05   -66.157   <2e-16 ***
## authority_score     1.895e-01  4.754e-03    39.861   <2e-16 ***
## closeness          -1.789e+01  7.874e+00    -2.272   0.0231 *
## in_degree           2.801e-03  6.819e-05    41.069   <2e-16 ***
## out_degree          5.646e-02  2.057e-04   274.476   <2e-16 ***
## nghb_mn_rating     -9.723e-03  1.253e-04   -77.613   <2e-16 ***
## nghb_mn_salesrank   2.057e-07  4.498e-09    45.733   <2e-16 ***
## nghb_mn_review_cnt  7.386e-04  1.969e-06   375.165   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 16968896  on 517  degrees of freedom
## Residual deviance: 15315200  on 505  degrees of freedom
## AIC: 15321778
##
## Number of Fisher Scoring iterations: 5
```

To generate a Poisson regression model to predict sales rank of the books in the dataset, we decided to use the following variables as predictors: Review count Ratings Hub score Betweenness Authority score Closeness In degree Out degree Nghb_mn_rating Nghb_mn_salesrank Nghb_mn_review_cnt The values of closeness, hub_score and authority_score is just too small, therefore, we decided to apply log function on these 3 variables. However, after running the log function, there are infinite values existing in our data frame, and those values would influence on our model's prediction. Therefore, we replaced inf value to NA value, and further removed those NA

values from our data frame. Now, our poisson data has total 159 observations. This is our final prediction model where the dependent variable is the sales rank and the independent variables correspond to the list above. After running the Poisson Regression model, we found that P values for all the variables are less than 2e-16, which indicate that all the variables are significant factors regarding the prediction of books' salesrank. Because "Lower salesrank" means "Higher Sales" for the book, there are 5 variables from our model having negative effects on salesrank, which are review_cnt, rating, betweenness, authority_score and ngnb_mn_rating. With the increasing percentage of these 5 variables, books' salesrank would be decreasing, which indicates that more customers choose to buy the book, further generating more revenues for the company.