Social Network Analysis: Lab 2

Mikael Brunila

2017-03-20

Contents

1. Describe the social network(s) to me, in terms of how it was collected, what it represents and so forth. Also give me basic topography of the network: the nature of the ties; direction of ties; overall density; and if attributes are with the network, the distribution of the categories and variables of those attributes.

As data I use the hashtag for a collective day of action that was organized by a number of housing groups in Spain, the US and some other countries in October 2015. This action targetted the private-equity fund Blackstone with the motivation that the fund was acting as a predatory landlord in communities that had been hit by the financial crisis. The action was spearheaded by la Plataforma de Afectados por la Hipoteca, a radical Spanish housing group that was formed in 2009 to fight against mortgage related evictions and that has since grown to be one of the largest social movements in Spanish history.

I was in Barcelona at this time, documenting the activites of la PAH for a the Learning in Social Movements project at the CRADLE research unit at the University of Helsinki. I used a Python scraper I had written and the Twitter API to gather all the tweets under the hashtag #StopBlackstone. The total flow of tweets (including retweets) during that day was 8310 strong and it seems I got all the tweets under the hashtags as no quota limits set by the Twitter API were reached.

I have since then used this data during a presentation at the Finnish Sociology Days in the winter of 2017 to discuss how one can overcome limitations in spatial data to still produce geospatial visualizations and analysis. Here I instead look at the activity around the hashtag as a social network. For this, I use a package I wrote for the Modern Data Structures class at QMSS in the autumn of 2017. The package extracts retweets in the form of edges from a list of tweets in the CSV format that the package Rtweets uses when returning a query. Becaus my tweets were gathered using the Python package Tweepy they were in JSON format. Hence, I had to reformat the data as CSV, and then run the function getRetweetEdges from my own package to get the edges.

```r
blackstone_df = data.frame(matrix(vector(), 0, 19,
                dimnames = list(c(),
                              c("status_id", "created_at", "user_id", "screen_name",
                                "text", "is_retweet", "favorite_count", "retweet_count",
                                "followers_count", "friends_count", "listed_count",
                                "user_created_at", "time_zone", "location",
                                "photo", "video", "amount_of_hashtags", "tweet_length",
                                "user_mentions"))),
                stringsAsFactors = F)

con <- file(description = "blackstone_STREAM.json", open = "r")

tweets_json <-list()
i = 0

# Solution borrowed from StackOverflow
# https://stackoverflow.com/questions/4106764/what-is-a-good-way-to-read-line-by-line-in-r
while (length(oneLine <- readLines(con, n = 1, warn = FALSE)) > 0) {
  i = i + 1
  myJson <- jsonlite::fromJSON(txt = oneLine)
  tweets_json[[i]] <- myJson
}

close(con)

for(i in 1:length(tweets_json)) {
  blackstone_df[i,]$status_id <- tweets_json[[i]]$id
  blackstone_df[i,]$created_at <- tweets_json[[i]]$created_at
  blackstone_df[i,]$user_id <- tweets_json[[i]]$user$id
  blackstone_df[i,]$screen_name <- tweets_json[[i]]$user$screen_name
  blackstone_df[i,]$text <- tweets_json[[i]]$text
  blackstone_df[i,]$is_retweet <- ifelse(length(tweets_json[[i]]$retweeted_status) > 0, "Yes", "No")
  blackstone_df[i,]$favorite_count <- tweets_json[[i]]$favorite_count
  blackstone_df[i,]$retweet_count <- tweets_json[[i]]$retweet_count
  blackstone_df[i,]$followers_count <- tweets_json[[i]]$user$followers_count
  blackstone_df[i,]$friends_count <- tweets_json[[i]]$user$friends_count
  blackstone_df[i,]$listed_count <- tweets_json[[i]]$user$listed_count
  blackstone_df[i,]$user_created_at <- tweets_json[[i]]$user$created_at
  blackstone_df[i,]$time_zone <- tweets_json[[i]]$user$time_zone
  blackstone_df[i,]$location <- tweets_json[[i]]$user$location
  if(length(tweets_json[[i]]$extended_entities$media$type) > 0) {
    blackstone_df[i,]$photo <- ifelse(tweets_json[[i]]$extended_entities$media$type[[1]] == "photo", 1,
    blackstone_df[i,]$video <- ifelse(tweets_json[[i]]$extended_entities$media$type[[1]] == "video", 1,
  } else {
    blackstone_df[i,]$photo <- 0
    blackstone_df[i,]$video <- 0
  }
  blackstone_df[i,]$amount_of_hashtags <- ifelse(length(tweets_json[[i]]$entities$hashtags) > 0, unlist
  blackstone_df[i,]$user_mentions <- ifelse(length(tweets_json[[i]]$entities$user_mentions) > 0, unlist
  blackstone_df[i,]$tweet_length <- nchar(tweets_json[[i]]$text)
}
```

The network has a total of 1224 users or nodes connected by a total of 7244 edges. There are 1224 unique users retweeting and 201 unique users being retweeted. The graph has a density of 0.004839167, meaning that it is connected only very sparsely. The graph below gives an overview of the network, with arrows pointing in towards users that were retweeted. Each node is a user, each edge a retweet.

```r
BlackstoneEdges <- getRetweetEdges(blackstone_df, rt_limit = 0)
BlackstoneEdgesFlipped <- as.data.frame(BlackstoneEdges)[c("V2", "V1")]

# Getting edge weights
BlackstoneEdgesWeights <- BlackstoneEdgesFlipped %>%
  as.data.frame() %>%
  group_by(V2) %>%
  count(V1)

el <- as.matrix(BlackstoneEdgesWeights)
g <- graph.edgelist(el[,1:2], directed = TRUE)
E(g)$weight <- as.integer(el[,3])
graph_density <- edge_density(g, loops = FALSE)

adj <- get.adjacency(g, attr = 'weight')
net <- network(adj, directed = TRUE)
```

```
vcount(g)
```

```
## [1] 1224
```

```
ecount(g)
```

```
## [1] 4392
```

```
length(unique(as.data.frame(BlackstoneEdges)$V1))
```

```
## [1] 1147
```

```
length(unique(as.data.frame(BlackstoneEdges)$V2))
```

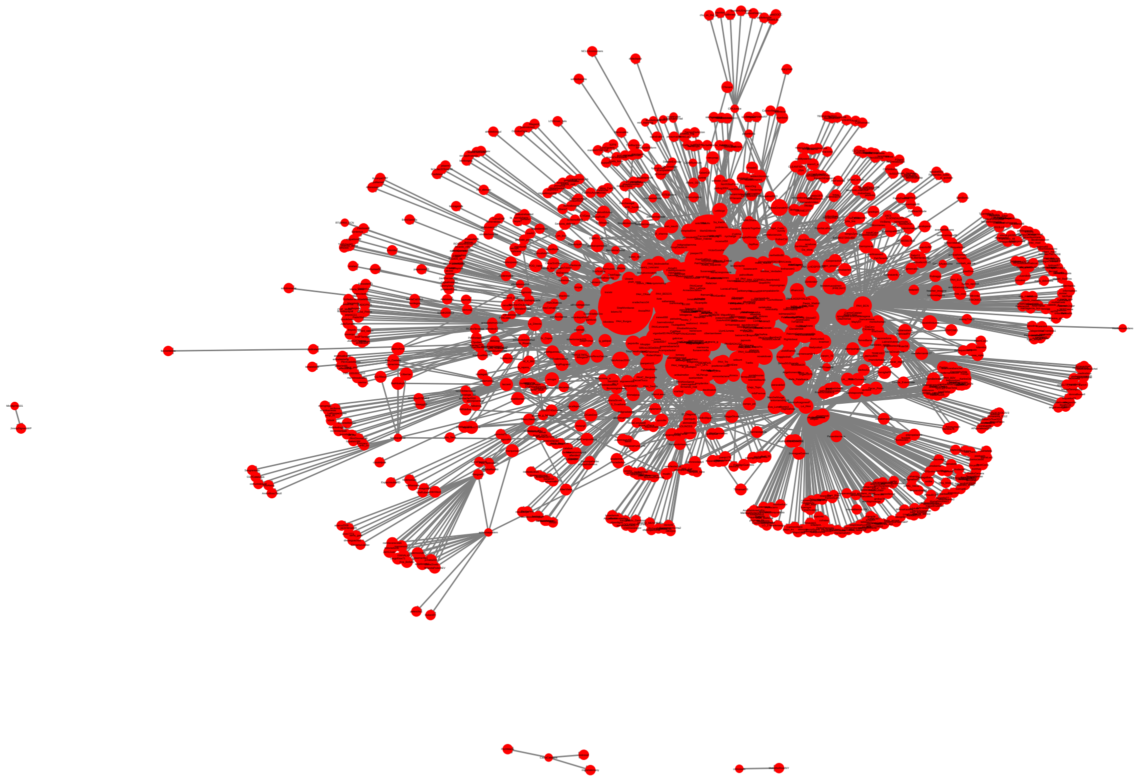
```
## [1] 201
```

```
graph_density
```

```
## [1] 0.002933962
```

```
#V(g)$label.cex = 0.1
#l <- layout_with_drl(g)
#plotGraphRetweets(BlackstoneEdgesWeights[,1:2], arrowsize = 2, color = "red", labelsize = 0.2)
#plotGraphRetweets(BlackstoneEdges, arrowsize = 2, color = "red", labelsize = 0.2)
#plot(g, edge.width = E(g)$weight/2, layout = l)

ggnet2(net, size = "indegree", arrow.size = 2, color = "red", label = TRUE, label.size = 0.5, legend.pos
```



Finally, I created a dataframe for the attributes of the vertices. This table contains more vertices, indicating that there were a number of nodes that were totally unconnected and therefore left out from the edgelist. This created some technical difficulties later on, but luckily I managed to solve them. The attributes I include are:

- screen_name
- user_id
- created_at
- photos
- vids
- amount_of_hashtags
- location
- time_zone
- followers_count
- friends_count
- listed_count
- total_tweets

```r
vertex_attributes <- data.frame(matrix(vector(), 0, 12,
                dimnames = list(c(),
                              c("screen_name", "user_id", "created_at", "photos",
                                "vids", "amount_of_hashtags", "location", "time_zone",
                                "followers_count", "friends_count", "listed_count", "total_tweets"))),
                stringsAsFactors = F)


V(g)$followers_count <- blackstone_df$followers_count[match(V(g)$name, as.character(blackstone_df$scree
V(g)$location <- blackstone_df$location[match(V(g)$name, as.character(blackstone_df$screen_name))]
V(g)$created_at <- blackstone_df$created_at[match(V(g)$name, as.character(blackstone_df$screen_name))]

vertex_attributes <- blackstone_df %>%
  group_by(screen_name) %>%
  summarise(photos = mean(photo), vids = mean(video), followers_count = mean(followers_count),
            friends_counts = mean(friends_count), listed_count = mean(listed_count),
            mean_hashtags = mean(amount_of_hashtags))

blackstone_df$is_retweet <- as.factor(blackstone_df$is_retweet)

tweet_counts <- blackstone_df %>%
  group_by(screen_name) %>%
  count(is_retweet) %>%
  spread(is_retweet, n)

tweet_counts$No <- ifelse(is.na(tweet_counts$No), 0, tweet_counts$No)
tweet_counts$Yes <- ifelse(is.na(tweet_counts$Yes), 0, tweet_counts$Yes)

vertex_attributes <- merge(vertex_attributes, count(blackstone_df, screen_name), by = "screen_name")
vertex_attributes <- merge(vertex_attributes, tweet_counts, by = "screen_name")

V(g)$total_tweets <- vertex_attributes$n[match(V(g)$name, as.character(vertex_attributes$screen_name))]

V(g)$photo <- vertex_attributes$photo[match(V(g)$name, as.character(vertex_attributes$screen_name))]
V(g)$video <- vertex_attributes$vids[match(V(g)$name, as.character(vertex_attributes$screen_name))]

V(g)$mean_hashtags <- vertex_attributes$mean_hashtags[match(V(g)$name, as.character(vertex_attributes$s
```

2. Calculate degree centrality (in- and out-degree, too, if you have such data); closeness centrality; betweenness centrality; and eigenvector centrality. Correlate those measures of centrality. Highlight which nodes are most central and least central, along different dimensions.

I calculated the centrality measures with in-degree, out-closeness, betweenness and eigen centrality. I then produced ggplots that contrast these measures (placing them arbitrarily on the x- or y-axis). The correlations between the measures seem weak, with a few exceptions: eigen centrality and degree are clearly linearly related and betweenness and degree might have some kind of relation, with several exceptions. Producing lists over the top performers for each centrality measure, it seems that there are some names shared by each category, except for betweenness and closeness. Finally, I also produced a network graph where nodes were colored by the number of centrality measures that the node scored in the top 20. No nodes were in the top 20 for all four measures, but some of the most graphically central nodes performed well on two or three lists.

```
centrality_measures <- data.frame(matrix(vector(), 1224, 5, dimnames = list(c(),
                                                         c("screen_name", "degree", "closeness",
                                                           "betweenness","eigen"))),
                                                            stringsAsFactors = F)

centrality_measures$screen_name <- V(g)$name
centrality_measures$degree <- igraph::degree(g, V(g), mode = "in")
centrality_measures$closeness <- igraph::closeness(g, V(g), mode = "out")
centrality_measures$betweenness <- igraph::betweenness(g, V(g))
centrality_measures$eigen <- igraph::eigen_centrality(g)$vector

range(centrality_measures$degree)
```

```
## [1]   0 120
```

```
range(centrality_measures$closeness)
```

```
## [1] 6.680241e-07 3.929289e-06
```

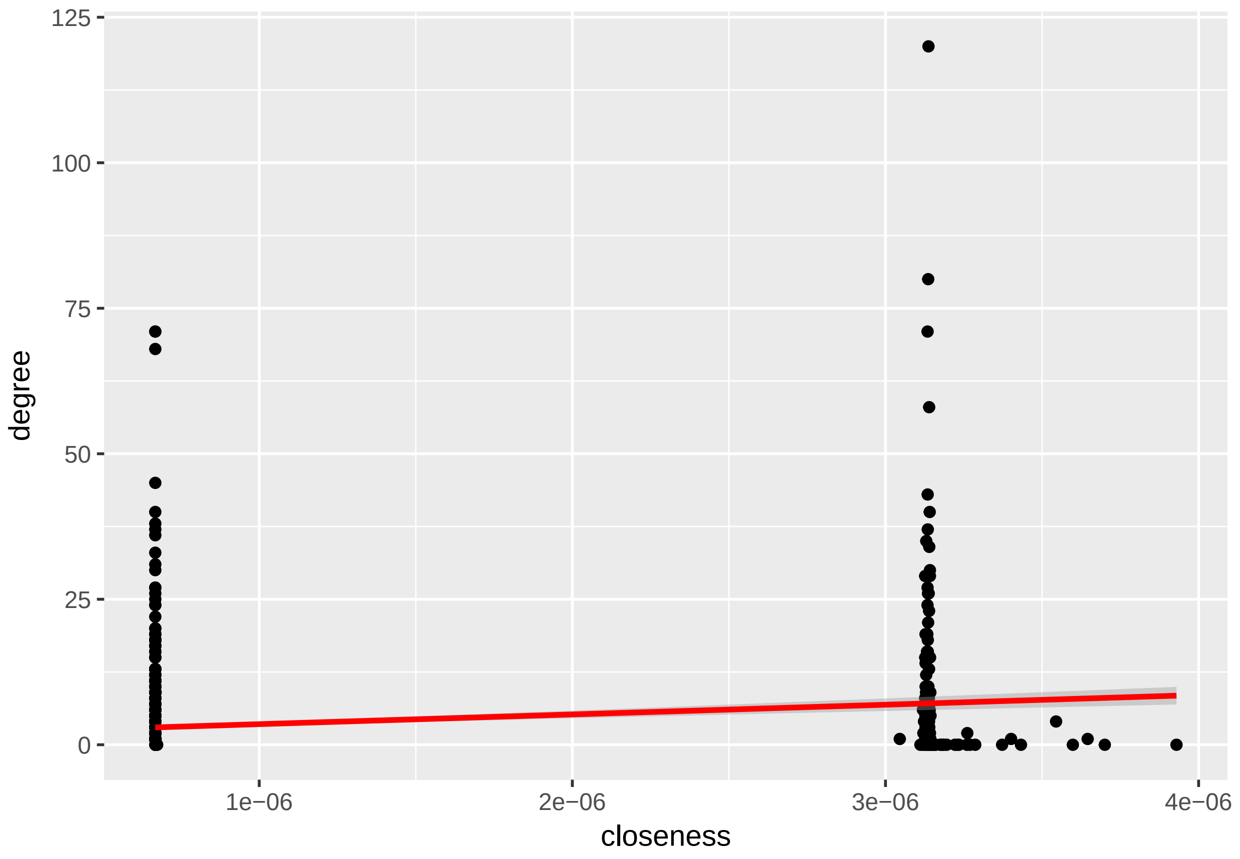
```
range(centrality_measures$betweenness)
```

```
## [1]     0.00 48189.62
```

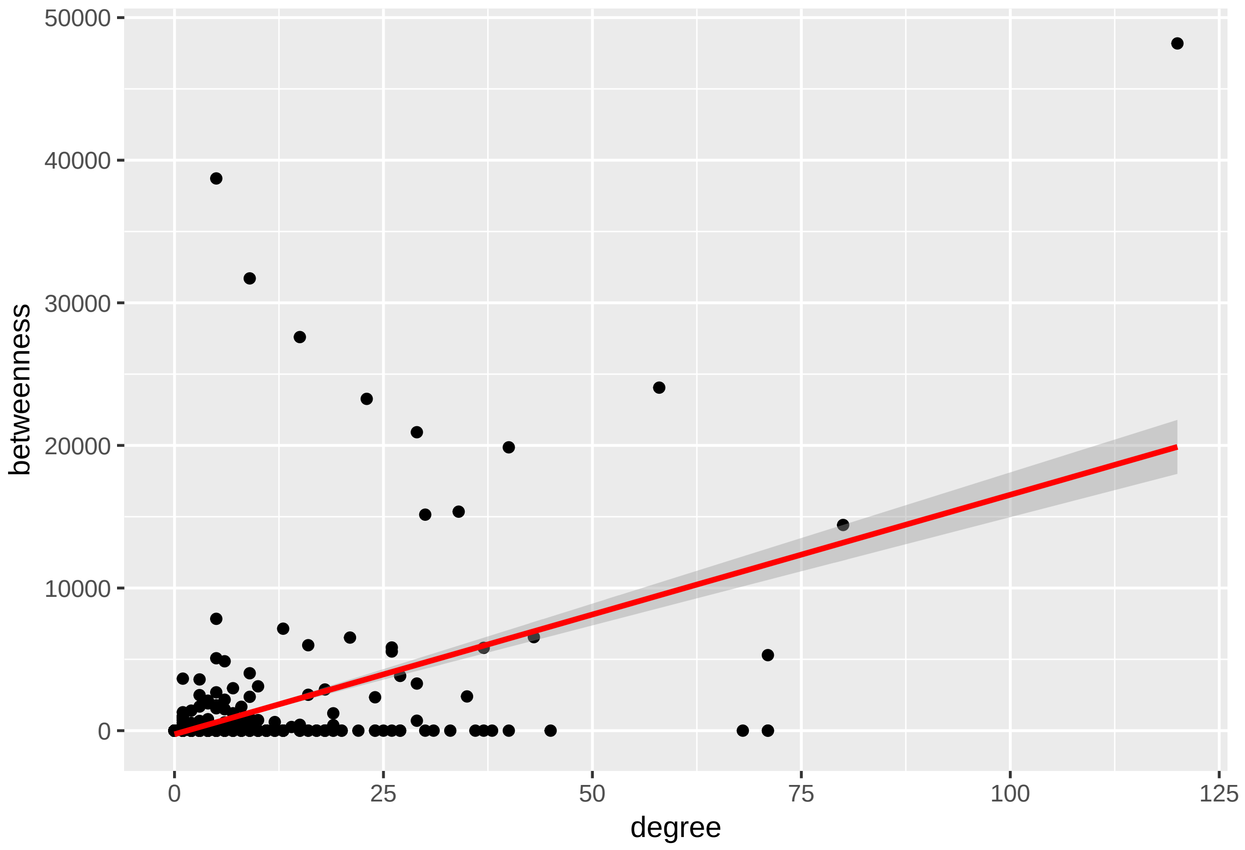
```
range(centrality_measures$eigen)
```

```
## [1] 0 1
```

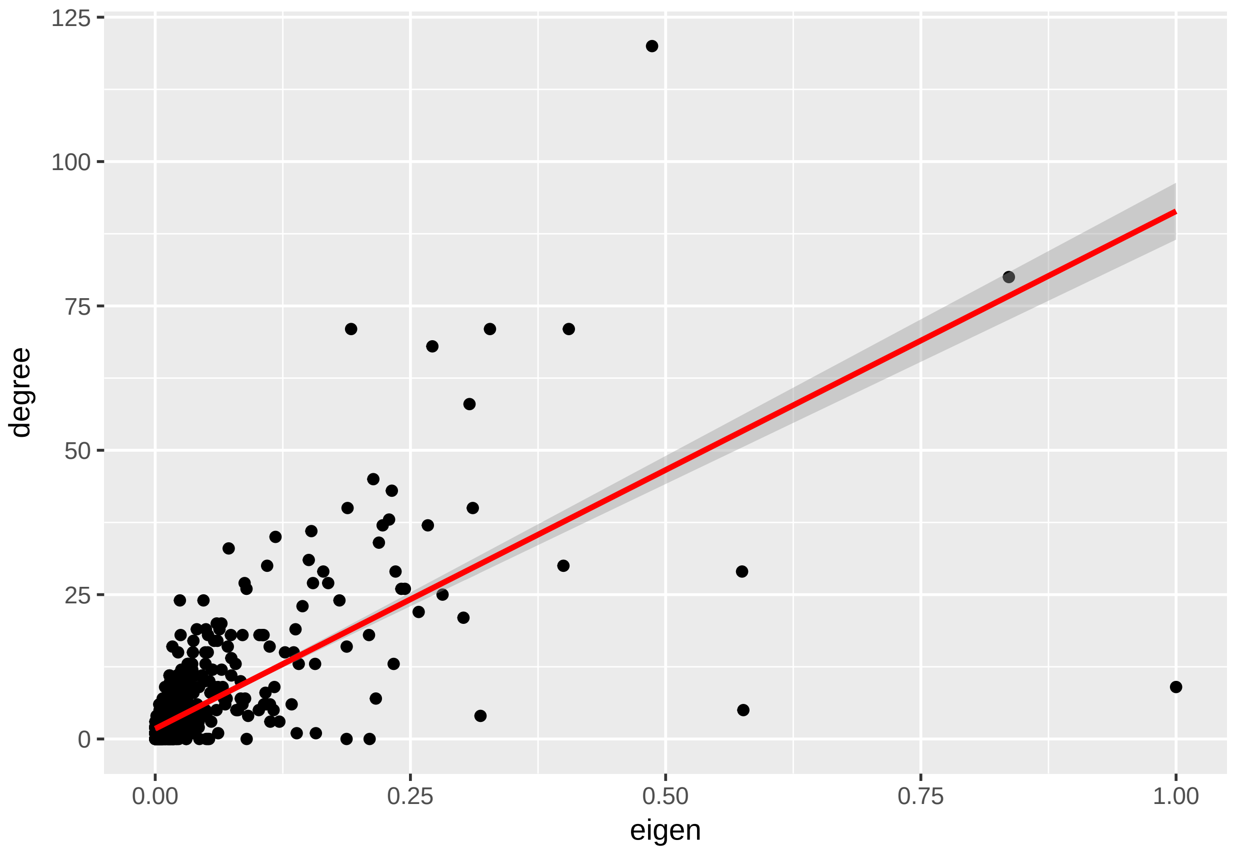
```
ggplot(centrality_measures, aes(x = closeness, y = degree)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```



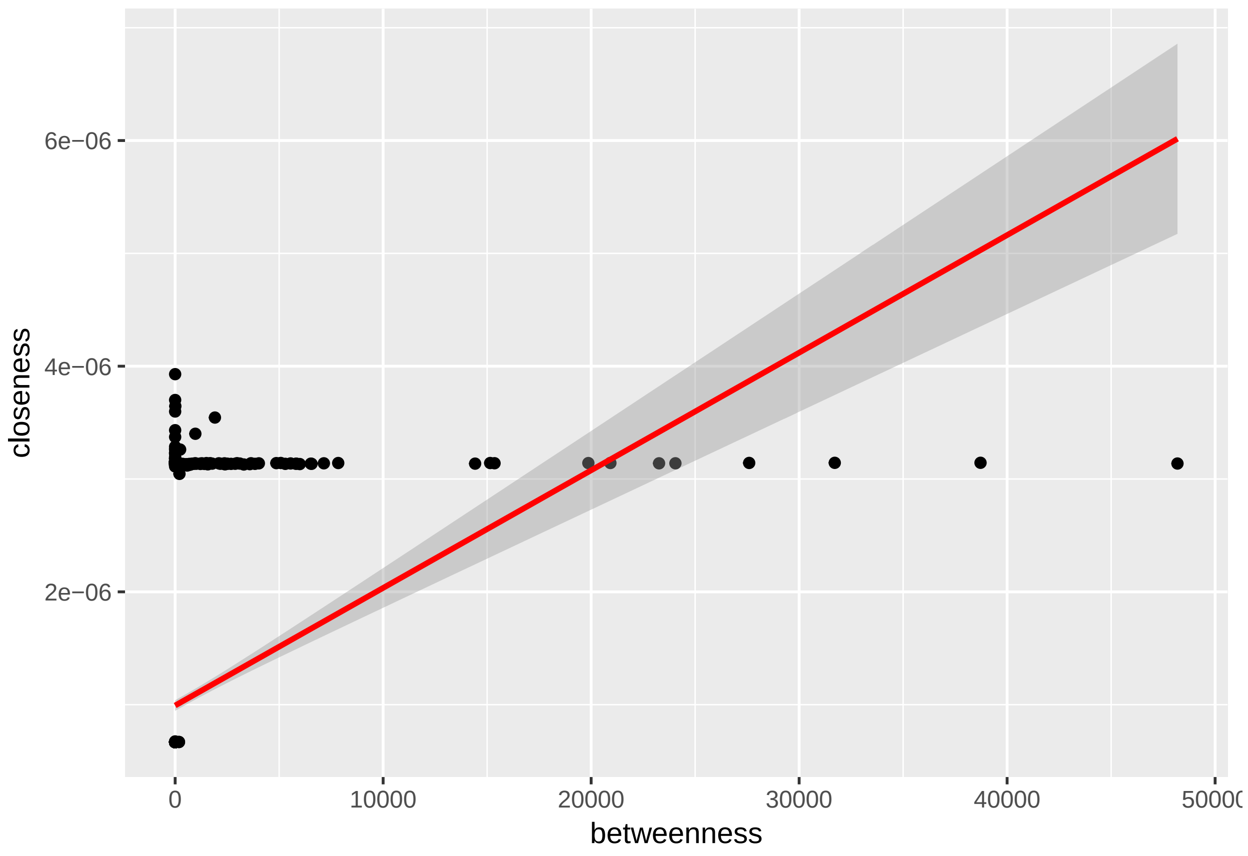
```r
ggplot(centrality_measures, aes(x = degree, y = betweenness)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```



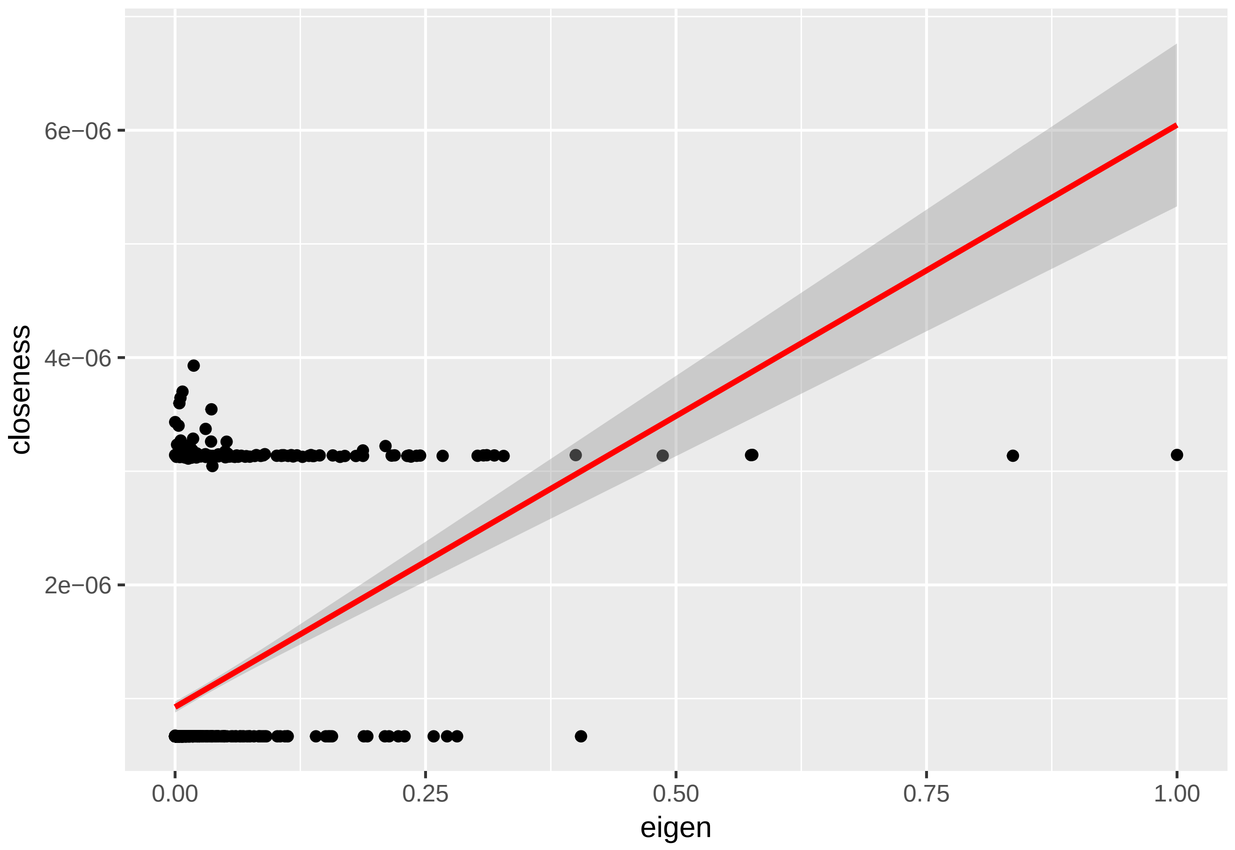
```
ggplot(centrality_measures, aes(x = eigen, y = degree)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```



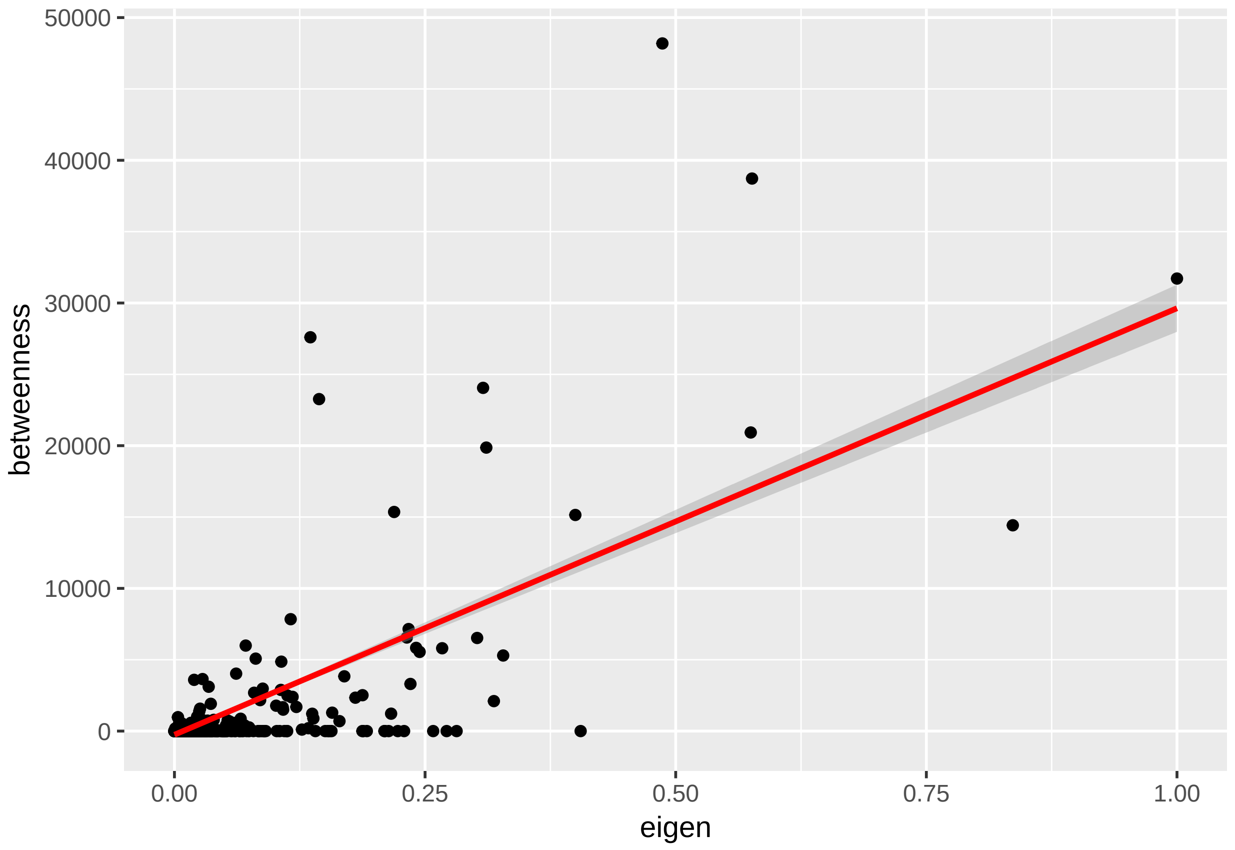
```r
ggplot(centrality_measures, aes(x = betweenness, y = closeness)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```



```r
ggplot(centrality_measures, aes(x = eigen, y = closeness)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```



```r
ggplot(centrality_measures, aes(x = eigen, y = betweenness)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```



```
top_degree <- head(arrange(centrality_measures, desc(degree)), 20) %>%
  select(screen_name, degree)

top_closeness <- head(arrange(centrality_measures, desc(closeness)), 20) %>%
  select(screen_name, closeness)

top_betweenness <- head(arrange(centrality_measures, desc(betweenness)), 20) %>%
  select(screen_name, betweenness)

top_eigen <- head(arrange(centrality_measures, desc(eigen)), 20) %>%
  select(screen_name, eigen)

top_nodes <- cbind(top_betweenness, top_closeness, top_degree, top_eigen)

vertex_attributes$centrality_rank <- data.frame(matrix(vector(), 1245, 1))
vertex_attributes$centrality_rank <- 0

# Count how many of the top 20 lists of centrality measures a node belongs to
for(name in vertex_attributes$screen_name) {
  if(name %in% top_betweenness$screen_name) {
    vertex_attributes$centrality_rank[vertex_attributes$screen_name == name] <- vertex_attributes$centra
  }
  if(name %in% top_closeness$screen_name) {
    vertex_attributes$centrality_rank[vertex_attributes$screen_name == name] <- vertex_attributes$centra
  }
  if(name %in% top_degree$screen_name) {
    vertex_attributes$centrality_rank[vertex_attributes$screen_name == name] <- vertex_attributes$centra
  }
  if(name %in% top_eigen$screen_name) {
    vertex_attributes$centrality_rank[vertex_attributes$screen_name == name] <- vertex_attributes$centra
  }
}
top_betweenness$screen_name[which(top_betweenness$screen_name %in% top_eigen$screen_name)]
```

```
##  [1] "StopMordazas"    "LA_PAH"          "PAH_BCN"
##  [4] "PahSantSadurni"  "PAH_Valencia"    "RafaUsui"
##  [7] "PahRipollet"     "ImPAHrable"      "LuchoMasan"
## [10] "PAHCBages"       "santimdx5"       "PAHSEGOVIA"
## [13] "CleliaVirginiaR" "JuanFcoSierra1"
```

```
top_betweenness$screen_name[which(top_betweenness$screen_name %in% top_closeness$screen_name)]
```

```
## character(0)
```

```
top_betweenness$screen_name[which(top_betweenness$screen_name %in% top_degree$screen_name)]
```

```
## [1] "StopMordazas"   "PahSantSadurni" "RafaUsui"       "PAHGirones"
## [5] "PahRipollet"    "ImPAHrable"     "PAHSantBoi"     "PAHSEGOVIA"
## [9] "JuanFcoSierra1"
```

```
top_betweenness$screen_name[which(top_eigen$screen_name %in% top_closeness$screen_name)]
```

```
## character(0)
```

```
top_betweenness$screen_name[which(top_eigen$screen_name %in% top_degree$screen_name)]
```

```
## [1] "LA_PAH"         "PahSantSadurni" "LlumLlumeta"    "PAH_Valencia"
## [5] "RafaUsui"       "PahRipollet"    "ImPAHrable"     "PAHSantBoi"
## [9] "PAHCBages"
```

```
top_betweenness$screen_name[which(top_closeness$screen_name %in% top_degree$screen_name)]
```

```
## character(0)
```

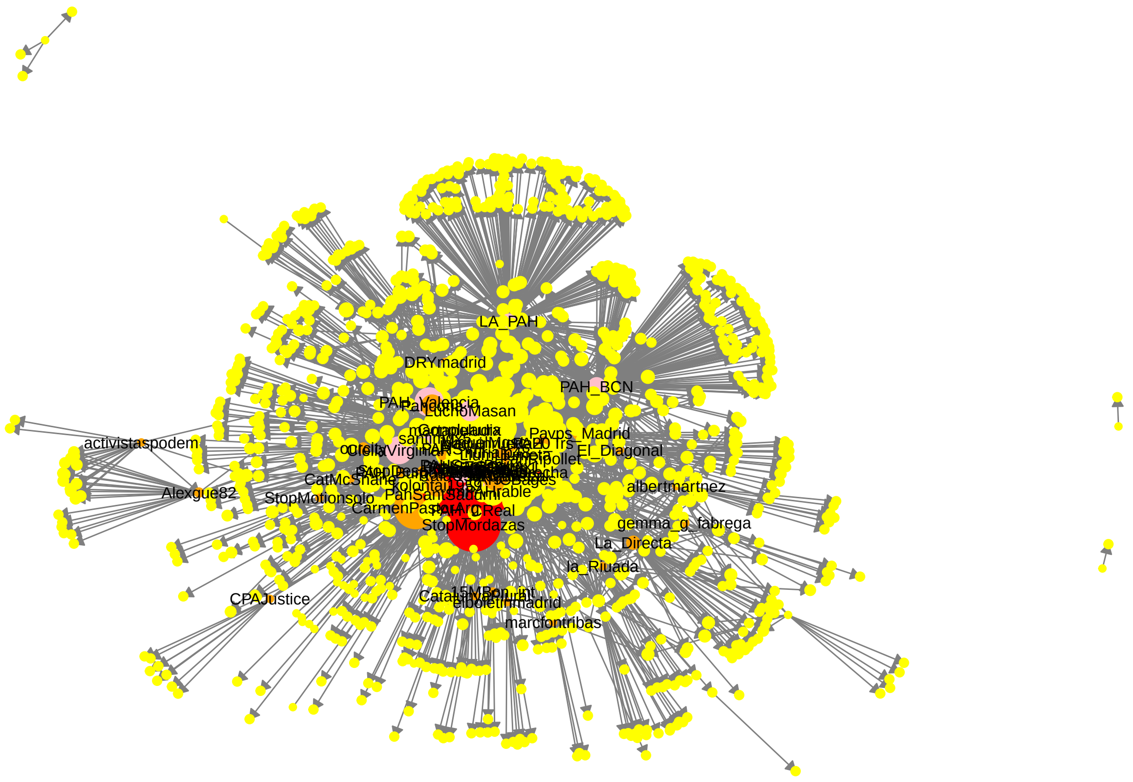
```
V(g)$centrality_rank <- vertex_attributes$centrality_rank[match(V(g)$name, as.character(vertex_attribute

V(g)[is.na(V(g)$centrality_rank)]$color <- "yellow"
V(g)[V(g)$centrality_rank %in% 0]$color <- "yellow"
V(g)[V(g)$centrality_rank %in% 1]$color <- "orange"
V(g)[V(g)$centrality_rank %in% 2]$color <- "pink"
V(g)[V(g)$centrality_rank %in% 3]$color <- "red"

V(g)$label <- vertex_attributes$screen_name[match(V(g)$name, as.character(vertex_attributes$screen_name

for(i in V(g)$name) {
  if(V(g)[V(g)$name == i]$centrality_rank %in% c(1, 2, 3)) {
    V(g)[V(g)$name == i]$label <- vertex_attributes$screen_name[vertex_attributes$screen_name == i]
  } else {
    V(g)[V(g)$name == i]$label <- ""
  }
}


ggnet2(net, size = "indegree", arrow.size = 4, color = "red", legend.position = "none", node.color = V(
```



3a. If you have a network with attribute data, then state some hypothesis about how an attribute may be related to some (or all of the) measures of centrality. Explains why you think these two variables should be related.

I start with the hypothesis, that the nodes with a high degree are also accounts with a lot of followers, friends or accounts that are featured on many lists. Running both the basic OLS and the boostrapped model, I was unable to find a statistically relevant relationship. Even if the relationship had been statistically significant, the coefficients were too small to draw any conclusions.

```r
vertex_attributes <- merge(vertex_attributes, centrality_measures, by = "screen_name")

lm <- lm(degree ~ followers_count + friends_counts + listed_count, vertex_attributes)
summary(lm)
```

```
##
## Call:
## lm(formula = degree ~ followers_count + friends_counts + listed_count,
##     data = vertex_attributes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -3.835  -2.742  -2.455  -0.736 116.639
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.809e+00  2.853e-01  13.348   <2e-16 ***
## followers_count  3.531e-05  5.373e-05   0.657    0.511
## friends_counts  -1.535e-04  2.006e-04  -0.765    0.444
## listed_count    -2.981e-03  2.821e-03  -1.057    0.291
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.968 on 1215 degrees of freedom
## Multiple R-squared:  0.002034,   Adjusted R-squared:  -0.0004301
## F-statistic: 0.8255 on 3 and 1215 DF,  p-value: 0.4798
```

```
boot <- lm.boot(lm, 50)
summary(boot)
```

```
## BOOTSTRAP OF LINEAR MODEL  (method = rows)
##
## Original Model Fit
## ------------------
## Call:
## lm(formula = degree ~ followers_count + friends_counts + listed_count,
##     data = vertex_attributes)
##
## Coefficients:
##     (Intercept)  followers_count   friends_counts     listed_count
##       3.809e+00        3.531e-05       -1.536e-04       -2.981e-03
##
## Bootstrap SD's:
##     (Intercept)  followers_count   friends_counts     listed_count
##    2.327132e-01     2.479661e-05     1.765452e-04     1.453421e-03
```

4. In either case, when you are done above, then considers alternate specifications of your variables and codings and decisions and models. What would you want to consider changing and why. If you can, report on what are the consequences of those changes?

Moving on, I instead tried if the type of content that a user posted affected their position in the retweet network. In this model, I tested how degree was affected by the average number of photos and videos posted by a user as well as the total amount of tweets they had posted and the average amount of hashtags they used in their tweets. This time the OLS model gave statistically relevan results for photos and the total amount of tweets and these coefficients remaind relevant also in the bootstrapped model.

```
lm_altered <- lm(degree ~ photos + vids + n + mean_hashtags, vertex_attributes)
summary(lm_altered)
```

```
##
## Call:
## lm(formula = degree ~ photos + vids + n + mean_hashtags, data = vertex_attributes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.868  -0.658  -0.355   0.182  17.537
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.802474   0.240916   3.331 0.000892 ***
## photos        0.399493   0.185206   2.157 0.031200 *
## vids          3.706052   4.222390   0.878 0.380273
## n             0.359553   0.003648  98.563  < 2e-16 ***
## mean_hashtags 0.096267   0.157151   0.613 0.540272
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 2.654 on 1214 degrees of freedom
## Multiple R-squared:  0.8894, Adjusted R-squared:  0.889
## F-statistic:  2441 on 4 and 1214 DF,  p-value: < 2.2e-16
```

```r
boot_altered <- lm.boot(lm_altered, 50)
summary(boot_altered)
```

```
## BOOTSTRAP OF LINEAR MODEL  (method = rows)
##
## Original Model Fit
## ------------------
## Call:
## lm(formula = degree ~ photos + vids + n + mean_hashtags, data = vertex_attributes)
##
## Coefficients:
##   (Intercept)         photos           vids              n  mean_hashtags
##       0.80247        0.39949        3.70605        0.35955        0.09627
##
## Bootstrap SD's:
##   (Intercept)         photos           vids              n  mean_hashtags
##    0.22383381     0.13575071     7.49093490     0.03181002     0.07685207
```

5. Lastly, give your best conclusion as to what you learned from your analysis. Did it make sense, given your initial expectations? Why? Why not?

The conclusion to draw from the analysis is that a user can be powerful in the network even if they do not necessarily have a lot of followers or friends from before. What they need to do is post a lot and feature images. This contradicts my expectations, which was that users with a lot of followers and friends from before would be more powerful in the retweet network *and* that the type of content and amount of tweets would also matter.