Final Project - Twitch Network Analysis

James Schwann & John Cappuzzello

2026-02-19

Contents

1 Networks Theory & Analysis: Final Project - Insight into Twitch Networks **1**

1.1 Commentary on Network Plot, Ego Network, Centrality Table, & Boxplot 13

1.2 Commentary on ERGM Analysis 15

1.3 Commentary on Diffusion Analysis 19

1 Networks Theory & Analysis: Final Project - Insight into Twitch Networks

My partner and I decided to use a Stanford network dataset. The description of the network is as follows: "A social network of Twitch users which was collected from the public API in Spring 2018. **Nodes are Twitch users** and **edges are mutual follower relationships between them.**"

Source citation: ***B. Rozemberczki and R. Sarkar. Twitch Gamers: a Dataset for Evaluating Proximity Preserving and Structural Role-based Node Embeddings. 2021.***

```
rm(list = ls())
gc()
```

```
##           used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells  579665 31.0    1298365 69.4         NA   715640 38.3
## Vcells 1117241  8.6    8388608 64.0      16384  2010404 15.4
```

```
cat("\014")
```

```r
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin20
## Running under: macOS Tahoe 26.2
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Indiana/Indianapolis
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.5.1    fastmap_1.2.0     cli_3.6.5         ragg_1.5.0
##  [5] tools_4.5.1       htmltools_0.5.8.1 rstudioapi_0.17.1 yaml_2.3.10
##  [9] rmarkdown_2.29    knitr_1.50        xfun_0.53         digest_0.6.37
## [13] textshaping_1.0.3 lifecycle_1.0.4   systemfonts_1.3.1 rlang_1.1.6
## [17] evaluate_1.0.5
```

```r
# Loading all libraries we accessed throughout course to have full "tool box"
# of analysis at disposal
library(igraph)
```

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

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

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

```r
library(tidygraph)
```

```
##
## Attaching package: 'tidygraph'

## The following object is masked from 'package:igraph':
##
##     groups
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(ggraph)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.5.2
```

```
library(ggpubr)
library(ggraph)
library(dplyr)
```

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

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

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

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

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:igraph':
##
##     crossing
```

```
library(igraph)
library(sna)
```

```
## Loading required package: statnet.common
```

```
## Warning: package 'statnet.common' was built under R version 4.5.2
```

```
##
## Attaching package: 'statnet.common'
```

```
## The following objects are masked from 'package:base':
##
##     attr, order, replace

## Loading required package: network

##
## 'network' 1.19.0 (2024-12-08), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information

##
## Attaching package: 'network'

## The following objects are masked from 'package:igraph':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute

## sna: Tools for Social Network Analysis
## Version 2.8 created on 2024-09-07.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.

##
## Attaching package: 'sna'

## The following objects are masked from 'package:igraph':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census
```

```
library(intergraph)
library(gplots)
```

```
## Warning: package 'gplots' was built under R version 4.5.2

##
## ---------------------
## gplots 3.3.0 loaded:
##   * Use citation('gplots') for citation info.
##   * Homepage: https://talgalili.github.io/gplots/
##   * Report issues: https://github.com/talgalili/gplots/issues
##   * Ask questions: https://stackoverflow.com/questions/tagged/gplots
##   * Suppress this message with: suppressPackageStartupMessages(library(gplots))
## ---------------------
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(ggplot2)
library(lme4)
```

```
## Warning: package 'lme4' was built under R version 4.5.2

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(lmerTest)
```

```
## Warning: package 'lmerTest' was built under R version 4.5.2

##
## Attaching package: 'lmerTest'

## The following object is masked from 'package:lme4':
##
##     lmer

## The following object is masked from 'package:stats':
##
##     step
```

```
library(ergm)
```

```
## Warning: package 'ergm' was built under R version 4.5.2

## Registered S3 methods overwritten by 'ergm':
##   method               from
##   simulate.formula     lme4
##   simulate.formula_lhs lme4

##
## 'ergm' 4.11.0 (2025-12-22), part of the Statnet Project
## * 'news(package="ergm")' for changes since last version
## * 'citation("ergm")' for citation information
## * 'https://statnet.org' for help, support, and other information
```

```
## 'ergm' 4 is a major update that introduces some backwards-incompatible
## changes. Please type 'news(package="ergm")' for a list of major
## changes.

##
## Attaching package: 'ergm'

## The following object is masked from 'package:statnet.common':
##
##     snctrl
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(btergm)
```

```
## Registered S3 methods overwritten by 'btergm':
##   method    from
##   plot.gof  ergm
##   print.gof ergm

## Package:  btergm
## Version:  1.11.1
## Date:     2025-03-19
## Authors:  Philip Leifeld (University of Manchester)
##           Skyler J. Cranmer (The Ohio State University)
##           Bruce A. Desmarais (Pennsylvania State University)

##
## Attaching package: 'btergm'

## The following object is masked from 'package:ergm':
##
##     gof
```

```
library(EpiModel) # for diffusion
```

```
## Loading required package: deSolve

## Warning: package 'deSolve' was built under R version 4.5.2

## Loading required package: networkDynamic
```

```
##
## 'networkDynamic' 0.11.5 (2024-11-21), part of the Statnet Project
## * 'news(package="networkDynamic")' for changes since last version
## * 'citation("networkDynamic")' for citation information
## * 'https://statnet.org' for help, support, and other information

## Loading required package: tergm

## Registered S3 method overwritten by 'tergm':
##   method                   from
##   simulate_formula.network ergm

##
## 'tergm' 4.2.2 (2025-06-15), part of the Statnet Project
## * 'news(package="tergm")' for changes since last version
## * 'citation("tergm")' for citation information
## * 'https://statnet.org' for help, support, and other information

##
## Attaching package: 'tergm'

## The following object is masked from 'package:ergm':
##
##     snctrl

## The following object is masked from 'package:statnet.common':
##
##     snctrl

##
## Attaching package: 'EpiModel'

## The following object is masked from 'package:magrittr':
##
##     set_attr

## The following objects are masked from 'package:igraph':
##
##     add_vertices, delete_edges, delete_vertices
```

```r
library(netdiffuseR) # for diffusion
```

```
## Warning: package 'netdiffuseR' was built under R version 4.5.2

## Thank you for using netdiffuseR! Please consider citing it in your work.
## You can find the citation information by running
##   citation("netdiffuseR")

##
## Attaching package: 'netdiffuseR'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode

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

```
library(parallel)
```

```
edges = read.csv("twitch_edges.csv") # Edges data
nodes = read.csv("large_twitch_features.csv") # Nodes data

## View(edges) # Roughly 6.7 M edges
## View(nodes) # Roughly 168k nodes - overall HUGE network

# To properly create network object, we need to make sure the first column
# of edges/nodes dataframes both are numeric_id. To do this, we use below

nodes_new = nodes %>%
  select(numeric_id, everything()) # Using piping operator and then reorder
# columns in new dataframe nodes_new so numeric_id leads first!

# This is an undirected network, so we will create an undirected network object
# using igraph package

network = graph_from_data_frame(edges, directed = F, vertices = nodes_new)
```

```
# Like an idiot, I tried plotting this whole network, which
# crashed my R Session abruptly. As such, I cannot do this. I must take samples
# and only graph subsets of the network
set.seed(1) # Setting seed to 1 for plot consistency - will do this for all
# code chunks

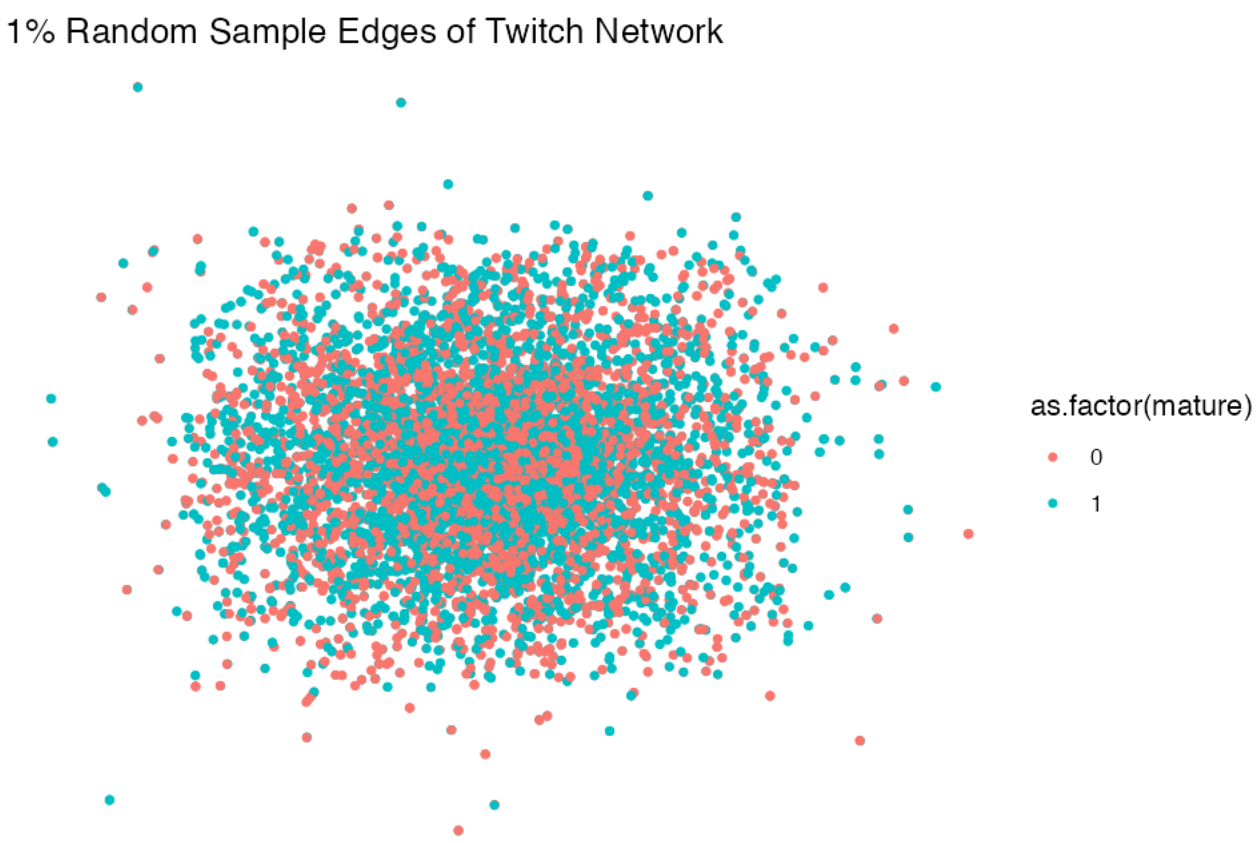

# Create 1% random sample of edges so R preview window can handle
# visualization

# Gemini Pro used to help create this code (see Appendix)
random_sample = sample(E(network), ecount(network) * 0.01)
sub_graph = subgraph.edges(network, random_sample)
```

```
## Warning: `subgraph.edges()` was deprecated in igraph 2.1.0.
## i Please use `subgraph_from_edges()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# In aesthetic function, setting color to factor mature --> which is a binary
# dummy variable. 0 means Twitch user does NOT stream or demonstrate explicit
# content. A 1 means Twitch user DOES stream explicit content
```

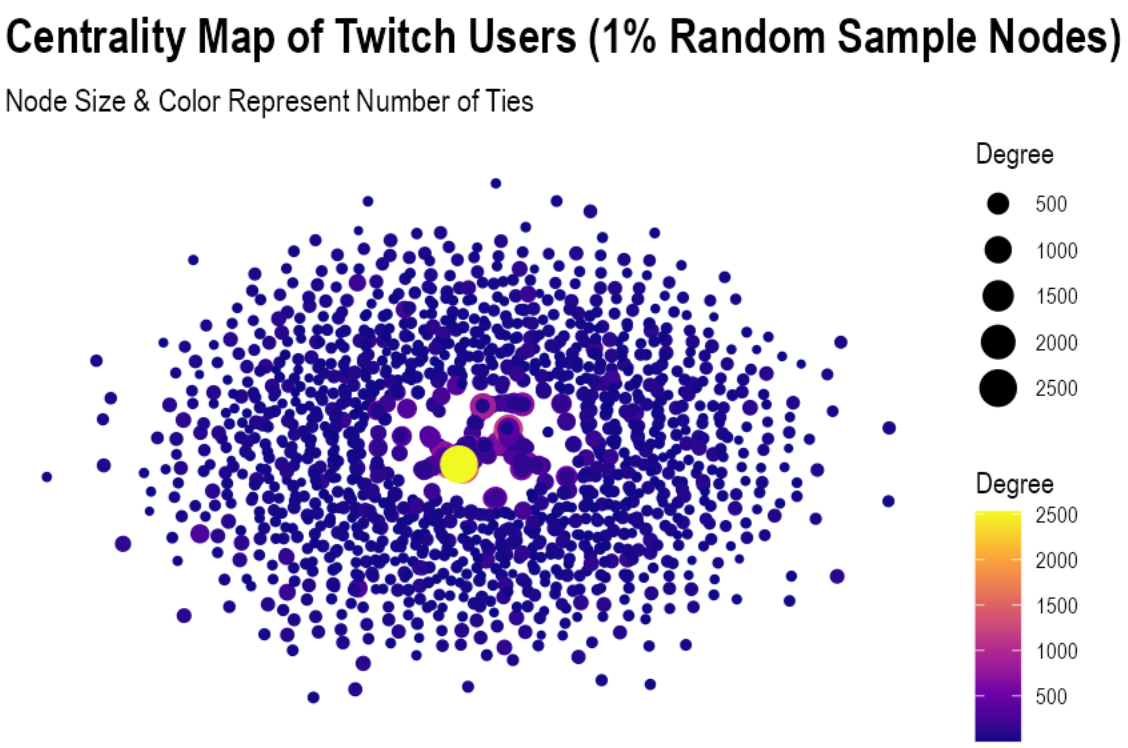
```r
# Plot visualization of 1% sample --> Super dense network
ggraph(sub_graph, layout = "fr") +
  geom_edge_link0(alpha = 0.1) +
  geom_node_point(aes(color = as.factor(mature)), size = 1) +
  theme_void() +
  labs(title = "1% Random Sample Edges of Twitch Network")
```




```r
# Gemini Pro used to help create code below (see Appendix)
# Calc centrality for whole graph
V(network)$centrality = igraph::degree(network)


random_nodes = sample(V(network), vcount(network) * 0.01) # 1% sample nodes
sample_graph = induced_subgraph(network, random_nodes)

ggraph(sample_graph, layout = "nicely") +
  geom_edge_link0(alpha = 0.05, color = "gray") +
  geom_node_point(aes(size = centrality, color = centrality)) +
  scale_color_viridis_c(option = "plasma") +
  theme_graph() +
  labs(title = "Centrality Map of Twitch Users (1% Random Sample Nodes)",
       subtitle = "Node Size & Color Represent Number of Ties",
       size = "Degree", color = "Degree")
```



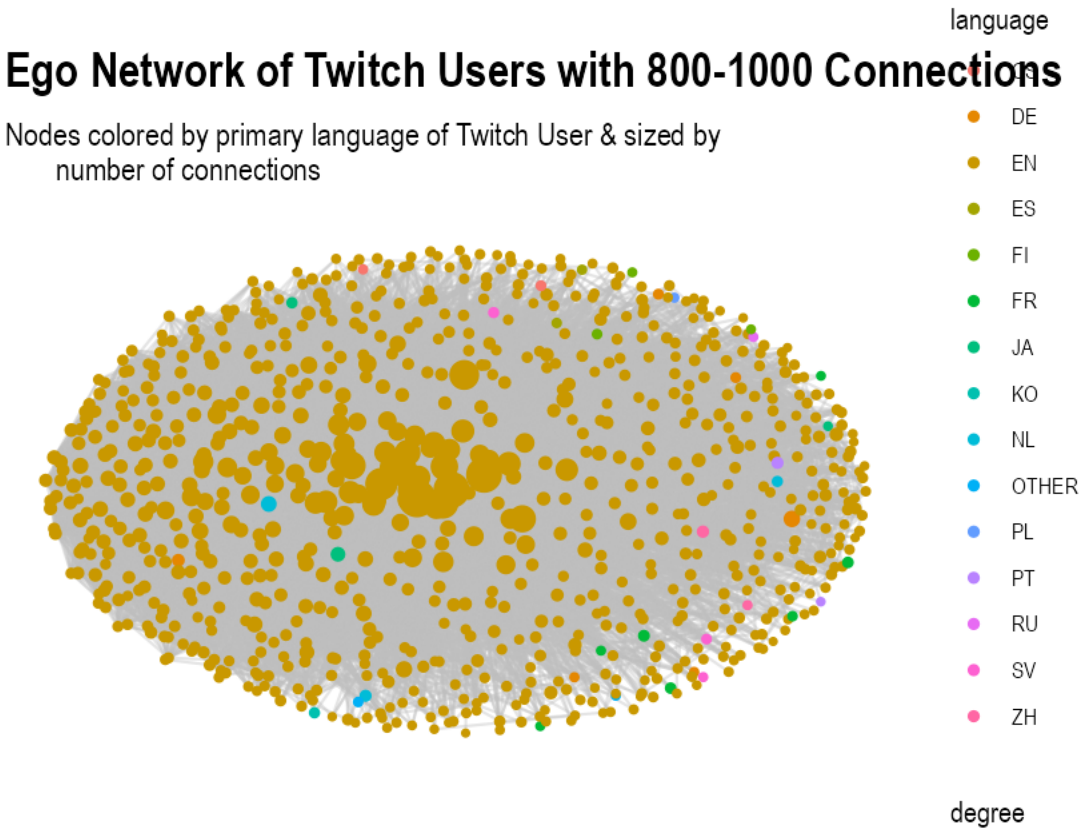

```
# Ego Network

# Gemini Pro helped with code here (see Appendix)

# Calc node with highest degree
V(network)$degree = igraph::degree(network)
mid_node = which(V(network)$degree > 800 & V(network)$degree<1000)[1]

ego_network = igraph::make_ego_graph(network, order = 1, nodes = mid_node)[[1]] %>%
  as_tbl_graph()

# Plot ego net
ggraph(ego_network, layout = "stress") +
  geom_edge_link0(alpha = 0.40, color = "gray") +
  geom_node_point(aes(color = language, size = degree)) +
  theme_graph() +
  labs(title = "Ego Network of Twitch Users with 800-1000 Connections",
       subtitle = "Nodes colored by primary language of Twitch User & sized by
       number of connections")
```




```
# Eigen V. Degree Centrality Rank Comparison

V(network)$degree = igraph::degree(network)
V(network)$eigenm = igraph::eigen_centrality(network)$vector

node_comparison = igraph::as_data_frame(network, what = "vertices")

# Get Top 5 for Degree from node_comparison dataframe
top_5_degree = node_comparison %>%
  select(name, views, mature, language, degree) %>%
  arrange(desc(degree)) %>%
  head(5) %>%
  mutate(Rank = 1:5, Type = "Top Degree Centrality")

# Get Top 5 for Eigen
top_5_eigen = node_comparison %>%
  select(name, views, mature, language, eigenm) %>%
  arrange(desc(eigenm)) %>%
  head(5) %>%
  mutate(Rank = 1:5, Type = "Top Eigenvector Centrality")

comparison = bind_rows(top_5_degree, top_5_eigen) %>%
  rename(User_ID = name)

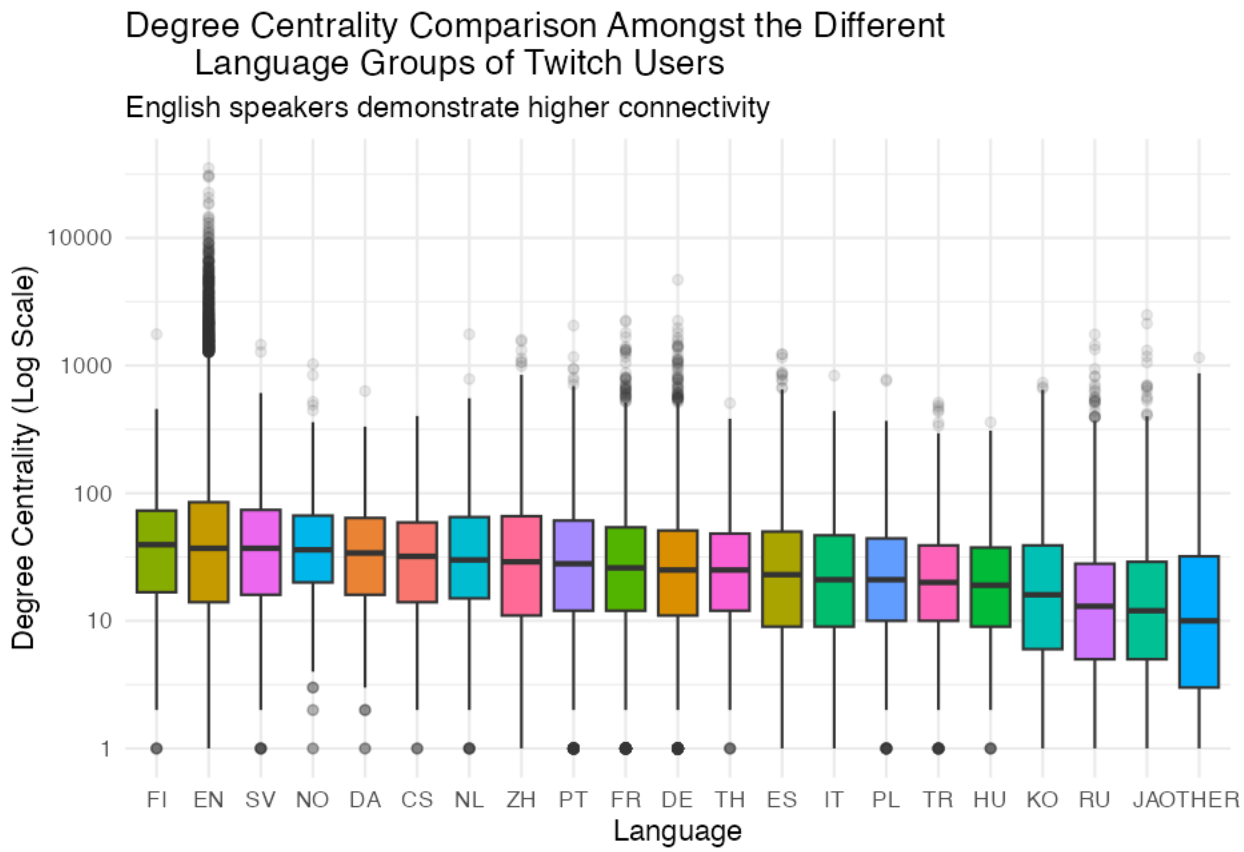
knitr::kable(comparison, caption = "Comparison of User Popularity
             (Degree Centrality) and User Influence (Eigen Centrality)")
```

Table 1: Comparison of User Popularity (Degree Centrality) and User Influence (Eigen Centrality)

	User_ID	views	mature	language	degree	Rank	Type	eigenm
61862...1	61862	340602050	0	EN	35279	1	Top Degree Centrality	NA
125642...2	125642	243451177	0	EN	31064	2	Top Degree Centrality	NA
32338...3	32338	202142952	0	EN	30499	3	Top Degree Centrality	NA
71050...4	71050	218559516	0	EN	29912	4	Top Degree Centrality	NA
110345...5	110345	264643346	0	EN	22675	5	Top Degree Centrality	NA
125642...6	125642	243451177	0	EN	NA	1	Top Eigenvector Centrality	1.0000000
32338...7	32338	202142952	0	EN	NA	2	Top Eigenvector Centrality	0.9512437
61862...8	61862	340602050	0	EN	NA	3	Top Eigenvector Centrality	0.9378773
110345...9	110345	264643346	0	EN	NA	4	Top Eigenvector Centrality	0.8620586
71050...10	71050	218559516	0	EN	NA	5	Top Eigenvector Centrality	0.8167397

```
# Analyze Degree Centrality Across language Groups
# We've already created a measure for degree in above code & stored in
# node_comparison dataframe

# Use ggplot to create boxplot of Centrality by Language
ggplot(node_comparison, aes(x = reorder(language, -degree, FUN = median),
                            y = degree, fill = language)) +
  geom_boxplot(outlier.alpha = 0.1) +
  scale_y_log10() + # Use log scale for degree for better measurement
  theme_minimal() +
  labs(title = "Degree Centrality Comparison Amongst the Different
       Language Groups of Twitch Users",
       subtitle = "English speakers demonstrate higher connectivity",
       x = "Language", y = "Degree Centrality (Log Scale)") +
  theme(legend.position = "none")
```



1.1 Commentary on Network Plot, Ego Network, Centrality Table, & Boxplot

Regarding our first visualization, we can see that even a 1% random sample of the edges/nodes creates a highly dense network object. Given our dataset has roughly 168,000 nodes and over 6 million edges, we had to take a small random sample to graph a portion of it successfully. This visualization seems to demonstrate that there seems to be an almost 50/50 split between explicit Twitch users (i.e. those playing explicit games or using explicit language) and non-explicit users.

Regarding our second visualization, we see the yellow node is the node of interest. This visualization makes both size of node and color correspond to degree centrality (number of ties). As such, we can see that the yellow node is the the most popular node within this sample. Most likely, this is a somewhat popular Twitch streamer/influencer. This node has over 2500 ties compared to the smaller blue nodes that have less than 500 ties.

The last visualization is very unique. It is an ego network (appropriate given the context of our dataset) that includes Twitch users with anywhere from 800-1000 ties (specifically 800<User Ties<1000, so non-inclusive endpoints) where color corresponds to language of user and size corresponds to number of connections/ties a Twitch user has. This graph demonstrates that English Twitch users dominate. Not only are these nodes more abundant than all the others, but they tend to have significantly more connections than others. Naturally, this leads us to ask the following: Is the measure of overall degree centrality highly related to Eigen-Vector centrality? In other words, are Twitch users who are super popular (have an enormous number of ties, i.e. high degree centrality) also very influential (i.e. high Eigenvector)?

As we can see via the table, it PAYS "richly" to be a Twitch User with a high following (a high degree centrality). The table demonstrates that the five most popular users in the dataset ALSO happen to be the same users with the greatest Eigen-Vector centrality. In other words, Twitch is not a social platform that

has hidden influencers: people with relatively low following but who influence greatly. Instead, this table demonstrates popularity is highly correlated to influence.

Lastly, the boxplot demonstrates a wonderful yet simple visualization showing log(degree centrality) of Twitch users grouped by their language bins. As we can see, English Twitch users have far more log(degree) outliers compared to all of the other Twitch users of different languages. This is very interesting considering the median log(degree) centrality for English Twitch users is roughly in line with the following languages: "FI," "SV," and "NO" –> Finnish, Swedish, and Norwegian, respectively. Trying to hypothesize a cause for this phenomenon is difficult. My partner and I believe it has nothing to do with the English language being more innately connective. Instead, English users are the dominant users on the platform meaning there is a greater probability that a super popular Twitch streamer is English. If this is the case, it is then very logical for the English bin to have significantly more outliers. The most popular Twitch streamers/users seem to be English.

```
# Set seed
set.seed(1) # For consistency

# Create SNA network object for ERGM analysis

network_sna_1pct = asNetwork(sample_graph) # Using 1% sample from previous
# code chunk as data is massive.

# Test homophily on categorical language and mature variables

model1_1pct = ergm(network_sna_1pct ~ edges +
               nodematch("language") + nodematch("mature"))
```

```
## Starting maximum pseudolikelihood estimation (MPLE):

## Obtaining the responsible dyads.

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Evaluating log-likelihood at the estimate.
```

```
summary(model1_1pct)
```

```
## Call:
## ergm(formula = network_sna_1pct ~ edges + nodematch("language") +
##     nodematch("mature"))
##
## Maximum Likelihood Results:
##
##                    Estimate Std. Error MCMC % z value Pr(>|z|)
## edges              -9.29360    0.14029      0 -66.247   <1e-04 ***
## nodematch.language  1.76025    0.14086      0  12.496   <1e-04 ***
## nodematch.mature    0.02933    0.09020      0   0.325    0.745
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1957503  on 1412040  degrees of freedom
##  Residual Deviance:    8583  on 1412037  degrees of freedom
##
## AIC: 8589  BIC: 8626  (Smaller is better. MC Std. Err. = 0)
```

```
# language - Definitely significant
# Mature - P-value is 0.745 so not significant

odds_language = exp(1.76025)
print(odds_language)
```

```
## [1] 5.813891
```

1.2 Commentary on ERGM Analysis

In the code chunk above, we ran an ERGM analysis on the 1% sample of the network (sample_graph from previous code chunk. To do this, we had to create an SNA network object). The coefficients that are in our output are log-odds meaning to convert them to odds, we do e^(coefficient). In our output, we see that the language coefficient of 1.76025 is statistically significant. This demonstrates the following: A Twitch user who shares the same language as another user (i.e. "EN - English", "FR- French," etc.) is roughly 5.81 times more likely to be connected to that user than a user who speaks a different language. Intuitively, this makes complete sense. It is very cool to be able to quantify this relationship. On the other hand, however, the coefficient on our mature variable is not significant. The 1% sample of our network demonstrates that there isn't a greater likelihood that two Twitch users will be connected based on the fact that both are non-explicit (mature = 0) or explicit (mature = 1).

```
network_sna_adj = as_adjacency_matrix(network)
n_nodes = nrow(network_sna_adj)
n_nodes # There are 168,114 nodes in the Twitch network ~ 168k Twitch users
```

```
## [1] 168114
```

```
# For diffusion net, let's again do it on 1% sample of graph
set.seed(1) # For consistency across code chunks

random_nodes_2 = sample(V(network), vcount(network) * 0.01)
sub_graph_1pct = induced_subgraph(network, random_nodes_2)

sub_sna = as_adjacency_matrix(sub_graph_1pct)

diffnet = rdiffnet(
  n = n_nodes, # Total no. nodes in network
  t = 20, # Doing 20 time periods
  seed.graph = sub_sna,
  seed.p.adopt = 0.10, # 10% early adopters
  rewire = F,
  threshold.dist = function(i) 0.10,
  exposure.args = list(normalized = T)
)
```

```
## Warning in rdiffnet(n = n_nodes, t = 20, seed.graph = sub_sna, seed.p.adopt =
## 0.1, : While the user set n=168114, nnodes(seed.graph)=1681. The later will be
## used.
```

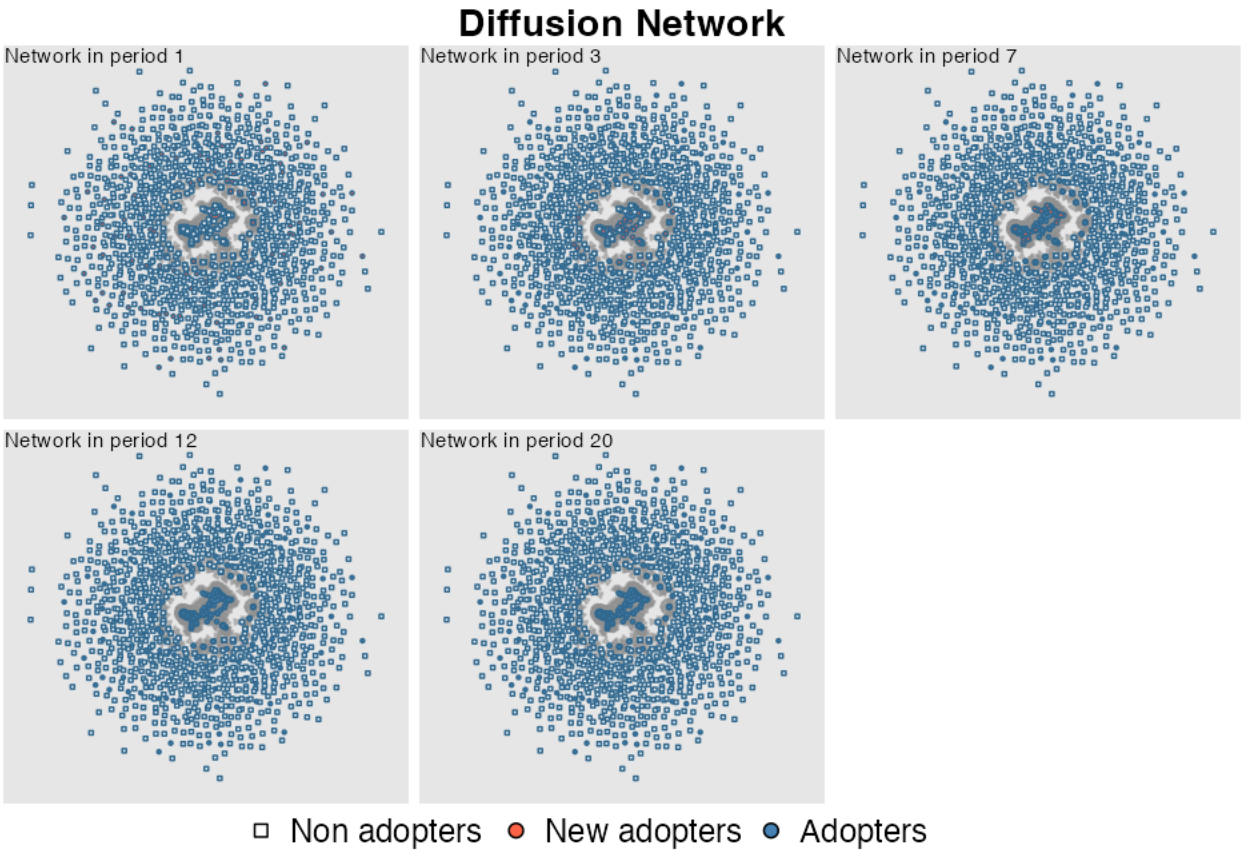
```
summary(diffnet)
```

```
## Diffusion network summary statistics
##  Name     :  A diffusion network
##  Behavior : Random contagion
## ---------------------------------------------------------------------------
##  Period   Adopters   Cum Adopt. (%)   Hazard Rate   Density   Moran's I (sd)
## -------- ---------- ---------------- ------------- --------- ----------------
##       1        168       168 (0.10)             -      0.00  0.02 (0.03)
##       2        154       322 (0.19)          0.10      0.00  0.57 (0.03) ***
##       3        132       454 (0.27)          0.10      0.00  1.03 (0.03) ***
##       4         91       545 (0.32)          0.07      0.00  1.39 (0.03) ***
##       5         42       587 (0.35)          0.04      0.00  1.55 (0.03) ***
##       6         14       601 (0.36)          0.01      0.00  1.59 (0.03) ***
##       7          2       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##       8          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##       9          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      10          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      11          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      12          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      13          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      14          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      15          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      16          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      17          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      18          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      19          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
##      20          0       603 (0.36)          0.00      0.00  1.59 (0.03) ***
## ---------------------------------------------------------------------------
##  Left censoring  : 0.10 (168)
##  Right centoring : 0.64 (1078)
##  # of nodes      : 1681
##
##  Moran's I was computed on contemporaneous autocorrelation using 1/geodesic
##  values. Significane levels  *** <= .01, ** <= .05, * <= .1.
```

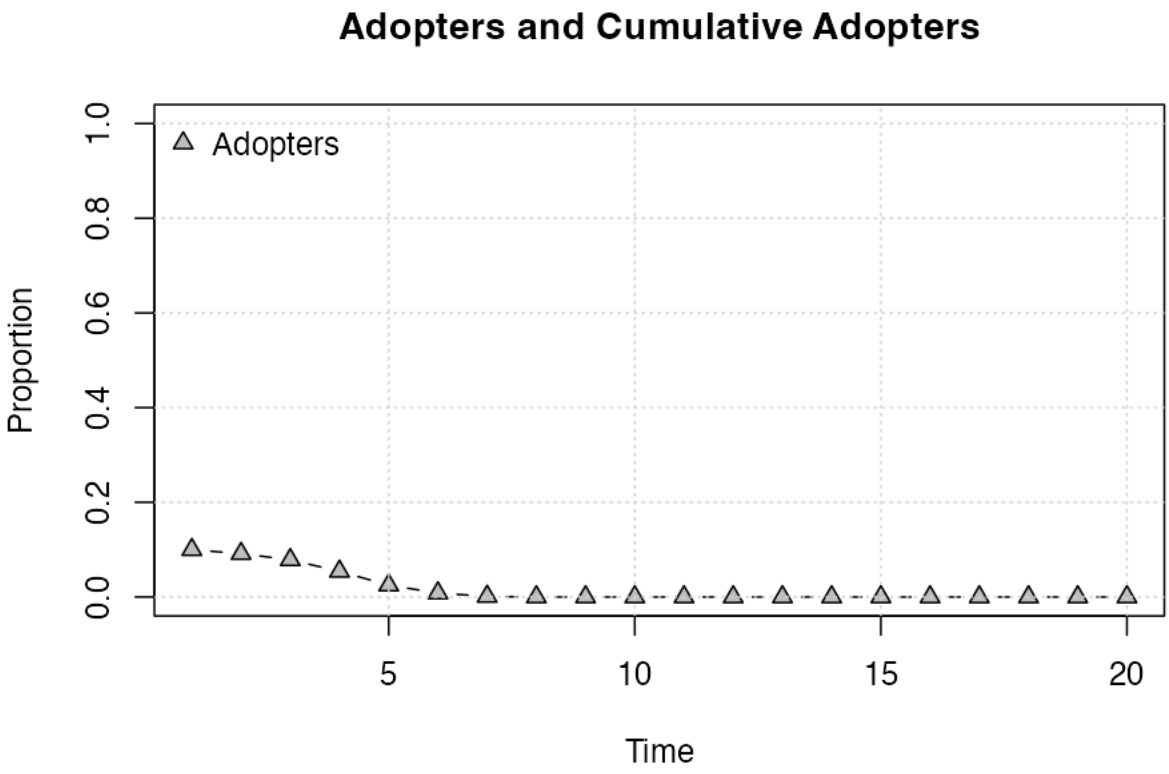
```
table(diffnet$toa) # time of adoption
```

```
##
##   1   2   3   4   5   6   7
## 168 154 132  91  42  14   2
```

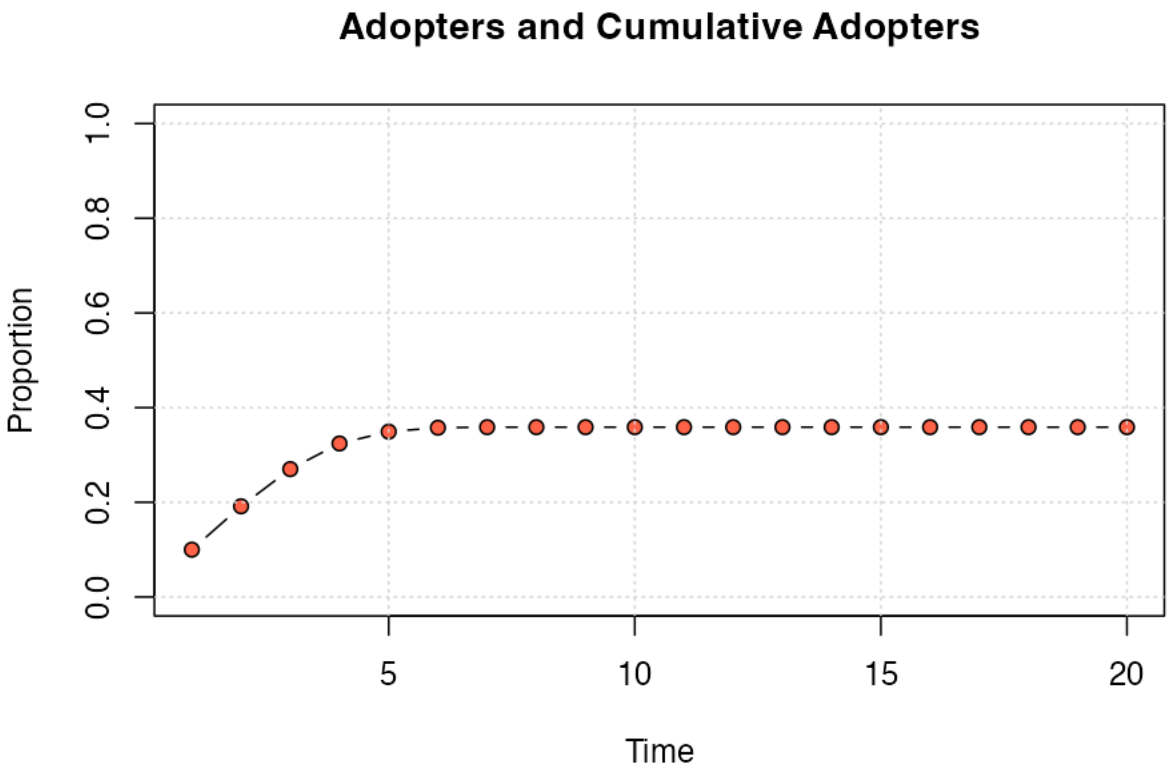
```
plot_diffnet(diffnet, slices = c(1, 3, 7, 12, 20))
```




```
plot_adopters(diffnet, what = "adopt")
```




```
plot_adopters(diffnet, what = "cumadopt", include.legend = FALSE)
```



1.3 Commentary on Diffusion Analysis

The diffusion analysis provides an interesting insight into the Twitch network. My partner and I set adoption to 10% and a time horizon of t = 20 periods. Despite Twitch being a super connected social network, cumulative adoption was under 40% by time period 20. We ran this diffusion on a 1% sample of the network. That being said, we're somewhat confident diffusion was low because several language clusters exist in this sample and the network overall. If Twitch users DO NOT speak the same language, diffusion across those users and language groups is likely to be very minimal/small. If we had run a diffusion model on a select language group (i.e. "EN," "FR," "JA," etc), we would have expected to see a much higher rate of diffusion compared to what we see here.