The idea of assortativity in a network is basically that popular people (or whatever a node is) hang out with eachother and form lots of connections.

First, let’s cut to the chase. Turns out that in fact, looking at the trimmed mean number of followers of the people that Twitter users follow, the relationship with the number of followers of the original person is not straightforward:

In fact, you could say that generally, the more followers you have, the *less* followers the people you follow have – until a threshold of around 100,000 followers, which makes you one of the giants of Twitter (for a reference point that might mean something to readers of this blog, [Hadley Wickham](https://twitter.com/hadleywickham) has around 58,000 followers, so more than 100,000 is really a lot). This finding holds with two different methods of sampling people from Twitter, and is confirmed by more sophisticated modelling that looks simultaneously at the number of people one is following and the number of followers they have.

One plausible explanation is that “number of followers” is a proxy for length of engagement on Twitter. When one starts out on Twitter, you are presented with a bunch of suggested popular accounts to follow (eg sports teams, entertainment celebrities, famous media outlets), typically with very high numbers of followers themselves. So on day one, the average number of followers of the people *you* follow is very high. Over time, the regular user of Twitter follows people (eg your friends) with less followers and the average followers of those people declines. Only for the super-famous, everyone you know or care about is famous and so the homogamy/assortativity thesis kicks in but *iff* you’re a bit selective about who you follow, and don’t just follow back all your non-entity fans.

The issue (I think) that stops Twitter being highly assortative is the one-way nature of following someone. Katy Perry’s 109 million followers don’t need to ask her permission to follow her; this makes the forming of edges between nodes in the Twitter network fundamentally different to marriage.

So what about the references mentioned in the Twitter exchange after Milanovic’s first tweet on this? What I see in the literature (admittedly after a very cursory glance – I’m pressed for time as I write this) is a bit different. There’s a couple of pieces suggesting *happiness* is assortative – people who write happy tweets associate with others who do the same, and the same applies to people who write unhappy tweets. Also, the action of replying and retweeting is assortative. But contrary to what some said in the Twitter exchange, I don’t see articles showing that the follow/followed network is assortative.

Let’s go into how I did this.

**A sampling problem**

The first challenge is to get a sample of Twitter users. This is harder than it might seem at first, if the aim is (as it should be) to be representative of the population at large. First challenge is defining that population. Do we mean every Twitter account, every human Twitter account, every account that is used for actively tweeting (why would you restrict it to this, as even reading tweets should surely count?).

There’s no conveniently published population of Twitter users. I’m aware of three broad ways one might go about getting a sample of users:

1. You could do the equivalent of “random digit dialling”, making up numeric Twitter identification numbers and checking them in the Twitter API for existence. This method is in fact what you find if you google “how do I get a sample of Twitter users” but Twitter have made it effectively impossible by the (hidden) way they assign IDs. I observed ID numbers up to 10^17 and as low as 10^6, and sampling random numbers between those extremes hoping to hit one of the 300 million or so actual users sounds like a recipe for getting nowhere.
2. You can pick a node and use a snowball sampling method; that is, follow an edge (either a person the first node follows, or a node who follows them) to another node, record what you need to about that person, then follow another edge to a third node, and so on until you have enough people. This is what I did with the sample labelled “snowball sampling along network”.
3. You can sample a bunch of actual tweets, and treat their authors as your sample. This is what I did with the sample labelled “sample of today’s tweeters”.

Method 1 is I think infeasible. Method 2 will oversample users with lots of followers and who follow lots of people – basically, the more networked you are, the more likely you are to be sampled. On the plus side for Method 2, quiet users who lurk but aren’t tweeting these days will have a chance of selection. Method 3 on the other hand will give a very particular slice of users; but it has the advantage of less obvious dependence between the nodes we pick (all they have in common is tweeting in the last couple of minutes of when we harvest them). I was unsure enough about sampling strategy to try them both.

Here’s code to do the sampling, using Jeff Gentry’s twitteR R package. Note that the code that follows isn’t very robust, and took days to run because of Twitter’s rules on maximum downloads via the public API. There’s lots that can go wrong with grabbing data from the Twitter API, hence the extensive and undisciplined use of try() in the code below in an effort to keep the data harvesting going in the face of various quirks .

First, setup. This requires four different pass phrases which associated with your Twitter account (for obvious reasons, blanked out in the below, which stops this script being fully reproducible as-is).

library(tidyverse)

library(twitteR)

library(GGally)

library(mgcv)

consumer\_key <- "XXXXXXX"

consumer\_secret <- "XXXXXXXXXX"

access\_token <- "XXXXXXXXXX"

access\_secret <- "XXXXXXXXXXX"

setup\_twitter\_oauth(consumer\_key, consumer\_secret, access\_token, access\_secret)

As I’m interested in the “average” number of followers of the people a sampled user follows, for each user I sample I’m going to need to find out everyone they follow and estimate how many followers *they* have. This is the thing that takes time. It also exposes a problem; Twitter won’t let you look at more than a certain number of users at once. That number turns out to be 75,000, as I find out from this experiment with [Todd Carey](https://twitter.com/search?q=toddcarey&src=typd):

many\_following\_user <- getUser("toddcarey")

many\_ids <- many\_following\_user$getFriendIDs()

length(many\_ids) # returns only 75000 ids; I think the most recent 75,000 he's followed

The problem here is that as Carey follows so many people (1.28 million) it’s infeasible to get the details of them all. In fact, it’s infeasible even just to get all their 1.28 million screen names and then sample from those (I’m very happy to estimate “average number of followers of people X follows” from a sample). I’ll have to deal with getting 75,000 users he follows, then sampling from those 75,000. The big problem here is that I think those are the most recent 75,000 people he’s followed. All else being equal, these are likely to be newer Twitter users than the overall population of people he follows, and hence likely to bias downwards my estimate of the average number of followers *they* have (as newer users will almost certainly have fewer followers on average).

[As an aside, one might wonder what is the point of following 1.28 million people on Twitter; it’s presumably part of a strategy, automated or not, of attracting followers by implicitly agreeing to follow them back. It’s not fully automatic – I established this by following him to see what happened, and nothing did.]

I don’t see what can be done about this, other than note the people who follow more than 75,000 people as potentially suspect in subsequent analysis. There’s not that many of them in my sample anyway.

Anywhere, here’s the code that does the snowball sampling. I start with myself. I take three types of average number of followers of the people X follows: mean (which is highly vulnerable to an arbitrarily large number getting in the sample), 20% trimmed mean and 50% trimmed mean (ie median); but I’m satisfied that 20% trimmed mean is robust and a good measure.

#=======================snowball sampling method================

follow\_data <- data\_frame(

number\_followers = integer(),

number\_following = integer(),

mean\_ff = numeric(),

trmean\_ff = numeric(),

median\_ff = numeric(),

screenName = character()

)

# Terminology

# A "friend" is someone you follow

# A "follower" is someone who follows you

current\_sn <- "ellis2013nz"

set.seed(124)

for(i in 1:1000){

# save latest copy of results in case of crash

save(follow\_data, file = "follow\_data.rda")

# number of calls left to make under Twitter's limits

lims <- getCurRateLimitInfo()

x <- as.numeric(filter(lims, resource == "/followers/ids")$remaining) \*

as.numeric(filter(lims, resource == "/friends/ids")$remaining)

while(x == 0) {

lims <- getCurRateLimitInfo()

x <- as.numeric(filter(lims, resource == "/followers/ids")$remaining) \*

as.numeric(filter(lims, resource == "/friends/ids")$remaining)

message("Waiting")

Sys.sleep(60) # wait 15 minutes (or 1 minute, 15 times)

}

cat(paste(i, current\_sn, " | "))

current\_user <- getUser(current\_sn)

follow\_data[i, "screenName"] <- current\_sn

# who this user follows (what happens if this is eg 1million people? I think it gets truncated at 75,000,

# which is not as bad as it sounds as we only need a sample; but the problem will be that this is their

# 75,000 most recent people they have followed, who might be biased to be newcomers?):

# certain privacy settings stop you seeing someone's friends and followers, in which case

# we try to sample another node from the last good person

all\_my\_friends <- character()

try(all\_my\_friends <- current\_user$getFriendIDs()) # about 1 second for c.1500 IDs, longer for more

if(length(all\_my\_friends) > 0){

# sample some of those friends

n <- min(2000, length(all\_my\_friends))

sample\_my\_friends <- sample(all\_my\_friends, n, replace = FALSE)

friend\_details <- lookupUsers(sample\_my\_friends) # about 15 seconds

friend\_details\_df <- twListToDF(friend\_details)

follow\_data[i, "number\_followers"] <- current\_user$followersCount

follow\_data[i, "number\_following"] <- current\_user$friendsCount

# mean number of followers of the people this user follows

follow\_data[i, "mean\_ff"] <- mean(friend\_details\_df$followersCount)

follow\_data[i, "trmean\_ff"] <- mean(friend\_details\_df$followersCount, tr = 0.2)

follow\_data[i, "median\_ff"] <- median(friend\_details\_df$followersCount)

}

# find the next user - flip a coin for it to be either a friend or a follower

current\_sn <- "non\_existent\_user"

while(current\_sn == "non\_existent\_user"){

if(sample(1:2, 1) == 1){

# 50% chance

# pick one of the user's friends (ie people they follow) at random as the new node to sample

# note that as a sampling strategy this will bias us towards people who are followed. People

# who have few followers have little chance of being selected here. Does this matter?

# Probably not for our main research question, but it does stop us doing meaningful inference

# on population totals of number of people followed and number of people following - unless

# we could estimate \*how much\* more likely people are to be followed, then we could estimate

# weights to make up for it.

cat("\nsampling a friend\n")

try(current\_sn <- sample\_n(friend\_details\_df, 1)$screenName)

} else {

# 50% chance

# pick one of the user's followers as the new node. This is to mitigate the problem

# noted above. We will still end up over-representing people with lots of followers

# (who will be sampled by the previuos method), and people who follow lots of people

# (more likely to be picked up in this alternative procedure in the next line), but

# that is better than \*only\* over sampling people with lots of followers. Probably.

cat("\nsampling a follower\n")

# problem here when current\_user has no followers:

# a random sample from the last 1000 people to follow this person (ie biased to more

# recent, for people with lots of followers, but c'est la vie - important to speed things

# up a bit)

try(follower\_samp <- sample(current\_user$getFollowerIDs(n = 1000), 1))

try(current\_sn <- lookupUsers(follower\_samp)[[1]]$screenName)

}

}

}

This gets me a sample that looks like the below. Note that this method is prone to sampling individuals more than once, particularly highly networked users (ie lots of followers and/or lots of people following). I’ll deal with that later by taking the average of the estimates of them.

> follow\_data %>%

+ arrange(desc(number\_followers))

# A tibble: 983 x 6

number\_followers number\_following mean\_ff trmean\_ff median\_ff screenName

<dbl> <dbl> <dbl> <dbl> <dbl> <chr>

1 108538969 203 6002702 1227629 450295 katyperry

2 71131344 1034 3113456 373193 140602 YouTube

3 44212387 183 3208975 594062 255161 BillGates

4 40436829 1868 810532 38519 17959 narendramodi

5 40410386 1867 810495 38519 17926 narendramodi

6 40409932 1867 810489 38519 17926 narendramodi

7 40400380 1868 809897 38479 17924 narendramodi

8 35470473 1661 1559447 334577 219023 SportsCenter

9 24954373 424 1182455 232537 96970 PMOIndia

10 21680806 771 900960 6736 2814 HillaryClinton

# ... with 973 more rows

The snapshot sampling method is a bit simpler. All I need is 1,000 random tweets, which I get by searching for tweets containing the letter “e” (as twitteR doesn’t facilitate a completely open search as far as I can see). This isn’t great – I think it eliminates people using some character sets – but is good enough for a blog.

#==================snapshot sampling method===========

# The first method of sampling will oversample people with lots of followers and people

# with lots of friends. As this is potentially linked to the response variable (indirectly)

# it might be better to try an alternative sampling method, by grabbing a bunch of tweets.

# This method will oversample people who are active tweeting today, which is a different

# sort of coverage problem, so still need to be careful.

follow\_data\_2 <- data\_frame(

number\_followers = integer(),

number\_following = integer(),

mean\_ff = numeric(),

trmean\_ff = numeric(),

median\_ff = numeric(),

screenName = character()

)

# 1000 random tweets with the letter "e" in them:

tweets <- searchTwitter("e", n = 1000)

# the users

users <- unique(sapply(tweets, function(x){x$screenName}))

for(i in 1:length(users)){

# save latest copy of results in case of crash

save(follow\_data\_2, file = "follow\_data\_2.rda")

# number of calls left to make under Twitter's limits

lims <- getCurRateLimitInfo()

x <- as.numeric(filter(lims, resource == "/followers/ids")$remaining) \*

as.numeric(filter(lims, resource == "/friends/ids")$remaining)

while(x == 0) {

lims <- getCurRateLimitInfo()

x <- as.numeric(filter(lims, resource == "/followers/ids")$remaining) \*

as.numeric(filter(lims, resource == "/friends/ids")$remaining)

message("Waiting")

Sys.sleep(60) # wait 15 minutes (or 1 minute, 15 times)

}

current\_sn <- users[i]

cat(paste(i, current\_sn, " | "))

try({

current\_user <- getUser(current\_sn) # this fails surprisingly often, not sure why

follow\_data\_2[i, "screenName"] <- current\_sn

# who this user follows (what happens if this is eg 1million people? I think it gets truncated at 15,000,

# which is not as bad as it sounds as we only need a sample; but the problem will be that this is their

# 15,000 most recent people they have followed, who might be biased to be newcomers?):

# certain privacy settings stop you seeing someone's friends and followers, in which case

# we try to sample another node from the last good person

all\_my\_friends <- character()

try(all\_my\_friends <- current\_user$getFriendIDs()) # about 1 second for c.1500 IDs, longer for more

if(length(all\_my\_friends) > 0){

# sample some of those friends

n <- min(2000, length(all\_my\_friends))

sample\_my\_friends <- sample(all\_my\_friends, n, replace = FALSE)

friend\_details <- lookupUsers(sample\_my\_friends) # about 15 seconds

friend\_details\_df <- twListToDF(friend\_details)

follow\_data\_2[i, "number\_followers"] <- current\_user$followersCount

follow\_data\_2[i, "number\_following"] <- current\_user$friendsCount

# mean number of followers of the people this user follows

follow\_data\_2[i, "mean\_ff"] <- mean(friend\_details\_df$followersCount)

follow\_data\_2[i, "trmean\_ff"] <- mean(friend\_details\_df$followersCount, tr = 0.2)

follow\_data\_2[i, "median\_ff"] <- median(friend\_details\_df$followersCount)

}

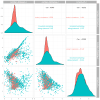
})

}

**Results**

The resulting numbers are all very skewed distributions. Visually, they look good when you take the logarithm of the original number plus 1 (needed to avoid turning people with 0 friends or 0 followers becoming -Inf). I can justify “+ 1” by saying that, in a way, everyone follows themself and is followed by themselves.

Here’s the distributions and relationships of the key variables, when transformed this way:



We see a strong positive relationship between number of your followers and number of people you are following – until we get to people with many followers (10,000 or more), when the relationship breaks down (visible only in the snowball sample, as the simpler “today’s tweeters” sampling method harvests few such people). Partly this comes from reciprocal follow-back arrangements, partly it is a general indicator of longevity.

We also see a strongly negative relationship between number of people one is following and the average number of their followers. This makes sense and fits in with the notion that if you want to be following people with lots of followers, you have to be quite selective in who you follow, and its best to follow big names in sport, entertainment and perhaps politics. Consider Trump supporter [@TheRoyalPosts](https://twitter.com/TheRoyalPosts) as an extreme example (not actually included in the final sample). With 41,700 followers of her own, she follows only 120 prominent political accounts, with an astonishing average number of followers themselves of 7 million.

At this point let’s have another look at the graphic I started the post with:

From these last couple of charts, I’m actually pretty happy with both of my sampling methods.

Here’s the code that combines the two samples and produces the graphics above.

#==================analysis========================

follow\_data\_proc <- follow\_data %>%

mutate(sampling\_method = "Snowball sampling\nalong network")

follow\_data\_proc\_2 <- follow\_data\_2 %>%

mutate(sampling\_method = "Sample of today's tweeters")

follow\_data\_comb <- rbind(follow\_data\_proc, follow\_data\_proc\_2) %>%

filter(!is.na(trmean\_ff)) %>%

group\_by(screenName, sampling\_method) %>%

summarise(

number\_followers = mean(number\_followers),

number\_following = mean(number\_following),

mean\_ff = mean(mean\_ff),

trmean\_ff = mean(trmean\_ff),

median\_ff = mean(median\_ff),

n\_obs = n()

) %>%

ungroup() %>%

mutate(many\_following = cut(number\_following, breaks = c(0, 1000, 5000, 15000, 75000, 10 ^ 9),

labels = c("0 - 1,000", "1,001 - 5,000", "5,001 - 15,000", "15,001 - 75,000", "75,001 or more")))

follow\_data\_trans <- follow\_data\_comb %>%

mutate(number\_followers\_l = log10(number\_followers + 1),

number\_following\_l = log10(number\_following + 1),

trmean\_ff\_l = log10(trmean\_ff + 1)) %>%

select(number\_followers\_l, number\_following\_l, trmean\_ff\_l, sampling\_method)

ggpairs(follow\_data\_trans, columns = 1:3, mapping = aes(colour = sampling\_method))

ggplot(follow\_data\_comb, aes(x = number\_followers, y = trmean\_ff)) +

facet\_wrap(~sampling\_method) +

geom\_point(aes(fill = many\_following), pch = 21) +

geom\_smooth(method = "loess") +

scale\_x\_log10("Number of followers, of this person", label = comma) +

scale\_y\_log10("Trimmed mean number of followers,\nof the people this person follows", label = comma) +

ggtitle("Do people with more followers follow people with more followers?",

"Apparently 'it depends'.") +

scale\_fill\_brewer("Following how\nmany people:", palette = "Spectral",

guide = guide\_legend(title.hjust = 0.5, override.aes = list(size = 5))) +

theme(legend.position = "right")

**Modelling**

Finally, I wanted to see if the apparent u-shaped relationship between number of followers and the average number of followers of people one follows was robust to a model that simultaneously modelled the strongly negative relationship between the number of people one follows and that same response variable. It turns out that this is the case. Here are the partial effects of the two variables in a generalized additive model without an interaction term:

And here is how the relationship looks when there is an interaction term. First, as a three-dimensional perspective plot:

… then, more usefully as a heatmap.

The interaction is significant so we’d keep it in even though it complicates the interpretation.

I think the heatmap is the best representation of the data. It shows clearly that the average number of followers of the people one follows is:

* high for people with few followers
* high for people with many followers who *don’t* follow many themselves
* low for people with many followers who *do* follow lots of people themselves

Additionally, there are are very few or no users who follow hundreds of thousands of accounts but have a low number of their own followers (hence the white space in the top left corner).

Here’s the code for the modelling:

mod\_full <- gam(trmean\_ff\_l ~ s(number\_followers\_l, number\_following\_l), data = follow\_data\_trans)

mod\_simp <- gam(trmean\_ff\_l ~ s(number\_followers\_l) + s(number\_following\_l), data = follow\_data\_trans)

anova(mod\_full, mod\_simp, test = "F")

# simple two variable model

par(bty = "l", font.main = 1)

plot(mod\_simp, pages = 1, main = "No-interaction model")

# perspective plot

plot(mod\_full, pages = 1, scheme = 1, main = ("Impact on the trimmed mean number\nof followers of people that are followed"))

# heat map

plot(mod\_full, pages = 1, scheme = 2, hcolors = topo.colors(50),

main = ("Impact on the trimmed mean number\nof followers of people that are followed"),

xlab = "Logarithm of number of followers",

ylab = "Logarithm of number people following")

legend("topleft", legend = c("low", "medium", "high"), fill = topo.colors(3), bty = "n", cex = 1.2)