R snowballing

Wouter van Atteveldt

February 8, 2017

Setup and authenticate

```
consumer_key = 'QvT64lks9xonDBeg6i7MSLlBv'
consumer_secret = 'ARwK8gyx79gk1PAiRO3addDaZQqXwXYdmWeL1Ry9CcCQ61SZto'
access_token = '155724548-sDvC64Nmu8I3EnBjI87gbckhS06vHfawmK5dMtVt'
access_secret = '...' #  secret!
```

```
library(twitteR)
twitteR::setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)
```

```
## [1] "Using direct authentication"
```

Create the follower graph

We start from a single user. You can use of course use any other user (yourself, some celebrity), I will just call this user 'me' in this demo for convenience.

```
u = twitteR::getUser("vanatteveldt")
followers = u$getFollowers()
followers.df = plyr::ldply(u$getFollowers(), as.data.frame)
knitr::kable(head(followers.df))
```

.id	description
4781723995	The official Twitter account for the Communication Science & Biology Interest Group of the @icahd
366561788	To be fabulous and classy. Dancing through piano sheets, CS scripts and political philosophy
229976596	City University of Hong Kong
2810168020	Data enthusiast. Research on how to support journalists in finding hidden stories within large docun
711064162412175361	
932421709	Political communication researcher at Mannheim Centre for European Social Research | Past Editor

Now, we want to get the followers of his followers. Since this is rate limited to 15 calls per 15 minutes, we need to be selective in who we sample.

Note that in an actual research project, we would wait automatically when we run out of calls so we can query our full (theoretically motivated) sample.

For this demo, let's select the most popular followers who I also follow back.

```
friends = u$getFriendIDs()
followers.df = subset(followers.df, id %in% friends)
followers.df = plyr::arrange(followers.df, -followersCount)
ids = head(followers.df$id, 10)
```

Now, to find out who follows these people, it's best to create a function that makes a dataframe of (leader, follower) for each user: Note that we use IDs rather than names to avoid looking up tens of thousands of twitter users.

```
get_followers = function(u) {
  message(u$screenName)
  f = as.numeric(u$getFollowerIDs())
  data.frame(leader=as.numeric(u$id), follower=f)
}
```

For example, let's look at the followers of our user u:

```
head(get_followers(u))
```

```
## vanatteveldt
```

```
##      leader           follower
## 1 155724548         4781723995
## 2 155724548          366561788
## 3 155724548          229976596
## 4 155724548         2810168020
## 5 155724548 711064162412175360
## 6 155724548          932421709
```

With this function, we can use 'ldply' to create a dataframe of the follower's followers. First, we need to select the user objects with our ids

```
users = c(u, followers[ids])
connections = plyr::ldply(users, get_followers, .id=NULL)
```

leader	follower
155724548	4781723995
155724548	366561788
155724548	229976596
155724548	2810168020
155724548	711064162412175361
155724548	932421709

These connections form a graph, but it is much too big to be useful, with (in my case) 22544 edges.

However, we can limit to people who I followed, and remove self-follows:

```
followed_by_me = connections$follower[connections$leader == u$id]
connections = subset(connections, follower %in% followed_by_me)
connections = subset(connections, follower != leader)
nrow(connections)
```

```
## [1] 222
```

That's better. Now, let's add the screen names:

```
userids = unique(c(connections$leader, connections$follower))
users = lookupUsers(userids)
users = plyr::ldply(users, as.data.frame)[c("id", "screenName")]
connections = merge(connections, users, by.x="leader", by.y="id")
connections = merge(connections, users, by.x="follower", by.y="id")
connections = connections[c("screenName.x", "screenName.y")]
head(connections)
```

```
##     screenName.x screenName.y
## 1   vanatteveldt kt_designbox
## 2      kenbenoit  philiphabel
## 3 GijsSchumacher  philiphabel
## 4     burtmonroe  philiphabel
## 5      p_barbera  philiphabel
## 6   vanatteveldt  philiphabel
```

```r
library(igraph)
g = igraph::graph_from_data_frame(connections, directed=F)
```

```r
plot(g)
```

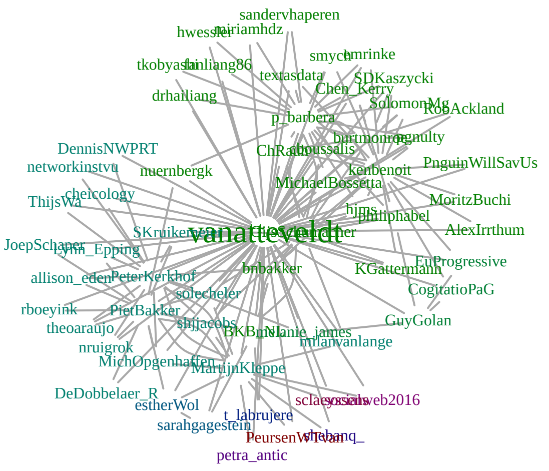
This works, but it is not a very pretty plot. To make it nicer, we can do a number of things:

```r
# keep only nodes with in-degree <= 1
indegree = degree(g, V(g), "in")
g2 =induced_subgraph(g, indegree>1)

# scale labels according to (bewteenness) centrality
centrality = betweenness(g2)
V(g2)$label.cex = 0.5 + 0.5 * centrality / max(centrality)

# color labels based on clustering
clusters = edge.betweenness.community(g2)$membership
pal = substr(rainbow(length(unique(clusters)), start=0.33, end=1, v=0.5), 1, 7)
V(g2)$label.color = pal[match(clusters, unique(clusters))]

# plot without arrows and without node shapes
plot(g2, vertex.shape = "none", edge.arrow.size=0)
```



Unsuprisingly, my name is most central in this graph, since that is the start of the snowballing (and we only sampled a very limited number of users). You can see a couple of other important users, such as Ken Benoit (LSE) and Peter Kerkhof (Amsterdam). By using betweenness centrality, the label size highlights which users are 'in between' most connections.

From the clustering, you can see two main clusters, which seem to be my colleagues and other contacts from the Netherlands (including Peter Kerkhof) and a set of international colleagues including Pablo Barbera and Ken Benoit.