Network Analysis Using R

Network Analysis Using R: Outline

- Introduction to the Digital Observatory
- The Australian Twittersphere
- What is a network?
  - Nodes, edges, attributes, layouts, directed/undirected
- Quantitative properties of networks
- Matrix representation of networks
- Prac
  - Research questions
  - Operationalisation and methods
  - Tasks
- References and resources

Digital Observatory

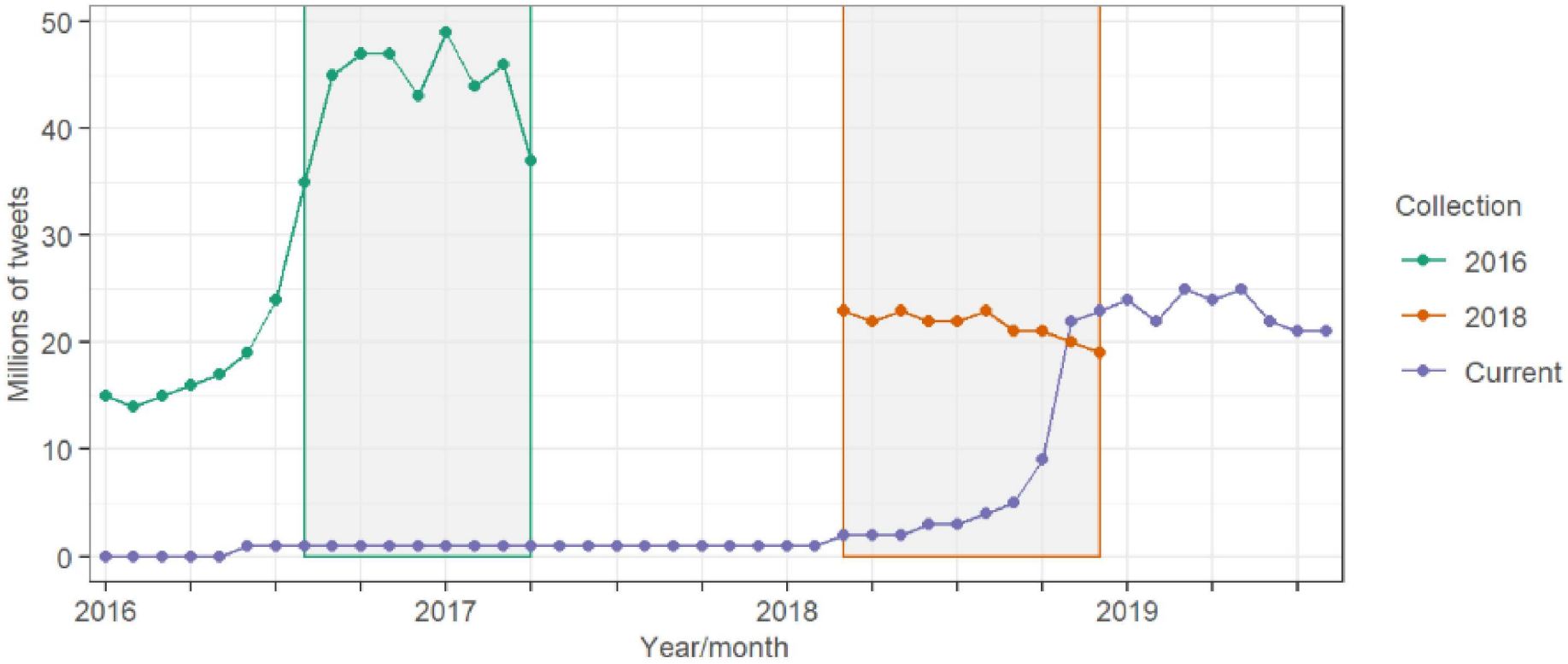
Mission: Enabling the understanding of the dynamic digital landscape by providing access to reliable and scalable research data infrastructure.

The Digital Observatory maintains a curated, longitudinal collection of “Australian” tweets known as the Australian Twittersphere.

- Marissa Takahashi
- Sam Hames
- Betsy Alpert
- Alice Miller

The Australian Twittersphere

Number of tweets per month for all three collections (2016 onwards):

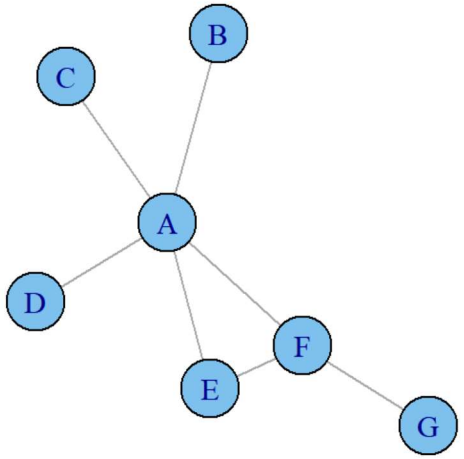


What is a network?

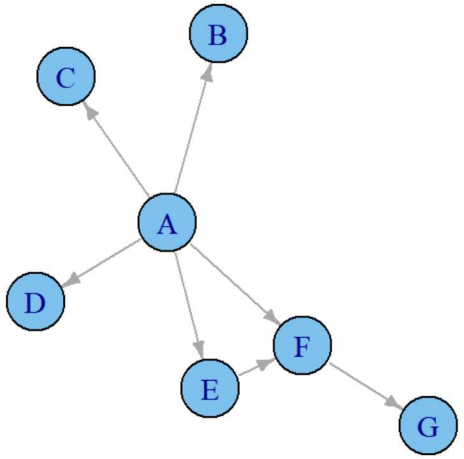
- A network is a set of points and lines connecting some pairs of the points (Voloshin, 2009).
- The points are known as **nodes** or vertices.
- The lines are known as **edges**.
- The positions of the nodes and edges are determined by a **layout**.
- Nodes and edges can have **attributes**.
- Networks can be **directed** or **undirected**.
- Networks have **quantitative properties** that are useful for descriptive analysis.
- Networks are sometimes referred to as **graphs**.

Let's take a look at some examples…

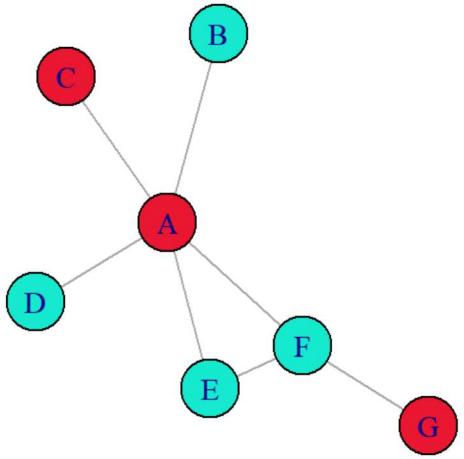
Undirected network



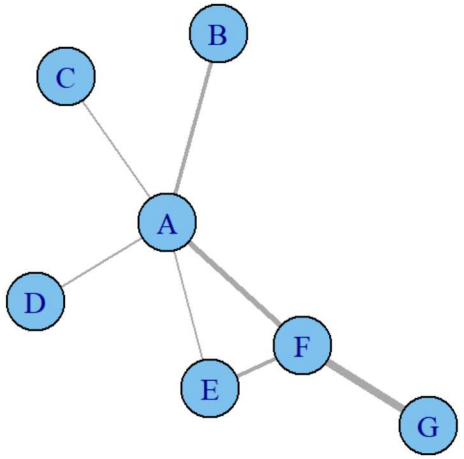
Directed network



Node attributes



Edge attributes



Layouts

- A layout specifies where nodes and edges are placed in space (usually two dimensional).
- Layouts affect the appearance and interpretation of network diagrams.
- Generating a layout often involves a random initialisation, meaning the same layout with the same data can appear differently with each generation.
- Common layouts include **circular** (left), **radial** (middle), and **tree** (right).

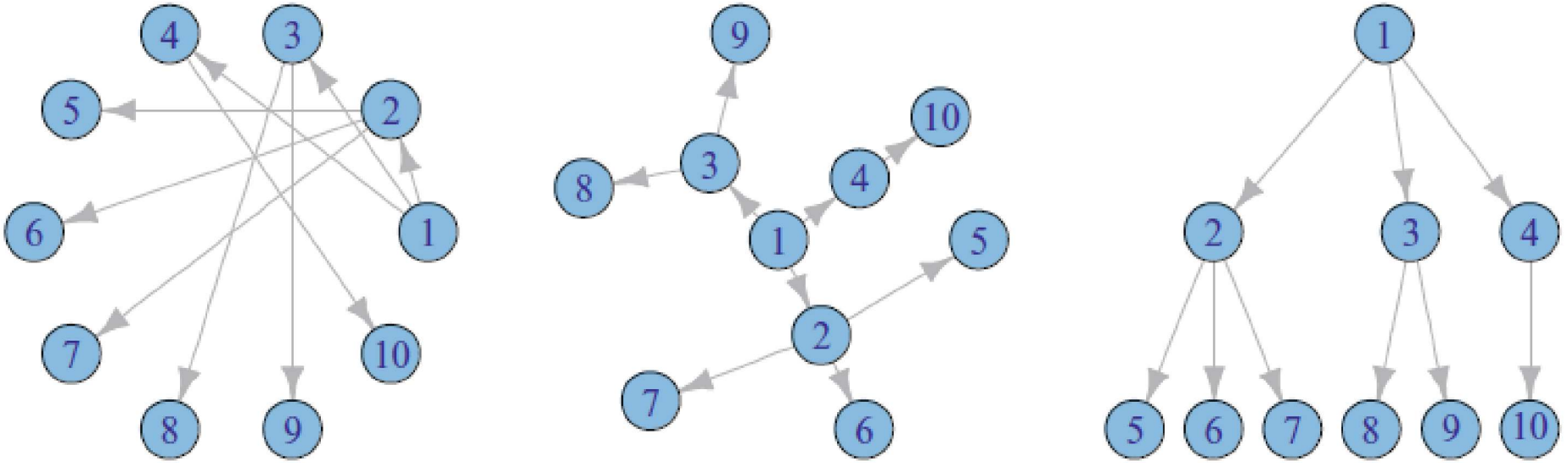


Image credit: Kolaczyk & Csárdi (2014)

Quantitative properties of networks

Common quantitative properties (there are many others) that are measures of centrality in a network:

- **Node degree**: Number of edges connected to the node. Directed networks have two node degrees: **out-degree** and **in-degree**.
- **Node closeness**: Calculates the “distance” between nodes using an algorithm. Nodes that are closer to other nodes are more central in the network.
- **Node betweenness**: Nodes that are part more shortest paths between pairs of other nodes are more central in the network.
- **Edge betweenness**: Number of shortest paths between pairs of nodes that include that edge.

Matrix representations of networks

Networks can be represented as matrices:

- **Adjacency matrix**: Matrix of 0's and 1's indicating which nodes are connected.
- **Edge list**: Two column matrix indicating which nodes are connected.

Comparison of adjacency matrix and edge List

Adjacency Matrix

	A	B	C	D	E	F	G
A	0	1	1	1	1	1	0
B	1	0	0	0	0	0	0
C	1	0	0	0	0	0	0
D	1	0	0	0	0	0	0
E	1	0	0	0	0	1	0
F	1	0	0	0	1	0	1
G	0	0	0	0	0	1	0

Edge List

V1	V2
A	B
A	C
A	D
A	E
A	F
E	F
F	G

Edge list (with and without attribute)

Edge List

V1	V2
A	B
A	C
A	D
A	E
A	F
E	F
F	G

… with Attribute

V1	V2	Frequency
A	B	2
A	C	1
A	D	1
A	E	1
A	F	3
E	F	2
F	G	4

Prac: Research questions

- Who is interacting with who on Twitter throughout the 2019 AFL Grand Final?
- Who is the most central Twitter account in the conversation (of replies) about the 2019 AFL Grand Final?

Prac: Operationalisation

- Original tweets from the Australian Twittersphere that were:
  - Tweeted in the 2.5 hour period corresponding to 28 September 2019 (AFL Grand Final); and
  - Contained the hashtag #AFLGrandFinal (including all upper and lower case variants); and
  - Received replies that were tweeted up to 2 hours after the match finished.
- Use the igraph package in R to create a reply network:
  - Nodes represent Twitter accounts.
  - Directed edges represent replies between Twitter accounts.
- Node degree will be used to establish the most central node.

Prac: Tasks

- Install R, R Studio, and the igraph package (if not done already)
- Import data from Australian Twittersphere
- Tidy data
- Create igraph object
- Create and experiment with a network diagram
- Extract node degree from the igraph object

Prac: Poll

- How many people have used R before?
- How many people have coding experience?
- How many people have tried network visualisation and/or analysis before?

Install R

- First, make sure you have local admin rights to your machine. If you're not sure how to do this, these instructions will help.
- Go to CRAN (comprehensive R archive network) to view the latest versions of R for each operating system.
- Select the appropriate version for your operating system (Windows, Mac, or Linux).
- Install R to the default location, usually:

```
C:\Program Files\R
```

Install R Studio

- Go to RStudio and select the free version of RStudio Desktop.
- Select the appropriate version for your operating system.
- Install R Studio to the default location, usually:

```
C:\Program Files\RStudio
```

Setup R Studio

- Download these two files from `bit.ly/network121219`
  - `query1_results_anon.csv`
  - `network.R`
- Save them in a folder called `network`
- Open R Studio
- File > New Project > Existing Directory > Browse
- Navigate to the `network` folder and click it once > Open > Create Project
- Click once on the `network.R` file in the files pane (bottom right)

Install and load libraries

Run the following code to install dplyr and igraph:

```
install.packages("dplyr")
install.packages("igraph")
```

Then run the following to load dplyr and igraph:

```
library(dplyr)
library(igraph)
```

Import data (1/4)

Extracting data from the Australian Twittersphere database using SQL:

```
query1_results <- tbl(con, sql ("select `timehashtagreply`.tweet_id as `original_tweet_tweet_id`,
                                  `timehashtagreply`.user_id as `original_tweet_user_id`,
                                  `timehashtagreply`.created_at as `original_tweet_created_at`,
                                  `timehashtagreply`.hashtag_id as `original_tweet_hashtag_id`,
                                  `timehashtagreply`.hashtag as `original_tweet_hashtag`,
                                   tweet.tweet_id as `reply_tweet_id`,
                                   tweet.user_id as `reply_user_id`,
                                   tweet.created_at as `reply_created_at`
                                 from oz_twittersphere.tweet
                                 right join (
                                              select `timehashtag`.tweet_id, `timehashtag`.user_id,
                                               `timehashtag`.created_at, `timehashtag`.hashtag_id, hashtag.hashtag
                                              from oz_twittersphere.hashtag
                                              inner join (
                                                           select `time`.tweet_id, `time`.user_id, `time`.created_at,
                                                            tweet_hashtag.hashtag_id
                                                           from oz_twittersphere.tweet_hashtag
                                                           inner join (
                                                                        select tweet_id, user_id, created_at
                                                                        from oz_twittersphere.tweet
                                                                        where created_at between '2019-09-28 04:30' and '2019-09-28 06:59'
                                                                      ) `time`
                                                           on `time`.tweet_id = tweet_hashtag.tweet_id
                                                         ) `timehashtag`
                                              on `timehashtag`.hashtag_id = hashtag.hashtag_id
                                              where lower(hashtag.hashtag) = 'aflgrandfinal'
                                            ) `timehashtagreply`
                                 on `timehashtagreply`.tweet_id = tweet.in_reply_to_tweet_id
                                 where tweet.created_at between '2019-09-28 04:30' and '2019-09-28 08:59'
                                 order by `original_tweet_tweet_id`, `reply_tweet_id`"
                               )
                    )
```

Import data (2/4)

Let’s have a look at the data that we will be working with:

	A	B	C	D	E	F	G	H
1	original_tweet_tweet_id	original_tweet_user_id	original_tweet_created_at	original_tweet_hashtag_id	original_tweet_hashtag	reply_tweet_id	reply_user_id	reply_created_at
2	1177802801310610000	783879853	28/09/2019 4:30	4885	AFLGrandFinal	1177803174968570000	95091601	28/09/2019 4:32
3	1177802801310610000	783879853	28/09/2019 4:30	4885	AFLGrandFinal	1177828555448730000	2493534578	28/09/2019 6:13
4	1177802837419380000	17331252	28/09/2019 4:30	4885	AFLGrandFinal	1177805771678960000	76555750	28/09/2019 4:42
5	1177803110397310000	160528143	28/09/2019 4:31	4885	AFLGrandFinal	1177803527088790000	207031278	28/09/2019 4:33
6	1177803130102110000	2928108217	28/09/2019 4:32	4885	AFLGrandFinal	1177806094644590000	454696257	28/09/2019 4:43
7	1177804300401000000	121953691	28/09/2019 4:36	4885	AFLGrandFinal	1177804497826920000	408106457	28/09/2019 4:37
8	1177805280681340000	141648982	28/09/2019 4:40	4885	AFLGrandFinal	1177805554875390000	34843277	28/09/2019 4:41
9	1177806019860180000	121953691	28/09/2019 4:43	4885	AFLGrandFinal	1177806333715730000	47844309	28/09/2019 4:44
10	1177806122536680000	2647441508	28/09/2019 4:43	4885	AFLGrandFinal	1177806484677120000	3972993374	28/09/2019 4:45

Import data (3/4)

Anonymised, minimum required data (`network.csv`):

	A	B	C	D
1	original_tweet_tweet_id	original_tweet_user_id	reply_tweet_id	reply_user_id
2	t001	u106	t005	u052
3	t001	u106	t115	u121
4	t002	u007	t011	u046
5	t003	u061	t006	u068
6	t004	u124	t013	u095
7	t007	u057	t008	u091
8	t009	u059	t010	u028
9	t012	u057	t015	u034
10	t014	u122	t016	u130

Import data (4/4)

Let’s import this .csv data into R Studio:

```
replies <- read.csv("query1_results_anon.csv",
                    header = TRUE,
                    fileEncoding = "UTF-8-BOM")
```

View the data to make sure it looks OK:

```
View(replies)
```

Create an edge list

Take the data that was imported and create counts of the number of replies between users:

```
reply_edges <- replies %>%
  group_by(original_tweet_user_id, reply_user_id) %>%
  summarise(nbr_replies = n())
```

View the results (number of replies between users) to make sure it looks OK. This will be used as input to create the igraph object.

```
View(reply_edges)
```

Create igraph object

Create igraph object using an edge list (reply_edges):

```
reply_g <- graph_from_data_frame(d = reply_edges, directed = TRUE)
```

Inspect igraph properties

```
reply_g
```

Output:

- First number is the number of nodes
- Second number is the number of edges

Plot the network diagram

Default plot settings:

```
plot(reply_g)
```

Doesn't look good! Let's try removing the node (vertex) labels:

```
plot(reply_g, vertex.label = NA)
```

Better, but the nodes are obscuring the arrows.

Reflexive nodes

Isolate the reflexive nodes in the edge list:

```
(reply_edges %>%
    filter(as.character(original_tweet_user_id)
        == as.character(reply_user_id)))
```

If we have time, we will remove these later.

Adjust nodes and arrows

Adjust the size and colour of the nodes:

```
plot(reply_g, vertex.label = NA,
     vertex.color = "SkyBlue2", vertex.size = 7)
```

Now we can see the arrows, but they need to be a little smaller. Let's adjust the arrow size:

```
plot(reply_g, vertex.label = NA,
     vertex.color = "SkyBlue2", vertex.size = 7,
     edge.arrow.size = 0.2)
```

Adjust edge thickness

Create a vector of edge weights based on number of replies:

```
weight1 <- E(reply_g)$nbr_replies
```

View the results:

```
View(weight1)
```

Plot the graph with the edge thickness determined by weights:

```
plot(reply_g, vertex.label = NA,
     vertex.color = "SkyBlue2", vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1)
```

Layouts (1/3)

Kamada and Kawai (a type of radial layout):

```
plot(reply_g, vertex.label = NA,
     vertex.color = "SkyBlue2", vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1,
     layout = layout.kamada.kawai)
```

Layouts (2/3)

Circle:

```
plot(reply_g, vertex.label = NA,
     vertex.color = "SkyBlue2", vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1,
     layout = layout.circle)
```

Layouts (3/3)

Tree:

```
plot(reply_g, vertex.label = NA,
     vertex.color = "SkyBlue2", vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1,
     layout = layout.reingold.tilford)
```

Layout nicely function

Use the layout_nicely() function to choose the most appropriate layout function (hopefully!) for a given graph object. Added bonus of providing consistent plotting of your network diagram.

First, save the results of the layout_nicely() function:

```
nice_layout <- layout_nicely(reply_g)
```

Plot the network diagram specifying your saved layout:

```
plot(reply_g, vertex.label = NA,
     vertex.color = V(reply_g)$color, vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1,
     layout = nice_layout)
```

Node degree (1/2)

Extract node degree (total) from the igraph object and save it:

```
reply_g_degree <- degree(reply_g)
```

Turn it into an easy to read data frame and sort it in descending order by degree:

```
reply_g_degree <- data.frame(user_id = names(reply_g_degree),
                             degree = reply_g_degree,
                             row.names = NULL) %>%
  arrange(desc(degree))
```

Node degree (2/2)

View the results:

```
View(reply_g_degree)
```

This shows us that the node with the highest degree (total) is “u085”.

Out degree (1/2)

Extract out degree from the igraph object and save it:

```
reply_g_degree_out <- degree(reply_g, mode = "out")
```

Turn it into an easy to read data frame and sort it in descending order by degree:

```
reply_g_degree_out <- data.frame(user_id = names(reply_g_degree_out),
                                 out_degree = reply_g_degree_out,
                                 row.names = NULL) %>%
  arrange(desc(out_degree))
```

Out degree (2/2)

View the results:

```
View(reply_g_degree_out)
```

In degree (1/2)

Extract in degree from the igraph object and save it:

```
reply_g_degree_in <- degree(reply_g, mode = "in")
```

Turn it into an easy to read data frame and sort it in descending order by degree:

```
reply_g_degree_in <- data.frame(user_id = names(reply_g_degree_in),
                                in_degree = reply_g_degree_in,
                                row.names = NULL) %>%
  arrange(desc(in_degree))
```

In degree (2/2)

View the results:

```
View(reply_g_degree_in)
```

Conditional colouring of nodes

Colour code nodes based on a condition (highlight user “u085”, the most central node according to node degree):

```
V(reply_g)$color <- ifelse(
  V(reply_g)$name == "u085", "red", "SkyBlue2"
)
```

Replot the graph to show the results:

```
plot(reply_g, vertex.label = NA,
     vertex.color = V(reply_g)$color, vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1)
```

Filtering nodes and edges (1/2)

Let's return to our earlier network diagram:

```
plot(reply_g, vertex.label = NA,
     vertex.color = V(reply_g)$color, vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1,
     layout = nice_layout)
```

Recall that these are the accounts that have replied to themselves (reflexive nodes):

```
(reply_edges %>%
    filter(as.character(original_tweet_user_id)
        == as.character(reply_user_id)))
```

Filtering nodes and edges (2/2)

We don't want to remove the node completely, because they may have replied to another account and we still want to keep that in our network diagram.

Turns out there is a very simple way to do this in igraph - use the simplify function:

```
reply_g2 <- simplify(reply_g, remove.loops = TRUE)
```

Let's take a look:

```
plot(reply_g2, vertex.label = NA,
     vertex.color = V(reply_g)$color, vertex.size = 7,
     edge.arrow.size = 0.2, edge.width = weight1,
     layout = nice_layout)
```

Recap

- Network theory
- Matrix representation of networks
- Installed R and R Studio
- Imported data
- Created an edge list
- Created an igraph object
- Plotted and manipulated a network diagram
- Extracted node degree

References and resources

Hacky Hour at QUT: Thursdays, 2pm, The Pantry, @GPHackyHour

Hacky Hour at UQ: Tuesdays, 3pm, Nano's Cafe, @HackyHourStLuc

Csárdi, G., & Nepusz, T. (2006). The igraph software package for complex network research. InterJournal, Complex Systems 1695 http://igraph.org

Kolaczyk, E., & Csárdi, G. (2014). *Statistical Analysis of Network Data with R (1st ed. 2014.).* https://doi.org/10.1007/978-1-4939-0983-4

Voloshin, V. I. (2009). *Introduction to graph theory.* Retrieved from https://ebookcentral.proquest.com

Wickham, H., & Grolemund, G. (2016). *R for Data Science (1st edition).* O'Reilly Media, Inc.

Contact information

Digital Observatory
Institute for Future Environments | QUT
digitalobservatory@qut.edu.au
@ObservatoryTeam
Ph: +61 7 3138 6849

QUT ife
Institute for Future Environments
Digital Observatory