Table of Contents

```
In [ ]: library(igraph)
```

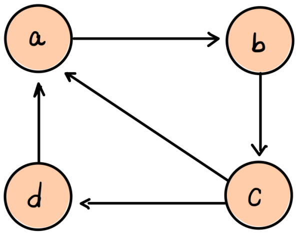
1 Network Metrics

Alpha centrality

If you are curious about this metric, you can find some notes here (https://nbviewer.jupyter.org/github/khof312/msba_network_analytics/blob/master/alpha_centrality.ipynb).

Hint: How to extract node characteristics from a sample graph

For a list of different characteristics we can extract, see the documentation (http://igraph.org/r/doc/).
You will save time by computing alpha centrality, PageRank, and in-degree using iGraph, rather than doing the calculations manually yourself.



In [2]:
```
# Step 1: Let's load a sample graph (pictured above)
uv <- matrix( c('a','b','b','c','c','a','c','d','d','a'), nc = 2, byrow = TRUE
)
g <- graph_from_edgelist(uv, directed=TRUE)

### IMPORTANT ###
# See note below <- if reading from a file, as in the assignment,
# it's easiest to use read.graph rather than graph_from_edgelist.
# I just used this method to make some fake data for the example.
#################


g
```

```
IGRAPH 5939469 DN-- 4 5 --
+ attr: name (v/c)
+ edges from 5939469 (vertex names):
[1] a->b b->c c->a c->d d->a
```

Note: In the cell above, I'm just creating some fake data to practice with. You will instead want to read your data from the files provided. Some notes:

- The files contain rows of data in the form " `a b` ", which implies that `a` follows `b` :
$$a \longrightarrow b$$
- You can re-use your code from the first homework, using the `read.graph` function with `ncol` format to import these files directly in graph format. Don't forget to specify an argument for `directed` .
- If you do use this approach, note that the nodes in the graph have both an R-assigned index (e.g. 1), and a name (e.g. "0"). As I've done below, you will want to call the node name in quotes in order to retrieve the correct node.

Check your work:

- The `twitter_graph_subset` file should have 1042 nodes and 734 edges.
- The `twitter_graph_complete` file should have 59973 nodes and 73277 edges.

```
In [3]: # Step 2: We can calculate a metric, e.g. closeness, over the whole graph
        closeness(g)
```

```
a   0.166666666666667
b   0.2
c   0.25
d   0.166666666666667
```

```
In [4]: # Step 3: We can use indexing to retrieve the value for a set of nodes node, e.g. a and c
        closeness(g)[c('a','c')]
```

```
a   0.166666666666667
c   0.25
```

2 Power Law Distributions

If you are curious for more details on the shape of power law distributions and why we take the logs, you can find some additional notes here (https://nbviewer.jupyter.org/github/khof312/msba_network_analytics/blob/master/power_law.ipynb).

Hint:

- You may want to use the `factor` function to count the number of times that each possible value of `ntweets` appears.
- It's fine if you want to use a scatter plot here.

3 Influential Nodes

Hint: How to aggregate over a dataframe

You may find it useful to use the `aggregate` function to get the average number of retweets per user.

In [5]:

```
# Step 1: For a simple example, let's use a subset of the mtcars sample data
data(mtcars)
test_df <- mtcars[0:5, c("cyl", 'wt')]
test_df
```

	cyl	wt
Mazda RX4	6	2.620
Mazda RX4 Wag	6	2.875
Datsun 710	4	2.320
Hornet 4 Drive	6	3.215
Hornet Sportabout	8	3.440

In [6]:

```
# Step 2: Let's get the maximum weight by cylinder count
maxwt<-aggregate(wt~cyl, test_df, max)

# Step 3: Sort the dataframe
maxwt[order(-maxwt$wt), ]
```

	cyl	wt
3	8	3.440
2	6	3.215
1	4	2.320

4 Distinguishing Homophily from Influence

Are users more likely to tweet with a hashtag, if their friends have already tweeted with that hashtag? This problem attempts to answer this question by studying the spread of four hashtags in a network of Twitter users. In other words, we want to estimate the effect of the treatment on the outcome of interest, where:

- **Treatment** = friends tweeted with a given hashtag before time t_h^* and user has tweeted since t_h^*
- **Outcome of interest** = adoption, defined as whether or not someone tweeted with the hashtag.

a) Conceptual overview

The adopter files (`worldcup.csv` , `love.csv` , `selfie.csv` , `tbt.csv`)

Essentially, we can consider every adopter who tweeted before time t_h^* to have "treated" all of his/her followers.

> **Example:**
> We see that 309 users adopted the hashtag `#worldcup` . We can get the median `timeStamp` from the dataframe of `#worldcup` adopters, and see that $t^*_{worldcup} = 154$. The timestamp is just a sequential ordering of users starting at 0, so we can confirm that 154 users adopted before $t^*_{worldcup}$ and had the potential to treat others.

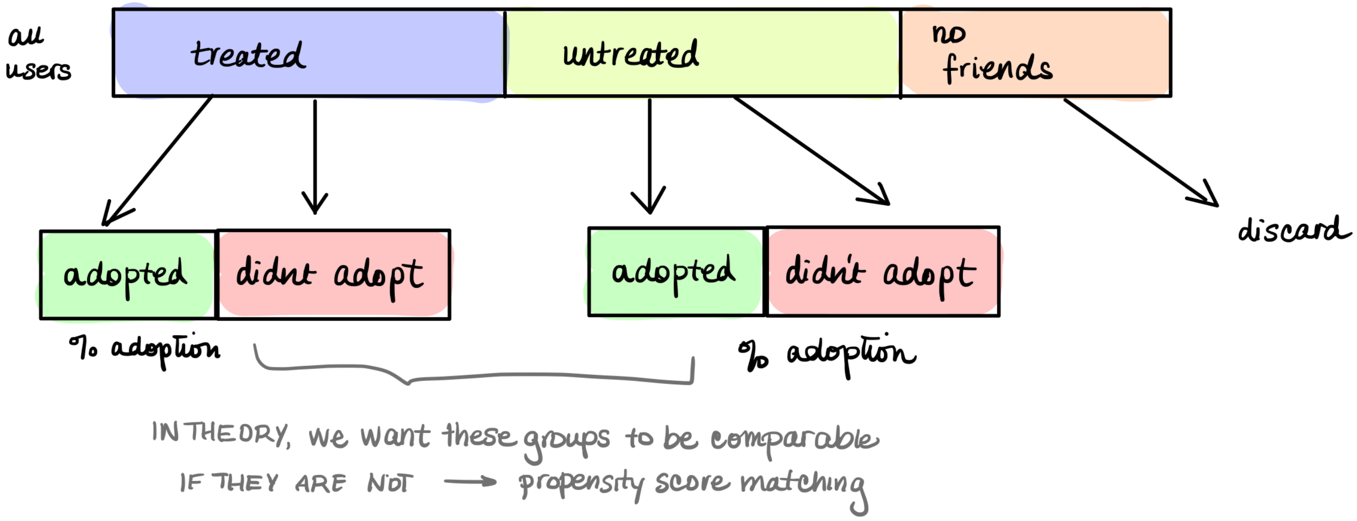
The file of all users (`all_users.csv`)

Note: When computing followers, `twitter_graph_complete.csv` is a file of links of the form "a b", where "a" follows "b". Therefore:

- If you see "a" multiple times, it means "a" follows more than one person.
- Treatment of "a" occurs when "b" tweets the hashtag, and adoption occurs when "a" tweets the hashtag (in addition to the other details described in the assignment).

The people in `all_users.csv` fall into three categories:

1. Those who were treated
2. Those who were untreated
3. Those who don't follow anyone and can't possibly have been treated ($\rightarrow$ discard)



> **Example:** After dropping users with no friends and calculating adoption and treatment status for everyone in `all_users.csv` , we wind up with the following distributions for `#worldcup` :

	didn't adopt	adopted	
untreated	200	30	= 13%
treated	214	61	= 22%

WHY DON'T MY VALUES MATCH THE TABLE?
Your values in this table will depend on how you define adoption. In the sample instructions, we have defined adopters as *anyone who tweets the hashtag, before or after time t*. If you prefer to define adopters as only those tweeting *after* time t, then your adoption counts will be lower. If that still doesn't work, you might also want to check that you discarded everyone who has no friends / follows no accounts.

The propensity score workflow

Ideally, treated and untreated users would be similar. Then, we could just compare the adoption rate across both groups. However, this is unlikely...

Due to homophily, the treatment group might suffer from selection bias. Specifically, this group of people might be more likely to adopt already, since they follow (and are likely similar to) the early adopters. In this case, they would be *both* more likely to be treated, *and* more likely to adopt.

We need to take extra steps so that we compare the *treated* users to the "right" *untreated* users who are just like them. We will do so with propensity score matching. A sample solution could adopt the following workflow:

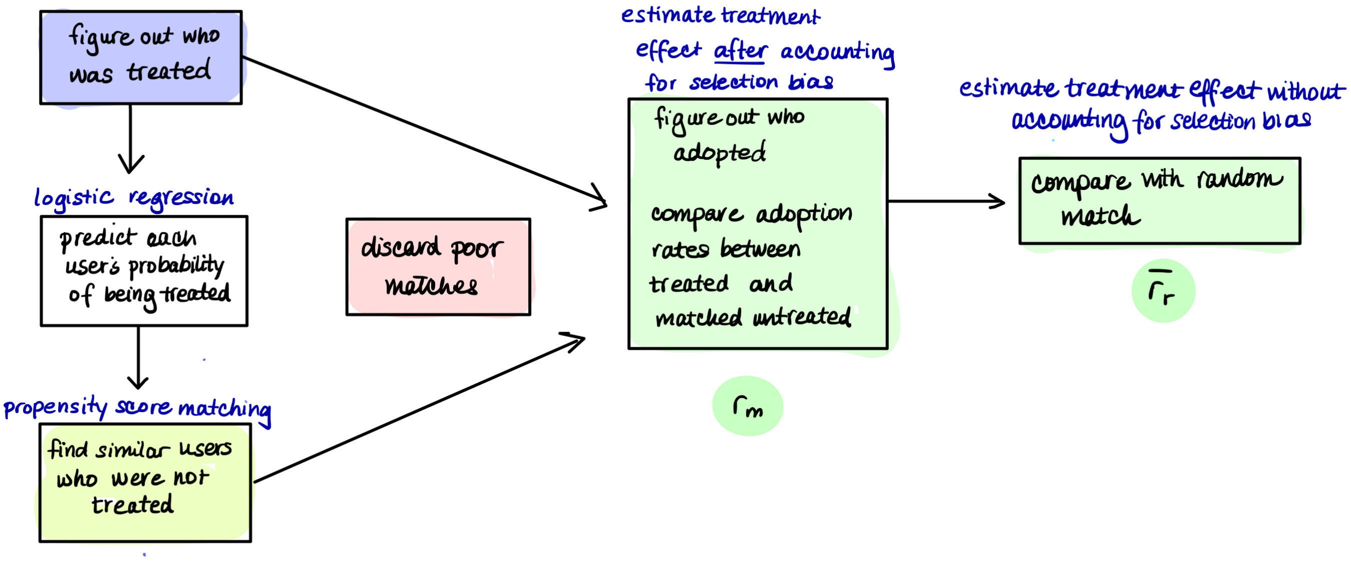


figure out who was treated
logistic regression
predict each user's probability of being treated
propensity score matching
find similar users who were not treated
discard poor matches
estimate treatment effect after accounting for selection bias
figure out who adopted
compare adoption rates between treated and matched untreated
r_m
estimate treatment effect without accounting for selection bias
compare with random match
$\bar{r}_r$

DO I RUN THIS PROCEDURE ONCE, OR ONCE *PER HASHTAG*?

Since there are four hashtags, you will want to repeat this process of computing $r+, r-$ four times, once per hashtag.

HOW DO I DEFINE MATCHED OBSERVATIONS?

- You will want to match each treated person to the untreated person that has the closest propensity score. This will be the person that is most similar with respect to the likelihood of being treated.
- You only need to match using the numeric propensity score that you generated with your logistic regression, since you've incorporated information about `location` , `followers` , `friends` , and `ntweets` when you calculated the score.

WHEN DOES A MATCH NOT EXIST? DO WE HAVE TO ESTABLISH A CUT OFF TO DEFINE WHO MATCHES?

Yes, you will want to make a cutoff to discard pairs whose propensity scores are too different. Following Aral, Muchnik, and Sundararajan (http://www.pnas.org/content/pnas/106/51/21544.full.pdf), the recommended threshold is to discard pairs where the observations are more than 2 standard deviations apart. We can define the standard deviation over all pairs of treated and untreated observations.

CAN I USE `MATCHIT` ?

In theory, yes, but I can't figure out exactly what the package means when it says it will discard units "outside of common support" (e.g. I don't know how it decides what the range of support will be). Instead, I would recommend:

1. Use `matchit` to get the best match for each treated unit
2. Calculate the absolute difference in propensity score between each treated unit and its untreated match.
3. Calculate two standard deviations over the distances in (2).
4. Then, discard any pair from (2) with a distance greater than the 2*SD you calculated in (3).

b) Programming overview

Hint: How to select a subset of a dataframe

In [7]:

```r
# Let's find all cars with 6 cylinders in mtcars
cars6cyl <- subset(mtcars, cyl==6)
```

Hint: How to loop over a dataframe by row index

In [8]:

```
# Let's find the qsec variable for every even-numbered row

for (row in 1:nrow(mtcars)) {            # Loop over dataframe
    if (row%%2==0) {                     # Select only even-numbered rows
        print(mtcars[row, c('qsec')])    # Print the "qsec" variable for that r
ow
        }
}
```

```
[1] 17.02
[1] 19.44
[1] 20.22
[1] 20
[1] 18.3
[1] 17.4
[1] 18
[1] 17.82
[1] 19.47
[1] 19.9
[1] 16.87
[1] 15.41
[1] 18.9
[1] 16.9
[1] 15.5
[1] 18.6
```

Hint: Initialize a dataframe and add rows

In [9]:

```
# Step 1: Initialize empty dataframe
df <- data.frame(matrix(ncol = 2, nrow = 0))

# Step 2: Name the dataframe columns
names(df) <- c("a", "b")

# Step 3: Add rows
df[nrow(df)+1,] = list('test1', 'test2')
df[nrow(df)+1,] = list('test3', 'test4')

df
```

a	b
test1	test2
test3	test4

Hint: How to fit a logistic regression in R, and use it for prediction

In [10]:

```
# Step 1: Let's use our mtcars data again
data(mtcars)

# Step 2: Fit a logit model to predict 'am' using 'cyl' and 'wt' variables
fittedlogit <- glm(am~cyl+wt, family=binomial(link='logit'), data=mtcars)

# Step 3: Get predicted values and assign them to a new column
mtcars$predicted <- predict(fittedlogit, newdata=mtcars, type='response')
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	predicted
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	0.95589722
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	0.74475046
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	0.94223476
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	0.16756939
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	0.32543790
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	0.02848076

c) Helpful resources

Some other resources that may be of use are:

- Propensity score section (https://ebookcentral-proquest-com.proxy.library.nyu.edu/lib/nyulibrary-ebooks/reader.action?docID=288457&ppg=232) from Gelman & Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models* (http://www.stat.columbia.edu/~gelman/arm/), with two caveats:
  - They use a function called `matching` but I think `MatchIt` is the standard
  - You don't need to fit the regression on p209 because you can do a simple means comparison of treatment vs. control
- KDnuggets tutorial (https://www.kdnuggets.com/2018/01/propensity-score-matching-r.html): Propensity Score Matching in R
- DataScience+ tutorial (https://datascienceplus.com/how-to-use-r-for-matching-samples-propensity-score/): How to use R for Matching Samples