Coding Task Data Generation

Table of Contents

Data Generation

Google Collab Setup

Uncomment and run this block of code only if you are attempting to run this notebook on google collab

In [1]:
```
"""
from google.colab import drive
drive.mount("/content/drive/")
%cd "/content/drive/MyDrive/vaccine_experiment_datatask"
!pip install stargazer
"""
```

Out[1]:
```
'\nfrom google.colab import drive\ndrive.mount("/content/drive/")\n%cd "/content/drive/MyDrive/vaccine_experiment_datatask"\n!pip install s
targazer\n'
```

Importing Needed libraries

This notebook assumes that the working directory is the root of the project folder. Otherwise, change the directory using `os.chdir()`.

In [2]:
```
# Importing libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.ticker import MaxNLocator
plt.style.use('ggplot')
import seaborn as sns

# Importing Custom Scripts For This Project
from scripts.network_generation import diffusion_network
from scripts.data_generation import generate_weights,generate_demographics,generate_attitudes
from scripts.treatment_simulation import apply_treatment, assign_block_treatment

# Set random seed for reproducibility
np.random.seed(111)
```

Generating Demographic Information

In [3]:
```
demographics_df = generate_demographics(n=5000)
demographics_df.head(1)
```

Out[3]:

	demographic_age	demographic_income	demographic_education	demographic_unobs_grp
0	53	137.883039	6	A

Generating Attitudes Towards Vaccination

In [4]:
```
attitudes_df = generate_attitudes(demographics_df,
                                  demographic_cont_vars=['demographic_age','demographic_income','demographic_education'],
                                  demographic_cat_vars='demographic_unobs_grp')
print(demographics_df.shape)
attitudes_df.head(1)
```

```
(5000, 4)
```

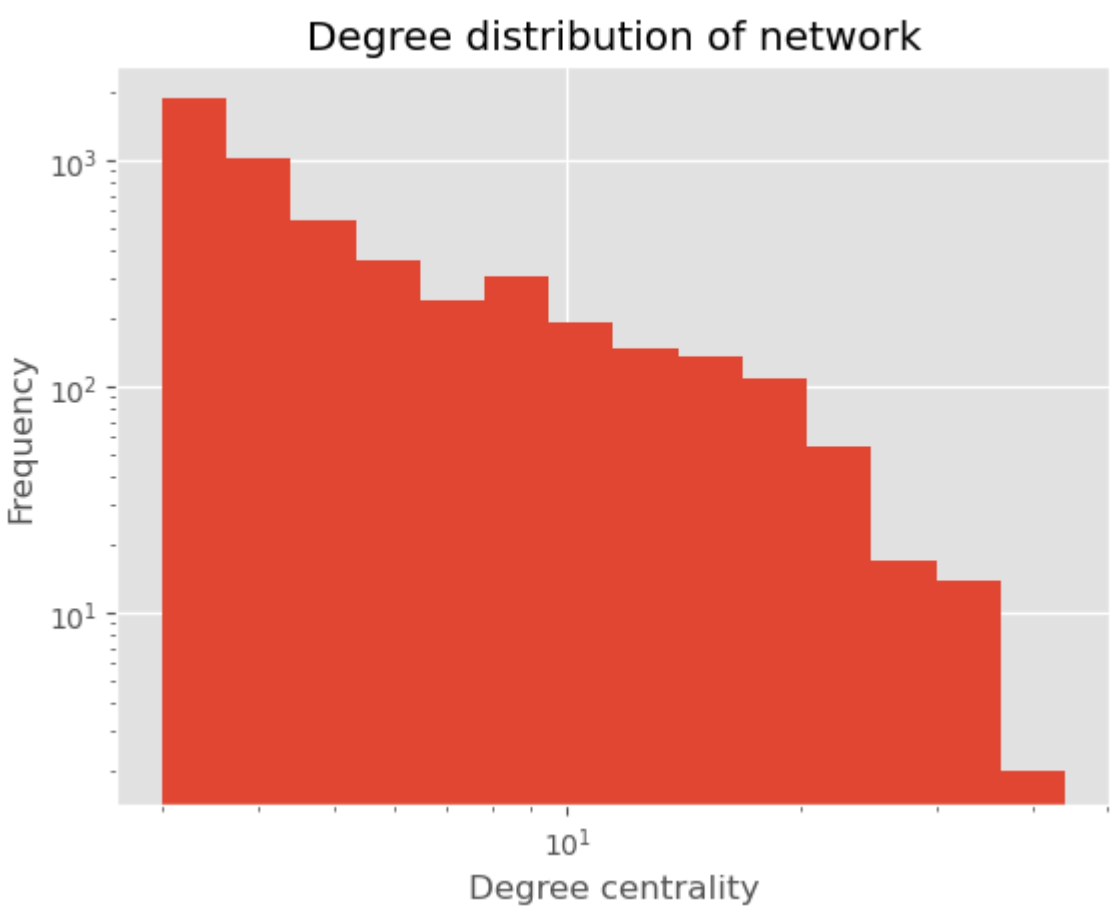
Out[4]:

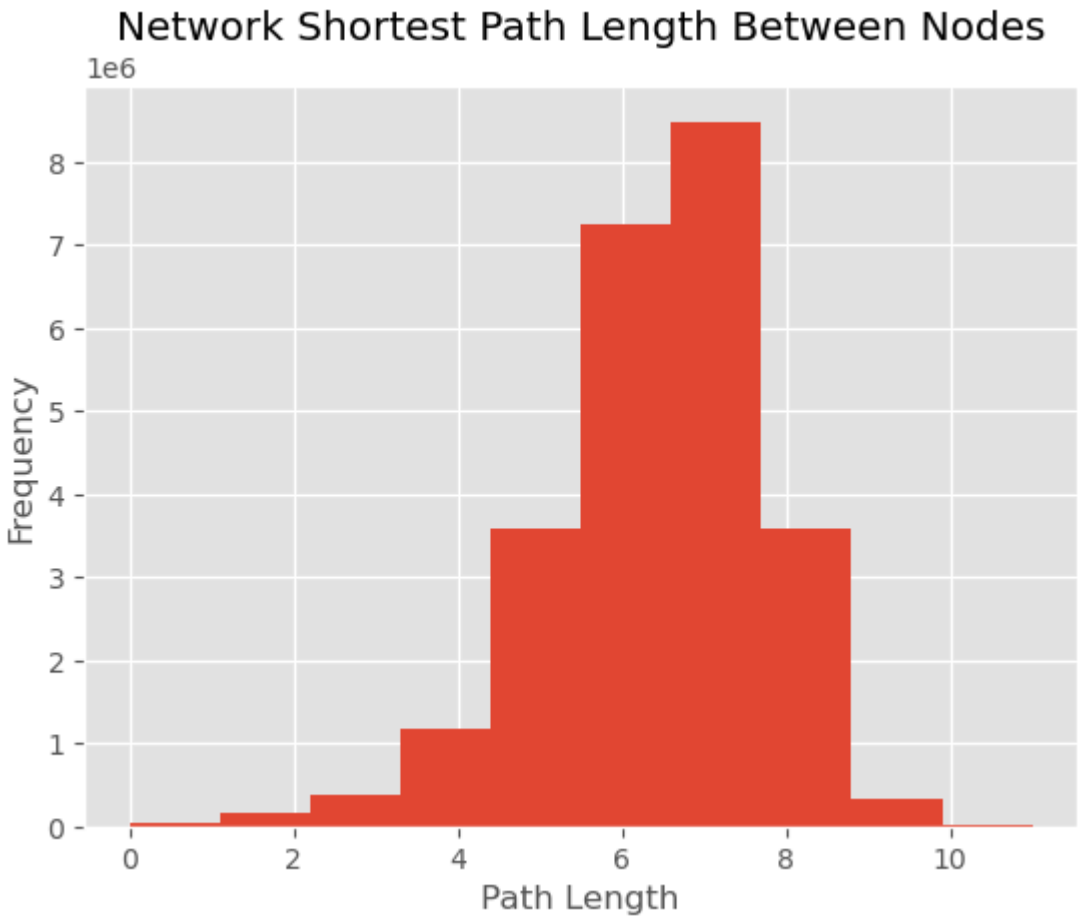
	att_covid	att_vaccine	att_safety	att_unobserved
0	2	5	3	5

Generating Network Structure

In [5]:
```
this_network = diffusion_network(attitudes_df)
```

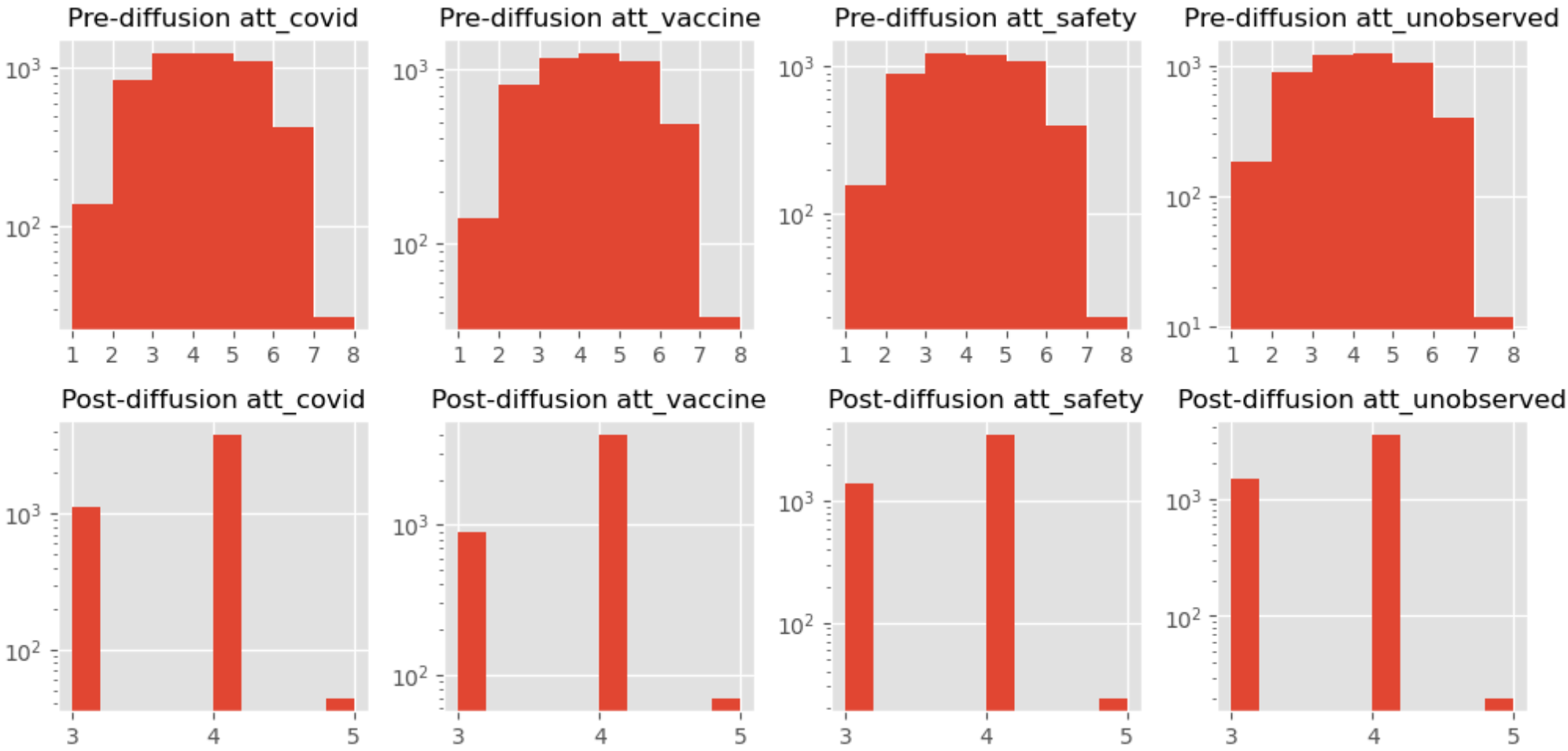
```
Network Generation Successful
Network Fully Connected: True
Number of Nodes: 5000
Number of Edges: 14400
Average Clustering: 0.19223701147097985
Modularity: 0.9027644410686728
```





Simulate the diffusion of initial attitudes

In [6]:
```
fig, ax = plt.subplots(nrows=2,ncols=4,squeeze=False,figsize=(10,5))
for i, attitude in enumerate(attitudes_df.columns):

    # Plot distribution of pre diffusion values
    ax[0,i].hist(attitudes_df[attitude],bins = range(1,9))
    ax[0,i].set_title('Pre-diffusion ' + str(attitude),fontsize=12)
    ax[0,i].xaxis.set_major_locator(MaxNLocator(integer=True))
    ax[0,i].set_yscale('log')

    # Plot distribution of post diffusion values
    attitudes_df[attitude] = this_network.fj_diffusion(attitudes_df[attitude])
    ax[1,i].hist(attitudes_df[attitude])
    ax[1,i].set_title('Post-diffusion ' + str(attitude),fontsize=12)
    ax[1,i].xaxis.set_major_locator(MaxNLocator(integer=True))
    ax[1,i].set_yscale('log')

fig.tight_layout(pad=1.0)
plt.savefig('figures/pre_treatment_diffusion.png')
plt.show()
```



Saving Data

Export edgelist and nodelist to csv for Gephi visualizations

In [7]:
```
# Export community list for Gephi visualization
this_network.node_community.to_csv('figures/gephi/community.csv')

# Export edgelist for Gephi visualization
this_network.export_edgelist(filename='figures/gephi/edgelist.csv') # export to gephi folder

# Export nodelist for Gephi
nodelist = attitudes_df['att_safety']
nodelist.index.name = 'id'
nodelist.to_csv('figures/gephi/nodelist.csv')
```

Export pre-treatment survey to csv

In [8]:
```
pre_treatment_df = pd.merge(attitudes_df,demographics_df,how='inner',left_index=True,right_index=True)
pre_treatment_df = pre_treatment_df.drop(columns=['att_unobserved','demographic_unobs_grp']) # Drop unobserved columnns
pre_treatment_df.head(1)
```

Out[8]:

id	att_covid	att_vaccine	att_safety	demographic_age	demographic_income	demographic_education
0	3.0	4.0	4.0	53	137.883039	6

In [9]:
```
# Save to file
pre_treatment_df.to_csv('data/pre_treatment.csv')
```

Simulating Treatment Effects

In [10]:
```
treatment_assignment_df = assign_block_treatment(this_network)
treatment_assignment_df.to_csv('data/treatment_assignment.csv') # Save to file
treatment_assignment_df.head(1)
```

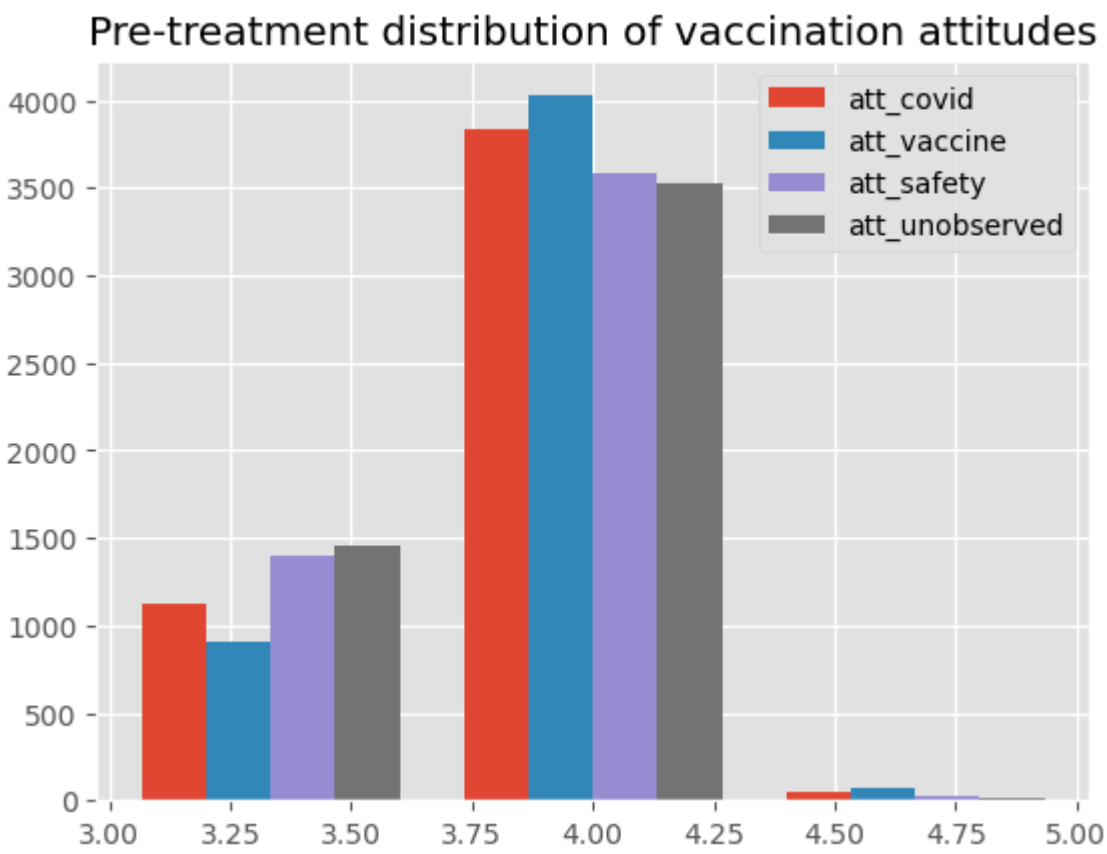
```
community
93       None
93       None
96    emotion
61     reason
65    emotion
       ...
25    emotion
17       None
17       None
46       None
42     reason
Name: treatment, Length: 5000, dtype: object
```

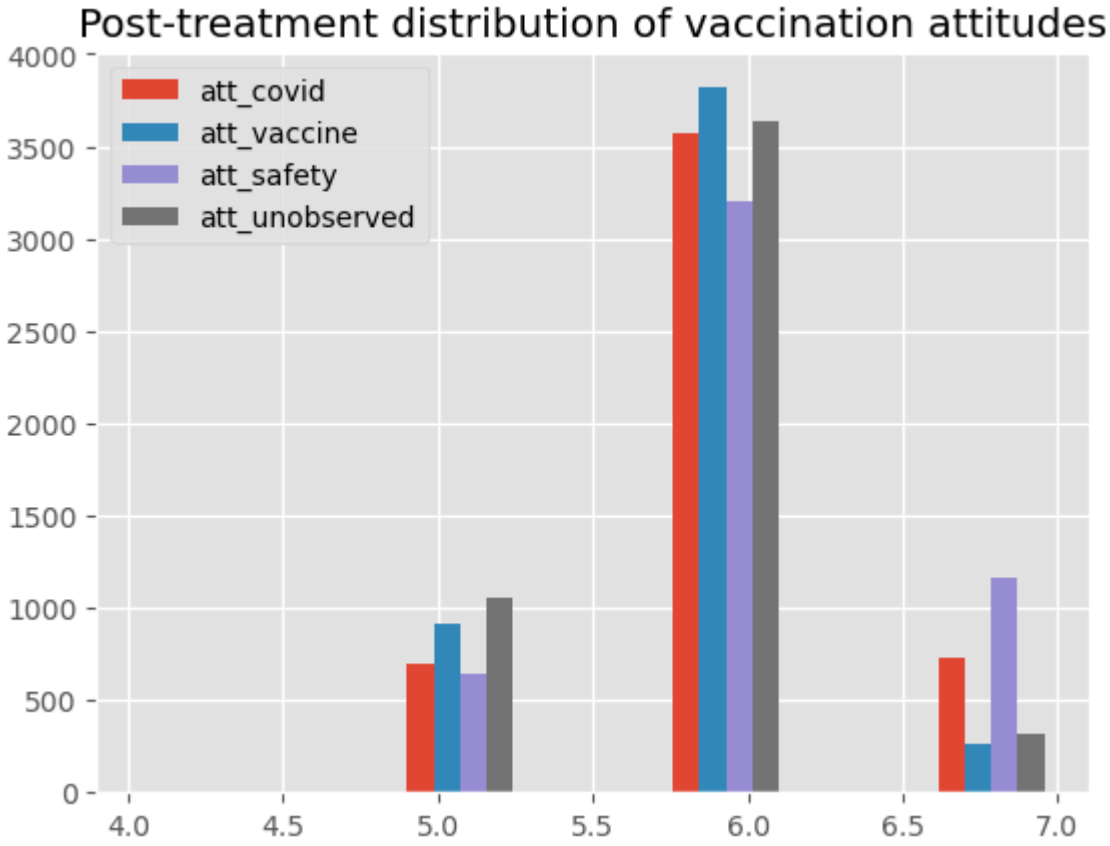
Out[10]:

id	community	treatment_emotion	treatment_reason
0	93	False	False

Model effects of treatment on attitudes

In [11]:
```
attitudes_df = apply_treatment(attitudes_df,treatment_assignment_df,this_network)
```





Model propensity to vaccinate

In [12]:
```
# Probability of vaccinating
weights = generate_weights(4)
vaccine_prob = np.zeros_like(attitudes_df.iloc[:,1])
for i,attitude in enumerate(attitudes_df.columns): # Weighted mean of each attitude
    vaccine_prob += attitudes_df[attitude] * weights[i]
vaccine_prob = vaccine_prob/10 # Normalize 10-point attitude scale into probability
vaccine_prob = vaccine_prob + np.random.normal(0,0.1) # Add random error

# Convert vaccine probability into vaccination outcome
post_survey = attitudes_df.copy(deep=True)
post_survey['vaccine'] = (vaccine_prob > np.random.uniform(0,1,5000))
post_survey.head(1)
```

Out[12]:

id	att_covid	att_vaccine	att_safety	att_unobserved	vaccine
0	5	5	6	5	True

Save post-treatment survey

Drop random rows to simulate 500 entries with attrition

In [13]:
```
# Mask 500 random entries
rows_to_drop = np.random.choice(range(0,5000),500,replace=False)
post_survey = post_survey.astype('int').astype('float')
post_survey.loc[rows_to_drop,:] = np.nan

# Drop unobserved variable
post_survey = post_survey.drop(columns=['att_unobserved'])

post_survey.head(1)
```

Out[13]:

id	att_covid	att_vaccine	att_safety	vaccine
0	5.0	5.0	6.0	1.0

In [14]:
```
post_survey.to_csv('data/post_treatment.csv')
```

Post-Simulation Analysis

Clear data cache and re-import packages

In [15]:
```
%reset -f

# Re-import packages
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from stargazer.stargazer import Stargazer
from sklearn.linear_model import LogisticRegression
import statsmodels.api as sm
import seaborn as sns
```

Combining the dataframes

In [16]:
```
pre_treatment_df = pd.read_csv('data/pre_treatment.csv',index_col='id')
random_assignment_df = pd.read_csv('data/treatment_assignment.csv',index_col='id')
post_treatment_df = pd.read_csv('data/post_treatment.csv',index_col='id')

# Rename variables to indicate if they're from pre or post_treatment
pre_treatment_df.columns = ['pre_' + x for x in pre_treatment_df.columns]
post_treatment_df.columns = ['post_' + x for x in post_treatment_df.columns]

# Merge all dataframes together
merged_df = pre_treatment_df.merge(random_assignment_df,left_index=True,right_index=True,how='inner',validate='1:1')
merged_df = merged_df.merge(post_treatment_df,how='inner',left_index=True,right_index=True,validate='1:1')
merged_df.head(50)

# Drop attritioned entries
merged_df = merged_df.dropna()

merged_df.head(1)
```

Out[16]:

id	pre_att_covid	pre_att_vaccine	pre_att_safety	pre_demographic_age	pre_demographic_income	pre_demographic_education	community	treatment_em
0	3.0	4.0	4.0	53	137.883039	6	93	

In [17]:
```
# Defining a few sets of variables
base_att = ['att_covid','att_safety','att_vaccine']
pre_att = ['pre_att_covid','pre_att_safety','pre_att_vaccine']
post_att = ['post_att_covid','post_att_safety','post_att_vaccine']
demographics = ['pre_demographic_age', 'pre_demographic_income', 'pre_demographic_education']
treatments = ['treatment_emotion','treatment_reason']
outcome = 'post_vaccine'
```

Basic regression table:

Analyzing the experiment's treatment effects on an individual-level suggests that the treatments **do** have a statistically significant effect on increasing vaccine uptake.

- The table below suggests that the emotion-based ads increased one's tendency to vaccinate by **~6.9%**
- The table below suggests that the reason-based ads increased one's tendency to vaccinate by **~8.3%**
- However, few covariances do a good job of predicting vaccination outcome, suggesting that treatment

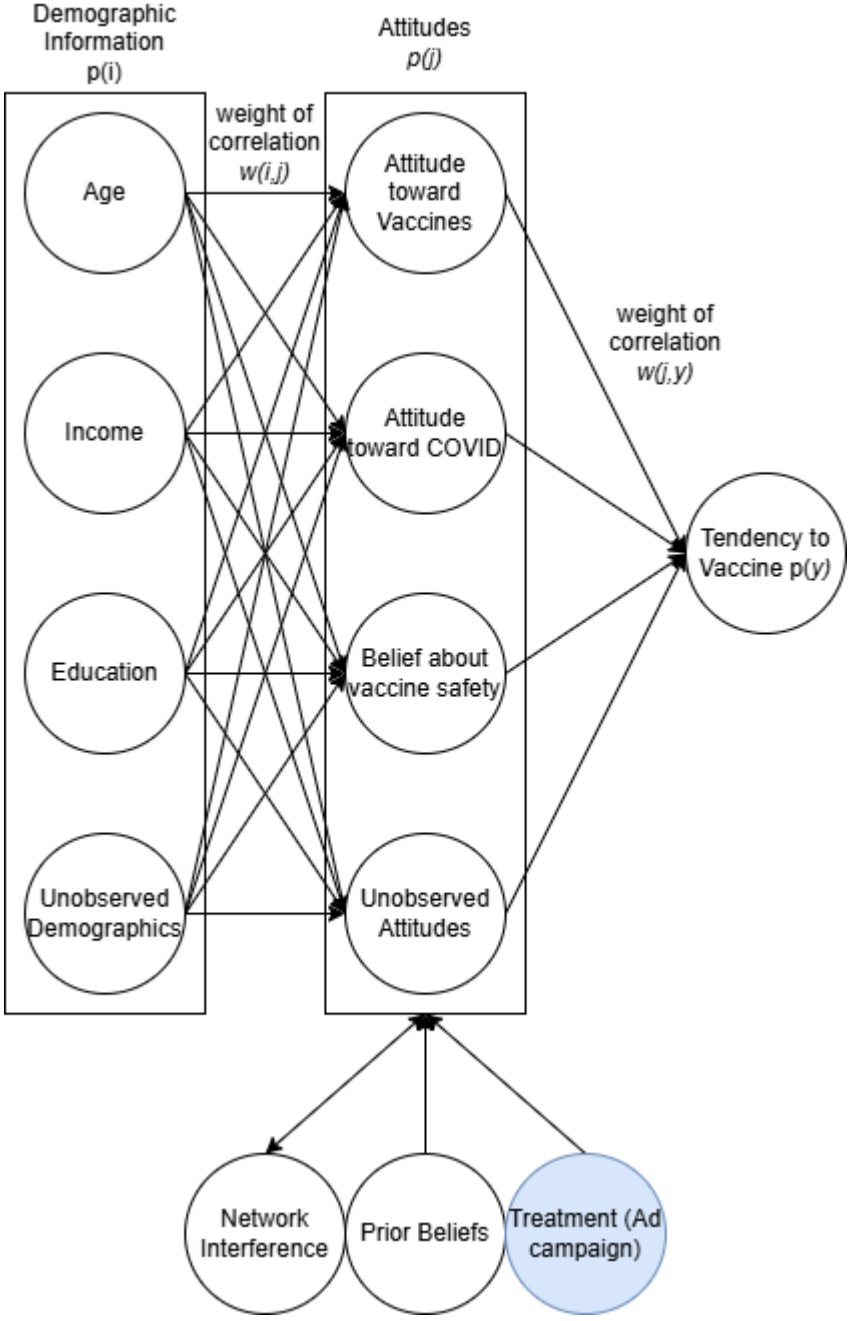
In [18]:
```
regressions = [treatments,demographics + treatments,pre_att + demographics + treatments]
reg_outcomes = []

for regression in regressions:
    X = sm.add_constant(merged_df[regression]).astype('float')
    Y = merged_df[outcome].astype('float')
    reg_outcomes.append(sm.OLS(Y,X).fit())

reg_table = Stargazer(reg_outcomes)
reg_table
```

Out[18]:

	Dependent variable: post_vaccine		
	(1)	(2)	(3)
const	0.506***	0.491***	0.283**
	(0.013)	(0.028)	(0.113)
pre_att_covid			0.014
			(0.017)
pre_att_safety			0.039**
			(0.016)
pre_att_vaccine			0.004
			(0.018)
pre_demographic_age		0.000	0.000
		(0.000)	(0.000)
pre_demographic_education		-0.004	-0.005
		(0.004)	(0.004)
pre_demographic_income		0.000*	0.000
		(0.000)	(0.000)
treatment_emotion	0.066***	0.066***	0.069***
	(0.018)	(0.018)	(0.018)
treatment_reason	0.084***	0.084***	0.083***
	(0.018)	(0.018)	(0.018)
Observations	4500	4500	4500
R^2	0.005	0.006	0.008
Adjusted R^2	0.005	0.005	0.006
Residual Std. Error	0.496 (df=4497)	0.496 (df=4494)	0.496 (df=4491)
F Statistic	12.154*** (df=2; 4497)	5.731*** (df=5; 4494)	4.395*** (df=8; 4491)
Note:			*p<0.1; **p<0.05; ***p<0.01

Attitudes as mechanism

As the Directed Acyclic Graph (DAG) below shows, we generate the data in a way that treatment affects vaccination propensity indirectly through component attitudes.



The below analysis firstly confirms a correlation between vaccinate attitudes, treatment, and vaccination propensity.

- The regression below suggests that the treatment did significantly push individual attitudes towards pro-vaccination by around **1-2 points** on the likert scale. This effect varied for each attitude towards vaccination.
- The regression table shows that **individuals who already have a high opinion of vaccination were less affected by the treatment**. This phenomenon is likely because the diffusion of opinions throughout the network homogenized in-group opinions, thus reducing extremely high opinions. The network structure is likely responsible for this phenomenon because this feature was not part of the treatment effect generation process.

In [19]:
```
# Initialize heterogenous treatment effect's interactive variables
regressions = []
for att in pre_att:
    merged_df[att +' * treatment_emotion'] = merged_df[att] * merged_df['treatment_emotion']
    merged_df[att + ' * treatment_reason'] = merged_df[att] * merged_df['treatment_reason']
    regressions.append(treatments + [att,att +' * treatment_emotion',att + ' * treatment_reason'])

outcome_vars = post_att
reg_outcomes = []

for i in range(0,len(outcome_vars)):
    X = sm.add_constant(merged_df[regressions[i]]).astype('float')
    Y = merged_df[outcome_vars[i]].astype('float')
    reg_outcomes.append(sm.OLS(Y,X).fit())

reg_table = Stargazer(reg_outcomes)
reg_table.custom_columns(outcome_vars)
reg_table
```

Out[19]:

	post_att_covid	post_att_safety	post_att_vaccine
	(1)	(2)	(3)
const	3.693***	3.857***	3.770***
	(0.095)	(0.094)	(0.086)
pre_att_covid	0.506***		
	(0.025)		
pre_att_covid * treatment_emotion	-0.199***		
	(0.034)		
pre_att_covid * treatment_reason	-0.258***		
	(0.036)		
pre_att_safety		0.479***	
		(0.025)	
pre_att_safety * treatment_emotion		-0.138***	
		(0.035)	
pre_att_safety * treatment_reason		-0.121***	
		(0.036)	
pre_att_vaccine			0.447***
			(0.022)
pre_att_vaccine * treatment_emotion			-0.329***
			(0.032)
pre_att_vaccine * treatment_reason			-0.276***
			(0.032)
treatment_emotion	1.419***	1.203***	1.817***
	(0.130)	(0.131)	(0.122)
treatment_reason	1.548***	1.195***	1.682***
	(0.138)	(0.137)	(0.125)
Observations	4500	4500	4500
R^2	0.393	0.432	0.418
Adjusted R^2	0.393	0.432	0.417
Residual Std. Error	0.416 (df=4494)	0.445 (df=4494)	0.357 (df=4494)
F Statistic	582.938*** (df=5; 4494)	684.307*** (df=5; 4494)	645.328*** (df=5; 4494)
Note:			*p<0.1; **p<0.05; ***p<0.01

Block-Level Treatment Effect

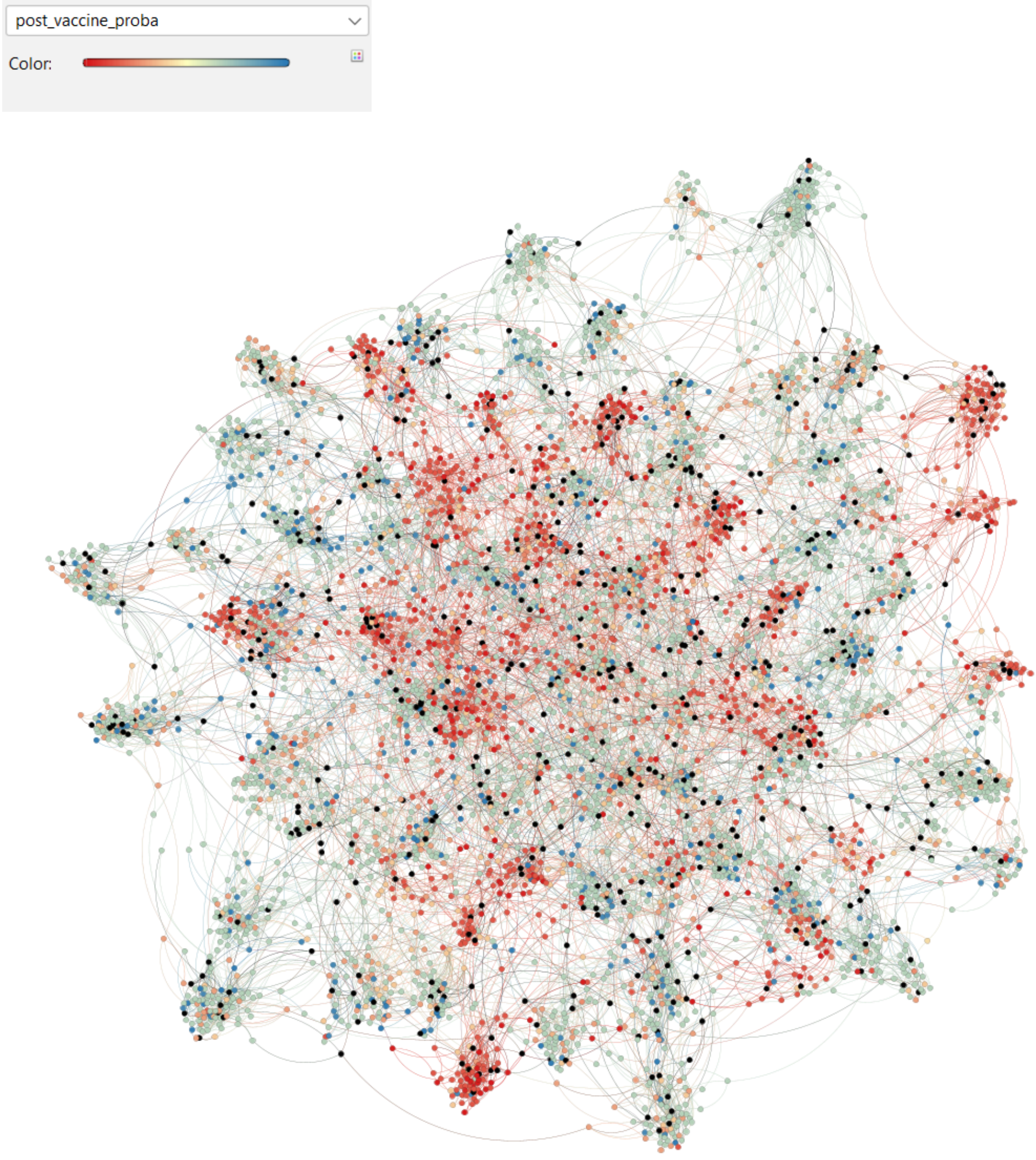
Observing the underlying network structure, one would realize that vaccination-rate is highly correlated with the community that the individual is part of. This suggests that block-level treatment has been effective because one primarily sees between-group differences, rather than in-group differences in vaccine propensity.

The following code allows us to shade a node based on its predicted propensity towards vaccination (PPV), rather than actual observed outcome. Visualizing the PPV on a network using Gephi, one can observe that vaccination propensity is highly homogenous within each community, supporting the hypothesis that the treatment primarily worked by converting the opinion of entire 'blocks'.

In [20]:
```
# Create a predicted probability of vaccination based on post-attitudes. Train model:
Y = merged_df['post_vaccine']
X = merged_df[post_att]
reg = LogisticRegression().fit(X, Y)

# Predict probability of vaccination
merged_df['post_vaccine_proba'] = reg.predict_proba(X)[:,1]

# Rank transform the probabilities to highlight contrast
merged_df['post_vaccine_proba'] = merged_df['post_vaccine_proba'].rank()
```

In [21]:
```
# Export vaccination information to Gephi for visualization
vaccine_nodelist = merged_df['post_vaccine_proba'].to_csv('figures/gephi/vaccine_nodelist.csv')
```

Gephi visualization: (black nodes represents nodes with missing data)



Running the same regression table on a block level also confirms the earlier hypothesis, showing a significant block-level treatment effect. The magnitudes of the treatment effect between block and individual-level analysis are also highly similar, which increases one's confidence in the magnitude of the estimate.

- The table below suggests that the emotion-based ads increased one's tendency to vaccinate by **7.3%**
- The table below suggests that the reason-based ads increased one's tendency to vaccinate by **9.3%**

In [22]:
```
merged_df=merged_df.groupby('community',observed=False).mean()
print(merged_df.shape)
merged_df.head(1)
```

```
(100, 19)
```

Out[22]:

community	pre_att_covid	pre_att_vaccine	pre_att_safety	pre_demographic_age	pre_demographic_income	pre_demographic_education	treatment_emotio
0	3.896552	4.0	4.017241	49.172414	136.317512	3.689655	0.

In [23]:
```
reg_outcomes = []
outcome_vars = post_att + ['post_vaccine',]
for att in outcome_vars:
    X = sm.add_constant(merged_df[treatments + pre_att]).astype('float')
    Y = merged_df[att].astype('float')
    reg_outcomes.append(sm.OLS(Y,X).fit())
```

```
reg_table = Stargazer(reg_outcomes)
reg_table.custom_columns(outcome_vars)
reg_table
```

Out[23]:

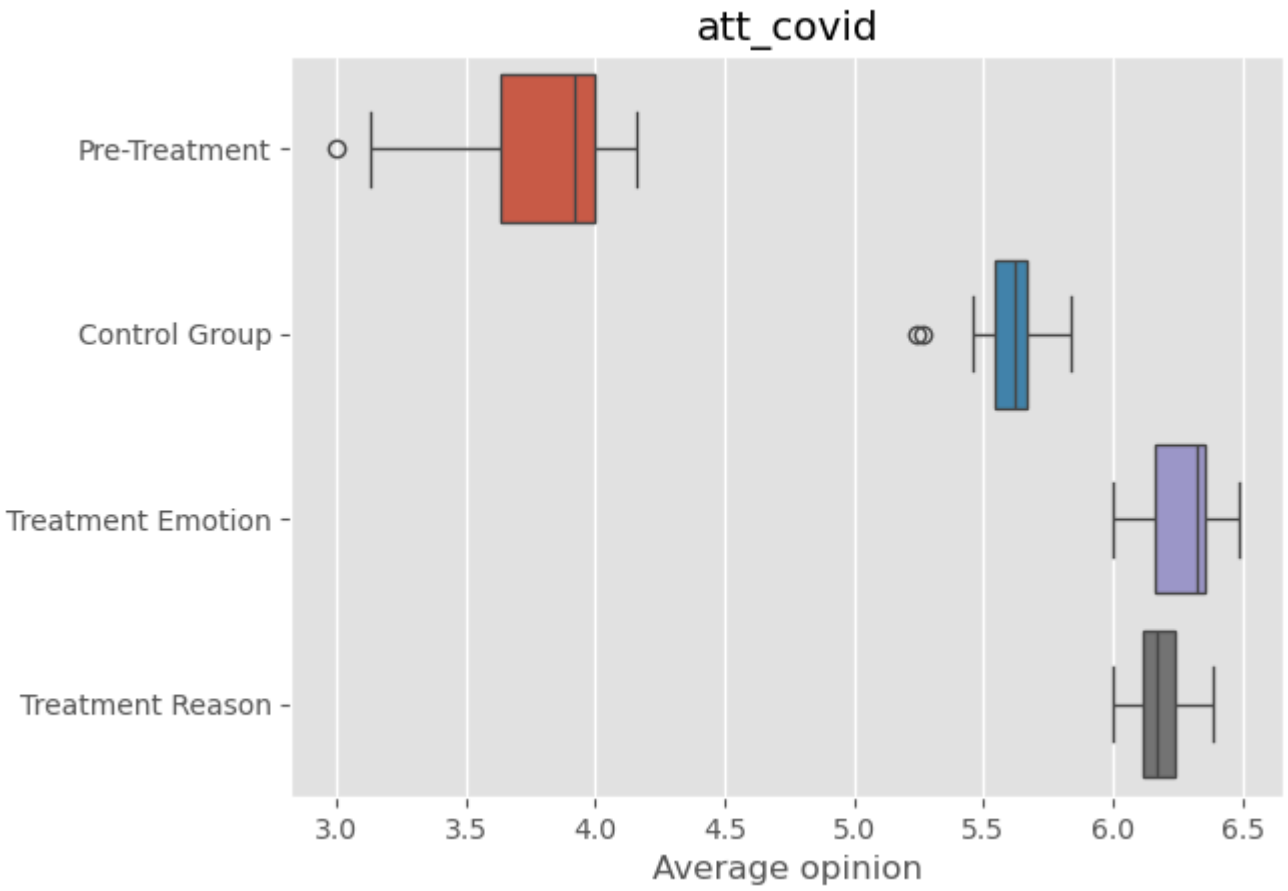
	post_att_covid	post_att_safety	post_att_vaccine	post_vaccine
	(1)	(2)	(3)	(4)
const	4.376***	4.239***	4.649***	0.566***
	(0.148)	(0.155)	(0.150)	(0.163)
pre_att_covid	0.334***	0.020	-0.004	0.000
	(0.023)	(0.024)	(0.023)	(0.025)
pre_att_safety	-0.005	0.359***	-0.004	0.017
	(0.021)	(0.022)	(0.021)	(0.023)
pre_att_vaccine	-0.004	-0.002	0.229***	-0.034
	(0.025)	(0.026)	(0.025)	(0.028)
treatment_emotion	0.666***	0.682***	0.548***	0.073***
	(0.017)	(0.018)	(0.017)	(0.019)
treatment_reason	0.569***	0.742***	0.616***	0.093***
	(0.017)	(0.018)	(0.017)	(0.019)
Observations	100	100	100	100
R^2	0.956	0.963	0.945	0.246
Adjusted R^2	0.954	0.962	0.942	0.206
Residual Std. Error	0.069 (df=94)	0.072 (df=94)	0.069 (df=94)	0.076 (df=94)
F Statistic	409.987*** (df=5; 94)	495.750*** (df=5; 94)	325.265*** (df=5; 94)	6.147*** (df=5; 94)
Note:				*p<0.1; **p<0.05; ***p<0.01

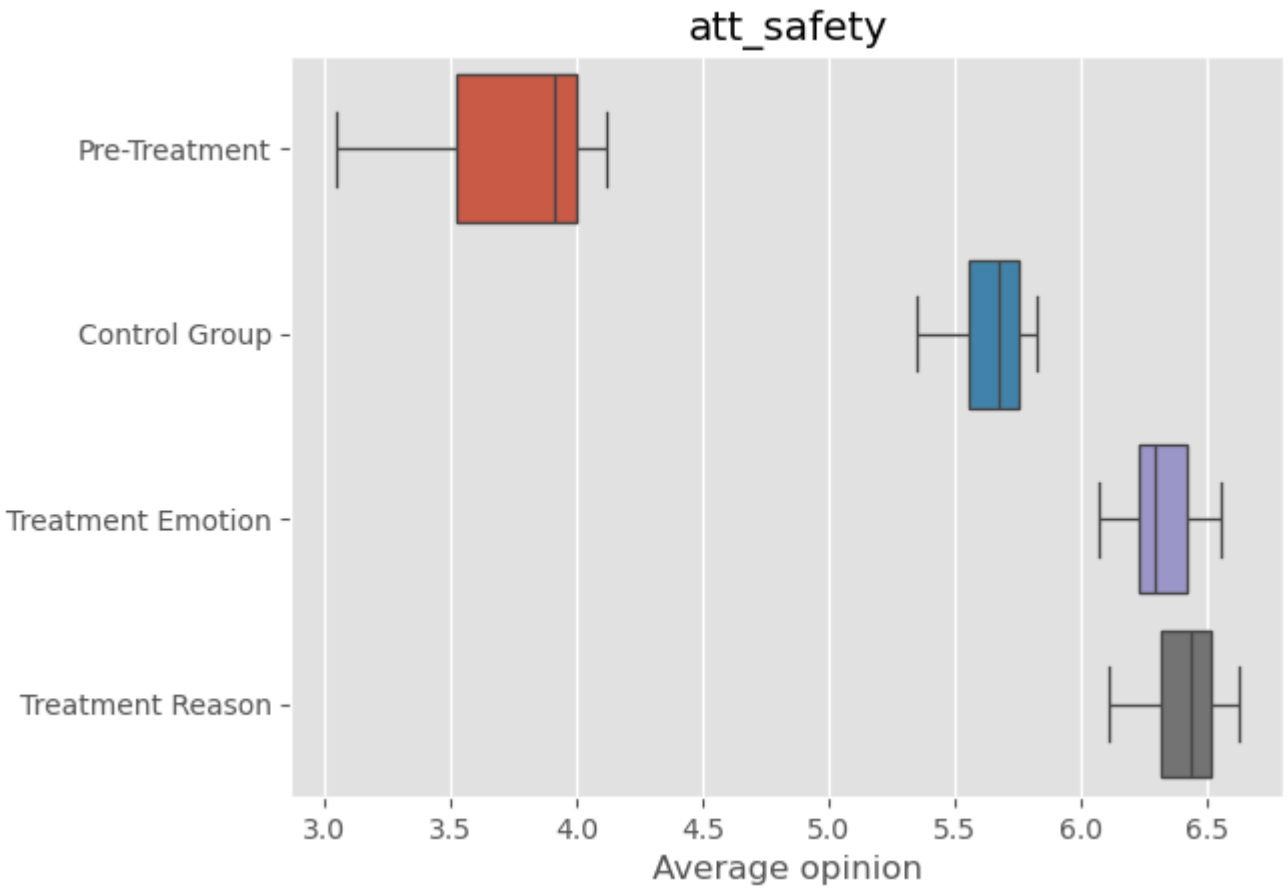
Network Interference

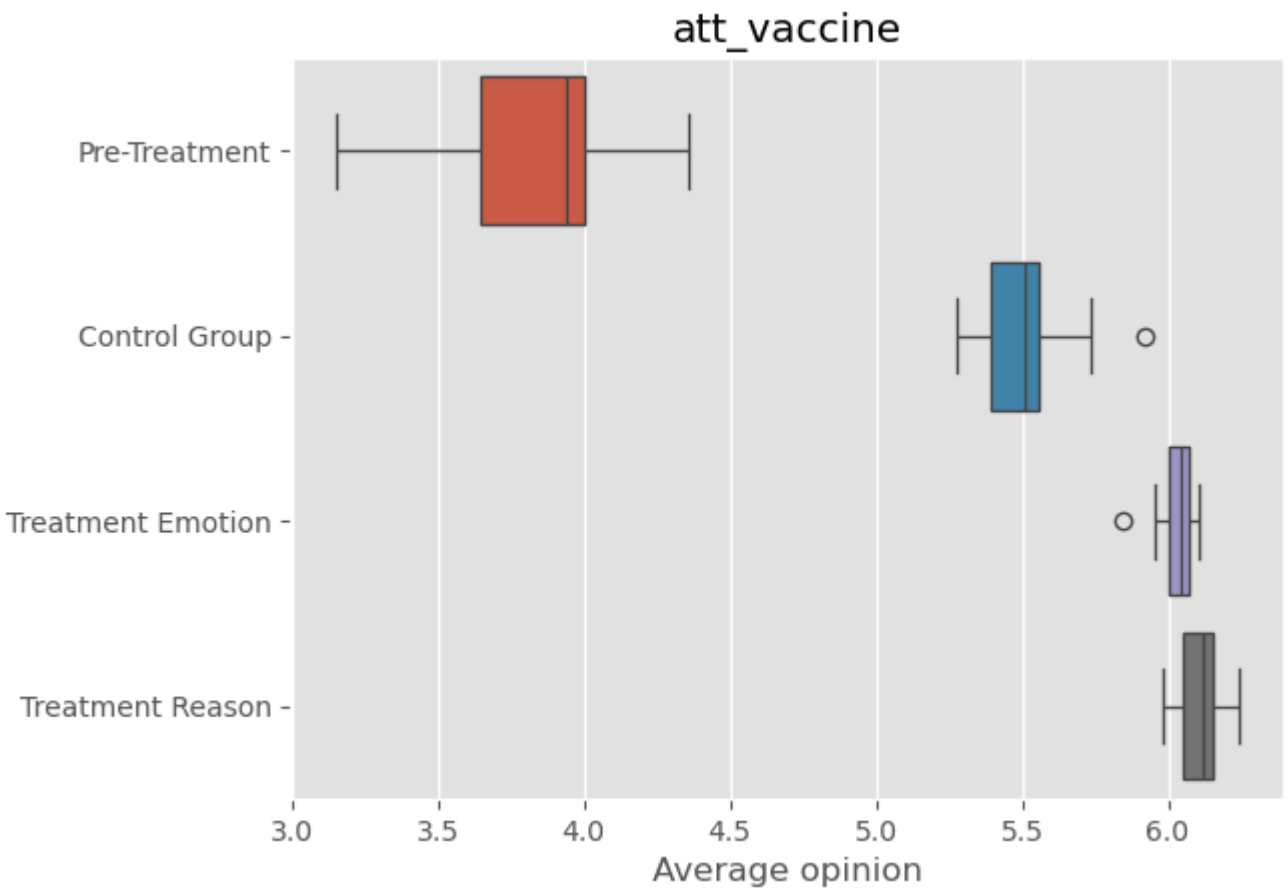
This section shows that due to the effects of interference, the treatment also had an effect on the control group.

- The graphs below shows that compared to pre-treatment levels, the treatment increased the control group's propensity to vaccinate by **about 17.5%**, while treatments increased the treated groups' propensity to vaccinate by **about 25%**.
- This means that network interference caused the previous analysis comparing control and treated groups to return a highly conservative estimate of the treatment effect, causing the project to underestimate the treatment effect by **about 17.5%.**
- It would be difficult to measure the true magnitude of underestimation due to network interference in a real-life situation because one cannot know/assume that no other factors (besides the treatment) intervened to cause this difference between pre-treatment and post-treatment. Thus, this simulation illustrates how real-life surveys need to use domain specific knowledge to determine whether network interference effects exist - it's hard to quantitatively measure interference effects unless the experiment takes place in a fully controlled lab environment.

In [25]:
```
for att in base_att:
    sns.boxplot(data=[merged_df['pre_'+att].rename('Pre-Treatment'),
                merged_df[~(merged_df['treatment_reason'].astype('bool') |
                            merged_df['treatment_emotion'].astype('bool'))]['post_'+att].rename('Control Group'),
                merged_df[merged_df['treatment_emotion'].astype('bool')]['post_'+att].rename('Treatment Emotion'),
                merged_df[merged_df['treatment_reason'].astype('bool')]['post_'+att].rename('Treatment Reason')],
                orient='h')
    plt.xlabel('Average opinion')
    plt.title(att)
    plt.show()
```







End of Notebook