5-1-2014

Community Detection as a Method to Control For Homophily in Social Networks

Hannah Worrall
Carnegie Mellon University

Community Detection as a Method to Control For Homophily in Social Networks

Hannah Worrall
Senior Honors/QSSS Thesis

April 30, 2014

Contents

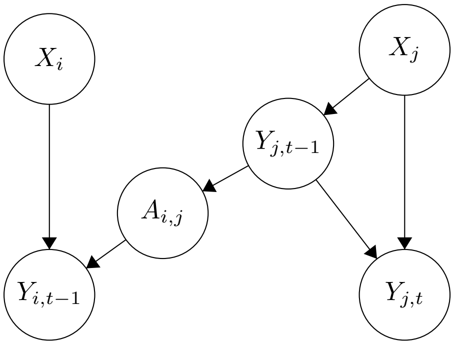
1 Introduction

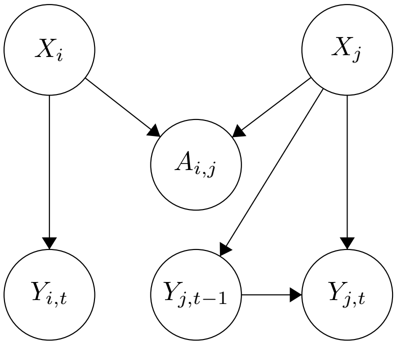
People tend to be very similar to their friends. Groups of friends often have similar political affiliations, exercise habits, and socioeconomic backgrounds. Is this similarity because people become friends and then start influencing each other, or because their friendship begins due to similar traits? If one person starts exhibiting similar behavior as their friend, is it because their friend caused them to behave that way or because their similar traits have caused them both to start exhibiting the trait? This question is difficult to answer, but researchers investigating how people interact with each other often encounter the issue. In this paper I put forth a method to help differentiate between one friend influencing another (contagion) and two friends behaving similarly because the cause of the behavior is the same reason they became friends in the first place (homophily).

1.1 Background

Many essential questions in the social sciences involve networks. A network is a structure composed of nodes and edges. A node can be thought of as an individual of interest and an edge is a connection between nodes. Social scientists study social networks in which each node is an individual actor and each edge is some sort of relationship between actors. For example, many studies have examined friendship networks, in which each node is a person and each edge indicates friendship between individuals. Much research has been done on networks in various fields. In the social sciences, one of the most important goals in analyzing social networks is determining causal inference within the network.

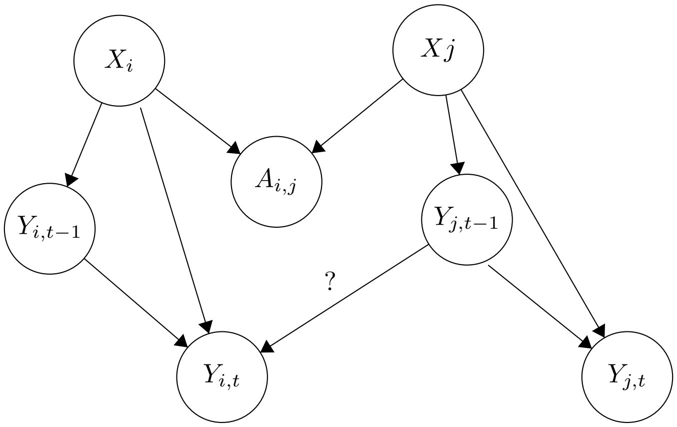
This is a difficult undertaking. A social scientist wants to know if attributes of individuals can be passed through the network. For example, he might want to know if an individual is more likely to become obese if his friends are also obese [2]. However, this question is complicated by the formation of friendship. Perhaps two people are friends because they both go to a burger restaurant every week. In this case both friendship and obesity have the same underlying cause. These two cases can be seen in the causal graphs below. The graph on the left depicts the case where one friend causes the other to become obese, while the graph at right depicts the case where the obesity is caused by the fondness for burger restaurants. The X are the individuals' underlying characteristics (in this case a strong liking for burger restaurants) and the Y are their observed characteristics (in this case obesity). The individuals are called i and j and their friendship is indicated by A_{ij}. Therefore X_i is the underlying trait of person i, Y_{jt} is the observed variable of person j at time t, and so on.





It is therefore very important for social scientists to be able to differentiate between homophily and contagion.

Shalizi and Thomas published a paper in 2011 which showed that it was impossible to differentiate between these two influences without making very strong assumptions about the underlying network structure [5]. The crux of their argument lies on the causal graph displayed below.



In this graph we can see homophily: the X_i and X_j are causing the formation of the friendship tie A_{ij}. If $Y_{j,t-1}$ and $Y_{i,t}$ are connected in the graph, then contagion is present. This edge may or may not exist, and determining if it does in specific applications is the goal of many studies. There exists a path between X_j and both variables whether or not contagion edge is present. Thus $Y_{j,t-1}$ and $Y_{i,t}$ are correlated and one contains information about the other [5]. Therefore homophily looks like contagion, unless you can control for every trait in X or make strong assumptions [5].

Shalizi and Thomas said it might be possible to determine bounds for the influences, which would help determine if the attribute in question was in fact being passed along friendship ties. In particular, they thought that grouping individuals in a network into communities would help control for latent characteristics.[1] The idea behind this is that people are likely to befriend people who have similar characteristics, and thus groups of friends likely have the same shared characteristics and form communities within the network whose members share these characteristics. Membership in a

[1] A community is a collection of nodes in which the probability of an edge forming between nodes within the community is greater than the probability of an edge forming between a node within the community and a node outside of it.

community would thus be an indicator of the similar characteristics. Controlling for community might thus be similar to controlling for the actual characteristics.

This report shows when this technique might work. If it is possible to partially control for latent characteristics by controlling for community this would greatly help social scientists determine the presence of causality in social networks. Since so many questions in the social sciences can be modeled by a social network, this would be very helpful across many disciplines.

Essential to this approach is community detection. It should only work if the communities detected by an algorithm are correct. I am defining correct communities to be communities which are consistent. In other words, if we let c be the true community of a node, and $\hat{c}$ to be the community detected by a community detection algorithm for the same node, the community detection is consistent if

$$P[c = \hat{c}] \to 1$$

as

$$n \to \infty$$

with n being the number of nodes in the network. Note that the method assumes communities will match up to the X_i if the method works. In other words, the method works if nodes with similar X_i have the same c. Also note that this definition describes asymptotic behavior. Therefore for any test that relies on community detection I will have to check its behavior as the network grows. Networks tend to grow in one of three ways, described in section 4.2, so I will check the method against each type of growth.

2 Literature Review

Many key conclusions in the social sciences have been based off of work with social networks. A commonly used example is Christakis and Fowler's 2007 paper on the spread of obesity through social networks [2]. This paper seemed to show that obesity could be thought of as an infectious disease [2]. Is this a true effect, or is homophily to blame? This is a very important question in the social sciences. The issue of homophily has been discussed in the literature. For example, McPherson et all wrote a paper in 2001 which discusses homophily and how it creates social cliques and communities [4].

Social scientists have thus been aware that homophily and contagion may be confounded in social networks for some time. One response to this problem has been to collect as much data as they can, and hope that the homophily is captured in one of the many variables they measure. Obviously this method has no guarantee of solving the problem, and if researchers assume that they have controlled for everything then there is a risk that they move forward with misleading and false results.

This paper is based off of the work of Shalizi and Thomas. In 2010 they finished a paper entitled "Homophily and Contagion Are Generically Confounded in Observational Social Network Studies" [5]. In this paper they show that "distinguishing [homophily and contagion] from one another requires strong assumptions on the parametrization of the social process or on the adequacy of the covariates used (or both)" [5]. Their paper examines graphical causal models to demonstrate this conclusion [5]. They then put forth several potential partial fixes to the problem.

The first potential solution involves not conditioning on the social network during inference, thereby not activating the collider [2] in the graphical causal model represented above [5]. They also suggest attempting to place bounds on the causal effect. This method has been attempted with some success [5]. Greg Ver Steeg and Aram Galstyan wrote a paper entitled "Statistical Tests for Contagion in Observational Social Network Studies" in April 2013 as a follow-up to Shalizi and Thomas's paper, in which they successfully place a upper bound on apparent contagion due to homophily, demonstrating the presence of contagion [6]. Lastly, Shalizi and Thomas suggest that controlling for cluster within the social network might help identify the presence of causality [5]. This is the method I will be testing.

3 Simulation Methods

In order to test if it is possible to control for homophily in a network by including community membership I simulated and evolved networks. This section describes the method I used.

3.1 Network Creation

First I generated the basic structure of my networks. I created n nodes. Each node in the network was assigned a latent variable. For simplicity, this variable was binary, with equal probability of being a 0 or a 1. These two numbers represent the two possible states of the variable. I then assigned each node an observed variable. This variable was also binary. There was a 70% chance of it being the same as the latent variable and a 30% chance of it being different. These probabilities were largely arbitrary. The 70% probability is large enough that most nodes have the same state for both their observed and latent variables, but not so large that there aren't a good number of nodes for which that isn't true. Further work might involve altering these probabilities and seeing what changes in the simulation. I then created the social network in the following way:

1. I assigned a base probability a of a connection between any two nodes, and an additional probability b for a connection between two nodes with the same latent variables. Therefore the total probability of a connection between nodes with the same latent variable was $a + b$, while the probability of a connection between nodes with differing latent variables was a. If b is greater than zero in this setup the network contains homophily, as nodes tend to form edges with nodes with the same underlying latent variable.

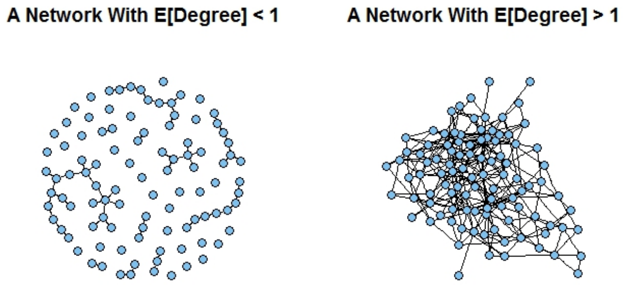
2. For each i, j pair of nodes, I simulated the outcome of a Bernoulli random variable with probability as described above. This outcome was binary. If the result was a 1, I created an edge between nodes i and j. If the result was 0 I did not. I did this only once for each pair and did not do it for a node and itself.

[2] A collider is a variable in a causal graph which has two variables which directly cause it. Conditioning on the variable allows information to pass through it, while not conditioning on the variable blocks the flow of information.

3. For parts of my analysis I required that each node be connected to at least one other node. This behavior was desired if I wanted to work with a small network. If this was the case, after I created my network I checked each node to see if it had a neighbor. If it did, I moved on to the next node. If it did not, I did one of the following:
    (a) With probability .7 I created an edge between the node and another node with the same underlying latent variable.
    (b) With probability .3 I created an edge between the node and another node with a different underlying latent variable [3].

These networks were represented by a $n \times n$ matrix. Each element A_{ij} is binary: It is a 1 if there is a connection between nodes i and j and a 0 otherwise. I used undirected graphs only. This means that each edge of the graph has no orientation. The connection between nodes is a symmetric relationship between them. Since my graphs are undirected the matrix containing the connections is symmetric ($A_{ij} = A_{ji}$).

In some cases, if the expected number of edges belonging to a node (also known as the degree of a node) is less than 1, the network will break apart into many small pieces. Below is a small network in which this occurs. Analyzing such networks is a waste of time. For an idea of why this is the case, examine the graph below. The network is broken into pieces, and it is impossible to find a path between most nodes. In addition none of the subnetworks have a significant number of nodes. While it is possible to analyze a subnetwork within the larger network, this should only be done if the largest subnetwork contains a large number of the total nodes in the whole network. I used a cutoff of one third, but it isn't essential this cutoff be used. Trying to analyze a network that is as scattered as the one below wouldn't be useful, as no subnetwork can contain enough information about the larger network to be useful.



[3] Again, the specific probabilities aren't important

3.2 Network Evolution

Social networks change over time. This change is called evolution. I evolved my network in one of two ways. In one way I evolved my network such that it would have contagion. In the other I evolved it so it did not. For networks with contagion I evolved them in the following way:

1. I selected how much time I wanted to elapse, t, and how often I got a snapshot of my network s. For example, I may want my network to evolve 1000 times, and get an idea of what it looked like every ten evolutions.

2. At each time point I selected a node at random to evolve.

3. I then forced the node to switch its observed variable to the value of one of its neighbors observed variables, chosen at random.

4. If this evolution was at a time point to be saved, I recorded the state of the network at this time point. In other words, I recorded the observed variable value for every node at this time point.

I ended up with an $n \times \frac{t}{s}$ matrix which I will call B. Each element B_{ik} is thus the value of node i at time k.

Similarly I evolved networks with no contagion in the following manner:

1. I selected how much time I wanted to elapse t, and how often I got a snapshot of my network s. For example, I may want my network to evolve 1000 times, and get an idea of what looked like every ten evolutions.

2. At each time point I selected a node at random to evolve.

3. I then had the selected node's observed variable match up with its latent variable with probability .7, and take the opposite value with probability .3.

4. If this evolution was at a time point to be saved, I recorded the state of the network at this time point. Again, I recorded the observed variable value for every node at this time point[4].

Like before I end up with a $n \times \frac{t}{s}$ matrix with elements defined the same way as in the network with contagion.

3.3 Community Detection

I then determined the community membership of each node. I used the R package igraph's [3] community detection algorithm spinglass.community. This package uses simulated annealing to find the optimal community structure. It does this by assigning each node a spin state, which can be thought of as a guess for its true community, and then finding the collection of spin states in which the most nodes have the same spin states as their neighbors but have different spin states from their non-neighbors. I used spinglass.community because it was able to detect the communities in my networks well. Any community detection algorithm should work, however. Since my network had binary variables I set the number of communities at two.

[4] Yet again, the exact probabilities used here aren't important

3.4 Creating A Table To Analyze The Network

After I initialized network and had evolved it over time I still needed a way to organize the information in a way that was easy to run regressions on. To do this I created a matrix which for every pair of i, j with nodes $i \neq j$ at each saved time point $\frac{t}{s}$ contained the following:

1. Node i's observed variable
2. Node j's observed variable
3. Node i's detected community
4. Node j's detected community

This matrix C could then be used to analyze the network.

3.5 Computational Difficulties

Determining the usefulness of using communities to control for homophily requires growing the networks, which will be described later. This can cause computational difficulties. C grows proportional to $n^2 \times \frac{t}{s}$, and thus becomes millions of rows long very quickly. For reference, a network I created with 1000 nodes had a C matrix that was 10,904,627 rows long. It isn't feasible to create this table and then run a logistic regression on its entirety with networks of more than ≈ 1000 nodes.

There are two key bottlenecks in this problem. The first is the creation of C itself. No matter how efficient the algorithm used to construct the matrix, at large network sizes it becomes incredibly time intensive to create the matrix. The matrix also begins to take up an incredible amount of memory.

The next bottleneck is the actual running of the logistic regressions. Putting the whole matrix into any logistic regression software results in a wait of hours before the computation is done. Even if one uses packages in R developed to run on large datasets this computation is extremely time intensive.

Both these issues appear when networks have only a thousand nodes. Many real problems require networks which contain many more nodes than this. Therefore it is necessary to speed up the analysis of a network.

I solved this problem by sampling from C without ever actually creating it. To do this I sampled from the n nodes and t time points m times with replacement. For each node selected, I chose one of its neighbors at random, then determined the value of each nodes$'$ observed variable at the randomly generated time t. In this way I could create a sample from the larger table without ever actually creating it. This sped up the process significantly. Creating this sample from the larger table is $O(m)$, but the amount of time it takes to run doesn't increase with network size. This method can also be combined with stochastic gradient descent [5] to find the regression coefficients which maximize the likelihood of the data. Instead of taking hours to run, this method takes only minutes while using much less

[5] Gradient descent is an optimization method in which steps are taken against the gradient of a function in order to find an optimum. Stochastic gradient descent does this optimization method with random samples of the data, which is useful for large datasets.

memory. Using this technique allowed for analysis of much larger networks. Other solutions to the computational difficulties exist, but I found that this one allowed me to grow my networks to a suitable size.

At this point I have described all the simulation techniques I used during this problem. I now move on to a discussion of controlling homophily by including community membership.

4 Proof Of Concept

The basic idea behind using community to control for homophily is that nodes that have clustered together tend to have the same underlying latent variables. If communities of nodes have the same underlying latent variables, then inserting each nodes' community into the regression should control for homophily. This idea has buried within it two critical assumptions.

1. Inserting the true latent variable into the analysis should control for homophily.
2. Communities align with latent variables

I will now check these assumptions.

4.1 Latent Variable Removes Predictive Power of Neighbor Nodes In Networks With No Contagion

The end goal is to be able to determine if there is contagion present in a network or not. Therefore one would like to run a regression with node i's observed variable as the response variable and its neighbors' observed variables as the explanatory variables. Inserting the true latent variables into this regression should remove the predictive power of the neighbor nodes' observed variables in the case where there is homophily and no contagion. Since all of my variables are binary, this regression is logisitc, and takes the form:

$$ln(odds(Y_{node} = 1)) = \beta_0 + \beta_1 * Y_{neighbor} + \beta_2 * X_{node} + \beta_3 * X_{neighbor} + e_{Y_{node}}$$

Each node will thus become many different observations. For each time step each node will have several observations, with one for each of its neighbors. If the method I am investigating works at all, the coefficient β_1 should be significant only if the network has contagion. Furthermore, in the network with no contagion the β_1 should become significant if the latent variables are left out of the regression. To test this, I simulated two networks. The first had homophily only, while the second had homophily and contagion.

1. First I checked to see if including the latent variables of the node and its neighbor in the regression did not make the β_1 insignificant in the network with contagion. The regression output is displayed below. As expected the observed characteristic of node j is still useful in explaining node i's observed variable.

```
Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.4529719  0.0124666 -36.335   <2e-16 ***
neighborsObserved   0.3160281  0.0120499  26.227   <2e-16 ***
nodeLatent          0.0005576  0.0121210   0.046    0.963
neighborsLatent     0.1864797  0.0121221  15.383   <2e-16 ***
```

2. I then began working with the network with no contagion. I ran a regression using just the observed variable for node j to predict the observed variable of node i. Note that the observed variable of node j is highly significant at any reasonable significance level. This indicates that contagion and homophily are confounded in this network.

```
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -0.029526   0.008619  -3.426 0.000613 ***
NeighborsObserved -0.051050   0.012358  -4.131 3.61e-05 ***
```

3. Finally I ran the regression with the latent variables included for the network with no contagion. The results are displayed below. Including the latent variables removes the significance of the neighbors' observed variables and directly accounts for the homophily in the network, as was expected.

```
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -0.79630    0.01267 -62.825  < 2e-16 ***
NeighborsObserved -0.01437    0.01407  -1.021  0.30702
NodeLatent        -0.04045    0.01407  -2.874  0.00405 **
NeighborsLatent    1.46704    0.01322 110.945  < 2e-16 ***
```

Much of the significance is placed on the neighbor$'$s latent variable rather than the node$'$s latent variable. This is random: the opposite could just as easily been true. This is because the true variables are very highly correlated. In any case, inserting the true latent variable into the analysis controls for homophily.

4.2 Communities Align With Latent Variables

If the community detection is consistent, i.e the detected community is the same as the true community, then these communities should line up with the latent variables, and therefore should control for homophily. Therefore it is worth considering when community detection is consistent. Yunpeng Zhao, Elizaveta Levina and Ji Zhu wrote a paper on this topic in 2012 entitled "Consistency of Community Detection In Networks Under Degree-Corrected Stochastic Block Models" [8]. They show that asymptotically community detection will be consistent if :

1. the community detection function is well behaved

2. under the stochastic block model, which states that the $E[A_{ij}] = P_{ij}$, with P_{ij} being the probability of a connection between i and j, community detection will be asymptotically consistent when:

   (a) Links within communities are more likely than links between communities

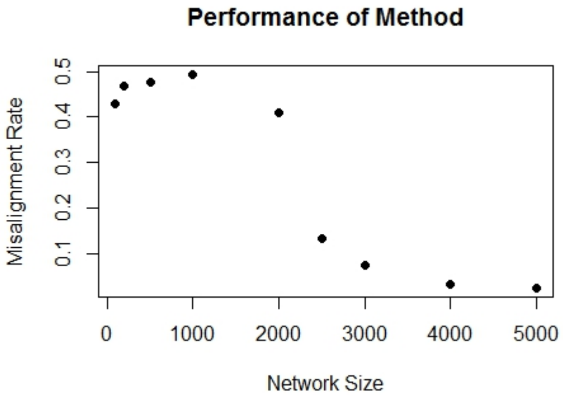
(b) expected degree/log(n)→infinity⇒ strongly consistent. A network is strongly consistent if every node in the community ends up in its true community.

(c) expected degree→infinity⇒ weakly consistent. A network is weakly consistent if the proportion of community detection errors becomes less than an arbitrary ϵ asymptotically [8].

In order to test if consistent community detection in my networks resulted in a match up with the latent variables I created networks with homophily, no contagion, and a probability of a connection between like nodes just .01 more likely than the probability of a connection between dissimilar nodes. I then grew my networks to see if community detection eventually became consistent. I grew my networks in three ways:

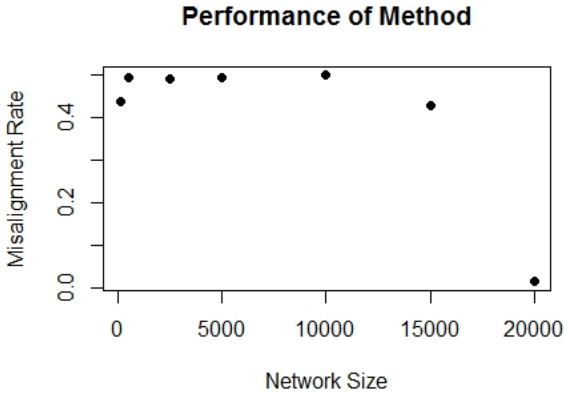
1. The probability of an edge is kept fixed, and the degree of the node grows with the size of the network.
2. The average degree of the network grows with log(n), and the probability of a connection between nodes is multiplied by $\frac{log(n)}{n}$
3. The average degree of the network is kept constant and the probability is multiplied by $\frac{1}{n}$.

Community detection is likely only consistent over the first two conditions. To test this I grew my networks until the rate of misalignment dropped off significantly. The rate of misalignment is the percent of nodes which are not assigned to the community in which the majority of the nodes share its latent variable. This drop off can be seen in this graph, which was created using the first condition. With a network size of 5000 nodes the communities align very with the underlying latent variables. The table below shows how each nodes' latent variable matches up with its detected community.



	Community	
Latents	1	2
0	5	2470
1	2418	7

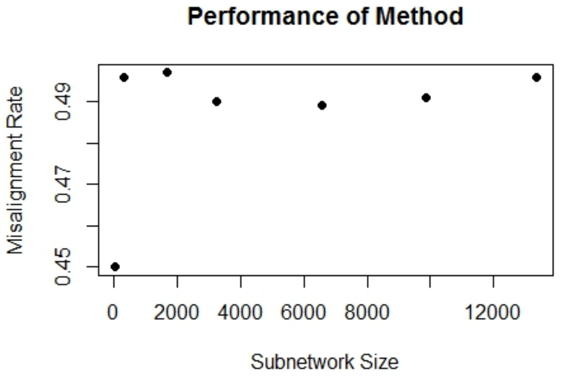
The misalignment rate has dropped to 1.4%.
For the second condition the misalignment rate drops off similarly, although



it takes quite a bit longer. At a network size of 20,000 nodes the community detection works very well:

	Community	
Latents	1	2
0	9888	57
1	51	9954

But for the third condition the community detection doesn't work well. At the largest size I was able to grow the network (20,000 nodes) the community detection wasn't lining up with the true latent variables at all:



	Community	
Latents	1	2
0	1323	2683
1	2550	1201

Therefore community detection is significant and lines up with the latent variables asymptotically when the networks are grown such that the expected degree of a node grows with log(n) or n.

5 Testing If Community Controls For Homophily

I tested if adding the community membership controls for homophily. I tested the method based on the three conditions for growing the network described above.

1. I first tested the condition where the expected degree of a node in the network grew with n. This is the condition in which each node has the most information. I used a network of 5000 nodes with homophily but no contagion. As expected, when the detected communities weren't included in the regression the neighbors' observed variables were significant:

   ```
   Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
   (Intercept)         -0.15690    0.02025  -7.746 9.48e-15 ***
   neighborObserved     0.35499    0.02840  12.499  < 2e-16 ***
   ```

   Next I included the detected communities:

   ```
   Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
   (Intercept)          2.58945    0.06756  38.327   <2e-16 ***
   neighborObserved     0.02589    0.03390   0.764    0.445
   nodeCommunity       -1.73332    0.03564 -48.641   <2e-16 ***
   neighborCommunity    0.01286    0.03819   0.337    0.736
   ```

   Including community controlled for the homophily: The neighbors' observed variable is no longer significant. Therefore the method worked.

2. Next I tested the condition where the expected degree of a node in the network grew with $\log(n)$. I used a network of 20,000 nodes with homophily but no contagion. As expected, when the detected communities weren't included in the regression the neighbor's observed variables were significant:

   ```
   Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
   (Intercept)        -0.29719    0.02022  -14.70   <2e-16 ***
   neighborObserved    0.60211    0.02861   21.05   <2e-16 ***
   ```

   Again, I next included the community variables.

   ```
   Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
   (Intercept)          2.44566    0.06139  39.840   <2e-16 ***
   neighborObserved     0.03937    0.03344   1.177    0.239
   nodeCommunity        -1.7366    0.07731 -22.463   <2e-16 ***
   neighborCommunity    0.09350    0.07843   1.192    0.233
   ```

   Including community controlled for the homophily: The neighbors' observed variable is no longer significant. Therefore the method worked.

3. Finally I tested the condition where the expected degree of a node in the network remained constant. Recall that community detection wasn't consistent under this condition. Like before when the detected communities weren't included in the regression the neighbor's observed variables were significant:

```
Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.18225    0.02040  -8.935   <2e-16 ***
neighborObserved    0.49630    0.02853  17.397   <2e-16 ***
```

However, adding community to the regression doesn't control for the homophily:

```
Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         -0.09247    0.05189  -1.782   0.0747 .
neighborObserved     0.49554    0.02853  17.368   <2e-16 ***
nodeCommunity       -0.01988    0.03671  -0.542   0.5881
neighborCommunity   -0.03957    0.03671  -1.078   0.2810
```

Since the community detection wasn't consistent and didn't match the latent variables the method didn't work.

6 A Note on Community Misspecification

The analysis so far has assumed knowledge of the correct number of communities. In real-world applications this may not always be the case. I will now take a look at what happens to the method when the number of communities is misspecified.

6.1 True Number of Communities is Less than Believed

One case of community misspecification occurs when there are fewer communities in a network than a user believes. To test how the method worked when the number of communities specified was not equal to the number of true communities in the network I simulated a network with 5000 nodes as described above, but asked my community detection algorithm to find 3 communities instead of 2. The algorithm found two large communities and one small community of 12 nodes. I then included community membership in the regression as defined above, and found that in this case community membership still controlled for homophily, even though the number of communities was misspecified.

6.2 True Number of Communities is Greater Than Believed

The other case of community misspecification occurs when there are more communities in a network than a user believes. In order to test this case I created a new simulation which has three states of observed and latent variables. Each variable can be high, middle, or low. Each latent variable has an equal probability of being high, middle, or low. I then chose the observed variables by having them align with their corresponding latent variable with probability .7 as before. With probability .3 I had them take on one of the other two states with a .5 probability of each. I then evolved the network as before. When I ran the community detection algorithm I told it to find two communities instead of the three that existed in the network. It correctly identified one community but combined the other two into one large community.

In order to determine if the method works in this case I ran two multinomial logistic regressions [7]. The first looked much like the previous regression models: I regressed each node i's observed variable at each time $\frac{t}{s}$ against each of its neighbors j's observed variable in addition to both nodes' detected community. I then ran a second multinomial logistic regression. This one excluded j's observed variable. Next I subtracted the two models' deviances. This result was 133.98. Finally I checked to see if this difference was significant. If it was, then j's observed variable is useful to the model and community doesn't control for homophily. The p-value I got was 1.07×10^{-20}. Therefore community doesn't control for homophily when the true number of communities is greater than one expects.

7 Conclusion

Through simulation I was able to show that if the probability of a connection between nodes remains constant as a network grows then homophily can be controlled for by including community membership in the regression equation. The method also works if the expected degree of the nodes grows with the log of the network size. The method didnt work when the expected degree of a network remained constant as the size of the network grew. This is as expected. The probability of a connection between nodes drops towards zero quickly when the expected degree is kept constant, and thus the nodes don't contain enough information for consistent community detection, and thus community can't be used to control for homophily.

This method can be used to determine if there is causation in a network. However, it requires consistent community detection. If community detection is consistent using community membership to control for homophily should be successful. Determining if community detection is consistent is still an open problem. Therefore determining when one can tell this method is working without already knowing the true communities could be an area of further study.

References

[1] Carter T. Butts, Mark S. Handcock, and David R. Hunter. *network: Classes for Relational Data.* Irvine, CA, March 15, 2013. R package version 1.7.2.

[2] Nicholas A. Christakis and James H. Fowler. The spread of obesity in a large social network over 32 years. *The New England Journal of Medicine*, 357:357 370–379, July 2007.

[3] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[4] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.

[5] Cosma Rohilla Shalizi and Andrew C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological Methods and Research*, 40:211–239, 2011.

[6] Greg Ver Steeg and Aram Galstyan. Statistical tests for contagion in observational social network studies.

[7] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[8] Yunpeng Zhao, Elizaveta Levina, and Zhu Ji. Consistency of community detection in networks under degree-corrected stochastic block models. *The Annals of Statistics*, 40, 2012.