Multidimensional Diffusion Processes in Dynamic Online Networks

David Easley* Eleonora Patacchini† Christopher Rojas‡

Abstract

We develop a dynamic matched sample estimation algorithm to distinguish peer influence and homophily effects on item adoption decisions in dynamic networks, with numerous items diffusing simultaneously. We infer preferences using a machine learning algorithm applied to previous adoption decisions, and we match agents using those inferred preferences. We show that ignoring previous adoption decisions leads to significantly overestimating the role of peer influence in the diffusion of information, mistakenly confounding influence-based contagion with diffusion driven by common preferences. Our matching-on-preferences algorithm with machine learning reduces the relative effect of peer influence on item adoption decisions in this network significantly more than matching on earlier adoption decisions, as well other observable characteristics. We also show significant and intuitive heterogeneity in the relative effect of peer influence.

KEYWORDS: machine learning, dynamic networks, matching algorithms

*Cornell University
†Cornell University
‡Cornell University

1 Introduction

In recent years, social scientists have become increasingly interested in the datasets generated by people interacting in online networks. When studying these networks, an important issue is to estimate whether and how individuals' decisions are affected by the decisions of their peers, a concept known as peer influence. Determining the impact of peer influence on the tendency of two individuals to behave similarly is challenging primarily due to social selection. Social selection, otherwise known as sorting or homophily, is the tendency of individuals to form and maintain links with those who are already like them [17], i.e. who have similar preferences. Due to social selection, individuals who are linked are already more similar than those who are not linked, and that similarity could be self-reinforcing even without the link. Hence, when we view behaviors or information spreading between connected individuals, we need to account for similar preferences when trying to estimate the impact of peer influence on contagion.

In this paper, we present a statistical framework for estimating the effect of peer influence on item (aka product) adoption decisions in large dynamic online networks. Our paper builds on a small literature in economics and computer science which uses matched sampling to estimate causal peer influence effects on product adoption. Matched sampling is a non-parametric method to control for confounding factors and to overcome selection bias by comparing treated and non-treated observations that have similar values of the matching covariates ([21]). The underlying assumption in this technique is that the treatment and the reaction to treatment should be independent conditional on the observable characteristics used in the matching process; i.e., that those characteristics are proxies for the drivers of social selection. There are two distinctive features of our method with respect to the existing studies ([4], [22]). [4] adapt propensity score matching for use in a dynamic online network to study the diffusion of a single item which is adopted by a large number of agents. In contrast to [4], our method can be used to study high-dimensional diffusion processes. Online social networks typically feature diffusion of a multitude of items at any given time, and an individual can adopt as many items as she would like. [22] study the effect of peer influence on the adoption of a large number of items, but they apply their method to a static online network, and they estimate peer influence for a single time-period. In our model, at each time period, each individual can be exposed to treatments on a large number of items (as many as the number of items which the agent has not adopted yet), and the effect of peer influence may vary depending on the item, the agent, her local network, and the time period. In order to deal with the individual- and time-specific nature of peer

influence, we develop a novel dynamic, distance-based matching procedure.

The key novelty of our method is that we use a machine learning algorithm to predict agents' adoptions by inferring a latent factor model of their preference types ([11]). Previous research has used demographic as well as behavioral variables to match agents, but has not matched agents with similar past adoption behaviors. In their application to the yahoo IM network, [4] include a large number of demographic and behavioral characteristics for matching. Demographic variables are not always available for online data, and even when they are, these variables alone are unlikely to adequately control for homophily. This is important because to the extent that the effects of homophily are underestimated, the impact of peer influence is over-estimated. Our method adopts the fundamental insight from collaborative filtering that users with similar past adoption behaviors have similar preferences. In addition, utilizing past adoption behaviors takes into account the importance of past contagion in shaping the evolution of preference types. However, directly utilizing past adoption behaviors to match agents may be ineffective because of the large number of items and the sparsity of individual adoption. [22] overcome this problem by matching dyads (pairs of agents) based on having an approximately equal measure of past adoption similarity for each dyad. This approach requires both of the agents in a dyad to have enough past adoption behaviors in order to infer their preferences. Moreover, just because two agents u, v are each equally similar to a third agent w does not imply that u and v adopted the same types of items in the past. Our method will instead first embed past adoption behaviors into a low-dimensional preference type vector, and then match agents with similar predicted vectors. The advantage of our method is that it ensures we match agents who adopted similar items in the past.

We illustrate our method using data from Github, the popular website for collaborative software development. On GitHub, users can create open-source software projects known as *repositories*, or repos for short, and push code to them. Other users can contribute to the repo (if given permission), or just use the repo for their own purposes. GitHub also has resources to help users find and share interesting repos. In particular, GitHub includes a directed social network, similar in design to Twitter, in which users can *follow* other users to receive notifications about some of their activities on repos. GitHub users can also *star* a repo they find interesting, which is similar to liking something on Facebook, and will result in a notification on the *activity feeds* of their followers. GitHub is known as the "Facebook for programmers" and prominent tech companies such as Microsoft want to reach communities of developers with their open-source software on GitHub ([26]). At the same time, rich longitudinal data concerning user activities on public

repos and social network links can be accessed and analyzed.[1] This makes GitHub a useful environment in which to study the diffusion of knowledge about software.

Our results show that the effect of peer influence on item adoption decisions, relative to common preferences, is much smaller than it would appear to be if we do not control for past adoption behaviors. In particular, we demonstrate that using our machine learning algorithm leads to a significantly lower estimate of peer influence. We also show that the marginal influence of additional adopting peers is diminishing, and we find that the peer influence effect is strongest immediately after a peer adopts. We further examine other ways that the importance of peer influence varies with the types of agents and items. Our algorithm detects lower peer influence in contexts where past influence is likely to be important in shaping current adoption behaviors, such as for items which are similar to those an agent has adopted in the past. It also detects lower peer influence in contexts where it is easier to learn about the item without the link, such as for highly popular items. Taken as a whole, our empirical findings suggest exposure is the main pathway for peer influence on Github. That is, peer influence occurs because the links spread information that the agent would not otherwise have access to, as opposed to other possible reasons for peer influence in product adoption choices, such as information cascades ([7]) and (local) network externalities ([24]).

The overview of our paper is as follows. In Section 2, we summarize our empirical method. Section 3 provides an overview of the data, as well as preliminary evidence of assortative matching and social selection. Section 4 is where we present our results, including our evidence of identification. Section 5 concludes.

2 Empirical Method

2.1 Preference Estimation

The *adoption* decision we focus on is whether or not to star a particular repo, and we will consider an agent to have adopted a repo when they star it. Let u denote agents, i denote items, and t denote periods (months). We will refer to the agents whom u follows as her *leaders*. Our primary definition of *treatment* is that one or more of agent u's leaders have recently adopted an item i, which agent u has not yet adopted. If agent u adopts the item

[1] GitHub does not require agents to provide detailed demographic data, and most do not. This is typical of many online networks.

during period t, then we measure treatment based only on her leaders who had adopted the item as of the time that u adopted; i.e., excluding those who adopt during period t and after agent u. By doing so, we alleviate concerns related to a possible simultaneity of adoption choices (e.g. reflection, [16]). Also, we fix the time of treatment for agent u and item i so that it is either the beginning of the month, or it is the most recent time at which a leader adopted the item, whichever comes later. Lastly, we restrict the duration of peer influence by only defining adoption by a leader as treatment during period t if it occurred since the beginning of period $t-3$, and before the end of period t.[2] Note that there are many items and a separate treatment occurs for each item and each agent who follows other agents who adopt the item.

The causal quantity we seek to identify is the *relative risk of adoption*, $RR_t = p_t^{(1)}/p_t^{(0)}$, where $p_t^{(1)}$ is the probability of adopting an item during period t for a treated agent, and $p_t^{(0)}$ is the probability of adopting an item during period t when not treated, for an agent who would have been treated. [3] Of course we do not actually observe the counterfactual quantity $p_t^{(0)}$ and it must be estimated from the data. Our definition of relative peer influence is the same as the definition used in [4], allowing us to compare our results to earlier findings.

Our estimation strategy is based on dynamic, distance-based matching. The propensity score $e(X_{u,i,t}) = \Pr(T_{u,i,t} = 1|X_{u,i,t})$ is the probability that agent u is treated on item i in period t, where treatment status is denoted by $T_{u,i,t}$ and $X_{u,i,t}$ are a set of time-varying covariates that describe the agent and item. Our identification strategy relies on the standard assumptions underlying matching studies. First, we need to assume *conditional unconfoundedness*, which is the assumption that the potential adoption outcomes $(Y_{u,i,t}^{(1)}, Y_{u,i,t}^{(0)})$ are independent of treatment $T_{u,i,t}$, conditional on the covariates $X_{u,i,t}$. The second assumption is *overlap*, which requires that units have positive probability of being treated and non-treated, $e(X_{u,i,t}) \in (0,1)$.

Conditional unconfoundedness in social network analysis is challenging primarily due to social selection, which results in connected agents having more similar preferences for items than non-connected agents. In order to identify the effect of peer influence, we would like to be able to control for preference types. However, we do not observe the true preference types of

[2] In the full dataset, leader adoption happens up to 32 months before follower adoption, but about 54% of the diffusion takes place within the current and previous 3 months. We have tried using varying the duration parameter, and we find only small differences in the peer influence estimate.

[3] We are interested in the effect of treatment on the treated, which is the (unobserved) effect of withholding a treatment that has in fact been implemented.

agents, and we will instead control for a set of proxy variables intended to predict preference types. Our set of matching covariates $X_{u,i,t}$ includes a set of *baseline characteristics* which provide information about the agent's account age and activity but do not tell us information about the types of items that the agent adopted in the past (we list the baseline covariates used for matching in the appendix Table A.2). We will then include in $X_{u,i,t}$ a vector of additional covariates which are meant to summarize the types of items that the agent has adopted in the past.

Note that, unlike [4], we cannot assume that social network link formation is independent of the adoption behavior. Furthermore, our definition of treatment includes adoption by leaders during past time periods, and it is possible that past contagion can affect the evolution of the social network and adoption choices. Our peer influence effect ignores any indirect effects which may occur because of a change in the matching covariates, as a result of the change in treatment status leading to a different equilibrium in which the matching covariates are different.[4] It captures only the direct effect of the change in treatment, conditional on everything in the past.

There are many different ways in which we can add covariates that capture the types of items adopted in the past. The simplest way is to use the *adoption vector*, which is a long, sparse, binary vector which has a 1 for the i-th component if the agent u adopted the item i by the beginning of period t, and a 0 otherwise. However, the downside of this approach is that it would require matching on a huge number of covariates, and matching estimators are known to have greater finite-sample bias when the set of matching covariates becomes too large, relative to the number of observations ([1]). In our results (Section 4.1) we will show that matching directly on all past adoption choices leads to poor matching performance. Another simple approach would be to utilize an observable characteristic of the past adoptions which allows us to reduce the dimensionality of previous choices; for example, one possible approach would be to utilize the programming languages of past adoptions. Hence, we could define a language vector, which is a (much shorter) vector in which each component represents the fraction of items an agent adopted which are (primarily) written in a particular programming language. The downside of this approach is that our measure of past adoptions only measures one characteristic of the items, and there could be many other characteristics that are important to agents. In our results (Section 4.1), we will also analyze the performance of past programming languages as

[4]A more exhaustive game-theoretic model of the problem would require a complex dynamic model of product adoption choices and social network formation, and the results would be conditional on the assumptions.

a low-dimensional method to control for common preferences.

Our preferred empirical method will be a two-step method in which we first infer vectors of agent characteristics which predict adoption choices. The *adoption matrix* is a matrix with a 1 in the (u, i) position if agent u has adopted item i before the beginning of period t, and 0 otherwise. We use a well-known collaborative filtering algorithm for implicit feedback data (clicks, views, purchases, etc.), based on the weighted, regularized matrix factorization (WRMF) of the adoption matrix ([11]).[5] We factorize the adoption matrix in a given period, to obtain a latent factor vector for each agent u and for each item i. Within the context of the model underlying WRMF, the latent factor vector for agent u represents u's preferences over characteristics of items and the latent factor vector for item i represents the characteristics of item i. Agent u's estimated payoff from any item is given by the inner-product of her latent factor vector with the item's latent factor vector. The latent factor vectors are chosen by the WRMF algorithm to best represent linear preferences over characteristics and the characteristics themselves, based on a static model of past adoption choices.

In the appendix we provide a detailed overview of the statistical model underlying WRMF (Section 6.2). Note that the preferences learned by WRMF do not have a structural interpretation in our setting. The model underlying WRMF is static whereas our network is dynamic, and the WRMF model does not incorporate past contagion (which we address in a robustness check). However, we do not attempt to interpret the parameters estimated by WRMF in a meaningful way, we only use them as inputs for matching. What is important for our purpose is that the "preference" types inferred by WRMF serve as effective proxy variables (in combination with the other baseline characteristics) for the true, unobserved preference types. We will provide evidence that this is indeed the case in our results (Section 4.1).

Under the assumptions of the WRMF model, the weighted maximum-likelihood estimates of preference types will minimize the weighted (squared-Euclidean) distance between the adoption matrix and the preference representation, using a penalty function for the complexity of both the preference and characteristic representations. The posterior weighted log-likelihood equation for WRMF is derived in the appendix (Section 6.2), and the estimation of the preference types is also detailed in the appendix (Section 6.3).

The WRMF algorithm has 4 *hyper-parameters* (aka tuning parameters), which must also be estimated from the data. In order to set the hyper-parameters, we use the approach which is typical in the machine learning

[5]The WRMF algorithm is based on the singular value decomposition of the adoption matrix.

literature. We partition our data into a training set and a test set; the training data is used to infer the model for a given value of the hyper-parameters, and the test data is used to compute a statistic which measures the accuracy of the predicted preferences at out-of-sample prediction. We use random search over a set of possible hyper-parameter values, and select for our analysis the values of the hyper-parameters which optimize the value of the statistic computed on the test data. Details are provided in the appendix (Section 6.4).

2.2 Matching Strategy

We propose a dynamic, distance-based matching strategy, with exact-matching on the item, and nearest-neighbor matching based on the distance between agent characteristics. Let $||\mathbf{X}||_{\mathbf{\Sigma}} = \sqrt{\mathbf{X}'\mathbf{\Sigma}\mathbf{X}}$ be the vector norm induced by the positive definite matrix $\mathbf{\Sigma}$. We define $||\mathbf{X}_{u,i,t} - \mathbf{X}_{v,i,t}||_{\mathbf{\Sigma}}$ as the distance between the vectors $\mathbf{X}_{u,i,t}$ and $\mathbf{X}_{v,i,t}$, where $\mathbf{X}_{v,i,t}$ represents the covariate values for a potential match for agent u on item i. For the weighting matrix $\mathbf{\Sigma}$ we use the inverse variance matrix. The inverse variance matrix is estimated once per period using all of the agents included in the data for the period.[6] This weighting accounts for differences in the scale of the covariates and it places more weight on differences in covariates which do not vary much in the population than it does on ones which do vary a lot. Agents who have different covariates that others are very similar over are thus treated as significantly different; while agents who have different covariates that vary a lot in the population are treated as being more similar.

Each period, we create a matched sample of treated and untreated agent-items, for each agent and each item on which they have been treated, but not yet adopted themselves. For each agent, we first compute their M nearest neighbors, based on the past data. Then, for each item i on which an agent u is treated during t, we find the matched pair of agents u and v such that u and v have the most similar preferences (as predicted by our covariates) for i, but only agent u is treated during t. We compute the nearest-neighbor for each treated observation (agent-item), with replacement. If we cannot find a valid non-treated agent in the first M matches, then there is no match for agent u and item i, and we remove this treated observation from consideration. M

[6] In a typical application of distance-based matching to estimate the average treatment on the treated for a single treatment, one would include only the non-treated agents to estimate $\mathbf{\Sigma}$ ([23]). We cannot do this because then we would need to estimate a separate $\mathbf{\Sigma}$ for each treatment. However, since very few agents are treated for the large majority of items, it should be the case that for almost all items, $\mathbf{\Sigma}$ for all agents is approximately equal to $\mathbf{\Sigma}$ for only the non-treated agents for a particular item.

needs to be large enough so that we are can match sufficiently many treated observations. We chose a value of $M = 50$ which allowed us to match over 99.99% of the observations. The benefit of our approach is that we only need to compute the nearest neighbors of each agent once per period, and so this approach can be used for the large number of items diffusing in our data.

Next, for each item i we can compute the fractions of treated ($\hat{p}_{i,t}^{(1)}$) and non-treated ($\hat{p}_{i,t}^{(0)}$) agents who adopt the item. Finally, the estimates for all of the items in the period are combined by weighting the estimate for each item by the fraction of exposed pairs for that item. Let $\alpha_{i,t}^{(1)}$ denote the fraction of treated observations in period t which correspond to item i. We have

$$\hat{p}_t^{(1)} = \sum_i \alpha_{i,t}^{(1)} p_{i,t}^{(1)} \quad \hat{p}_t^{(0)} = \sum_i \alpha_{i,t}^{(1)} p_{i,t}^{(0)} \tag{1}$$

In our results, we report the ratio $\hat{p}_t^{(1)}/\hat{p}_t^{(0)}$. Note that $\hat{p}_t^{(1)}/\hat{p}_t^{(0)}$ simplifies to the number of treated agents who adopt an item, relative to the number of matched, non-treated agents who adopt an item. The probability of a follower adopting any given item is very small, but our statistic is uncontaminated by observations in which an agent does not adopt an item. In our results (Section 4) we will refer to the estimated treatment effect simply as n_+/n_-, ignoring the time period subscript.

Also, note that although we match pairs of agents, the statistic we use to measure peer influence relative to the effect of homophily uses only data about populations of matched pairs. That is, we do not directly compare matched agent-pairs; rather, we compare matched populations. We could instead match agents based on binning, and then directly compare binned populations, but this would require us to define bins of similar agents and items. Binning is problematic as the results may depend on the definitions of the bins and how they are filled. We view it as a benefit of our matching method that we do not need to specify bins.

Since our treated observations are not independent, we cannot rely on standard formulas to compute the asymptotic variance of our estimator. Instead, we will compute confidence intervals using bootstrap re-sampling simulations applied to the matched sample.[7] [8] We draw 100 samples from the matched pairs and compute the test statistic on each sample. We then use

[7] A similar inferential methodology has been used by [5] for the case of a single, binary treatment.

[8] [2] show that matching estimators do not satisfy the required smoothness conditions for consistency of the bootstrap. We include as a robustness check a type of bootstrap which is known to produce confidence intervals that are too large, in our setting (see appendix Section 6.6).

the 2.5 and 97.5 percentiles of the empirical distribution of the test statistic over the generated samples as the endpoints of our 95% confidence interval. A confidence interval with lower endpoint greater than 1 implies that the treatment has a significant effect.

3 Data and Descriptive Evidence

Our data is time-stamped GitHub data collected from the GitHub Archive that includes user activities on public repos and social network links created over a 32-month period, from February, 2011 to October, 2013. We provide a more detailed background on GitHub in the Appendix (Section 6.1). We estimate peer influence during the 10 months from January to October, 2013; earlier data will be used for learning preferences, observing the evolution of the social network, and observing item adoptions. We define period 1 to be January, 2013; earlier months have period less than 1, and the final month is period $T = 10$. Since we will use earlier starring behaviors to learn preferences, we will restrict our data to include only agents who have enough stars to accurately infer their preferences. In order to do this, in a given period $t \geq 1$ we will estimate peer influence effects on the adoption behaviors of agents who have at least 10 stars prior to the start of the period.[9] Our final sample consists of 163,458 unique agents for whom we are able to estimate social influence in at least one month from January to October, 2013. These agents create a total of 10,036,987 stars of 855,317 unique repos, and they create a total of 1,662,393 links to 363,598 unique leaders.[10] Summary statistics for our sample are provided in the Appendix (Table A.1). On average, an agent stars 61 repos, and follows 10 leaders. A repo receives on average 12 stars, with a large dispersion about this mean value. We will also use the (primary) programming languages of repos in our analysis, to provide a benchmark for how well we can match on preferences using only the features available in the data, without collaborative filtering.[11] We graph the composition of stars by language over time in the Appendix (Figure A.1) According to this measure, the most popular language is Javascript, followed by Ruby, and the popularity of the languages is fairly stable over time.

[9]We exclude inactive agents, that is those that do not take any action for six subsequent months.

[10]The agents in our sample are responsible for approximately 86% of the stars in our dataset, and they create 56% of the links.

[11]If a repo contains multiple programming languages, then GitHub sums up the file size associated with each programming language, and the largest one is designated as the primary programming language.

Figure 1 (a) plots the fraction of adopters and non-adopters of a repo during period T, as a function of the number of leaders in their local network who have adopted the repo (since the beginning of period $T-3$). Figure 1 (b) depicts the probability of adoption as a function of the number of leaders who have adopted the repo (since the beginning of $T-3$). In Figure 1 (a), we see that agents who star a repo during T tend to follow significantly more agents who also starred that repo recently. In Figure 1 (b), we see that the probability of adoption increases significantly when 1 or more agents whom an agent follows have starred the repo recently. Interestingly, the effect is non-linear. The probability of adoption decreases when more than 5 leaders whom an agent follows have adopted the repo, though it remains significantly higher than when 0 leaders have adopted.

In Figure 1 (c) we implement a version of the well known "shuffle test" of social influence ([3]), modified for multi-dimensional diffusion and for a directed network. We randomly reassigned all of the adoption times for all items, so that the adoption frequency over time remained constant. We then compared the fraction of observed dyadic differences in adoption times (including only the observations for which the follower adopts after the leader) for the original network relative to the network with the shuffled adoption times. In comparison to the randomly shuffled adoption times, we find that followers are about 1,788% more likely to adopt an item within the same week as the leader, and that the temporal interdependence persists; the follower is about 205% more likely to have adopted the item during the first 16 weeks with the actual adoption times.

Figure 1 (a-c) shows us that connected agents tend to adopt more similar items than non-connected agents, and that connected agents who adopt the same items do so closer together in time than non-connected agents. Both of these findings are consistent with peer influence. However, this evidence is not sufficient to conclude that peer influence affects followers' adoption decisions. The choice of which repos to adopt and the choice of who to follow are endogenous. Time-varying homophily could cause connected agents to adopt similar items and to adopt them closer together. In Figure 1 (d), we compute the cosine similarity of the adoption vectors for pairs of agents who form a link and for random pairs of agents who do not form a link. An adoption vector is a vector with a 1 in the i-th position if agent u has adopted item i, and a 0 otherwise. For the agents who do form a link, we compute the similarity of their adoption vectors at 30-day intervals leading

up to and after the moment of link formation.[12] [13] For the agents who do not form a link, we fix an arbitrary date of 5/1/2013, and compute the cosine similarity during 30 day intervals before and after this date. We see clearly that the adoption behaviors of agents who become linked are far more similar than the behaviors of agents who do not form a link. Moreover, the agents who form a link were already becoming more similar during the time before they formed the link. Even though they continue to grow more similar after link formation, that additional growth may be caused by homophily rather than influence. In order to determine the impact of peer influence on adoption decisions, we need to use our empirical methodology which allows us to account for the impact of common preferences on adoption decisions for a multidimensional diffusion process.

4 Results

4.1 Baseline

We will now show that our method is effective at controlling for common preferences. First, we use simulations to analyze the finite-sample bias of our matching algorithm, under the assumption that homophilous adoption choices are determined by the WRMF preference types with the optimal hyper-parameter values. We simulate the adoption choices during the first period by assuming that each observed adoption is based on one of three possible mechanisms:

1. **Influence**: An agent adopts the item most-recently adopted by one of her leaders, which she has not already adopted.

2. **Homphily**: An agent adopts her most preferred but not yet adopted item, as predicted by WRMF.

3. **External Exposure**: An agent adopts the most popular item (based on adoptions since the beginning of period -2) which she has not already adopted.

[12] We include all of the dyads in our data for which the follower is one of the valid agents as defined above, and which we can observe for 180 days before and after the link forms.

[13] Cosine similarity between two non-zero vectors v_1, v_2 is defined as

$$\cos(v_1, v_2) = \frac{v_1 \cdot v_2}{||v_1||_2||v_2||_2}.$$

In the simulations we randomly assign 90% of the adoptions during the first period to be determined by the homophily rule, 5% to be determined by the influence rule, and 5% to be determined by the external exposure rule.[14] We then replace the actual adoption choice in the data with the adoption choice generated by the randomly selected rule, while keeping the timing the same. We also estimate the nearest-neighbor for each agent, based on our preferred set of matching covariates which includes WRMF, prior to the beginning of period 1. We are interested in the ability of our matching method to estimate the counterfactual number of agents who adopt the item; i.e., the number of agents who adopt the item when not treated, for agents who would have been treated. In our simulation, the true counterfactual is given by the number of treated items adopted by the treated agent due to the homophily rule (10,406) plus the number of treated items adopted by the treated agent due to the external exposure rule (2,449). The estimated counterfactual is given by the number of matched, non-treated agents who adopt the item (10,907). Thus, we see that the estimated counterfactual performs fairly well; it generates approximately 84.8% as many adoptions as the true counterfactual. Table 1 shows the confidence intervals for the number of adoptions from homophily and external exposure, as well as from the estimated counterfactual, based on bootstrap resampling simulations.

Of course, as we stated earlier, the true underlying preference types are different from WRMF and so simulations alone can only establish that our matching algorithm has low finite-sample bias when we observe preference types. We still need to establish that our matching covariates are effective proxies for the true unobserved preference types. We do this by providing evidence that our covariates used to match agents in the first period are related to adoption choices during the first period. Figure 2 (a) provides an overview of the most-preferred items in period 1, as predicted by WRMF with the optimal hyper-parameter values, for each agent and each item, based on adoption behaviors prior to the beginning of period 1. Figure 2 (a) shows that the preference types learned by WRMF are reasonable, in the sense that agents are more likely to adopt items for which they had a higher predicted preference. The correlation between predicted ranking and adoption probability is negative and highly significant ($-0.013, p < 0.001$). Figure 2 (b) provides an overview of adoption similarity during period 1 for an agent and her nearest neighbors. In Figure 2 (b), we use the adoption vector during period 1, which is a vector with a 1 in the i-th position if the

[14] In ([22]), the authors use a simulation with 90% homophily, 10% influence. We modify their simulation to include all 3 mechanisms. We may want to add additional tests with different fractions of adoptions determined by each mechanism.

agent adopts i during period 1, and a 0 otherwise. We see that the cosine similarity of items adopted during period 1 is higher for pairs of agents who are lower-ranked neighbors than for pairs of agents who are higher-ranked neighbors. Once again the correlation between distance-ranking and adoption similarity is negative and highly significant ($-0.012, p < 0.001$). Figures 2 (a) and 2 (b) provide evidence that our covariates are effective proxies for the true adoption preferences during the first period. The low p-values for the correlation coefficients indicate a real relationship between similarity in terms of our matching covariates, and revealed adoption preferences.[15]

Finally, to demonstrate that WRMF improves our ability to reduce homophily bias in the estimation of peer influence, we will compare several different matching algorithms. Homophily generally leads to an upward bias in the estimated peer influence effect, and our goal is to compare the ability of the different algorithms to attenuate this bias. We consider the following five algorithms:

- **Random matching**. The matching process selects random, non-treated agents as controls for each treated observation.[16] Since links are not created randomly, the use of random matching cannot control for homophily.

- **Matching based on baseline agent's characteristics (Baseline)**. Nearest-neighbor matching using the covariates listed in the appendix Table A.1. Importantly, the covariates do not measure the types of items adopted in the past by agents. We include the nearest-neighbor matching on baseline characteristics in order to understand the bias created by excluding past adoption behaviors.

- **Baseline+Adoptions** Nearest-neighbor matching using the baseline variables plus the adoption vectors for past adoptions. We include this approach to demonstrate that matching performs poorly when we do not reduce the dimensionality of the adoption vectors.

- **Baseline+Languages**. Nearest-neighbor matching using the baseline variables plus the programming languages of past adoptions. We measure agents' preferences by using the programming languages of their

[15]We can redo the regressions, excluding any observations on which an agent has been treated, and the results are virtually identical.

[16]In order to reduce variance, we will use the adoption choice averaged over multiple random neighbors, for each treated observation, as our counterfactual. For each treated observation, we start with $M = 50$ random neighbors, and then remove any neighbors who have already adopted the item or are themselves treated on the item. We average over the remaining neighbors.

adopted items. We encode the top-100 most popular programming languages using a one-hot scheme, and measure the fraction of an agent's adopted items in each language. [17] We include this approach to investigate whether our machine learning algorithm reduces homophily bias more than any directly observable features of the data.

- **Baseline+WRMF**. Nearest-neighbor matching using the baseline variables plus the (normalized) WRMF preference type vectors.[18]

We present the results for period $t = 1$ in Table 2, which show that the use of WRMF preferences results in the lowest estimate of peer influence, significantly outperforming the other candidate sets of matching variables.[19] With random matching during the first period, the fraction of treated adopters is 201.84 times greater than the fraction of non-treated adopters. When we compare these results to our nearest-neighbor matching algorithm based on WRMF, we see that the estimated peer influence effect is dramatically reduced. During the first period the fraction of treated adopters is only 2.05 times greater than the fraction of non-treated adopters under matched sampling with WRMF. The random matching result suggests that failing to account for selection into treatment results in an overestimation of the treatment effect by 9,744%. The language vectors result in the second lowest estimate, but it is still 42.42% higher than with WRMF. The inclusion of the adoption vectors actually causes the matching estimator to perform worse than the baseline matching approach which does not control at all for the types of items agent adopt in the past.

In Figure 3 we depict the total amount of contagion for treated agents, and the total amount of contagion which can be attributed to homophily; i.e., the total amount of adoption of items by (WRMF) matched, non-treated agents, versus the total amount of adoption by treated agents. It appears that homophily can explain over half of the contagion that we see in the data (50.47%), which is similar to what was found in [4]. The estimated homophily without collaborative filtering would be much lower (0.8% with Random, 5.8% with Baseline+Adoption Vectors, 20.69% with Baseline, 32.30% with Baseline+Languages), thus producing an overestimation of the peer influence effect on contagion.

[17] We use the same dimensionality as the preference types inferred by WRMF.

[18] The magnitude of the WRMF preference type vectors is proportional to the number of items adopted by the agent. Therefore, we normalize the WRMF preference type vectors, so they have unit length.

[19] We only include $t = 1$ for brevity; however, the results are similar in all of the time periods.

4.2 Treatment Intensity

Next we turn our focus to different intensities of influence, based on the number of adopting leaders. Figure 4 (a-b) shows a comparison of WRMF-based matching to random matching over time and for different numbers of adopting leaders. With random matching, the estimated treatment effect of having 2 and 3 (or more) adopting leaders is higher than that of having a single adopting leader (Figure 4 a), and the results exhibit a downward time-trend. Matched sampling with WRMF results in peer influence estimates that are significantly lower, and the downward time-trend disappears (Figure 4 b). The marginal influence of adopting leaders is actually diminishing, and the treatment level of 3 (or more) adopting leaders does not have a statistically significant peer influence effect in most of the time periods. One reason why random matching incorrectly suggests that marginal influence is increasing may be that it fails to control for the greater homophily that exists with larger groups of adopting leaders (inset in Figure 4 b).[20]

We can also examine how peer influence varies based on the timing of adoption decisions, by defining treatment based on the most recent period during which an agent's leader adopted an item. Let $\Delta t_{u,i}$ denote the number of periods between the current period (t), and the most recent period that one of agent u's leaders adopted the item. Figure 4 (c-d) shows a comparison of random matching to WRMF-based matching over time and for items which an agent's leader(s) most-recently adopted during the current period ($\Delta t_{u,i} = 0$), or during an earlier period ($0 < \Delta t_{u,i} \leq 3$). With both random matching (Figure 4 c) and WRMF-based matching (Figure 4 d), the estimated treatment effect is larger when $\Delta t_{u,i} = 0$, and smaller when $\Delta t_{u,i} < 0$. In addition, random matching overestimates peer influence by more when $\Delta t_{u,i} = 0$ (17,000% in period 1) than when $\Delta t_{u,i} < 0$ (6,500% in period 1). This may be because there is greater homophily between agents who adopt closer together in time (inset in Figure 4 d). Overall, the results show that both homophily and peer influence are strongest immediately after a leader adopts an item.

4.3 Treatment Heterogeneity

We next use nearest-neighbor matching with WRMF to evaluate the treatment effect (of at least 1 adopting leader versus none) under various types of heterogeneity. To do this, we compare the fractions of treated and non-

[20]We measure the homophily between a pair of agents based on the cosine similarity of their normalized attributes, which includes their learned preference types. See the Appendix (Table A.3) for the full list of attributes used to measure homophily.

treated adopters for observations with different types of followers, leaders and/or items.

In Figure 5 (a) we examine whether or not peer influence plays a greater role in adoption of items for agents who have adopted similar items in the past. We estimate similarity by using the cosine similarity between WRMF preference and characteristic type vectors. We group observations into those for which a treated agent had similarity above or below the average similarity for her treated items in the period.[21] In Figure 5 (b) we analyze heterogeneity by agents' levels of experience, measured using the total number of adopted items. We compare observations in which the follower has adopted at most 50 items versus observations in which the follower has adopted at least 75 items, measured as of the beginning of each period.

The results in Figures 5 (a, b) are consistent with our algorithm successfully controlling for common preferences in instances when past influence may affect the evolution of preferences. In 5 (a) we see that adoption of items which are more similar to those an agent has adopted in the past tends to be driven less by peer influence (although the difference is not statistically significant in periods 5, 9 and 10). This result is in fact the opposite of what was found in [4], in which it was found that agents with a greater interest in news content are more susceptible to peer influence on an item that provides news content. In 5 (b) we see that agents who have adopted fewer items overall are more susceptible to influence than agents who have adopted more items. The difference between our methodology and the one adopted by [4] is that our algorithm takes into account past peer influence in shaping the evolution of preferences. Agents with high preference similarity to an item, as measured by WRMF, may have leaders who adopted more similar items in the past, so that past influence already caused them to adopt these types of items before. And agents who have adopted more items overall are also more likely to have been already exposed to peer influence on similar items, which affects the evolution of their preferences.

Previous research ([4]) suggests that homophily is greater amongst early adopters of an item, but has not found that peer influence varies during an item's life cycle. We investigate how peer influence and homophily vary for newer versus older items in our data. To do so, we define an item's age in a given period as the number of days since the item first appeared in the data, computed at the end of the period. We compare items which first appeared at most 180 days ago against items which first appeared at least 360 days ago. We do indeed find that adoption of newer items is driven significantly less by peer influence (Figure 5 c), and we find that greater homophily exists

[21] So we only include agents treated on at least two items.

amongst the early adopters of an item (inset in Figure 5 c).[22]

We have already seen that peer influence plays a smaller role in the adoption of items more similar to an agent's preference type, which is consistent with exposure as the primary reason for peer influence. On GitHub there is a huge number of items diffusing simultaneously, making it difficult for an agent to discover an item less similar to her previously adopted items, without contagion. We look for additional evidence for exposure by examining heterogeneity with respect to popularity, as well as the number of items adopted by an agent's leaders. As mentioned earlier, popular items may be easier for agents with similar interests to learn about, because more popular items have avenues other than contagion through which agents can learn about them. For example, GitHub (similar to other online platforms) has a special page listing recently popular repos, which are known as trending repos. Popular repos may also be easier to find outside of GitHub, on social media and independent websites. We measure the popularity of an item in period t based on whether or not the item has (log+1) number of adopters above or below the median for all observations during the period. We find that peer influence is significantly lower for more popular items than for less popular items (Figure 5 d). It may also be easier for agents to learn about the items adopted by popular agents, even without following them, because of the availability of alternative channels. In addition to listing trending repos on a special page, GitHub also highlights trending programmers, defined as those agents who receive many new followers. Those agents may be well-known on other platforms popular with programmers too, such as Twitter or Stack Overflow. Previous research into Twitter has shown that popular agents are not necessarily more influential in terms of spawning retweets or mentions ([6]), but this research does not estimate causal peer influence effects, which we now do. For each observation in a period, we define popular leaders to be those whose average number of followers is above the median for all observations in the period. We find that peer influence is significantly lower for items adopted by more popular leaders than for items adopted by less popular leaders (Figure 5 e).

In Figure 5 (f) we compare the behavior of followers in each period for whom the average (log+1) number of adoptions by their leaders, as of the end of the previous period, is either above or below the median for all observations that period. The results show that agents who follows leaders that adopt many items are less influencible than agents who follow leaders that adopt fewer items. On GitHub, followers get a notification for each star by one of their leaders. When an agent follows leaders who are creating a large number

[22]The results for peer influence are significant in every period, including period 6.

of stars, it seems reasonable to hypothesize that she is proportionally less likely to pay attention to each star created by a leader, because the higher volume of information being transmitted to the follower makes it more costly for her to find each individual item.[23]

We believe that our findings with respect to the popularity of items and leaders, as well as the number of items adopted by an agent's leader(s), support the hypothesis that exposure is the main pathway underlying peer influence. It seems highly plausible that for each of these types of heterogeneity, the heterogeneous covariate affects the observed adoption outcomes primarily through its effect on whether an agent is aware of the item at all. Hence, the main reason that we observe peer influence is because a leader's adoption of an item increases the likelihood that the follower is exposed to an item to which she would not otherwise have been exposed. For items to which an agent was already likely to be exposed, even without leader adoption, the peer influence effect is smaller.

4.4 Robustness Checks

Finally, we provide a couple of robustness checks, which are summarized in greater detail in the appendix. First, we consider an extension of the WRMF algorithm which incorporates past exposure, the details of which are provided in the Appendix (Section 6.5). The results for the first period are in Table 3, which shows that the extension reduces homophily bias less than our main specification. Second, we consider an alternative bootstrap technique which accounts for two-way random effects, which is discussed in greater detail in the Appendix (Section 6.6). The results for the first period are in Table 4, which shows that the extension does not change our main result.

5 Conclusion

We present a novel statistical framework to estimate peer influence effects for a multidimensional diffusion process that uses machine learning to control for common preferences that drive adoption decisions. Using this technique allows us to account more effectively for the drivers of social selection. We show that ignoring common preferences leads to significantly overestimating the role of peer influence in the diffusion of knowledge on GitHub, mistakenly identifying homophily-driven diffusion as influence-based contagion. In

[23]It has been demonstrated in lab experiments that, when an individual's peers are each an equal source of information, then as the size of an individual's peer group increases, each peer has less of an impact ([13]).

particular, we show that our preference-based matching algorithm is able to reduce the estimate of peer influence effects significantly more than matching with other observable variables in the data. We also find significant heterogeneity in peer influence for different types of items, leaders, and followers. Peer influence is lower for agents who have been influenced on similar items in the past, and it is lower for agents who are more likely to be exposed to an item even in the absence of peer adoption. Our findings thus point to the importance of exposure for diffusion processes in online networks. Although we cannot be certain that we have captured all of the intricacies driving selection bias, our approach provides a significant improvement in the ability to sort out homophily and influence in a social network using observational data.

References

[1] Abadie, A., & Imbens, G. W. (2006). Large sample properties of matching estimators for average treatment effects. Econometrica, 74(1), 235-267.

[2] Abadie, A., & Imbens, G. W. (2008). On the failure of the bootstrap for matching estimators. Econometrica, 76(6), 1537-1557.

[3] Anagnostopoulos, A., Kumar, R., & Mahdian, M. (2008). Influence and correlation in social networks. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 7-15). ACM.

[4] Aral, S., Muchnik, L., & Sundararajan, A. (2009). Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. Proceedings of the National Academy of Sciences, 106(51), 21544-21549.

[5] Austin, P. C., & Small, D. S. (2014). The use of bootstrapping when using propensity-score matching without replacement: a simulation study. Statistics in Medicine, 33(24), 4306-4319.

[6] Cha, M., Haddadi, H., Benevenuto, F., & Gummadi, P. K. (2010). Measuring user influence in twitter: The million follower fallacy. ICWSM, 10(10-17), 30.

[7] Easley, D., & Kleinberg, J. (2012). Networks, crowds, and markets. Cambridge Books.

[8] Frederickson, B. (2018). Implicit: Fast Python Collaborative Filtering for Implicit Feedback Datasets. GitHub repository, https://github.com/benfred/implicit

[9] Heater, B. (2018). Microsoft is reportedly acquiring GitHub. Techcrunch. https://techcrunch.com/2018/06/03/microsoft-is-reportedly-acquiring-github/

[10] Heil, B. & Piskorski, M. (2009). New Twitter Research: Men Follow Men and Nobody Tweets. Harvard Business Review. https://hbr.org/2009/06/new-twitter-research-men-follo

[11] Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback. In IEEE International Conference on Data Mining (pp. 263-272). IEEE.

[12] Kalervo J., & Jaana K. (2002). Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4), 422-446.

[13] Latane, B. (1981). The Psychology of Social Impact. American Psychologist 36(4), 343-356.

[14] Lazarsfeld, P., & Merton, R. (1954). Friendship as a Social Process: A Substantive and Methodological Analysis. In Bergen, M., Abel, T., & Page, C., editors, Freedom and Control in Modern Society (pp 18-66). Van Nostrand, Inc.

[15] Lima, A., Rossi, L., & Musolesi, M. (2014). Coding together at scale: GitHub as a collaborative social network. ICWSM, 295-304.

[16] Manski, C. F. (1993). Identification of Endogenous Social Effects: The Reflection Problem. The Review of Economic Studies, 60(3), 531-542.

[17] Mcpherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a Feather: Homophily in Social Networks. Annual Review of Sociology, 27, 415-444.

[18] Ochoa, X. & Duval, E. (2008). Quantitative Analysis of User-Generated Content on the Web. In Proceedings of the First International Workshop on Understanding Web Evolution (pp 19-26).

[19] Owen, A. B. (2007). The pigeonhole bootstrap. The Annals of Applied Statistics, 1(2), 386-411.

[20] Owen, A. B., & Eckles, D. (2012). Bootstrapping data arrays of arbitrary order. The Annals of Applied Statistics, 6(3), 895-927.

[21] Rubin, D. (2006). Matched Sampling for Causal Effects. Cambridge University Press.

[22] Sharma, A., & Cosley, D. (2016). Distinguishing between personal preferences and social influence in online activity feeds. Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing. ACM.

[23] Stuart, E. (2010). Matching methods for causal inference: A review and a look forward. Statistical Science, 25(1), 1-21.

[24] Sundararajan, A. (2007). Local network effects and complex network structure. The BE Journal of Theoretical Economics, 7(1).

[25] Tapscott, D. & Williams, A. (2008). Wikinomics: How Mass Collaboration Changes Everything. Portfolio Hardcover.

[26] Weinberger, M. (2016). Microsoft just edged out Facebook and proved that it's changed in an important way. Business Insider. http://www.businessinsider.com/microsoft-github-open-source-2016-9

[27] Zlotnick, F. (2017). GitHub Open Source Survey. GitHub, Inc. http://opensourcesurvey.org/2017/

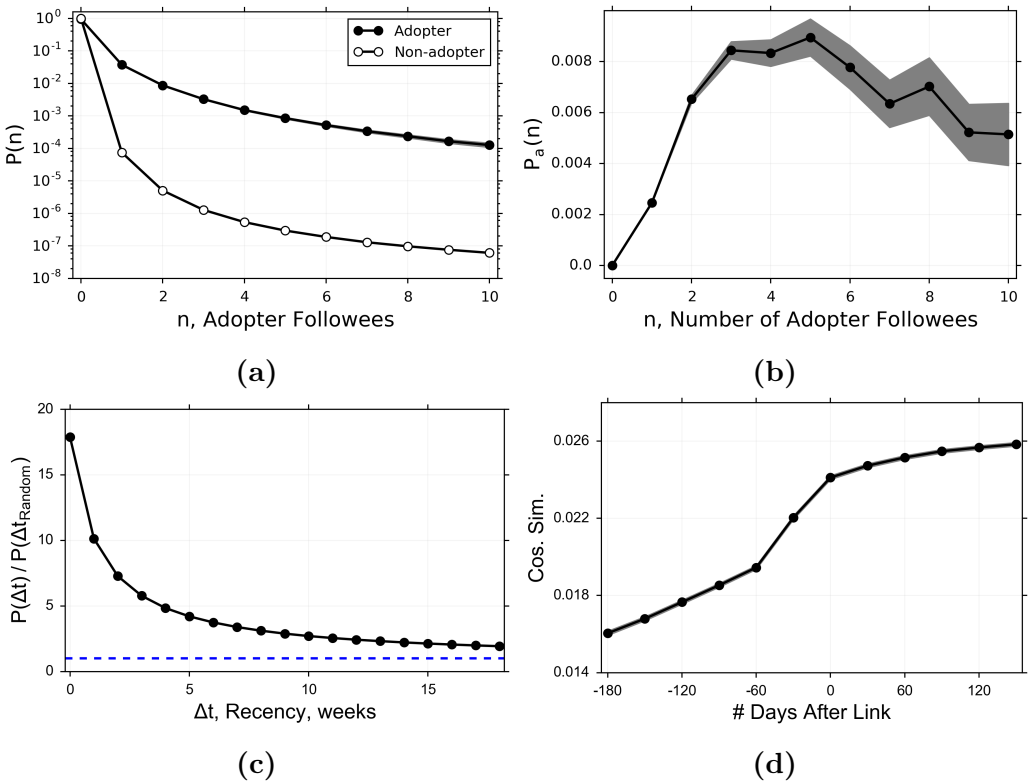
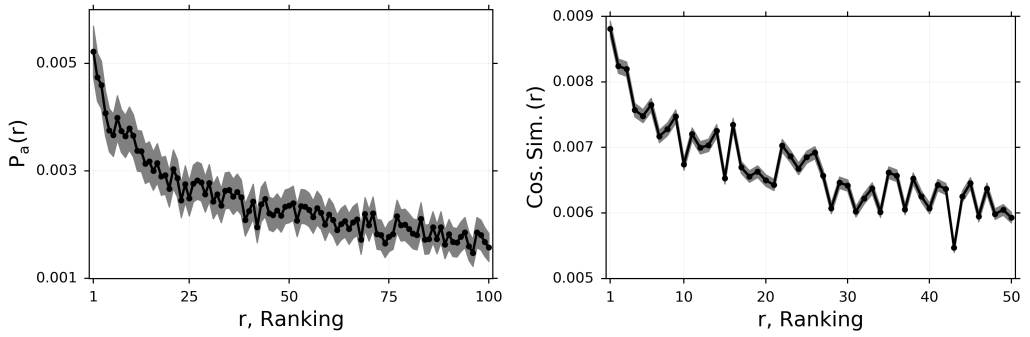
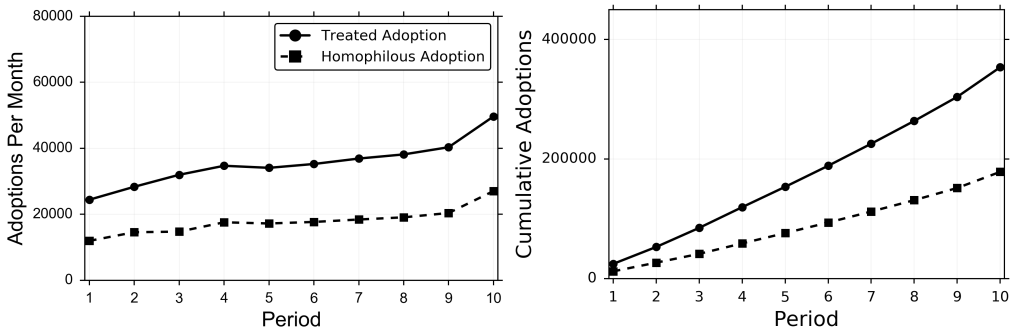


Figure 1: Suggestive evidence of social influence and social selection. (a) The fraction of adopters and non-adopters of a repo for whom n of their leaders have adopted by the end of the period. (b) The likelihood an agent adopts a repo given n of her leaders have adopted. The number of leaders who adopt is measured at the time the agent adopts. In (a) and (b), the results use data during period T. (c) The frequency of observed dyadic differences in adoption times between leader and follower with actual adoption times relative to randomly assigned adoption times. We include all adoptions for which the follower adopts an item after the leader. $\Delta t = t_i - t_j$, where t_i is agent i's (follower) adoption time, and t_j is agent j's (leader) adoption time. The dashed blue line indicates a ratio of 1, which means that there is no temporal clustering in adoption decisions. (d) The average cosine similarity of pairs of agents who form a link during 30-day time periods measured before and after link formation. For each dyad, the time periods are centered so that period 0 is the period which begins at the moment of link formation. In (a) - (d), the gray shaded area indicates the 95% confidence interval.



(a) **(b)**

Figure 2: Evidence that WRMF and other matching covariates are proxies for true, underlying preferences. (a) The likelihood an agent adopts her r ranked repo, as predicted by WRMF, during the period. (b) The cosine similarity of an agent's r-th nearest-neighbor, as measured by our distance function including WRMF and other matching covariates. The gray shaded areas indicate the 95% confidence interval.



(a) **(b)**

Figure 3: Aggregate influence and homophily in multi-dimensional diffusion. Treated and homophilous adoptions per month (a) and cumulatively over time (b). Homophilous adoption is adoption by a matched, non-treated agent, based on nearest-neighbor matching with WRMF.

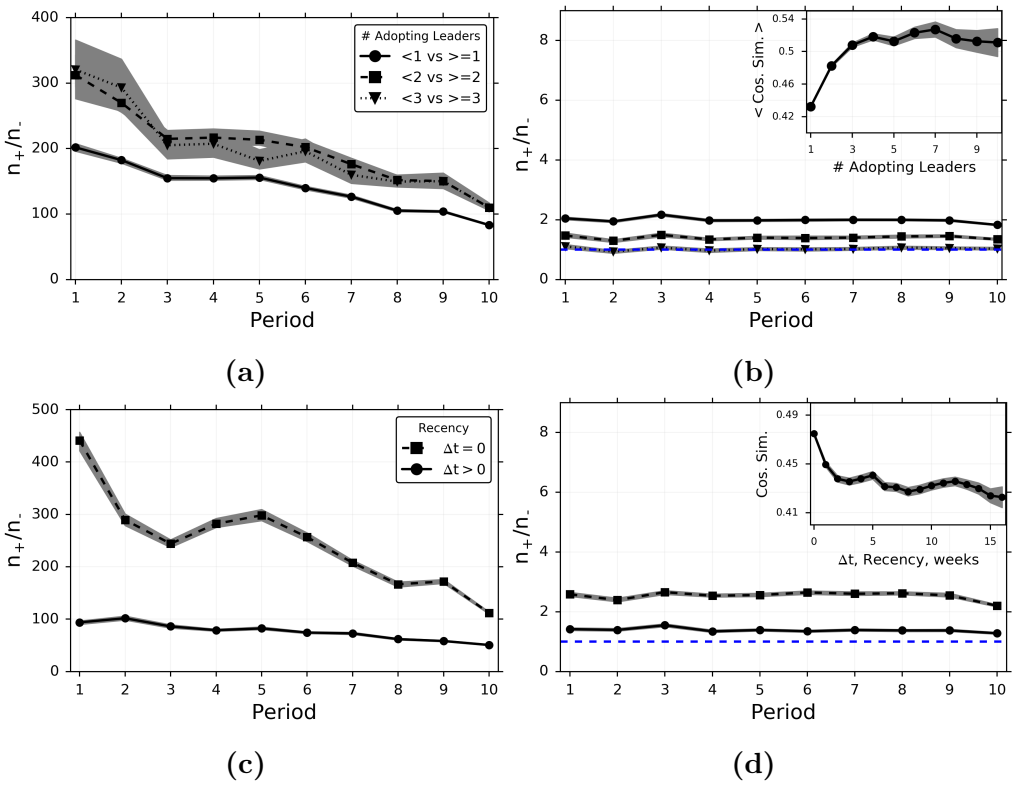


Figure 4: Peer influence and homophily for different intensities of the treatment. (a)-(b) The fraction of observed treated to untreated adopters (n_+/n_-) over time under random matching (a) and nearest-neighbor matching with WRMF (b), for different treatments defined by the number of adopting leader(s). The *inset in (b)* graphs the average cosine similarity between the attributes of an adopter and her adopting leaders, for different numbers of adopting leaders. We only include an observation if we can estimate preferences with WRMF for at least 75% of a follower's adopting leaders. (c)-(d) The fraction of observed treated to untreated adopters (n_+/n_-) over time under random matching (c) and nearest-neighbor matching with WRMF (d), for different treatments defined by the recency of leader adoption. In (c) and (d), Δt refers to the number of periods between the current period and the most recent leader adoption. The *inset in (d)* graphs the dyadic cosine similarity between an adopting follower and her adopting leader, based on the amount of time between adoptions. In (b) and (d), the dashed blue line indicates a ratio of 1, which means the treatment has no effect. In (a)-(d), the 95% confidence intervals are computed using the non-parametric bootstrap. In the *insets in (b) and (d)*, the 95% confidence intervals are based on the formula for standard error.

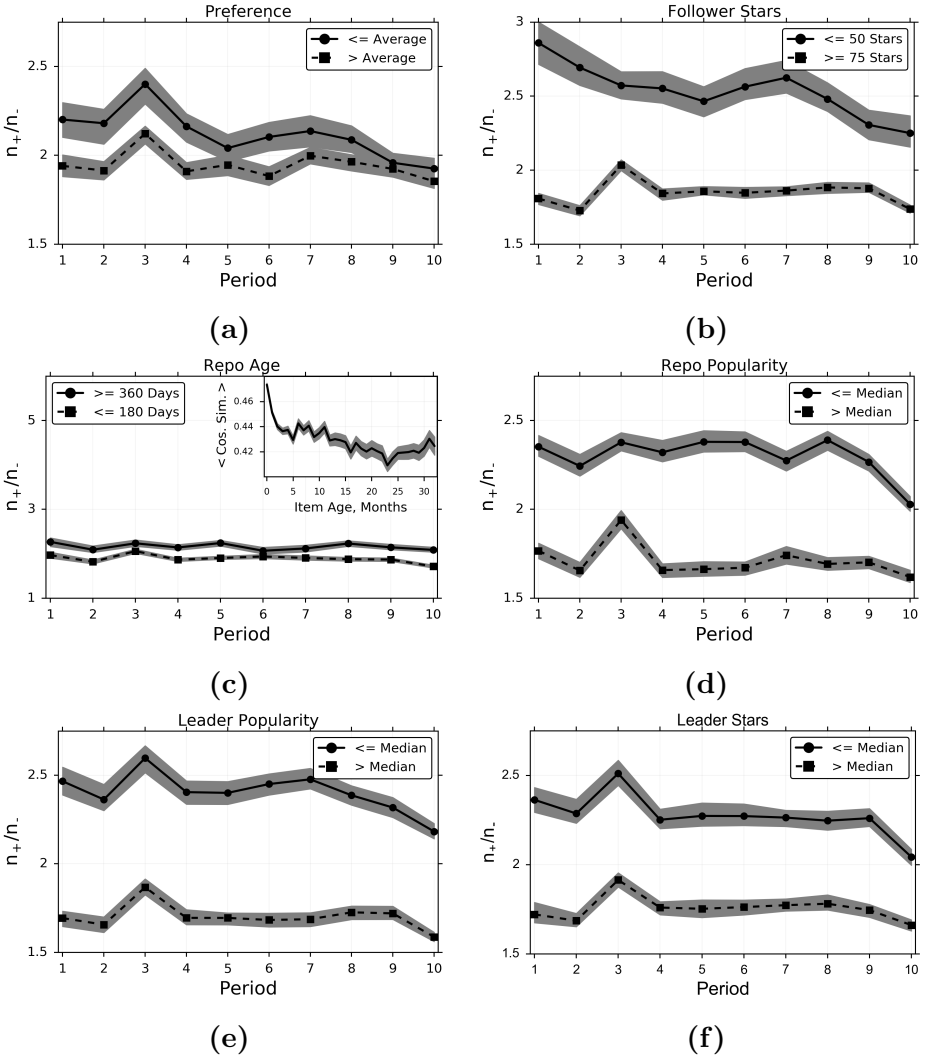


Figure 5: Heterogeneity in peer influence and homophily. (a) An agent's preference for a repo is above or below her average preference for treated repos during the period. We only include agents who are treated in a period on at least 2 repos. (b) The number of stars by the follower up to the end of the period $t-1$ is at most 50, or at least 75. (c) The repo first appeared at most 180 days ago, or at least 360 days ago, measured at the end of the period. The *inset in (c)* graphs the average similarity of an adopter to each of her adopting leaders, based on the number of months since the item first appeared in the data. (d) The number of stars of the repo, as of the end of period $t-1$, is above or below the median for all observations in the period. (e) The average (log+1) number of followers of the adopting leaders, as of the end of period $t-1$, is above or below the median for all the observations in the period. (f) The average (log+1) number of stars by all of an agent's leaders, as of the end of period $t-1$, is above or below the median for all observations in the period. In (a)-(f), treatment is defined as having at least one leader who adopts the repo, and the 95% confidence interval is computed using the non-parametric bootstrap. In the *inset in (c)*, the 95% confidence is based on the formula for standard error.

Adoption Type	Treated	# Adoptions	95% C.I.
Homophily	Yes	$10,406$	$[10,229.93, 10,572.28]$
External Exposure	Yes	$2,449$	$[2,368.48, 2,547.58]$
Est. Counterfactual	**No**	$\mathbf{10,907}$	$[\mathbf{10,707.95}, \mathbf{11,064.1}]$

Table 1: Simulation Results. The number of items adopted by agents for the true counterfactual and the estimated (est.) counterfactual. Homophily refers to the treated agents who adopt an item due to preference, as measured by WRMF. External Exposure refers to the treated agents who adopt an item due to its recent popularity. Est. Counterfactual refers to the matched, non-treated agents who adopt the item. The 95% C.I. is estimated using the non-parametric bootstrap.

Matching Variables	n_+/n_-	95% C.I.
Random	201.84	$[196.5, 207.33]$
Baseline	5.57	$[5.38, 5.75]$
Activities+Baseline	18.18	$[17.4528, 18.862]$
Languages+Baseline	2.91	$[2.85, 2.99]$
WRMF+Baseline	$\mathbf{2.05}$	$[\mathbf{2.01}, \mathbf{2.09}]$

Table 2: The performance of random matching, and nearest-neighbor matching with different sets of variables, for items on which an agent was treated during period 1. Treatment is defined as following at least one agent who adopted the item, versus none. The outcome measure is the fraction of treated to untreated adopters (n_+/n_-). The 95% confidence interval is computed using the non-parametric bootstrap.

Exposure	n_+/n_-	95% C.I.
No	$\mathbf{2.05}$	$[\mathbf{2.01,2.08}]$
Yes	2.17	$[2.12, 2.22]$

Table 3: The performance of nearest-neighbor matching, for items on which an agent was treated during period 1, with WRMF that either does (Exposure=Yes) or does not (Exposure=No) allow for higher confidence for items that a follower does not adopt, but one of her leaders does. Treatment is defined as following at least one agent who adopted the item, versus none. The outcome measure is fraction of treated to untreated adopters (n_+/n_-). The 95% confidence interval is computed using the non-parametric bootstrap.

Matching Variables	n_+/n_-	95% C.I.
Random	201.84	$[181.81, 224.7]$
Baseline	5.57	$[5.05, 6.24]$
Languages+Baseline	2.91	$[2.69, 3.19]$
WRMF+Baseline	$\mathbf{2.05}$	$[\mathbf{1.92}, \mathbf{2.24}]$

Table 4: Reproduction of Table 3, except that the 95% confidence interval is computed using the non-parametric pigeonhole bootstrap.

6 Appendix

6.1 Background on GitHub

GitHub is an online platform for collaborative software development, based on the Git version control system. On GitHub, users *create* software projects known as repos. Each repo is a directory which holds all of the files associated with the project, such as source code and documentation files. The repo exists both locally on a user's machine, and remotely on GitHub. The user makes changes to the repo on her local machine, such as adding code, and then *pushes* the changes to the repo on GitHub, so that the online repo matches her local copy. The user who creates the repo also designates a list of collaborators; these are other users who can push changes to the content of the repo. If the repo is public, then the collaborators are not the only users who can contribute to the repo. In fact, any user on GitHub can *fork* a public repo to create a copy of the repo with which they can freely experiment. The user can then push changes to their forked copy without affecting the original repo. The forked repo can be the starting point for a new independent project, or it can be used to propose changes to the original repo. If a user wishes to merge the changes in her forked repo back into the original repo, and she is not a collaborator, then she can submit a *pull* request. Any of the collaborators may decide whether to approve or reject the requested changes. If approved, then those changes are merged back into the original repo.

However, GitHub is more than just a code hosting service. It also offers social networking features to help users find interesting projects and to support the community of developers.[24] Users can *star* interesting repositories that they want to bookmark for later reference, and/or to show their appreciation to the project's developer(s). The total number of forks and stars of a repo are prominently displayed on the repo's GitHub page. In addition to that, GitHub users can *follow* other users, to be notified of some of their activities. GitHub users can follow other users without the other user's consent, similar to following someone on Twitter. Once a following relationship is established, the follower will receive notifications on her GitHub homepage when the leader creates, forks, or stars a repo, when the leader is added as a collaborator to a repo, and when the leader follows another user. However, the leader will not receive notifications about the follower. Every GitHub user has a profile page with optional information about the user, such as the company for which the user works and her physical location. The user's

[24] At the time of our study, GitHub did not have a recommendation system to help users find interesting projects.

profile page includes information about the public repos that the user has created, forked or starred, and the contributions to public repos which she has made. The user's profile page also lists her total number of followers and leaders, and links to the profile pages of all of the user's followers and leaders.

Besides its public open-source repositories, GitHub also sells private repos which can only be accessed by collaborators, and it sells on-premise instances of its software for businesses. However, we will focus exclusively on public repos, because these are the repos for which we have data, and more importantly, they are the repos for which GitHub's social networking features are relevant.

Like many online networks, GitHub is popular with young people, but it has some unique features. We do not have detailed demographic information for most of the users in our main dataset, but a recent survey of GitHub users can tell us about the types of individuals who currently use GitHub, which is likely similar to the users in our data[27]. The survey results suggest that the overwhelming majority of GitHub users are men, with 90.95% of survey respondents identifying as male and only 3.36% identifying as female. This is similar to the collaborative content creation sites Wikipedia and StackOverflow. GitHub users are young, with over 63.45% of respondents being under 34 years old. They are also well-educated, with 65.7% of respondents having at least a bachelor's degree. Results from [15], which were collected in 2011-2012, show that while GitHub users are located all over the world, the majority of users are in North America or Western Europe, with the U.S.A. having the largest fraction at 30.14%. Aside from gender, the demographics of GitHub users are similar to those found in other online networks targeting young people. GitHub is also a very prominent technology company, used by companies including Google, Facebook, Twitter and Microsoft (and many others) to house their open-source software projects and to reach out to the developer community. As of early June, 2018, Github had over 27 million active users ([9]), and according to the website traffic monitor Alexa, it is the 60th most popular website on earth.[25]

The content creation patterns on GitHub are broadly similar to what has been observed in other online social networks. The so-called "90-9-1" rule is a commonly observed pattern of online user-generated content distributions, in which 90% of users do not contribute anything, 9% of users occasionally create content and 1% of users drive a large amount of the activity [18]. In

[25]www.alexa.com/siteinfo/github.com. The ranking is computed using a combination of average daily visitors and pageviews on GitHub, over the 3-month period ending on 7/30/18.

our data we cannot see the users who do not contribute anything, but we can see that a small fraction of very active users create a large fraction of the content. For instance, when we look at the 1,763,923 unique users in our data who created at least one repo, 46.31% of the repos are created by the top 10% of the users. For the 785,584 users who starred at least one repo, the top 10% of users were responsible for 73.5% of the stars. The remaining behaviors in we analyze in this paper fall in between these two extremes. Compare these results to other platforms such as Wikipedia, in which the top 2.5% of users contribute 80% of the content [25], and Twitter, in which the top 10% of users write 90% of the Tweets [10]. If anything, the distribution of behaviors is less skewed on GitHub than on these other platforms.

However, there are important differences between GitHub and more typical online social networks primarily about interaction, such as Twitter. Many GitHub users do not even participate in the social network at all; for instance, of the GitHub users in our data who created at least one repo, only 31.54% are involved in the social network at all. For users in the social network graph, their average degree of 3.65 is much lower than on Twitter [15]. Also, only 9.6% of the pairs of connected users in our social network have a reciprocal relation between them, while the rest are one-sided. This is again much lower than the rate on Twitter of 22% [15]. These numbers support the idea that the social network is less central to the purpose of GitHub than it is for Twitter, and that following users on GitHub can be more costly in the sense that following many other developers on GitHub results in receiving many notifications from them, which can be distracting. Even though the social network is less central to GitHub's purpose, it still is large and plays significant role in connecting the most active users; the 860,071 users involved in the social network generate the majority of creates, forks, stars and pushes in our data.

6.2 Maximum Posterior Weighted Likelihood based on the WRMF Model

We will infer estimates of preferences based on the statistical model used by the popular collaborative filtering algorithm referred to as WRMF ([11]). The model is static and there is no social network. Let m_t denote the number of items on the platform by the end of period t, and let n_t denote the number of agents who belong to the platform by the end of period t. The adoption matrix $\mathbf{Y}_t = \{Y_{u,i,t}\}$ at the end of period t is the $n_t \times m_t$ matrix with a 1 in its (u, i)-th position if agent u has adopted item i by the end of period t, and zero otherwise. We assume that at the end of period t, each agent

u has linear preferences over characteristics of items, represented by (N-dimensional vector) $\mathbf{W}_{u,t}$, and that each item has characteristics represented by (N-dimensional vector) $\mathbf{V}_{i,t}$. Hence, the utility for agent u from adopting item i would be $\mathbf{W}_{u,t} \cdot \mathbf{V}_{i,t}$. Let $\mathbf{W}_t, \mathbf{V}_t$ denote the matrices of stacked row vectors for $\mathbf{W}_{u,t}, \mathbf{V}_{i,t}$, respectively.

WRMF is estimated at the end of period t by using only the adoption matrix; the algorithm assumes that the adoption of item i by agent u by the end of period t is based on the following equation:

$$Y_{u,i,t} = \mathbf{W}_{u,t} \cdot \mathbf{V}_{i,t} + \mu_{u,i,t}, \quad \mu_{u,i,t} \sim N(0, \sigma_\mu) \tag{A5}$$

Note that WRMF assumes that agents have a negative preference towards any items which they have not adopted, but we can choose how confident we are in that assumption through the use of confidence weights. Let $C_{u,i,t}$ denote the confidence weight for agent u and item i. WRMF assumes that the confidence weights take the following form:

$$C_{u,i,t} = 1 + \alpha Y_{u,i,t} \tag{A6}$$

We incorporate the confidence weights by letting each observation serve as $C_{u,i,t}$ independent observations.[26] By assuming that all of the observations are independent, we can derive the weighted likelihood of our observations $\mathbf{Y}_t$, given the parameters $\mathbf{W}_t, \mathbf{V}_t$:

$$\mathcal{L}(\mathbf{Y}_t | \mathbf{W}_t, \mathbf{V}_t) = \prod_{u,i} g(Y_{u,i,t} | \mathbf{W}_{u,t}, \mathbf{V}_{i,t})^{C_{u,i,t}} \tag{A7}$$

g is the probability density function for a normal distribution.[27] WRMF assumes zero-mean spherical Gaussian priors for the preference and characteristic vectors:

$$\mathbf{W}_{u,t} \sim N\left(\mathbf{0}, \tilde{\rho}^{-1}\mathbf{I}_N\right), \quad \mathbf{V}_{i,t} \sim N\left(\mathbf{0}, \tilde{\rho}^{-1}\mathbf{I}_N\right). \tag{A8}$$

We take the log of the posterior weighted likelihood and replace constant terms with λ to get the following:

$$\ln \mathcal{L}\left(\mathbf{W}_t, \mathbf{V}_t | \mathbf{Y}_t\right) = -\sum_{u,i} C_{u,i,t} \left(Y_{u,i,t} - \mathbf{W}_{u,t} \cdot \mathbf{V}_{i,t}\right)^2 - \lambda \left(\sum_u ||\mathbf{W}_{u,t}||_2^2 + \sum_i ||\mathbf{V}_{i,t}||_2^2 \right). \tag{A9}$$

We learn $\mathbf{W}_t, \mathbf{V}_t$ to maximize the posterior weighted likelihood.

[26] This assumes that $C_{u,i,t}$ is an integer, although that is not necessary to estimate the model.

[27] Note that the data is binary but the distribution of our unobservable is normal. There are other versions of collaborative filtering that use a logistic function to model the binary outcomes.

6.3 Details of Estimation for WRMF

Solving the log-likelihood equation for WRMF (Section 6.2) for a global maximum is not feasible with large datasets. Instead, the values of $\mathbf{W}_t, \mathbf{V}_t$ can be found by alternating least squares, because when we fix either $\mathbf{W}_t$ or $\mathbf{V}_t$ the equation becomes quadratic and so its maximum can be easily computed. To do alternating least squares, begin by randomly guessing values for $\mathbf{W}_t$, and then solve for the $\mathbf{V}_t$ that maximizes the log-likelihood. Then, fix the values of $\mathbf{V}_t$ at what was just found, and solve for the $\mathbf{W}_t$ that maximizes the log-likelihood. Repeat this process a pre-specified number of times, which we will denote by A. α, λ, N and A are all different hyper-parameters which must be set before we can do alternating least squares.

The models in this paper were estimated using the implementation of alternating least squares available in the repo *benfred/implicit* on GitHub ([8]).

6.4 Details of Hyper-parameter Setting

The statistic we compute to set the hyper-parameters is known as the Discounted Cumulative Gain, or DCG ([12]). Let $\text{Rank}(u, i)$ denote the predicted ranking of an item adopted in the test set. We compute the following:

$$\text{DCG} = \sum_{u,i} \frac{1}{\log_2(\text{Rank}(u, i))} \tag{2}$$

The reason we favor this statistic is because it places more emphasis on getting rankings near the top correct. We drop any observations in the test set with a predicted ranking greater than 100. The objective in selecting hyper-parameters is to maximize the value of this statistic.

Preliminary tests revealed that a greater number of factors tends to lead to a higher value of the statistic. Hence, to reduce the number of hyper-parameters over which we need to search, we fixed the number of latent factors at the largest value we tried, $N = 100$. We also fixed the number of alternations at $A = 15$. We then tested random values of the hyper-parameters α, λ. Each value was drawn from a uniform distribution over the set of integers $\{1, ..., 1000\}$. We tested 120 pairs of values. In the end, we use the values $\alpha = 95, \lambda = 864$, which results in DCG $= 0.468$

6.5 Extension of WRMF for Exposure by Leader Adoption

For learning preferences in period t, define the exposure indicator $E_{u,i,t}$ as a variable which is equal to 1 if agent u would have been treated in the past on item i, according to our definition of treatment (Section 2.1), and if the treatment expired by the end of period t. In other words, we have the following:

$$E_{u,i,t} = \begin{cases} 1, & \text{if } Y_{u,i,t} = 0, \quad T_{u,i,s} = 1 \text{ for some } s < t-3, \quad T_{u,i,t} = 0 \\ 0, & \text{otherwise} \end{cases}$$

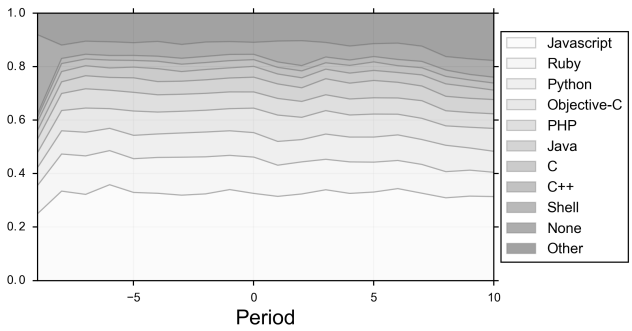
We generalize $C_{u,i,t}$ to be the following:

$$C_{u,i,t} = \begin{cases} 1, & \text{if } Y_{u,i,t} = 0, \quad E_{u,i,t} = 0 \\ 1+\alpha_1, & \text{if } Y_{u,i,t} = 0, \quad E_{u,i,t} = 1 \\ 1+\alpha_2, & \text{if } Y_{u,i,t} = 1 \end{cases}$$

We can now adjust the confidence in the case where an agent has not adopted a given item, but was treated on the item sufficiently long in the past. We use a value of $\alpha_1 > 0$, so that if agent u's leaders adopted the item, then we are more confident that a negative preference revealed by agent u for item i ($Y_{u,i,t} = 0$) is measured correctly. We also assume that $\alpha_1 <= \alpha_2$, since we don't know for certain if an agent was exposed to an item whenever the revealed preference is negative.

6.6 Pigeonhole Bootstrap

We do bootstrap re-sampling on the matched pairs, as before. For each of 100 bootstrap replicates, we re-weight observations according to the following procedure ([19], [20]). For a particular replicate, each agent is assigned the outcome of a Bernoulli($p = 0.5$) draw, and each item is also assigned the outcome of a Bernoulli($p = 0.5$). Each observation is then assigned the product of the draws for the agent and item of that observation as its weight. Hence, an agent-item pair appears in a bootstrap replicate if and only if both the agent and the item are in the replicate.



(a)

Figure A.1: The fraction of stars each period (beginning in period -9) of repos from various programming languages. Missing means that the language field is empty, and Other means that the language is some other language besides the ones listed.

(a)

	Stars Per Agent	Stars Per Repo	Follows Per Agent
Count	163,458	855,317	163,458
Mean	61.40	11.73	10.17
Std. Dev.	146.31	119.78	114.70
Min.	10	1	0
25%	16	1	1
50%	28	1	4
75%	60	3	10
99%	510	179	87
Max	28,148	37,959	38,541

(b)

	Stars Per Period	Follows Per Period
Count	33	33
Mean	304,151.12	50,375.55
Std. Dev.	151,236.57	19,769.92
Min.	67,443	5,724
50%	292,075	53,213
Max	537,477	93,400

Table A.1: Overview of the stars and follows by valid agents. (a) Summary statistics of the distribution of stars per agent, stars per repo, and follows per agent, for the valid agents. (b) Summary statistics of the distribution of total number of stars per period, and follows per period, by the valid agents.

Characteristic	Type
Personal	Experience (Months)
Behavior	Number of followers
	Number of followees
	Number of Repos Starred
Average leader Personal	Average leaders' experience
Average leader behavior	Average leaders' number of followers
	Average leaders' number of followees
	Average leaders' number of repos starred

Table A.2: The demographic and behavioral covariates used for nearest-neighbor matching, aside from previous adoption behaviors. We use logarithms (log + 1) of all covariates except for experience.

Characteristic	Type
Demographic	Experience (Months)
Behavior	Number of followers
	Number of followees
	Number of Repos Starred
	WRMF Agent Preferences

Table A.3: The demographic and behavioral covariates used for estimating homophily. We use logarithms (log +1) of number of followees, number of follwers, and starred repos. All of the covariates are normalized before estimating homophily.