CS Educators Stack Exchange: Exploring User Interactions

Sukanya Moudgalya & K. Bret Staudt Willet

11/20/2018

Introduction

The Stack Exchange Network is one of the largest question-and-answer platforms in the world, founded in 2009 and boasting over 100 million unique monthly visitors by 2015. It has many subject-specific forums for coding related topics, Mathematics, Engineering, Photography, etc.—133 different sites as of October 2018. More recently, Stack Exchange has recently opened up discussion forums for Math educators (2013) and Computer Science educators (2016) as well. For this project, we will focus on the users of the discussion forum Computer Science Educators' Stack Exchange (CSEd SE). The description of the forum calls it 'A question and answer site for those involved in the field of teaching Computer Science.'

In this forum, registered users can ask questions on a wide range of topics related to Computer Science (CS) education. Once they ask a question, they can assign it 'tags', so that similar groups of questions can be identified by a common theme or idea. Some tags that are currently present on the forum include 'curriculum design,' 'student motivation,' 'algorithm,' etc. Users can also reply to (i.e., answer) questions posed by other users, comment on both questions and answers, upvote and downvote questions and answers, and so on—through the voting mechanism, users are held accountable to what they post on the forum. The content on the forum is thus regulated by the forum users and members to a large extent.

In addition to 'normal' user regulation, a set of registered users also act as 'moderators' or 'editors'. Editors can edit the text of what other people post. Moderators, on the other hand, have more power. They can follow up with posts that are flagged as spam or offensive, delete and manage other users, combine or collapse tags, lock posts so that no one can change them or vote on them, and so on (read more here).

The users on this forum range from actual K-12 and college-level CS educators to working professionals in the software industry, students, and other people interested in CS education. The users also belong to many different countries, such as the USA, UK, Canada, Ireland, India, Finland, etc. Each user has reputation points primarily based on the quality of their posts, such as user-generated votes their posts receive. The reputation changes across time are accessible for people to see.

Purpose

This study explores the kinds of interaction between the users, with a focus on the top voted questions. It seeks to describe the kind of users, based on their profession, location, reputation, etc., that tend to ask questions or respond to the top questions. It also seeks to understand if the users tend to cluster around certain question tags (topic area) or if the clusters are agnostic to the topics and user characteristics.

In the second part of the study, we will focus on trying to interpret the reasons for users choosing to answer the questions. It will explore if the reasons are dependent to the topic tag or some characteristic of the users who ask the question.

Theoretical Framework and Research Questions

Communities of practice are "....groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly" (Wenger, 2011, p. 1). These communities have the following traits: (a) a domain of shared interest; (b) people who engage in joint activities and discussions; and (c) members who are practitioners (such as educators). Further, according to Wenger, they should demonstrate (a) mutual engagement (such as a dialogue instead of one directional information flow); (b) a joint enterprise; and (c) a shared repertoire. These three traits represent "...three dimensions of the relation by which practice is the source of coherence of a community" (Wenger, 1998, p. 72).

The CSEd SE has many elements that could potentially make it a 'community of practice'. For instance, there are clear boundaries in terms of membership to the forum. The forum has a unique purpose, that is to build knowledge and dialogue in the domain of CS education. Further, the discussions on the forum are not generally a unidirectional venture. The commenting feature, for instance, allows scope for a back and forth dialogue. Finally, the presence of moderators and editors for maintenance of the forum, scope of democratic elections for the selection of moderators, reputation points based on user-generated voting system, gives CSEd SE the traits of a community of practice. Indeed, a literature review of informal online teacher communities revealed that the communities of practice framework has often been used to study teacher interactions in online forums (Macià & García, 2016).

Using this communities of practice framework, we focused our study to answer three research questions:
1. What are the characteristics of users who contributed to the 50 highest-voted questions in the CSEd SE?
2. Does network clustering occur in the context of users who contributed to the 50 highest-voted questions? What are some characteristics of these clusters?
3. From all the questions that users are exposed to, what factors predict the questions that users to respond to?

Method

Data Collection

The data from the CSEd SE forum were collected through data mining, using the statistical software *R* (R Core Team, 2018). In particular, we used the R package *stackr* (Robinson, 2018) to extract data from the CSEd SE website. The stackr package helps obtain the "read-only features of the Stack Exchange API with the ability to download information on questions, answers, users, tags, and other aspects of the site so that they can be analyzed in R".

Using stackr, we were able to collect the metadata for CSEd SE questions, answers, comments, tags, and users in the week of 7th-13th June 2018. These metadata included information such as question identity number, answer identity number, user identity number, tag name, reputation scores of users, the date they joined the forum, and so on. This data collection resulted in a corpus of 559 questions, 2,675 answers, and 7,209 comments from CSEd SE.

Data Analysis

First, we loaded the necessary R packages and our data:

```
#library(stackr)  # for collecting data from Stack Exchange
library(tidyverse)  # for data manipulation; includes library(dplyr); library(ggplot2)
library(stringr)  # for ease of working with string and character varbiables
library(lubridate)  # for ease of working with dates
library(gridExtra)  # for working with "grid" graphics, notably to arrange multiple grid-based plots on
library(igraph)  # for processing social network
```

```r
library(ggraph)  # for visualizing social network
library(moments)  # for skewness and kurtosis statistics
library(lme4)  # for building selection model using linear mixed-effects
library(ergm)  # for building selection model using exponential-family random graph models
```

```r
## merging the answers to the top 50 questions
csed <- csed_ques_top %>% inner_join(csed_og[[1]], by = "question_id") %>%
        as.data.frame

#csed$question_id %>% unique %>% length   # these are the number of different questions

csed$askers <- csed$owner_display_name.x %>% as.character
csed$responders <- csed$owner_display_name.y %>% as.character

#csed$askers %>% unique %>% length  # these are the different question askers
#csed$responders %>% as.character %>% unique %>% length  # these are the different responders

#csed %>% group_by(owner_display_name.y) %>% summarize(view_count %>% mean)


## ------------------------------------------------------------------
## posts per contributor
## ------------------------------------------------------------------

responders <- csed$responders %>% table %>% as.data.frame %>% arrange(desc(Freq))


## ------------------------------------------------------------------
## calculating descriptive statistics
## ------------------------------------------------------------------

library(moments)
responders$Freq %>% mean %>% round(2)
```

```
## [1] 1.86
```

```r
responders$Freq %>% sd %>% round(2)
```

```
## [1] 3.33
```

```r
responders$Freq %>% median %>% round(2)
```

```
## [1] 1
```

```r
responders$Freq %>% min
```

```
## [1] 1
```

```r
responders$Freq %>% max
```

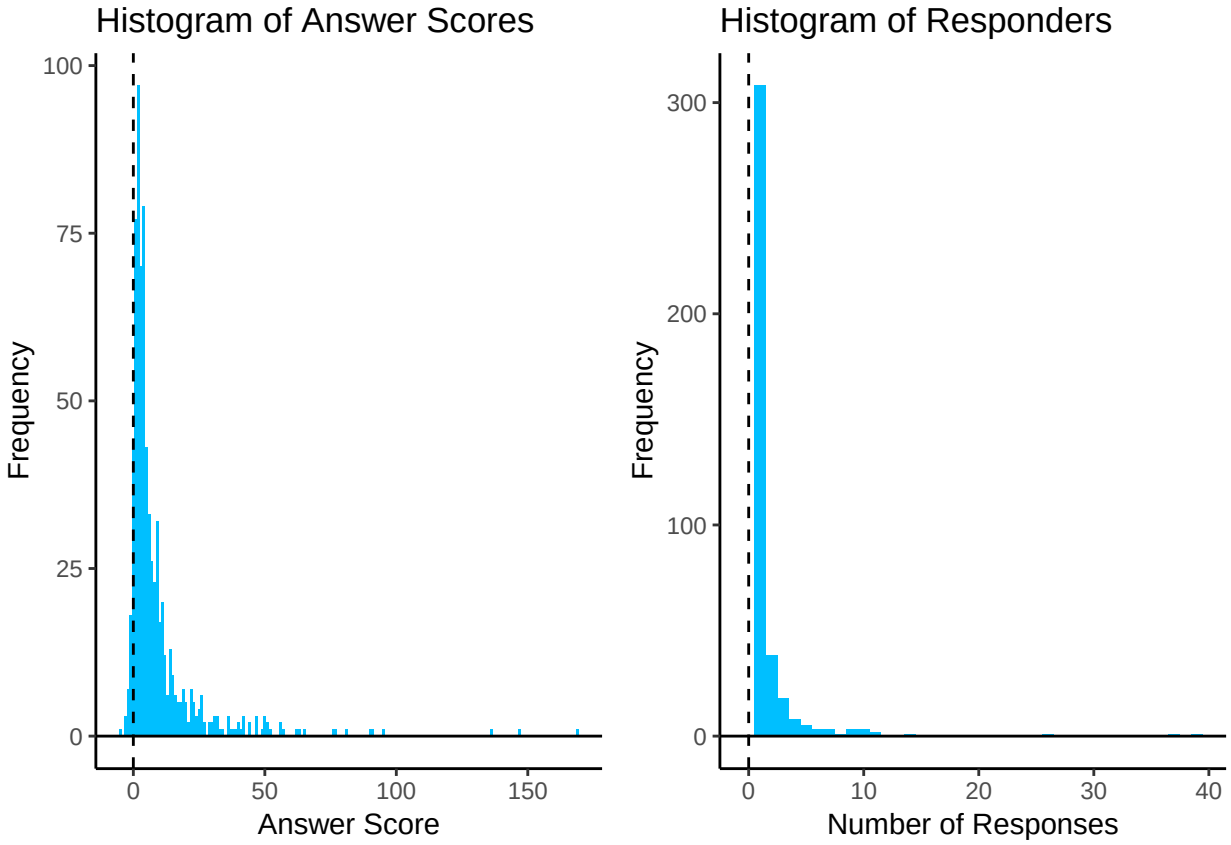
```
## [1] 39
```

```r
responders$Freq %>% skewness %>% round(2)
```

```
## [1] 7.97
```

```r
responders$Freq %>% kurtosis %>% round(2)
```

```
## [1] 79.41
```



Create the Network Graph

```
library(tidyverse); library(igraph)
csed_graph <- csed %>% dplyr::select(responders, askers) %>%
        as.matrix %>% graph_from_edgelist(directed=TRUE)
# csed_graph %>% summary
```

Network Statistics

```
csed_graph %>% gsize  # number of edges
```

```
## [1] 737
```

```
csed_graph %>% V %>% length  # number of vertices/nodes
```

```
## [1] 416
```

```
csed_graph %>% edge_density  # the ratio of the number of edges and the number of possible edges.
```

```
## [1] 0.004268999
```

```
        # Also called cohesion.
csed_graph %>% transitivity("global")  # the probability that the adjacent vertices of a vertex are con
```

```
## [1] 0.05006284
```

```
        # Also called the clustering coefficient. The balance of connections.
csed_graph %>% vertex_connectivity  # If the graph is not (strongly) connected then the connectivity is
```

```
## [1] 0
```

```
csed_graph %>% diameter  # the length of the longest geodesic (max distance between two vertices)
```

```
## [1] 5
```

```
csed_graph %>% reciprocity  # defines the proportion of mutual connections, in a directed graph.
```

```
## [1] 0.03867403
```

```
        # most commonly defined as the probability that the opposite counterpart of a directed edge is
csed_graph %>% eigen_centrality %>% '$'(value)  # The eigenvalue corresponding to the calculated eigenv
```

```
## [1] 25.81937
```

```
        # Eigenvector centrality scores correspond to the values of the first eigenvector of the graph
        # these scores may, in turn, be interpreted as arising from a reciprocal process in which the c
        # vertices with high eigenvector centralities are those which are connected to many other verti
csed_graph %>% authority_score %>% '$'(value)  # The corresponding eigenvalue of the calculated princip
```

```
## [1] 316.4608
```

```
        # The authority scores of the vertices are defined as the principal eigenvector of AT A, where
csed_graph %>% hub_score %>% '$'(value)  # The corresponding eigenvalue of the calculated principal eig
```

```
## [1] 316.4608
```

```
csed_graph %>% centr_clo_tmax  # Theoretical maximum for closeness centralization
```

```
## [1] 414.0024
```

Clustering

A community is a set of nodes with many edges inside the community and few edges between outside it (i.e. between the community itself and the rest of the graph).

The *spinglass clustering* algorithm maps community detection onto finding the ground state of an infinite range spin glass. Csárdi, Nepusz, and Airoldi (2016, pp. 132-133) explained:

"The clustering method of Reichardt and Bornholdt (2006) is motivated by spin glass models from statistical physics. Such models are used to describe and explain magnetism at the microscopic scale at finite temperatures. Reichardt and Bornholdt (2006) drew an analogy between spin glass models and the problem of community detection on graphs and proposed an algorithm based on the simulated annealing of the spin glass model to obtain well-defined communities in a graph. A spin glass model consists of a set of particles called spins that are coupled by ferromagnetic or antiferromagnetic bonds. Each spin can be in one of k possible states. The well-known Potts model then defines the total energy of the spin glass with a given spin configuration... Spins and interactions in the Potts model are very similar to graphs: each spin in the model corresponds to a vertex, and each interaction corresponds to an edge... Reichardt and Bornholdt (2006) gave efficient update rules for the above energy function, making it possible to apply a simulated annealing procedure to find the ground state of the model that corresponds to a low energy con- figuration. Their algorithm starts from a random configuration of spins and tries to flip all the spins once in each time step. After each individual spin flip, the energy of the new configuration is evaluated."

In other words, the spinglass clustering algorithm partitions the vertices into communities by optimizing an energy function. The initial R code to produce spinglass clusters is straightforward:

```
csg <- csed_graph %>% cluster_spinglass  # creates the clusters
csg$membership %>% unique %>% length  # this is the number of clusters/communities/groups
```

```
## [1] 15
```

One of the important outcomes of this method is the *modularity* value. Modularity measures how good the division is, or how separated are the different vertex types from each other. The spinglass algorithm looks for the modularity of the optimal partition. For a given network, the partition with maximum modularity corresponds to the optimal community structure.

Note that if $M = 0$, all nodes belong to one group; if $M < 0$, each node belongs to separate community.

```
csg$modularity  # The modularity of a graph with respect to some division (or vertex types)
```

```
## [1] 0.5219158
```

Test of Statistical Significance

The test for statistical significance for spinglass clustering is a bit different than the familiar tests that return p-values (Csárdi, Nepusz, & Airoldi (2016, pp. 132-138).

The idea behind this test of siginficance is that a random network of equal size and degree distribution as our observed network should have a lower modularity score—that is, if the observerd network does in fact have statistically significant clustering.

The following R procedure generates 100 randomized instances of our network (with the same size and degree distribution) using the `sample_ degseq()` function. The `method = 'vl'` ensures that there are no loop edges in the randomly generated networks.

A '0' result from this procudure indicates that no randomized networks have community structure with a modularity score that is higher than the one obtained from the original, observed network. Hence a '0' result means that our network has significant community structure.

```
#degrees <- igraph::degree(csed_graph)
#q <- csg$modularity
#q_rand <- replicate(100, sample_degseq(degrees, method = 'vl'),
#                simplify = FALSE) %>%
#        lapply(cluster_spinglass) %>%
#        sapply(modularity)
#sum(q_rand > q) / 100
```

Identifying the "Typical" Number of Clusters Returned with the Spinglass Algorithm

It is important to note that a different result is returned each time the spinglass clustering algorithm is run. For this reason, we need to run a number of simulations to see what the "typical" number of clusters are. For now, we run the algorithm 100 times and then look at the mean and median number of clusters obtained; for publication, we would run the algorithm 1,000 times.

We made a note of a `seed` reproduces the median number of clusters, confirmed that this is reproducible, and then set this seed so that all future work will be run with this same clustering configuration.

```
#csg_matrix <- matrix(NA, nrow=1, ncol=100)
#for (i in 1:100) {
#        set.seed(i)
#        csg <- csed_graph %>% cluster_spinglass
#        csg_matrix[1,i] <- max(csg$membership)
#}

#csg_matrix %>% as.vector %>% mean
#csg_matrix %>% as.vector %>% sd
```

```
#csg_matrix %>% as.vector %>% min
#csg_matrix %>% as.vector %>% max
#csg_matrix %>% as.vector %>% median
#csg_matrix %>% as.vector %>% skewness
#csg_matrix %>% as.vector %>% kurtosis
```

```
set.seed(2)  # the solutions look different with a different seed
csg <- csed_graph %>% cluster_spinglass
csg$membership %>% unique %>% length  # number of clusters
```

```
## [1] 15
```

```
csg$csize  # size of each cluster
```

```
##  [1] 19 21 27 33 27 18 37 38 21 40  8 13 57 11 46
```

```
csg$modularity  # modularity
```

```
## [1] 0.5238867
```

```
csg$temperature  # temperature
```

```
## [1] 0.02376045
```

```
csg$vcount  # number of vertices (nodes)
```

```
## [1] 416
```

```
#csg$names  # names of each of the vertices (nodes)
```

Results

Network Visualization

Finally, we created a visualization of our network structure, using the color palette generated by our spinglass clustering. Here, we used the `layout = 'fr'` algorithm, which is appropriate for large (but still with less than 1,000 nodes), potentially disconnected networks.

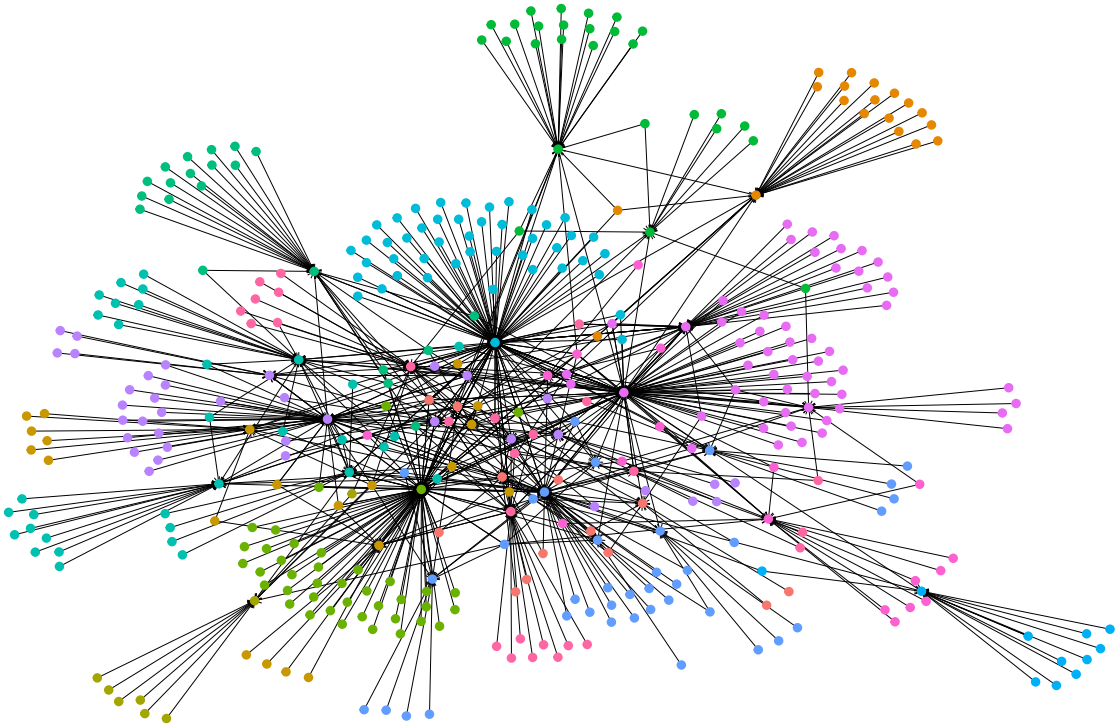
```
## see https://www.data-imaginist.com/2017/ggraph-introduction-edges/

library(ggraph)
layout <- csed_graph %>% create_layout(layout = 'fr')

ggraph(layout) +
        geom_edge_link(width=.1,
                       arrow = arrow(length=unit(1, 'mm'))
                       ) +
        geom_node_point(alpha=1,
                        size=1,
                        aes(color=node_palette)
                        ) +
        ggtitle("CSEd SE Network Visualization") +
        theme_bw() +  # makes background white (not gray)
        theme(plot.background = element_blank(),
              panel.grid.major = element_blank(),
              panel.grid.minor = element_blank(),
```

```
            panel.border = element_blank(),
            axis.title = element_blank(),
            axis.text = element_blank(),
            axis.ticks = element_blank(),
            legend.position="none"
        )
```

CSEd SE Network Visualization



```
ggsave("model-output/csed_se_network_visualization.png", width = 1.618 * 10, height = 1 * 10)
```

Selection Model

Here we describe and build a selection model of users on CSEd SE choosing to answer certain questions. We used *linear mixed-effects* models to construct our selection model.

Null p_2 Selection Model

First we built the null p_2 model (i.e., only using edges) for predicting ties.

We then compared the Akaike information criterion (AIC) and the Bayesian information criterion (BIC), keeping in mind that smaller values are better. We have also included the Intraclass-Correlation Coefficient (ICC) scores for responders and askers.

```
p2_0 %>% summary
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
```

```
## Formula: tie ~ 1 + (1 | responder) + (1 | asker)
##    Data: csed_full_weighted
##
##      AIC      BIC   logLik deviance df.resid
##   5150.0   5180.2  -2572.0   5144.0   173053
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.6591 -0.0007 -0.0006 -0.0006  9.8575
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  responder (Intercept)   0.5369  0.7327
##  asker     (Intercept) 104.6900 10.2318
## Number of obs: 173056, groups:  responder, 416; asker, 416
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.8170     0.4182  -35.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
sjstats::icc(p2_0)
```

```
##
## Generalized linear mixed model
##
## Family : binomial (logit)
## Formula: tie ~ 1 + (1 | responder) + (1 | asker)
##
##   ICC (responder): 0.0049
##       ICC (asker): 0.9647
```

We have more work to do to build additional p_2 models; this is next our to-do list.

Level 1 Selection Model

The selection model at level 1, for responder i, answering question q, asked by user $i\prime$:

$$log(\frac{p(\text{Answering a certain question} = 1)}{1 - p(\text{Answering a certain question} = 1)}) = \theta_i + \theta_{i\prime} + \theta_q + \theta_1(\text{Prior relationship of } i \text{ and } i\prime) + \theta_2(\text{Prior connection between } i$$

Where

1. θ_i = Responder effect
2. $\theta_{i\prime}$ = Asker effect
3. θ_q = Question effect
4. Prior relationship of i and $i\prime = 1$ if i had answered one of $i\prime$'s question or if $i\prime$'s had answered one i's question previously; 0 if i and $i\prime$ had no such interaction.
5. Prior connection between i and question tag $q = 1$ if i had previously answered any question with the same topic tag(s) as question q; 0 if i had no such connection with q.

Constructing the Level 1 ERGM

Coming soon!

Level 2 Selection Models

The selection model at level 2, for responder i;

$\theta_i = \alpha_0 + \alpha_1$(Reputation points)$+\alpha_2$(Role)$+\alpha_3$(Duration of membership)$+\alpha_4$(Number of questions previously answered)$+\epsilon_1$

Where

1. Reputation points = The numerical reputation points associated with each user.
2. Role = 1 if the responder answering question is an editor or a moderator. 0 otherwise.
3. Duration of Membership = Time in months of being a member of CSEd SE
4. Number of questions previously asked = Total number of questions asked up until present time.

The selection model at level 2, for user $i\prime$ asking the question;

$\theta_{i\prime} = \beta_0 + \beta_1$(Reputation points)$+\beta_2$(Duration of membership)$+\beta_3$(Number of questions previously asked)$+\epsilon_2$

Where

1. Reputation points = The numerical reputation points associated with each user.
2. Duration of Membership = Time in months of being a member of CSEd SE.
3. Number of questions previously asked = Total number of questions asked up until present time.

The selection model at level 2, for question q;

$$\theta_q = \gamma_0 + \gamma_1(\text{Number of votes}) + \gamma_2(\text{Number of answers}) + \gamma_3(\text{Number of views}) + \epsilon_3$$

Where

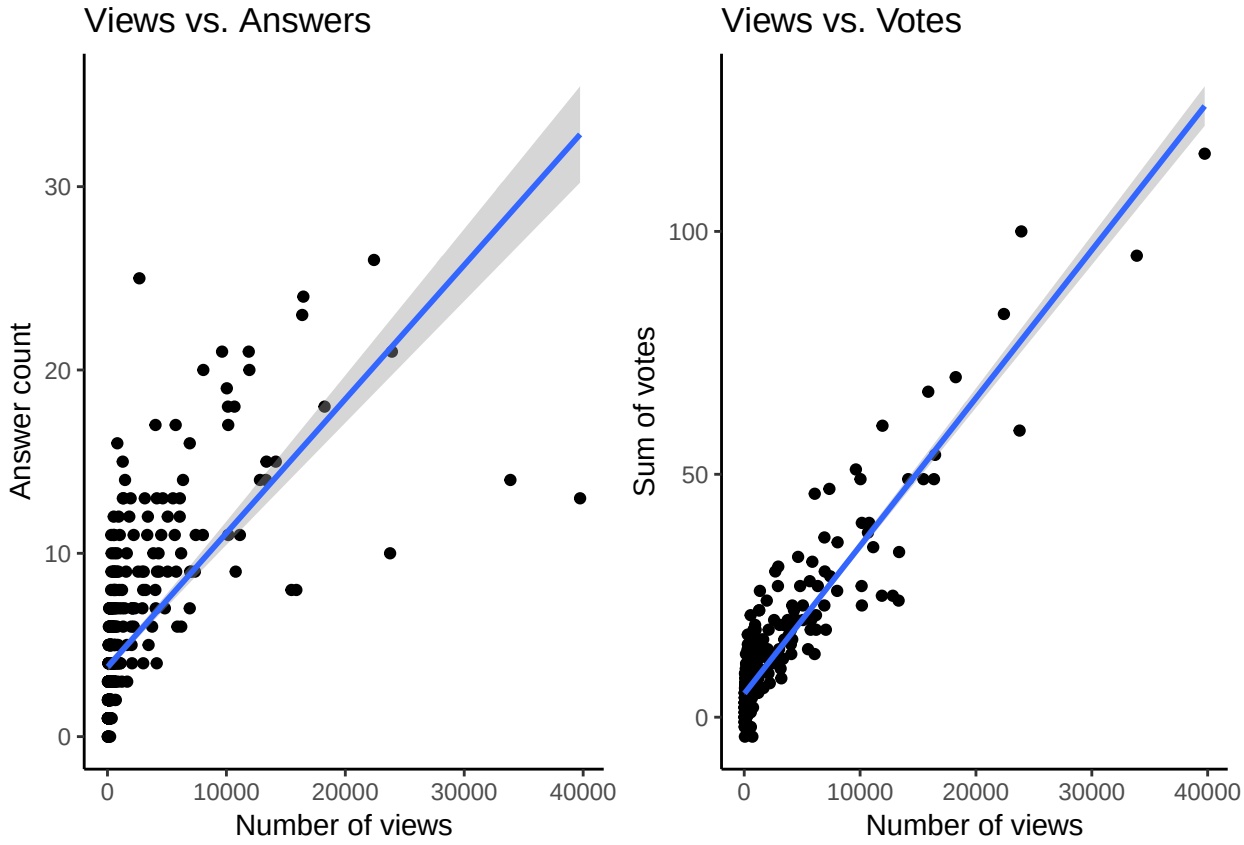
1. Number of votes = Total up-votes the question got on the forum prior to i answering the question.
2. Number of answers = Total answers the question had prior to i answering the question.
3. Number of views = Total views the question got on the forum prior to i answering the question.

Possible sources of multicollinearity

We tested to see if these variables are highly correlated and if we needed to change the models that we proposed. These are the correlations that we tested:

1. Correlations between number of votes, answers, and views
2. Correlations between reputation points and number of questions answered/asked

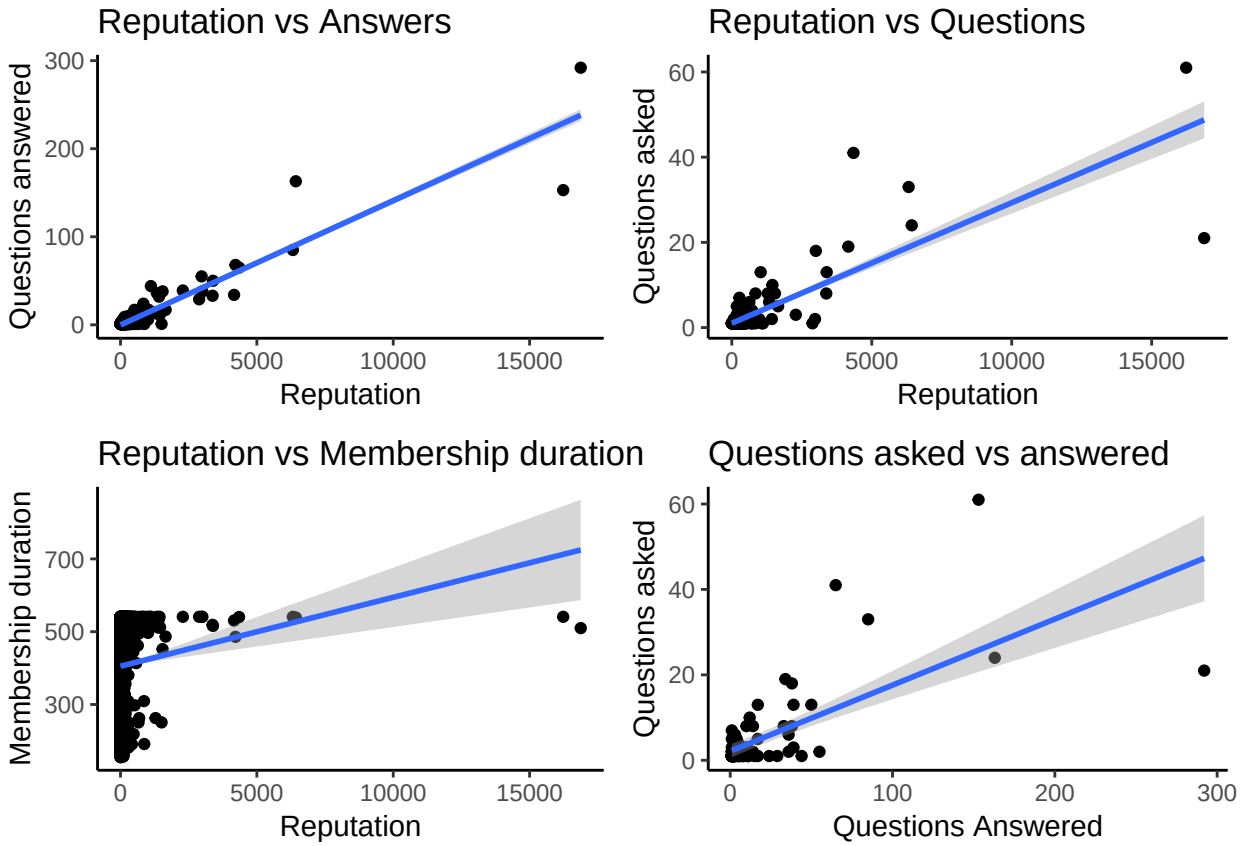
First, we explored the correlations between the number of votes, answers, and views. Please see the plots and the correlation matrix below. As these variables were highly correlated, we decided to use only one, the number of votes.




```
cor(stack.questions[c(4, 5, 6)], use = 'complete.obs') %>% round(2)
```

```
##              view_count answer_count score
## view_count         1.00         0.67  0.92
## answer_count       0.67         1.00  0.74
## score              0.92         0.74  1.00
```

```
## Warning: Removed 4540 rows containing non-finite values (stat_smooth).
## Warning: Removed 4540 rows containing missing values (geom_point).
## Warning: Removed 4981 rows containing non-finite values (stat_smooth).
## Warning: Removed 4981 rows containing missing values (geom_point).
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
## Warning: Removed 1 rows containing missing values (geom_point).
## Warning: Removed 5104 rows containing non-finite values (stat_smooth).
## Warning: Removed 5104 rows containing missing values (geom_point).
```




```
cor(usr.plus.data[c(11,25,26,27)], use = 'complete.obs') %>% round(2)
```

```
##                          reputation answered_ques questions_asked
## reputation                     1.00          0.94            0.81
## answered_ques                  0.94          1.00            0.68
## questions_asked                0.81          0.68            1.00
## membership_duration_days       0.19          0.18            0.18
##                          membership_duration_days
## reputation                                   0.19
## answered_ques                                0.18
## questions_asked                              0.18
## membership_duration_days                     1.00
```

UPDATED Level 2 Selection Models

New model at level 2, for responder i;

$$\theta_i = \alpha_0 + \alpha_1(\text{Reputation points}) + \alpha_2(\text{Role}) + \alpha_3(\text{Duration of membership}) + \epsilon_1$$

(We removed 'number of questions previously answered' as it was highly correlated with reputation points)

New model at level 2, for user $i\prime$ asking the question;

$$\theta_{i\prime} = \beta_0 + \beta_1(\text{Reputation points}) + \beta_2(\text{Duration of membership}) + \epsilon_2$$

(We removed 'number of questions previously asked' as it was highly correlated with reputation points)

New model at level 2, for question q;

$$\theta_q = \gamma_0 + \gamma_1(\text{Number of votes}) + \epsilon_3$$

(We removed 'number of answers' and 'number of views' as they were highly correlated with number of votes)

Constructing the Level 2 ERGMs

Coming soon!

Future Research

It is highly possible that in future research, we may add other variabes to the model, such as geographical location. Also, the variables we chose in this model reflect prior research on teacher interactions where things such as 'reputation', years of service (in CSEd SE it is 'membership duration', formally designated leaders (in CSEd SE it is 'editor/moderator'), previous ties, etc mattered for selection model.

References

Csárdi, G. (2018). igraph: Network analysis and visualization (Version 1.2.2) [R package]. Retrieved from https://CRAN.R-project.org/package=igraph

Macià, M., & García, I. (2016). Informal online communities and networks as a source of teacher professional development: A review. Teaching and Teacher Education, 55, 291-307.

Pedersen, T. L. (2018). ggraph: An implementation of grammar of graphics for graphs and networks (Version 1.0.2) [R package]. Retrieved from https://CRAN.R-project.org/package=ggraph

R Core Team. (2018). R: A language and environment for statistical computing (Version 3.5.0) [Computer software]. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Robinson, D. (2015). stackr: An R package for connecting to the Stack Exchange API [R package]. Retrieved from https://github.com/dgrtwo/stackr

Wenger, E. (1998). Communities of practice: Learning, meaning, and identity. New York, NY: Cambridge University Press.

Wenger, E. (2011). Communities of practice: A brief introduction

Wickham, H., Chang, W., & RStudio. (2016). ggplot2: Create elegant data visualisations using the grammar of graphics (Version 2.2.1) [R package]. Retrieved from https://CRAN.R-project.org/package=ggplot2

Wickham, H., François, R., Henry, L., Müller, K., & RStudio. (2018). dplyr: A grammar of data manipulation (Version 0.7.6) [R package]. Retrieved from https://CRAN.R-project.org/package=dplyr