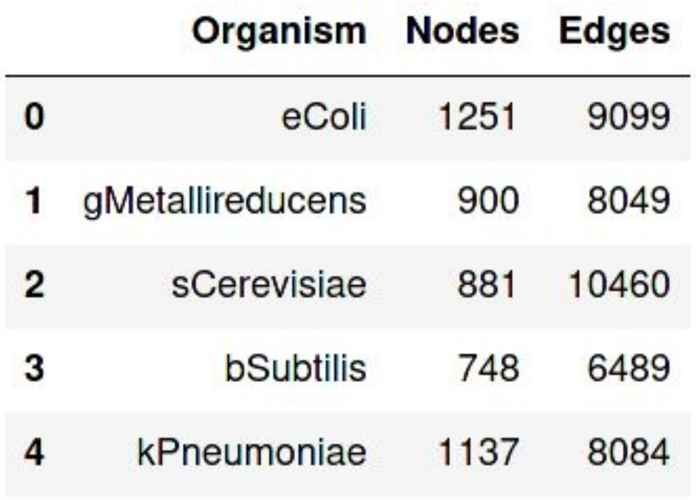
Identification of critical connectors in the directed reaction-centric graphs of microbial metabolic networks

Maggie Tsui
Miguel Guardado
Noam Teyssier
Silver Alkhafaji

November 18th 2020

Background/Description of Data

- Data is an **adjacency matrix** from a directed microbial metabolic network
  - Nodes are metabolic reactions and edges are metabolites
  - Binary NxN matrix, N is the number of reactions
- Matrices for five microorganisms: *E. coli, K. pneumoniae, B. subtilis, G. metallireducens,* and *S. cerevisiae*
- 4917 total nodes (mean = 983.4, std = 183.3) in the reaction graphs with 42181 total arcs (mean = 8436.2, std = 1311.5) across 5 organisms



	Organism	Nodes	Edges
0	eColi	1251	9099
1	gMetallireducens	900	8049
2	sCerevisiae	881	10460
3	bSubtilis	748	6489
4	kPneumoniae	1137	8084

Analysis Plan

- Recreate the results found in figure 2 of the paper showing that the reaction-centric networks are scale free (degree distribution follows a power law)
- Perform regression analysis (linear model) of node degrees for each microorganism
- Create our own cascade number algorithm
- Calculate correlation matrix with the different centrality metrics using networkx package in Python

Paper’s Fig 2.

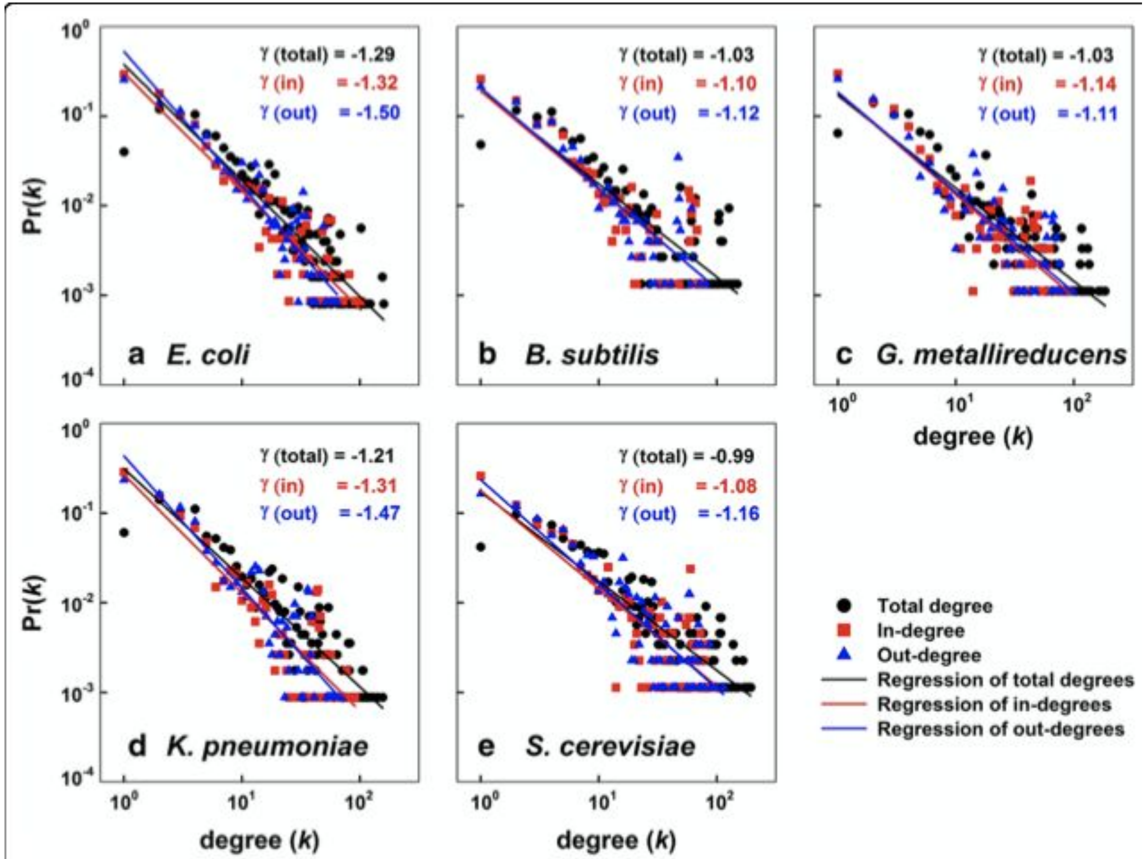


Fig. 2 Degree distribution in the reaction-centric metabolic networks. (**a**) *Escherichia coli* (iJO1366), (**b**) *Bacillus subtilis* (iYO844), (**c**) *Geobacter metallireducens* (iAF987), (**d**) *Klebsiella pneumonia* (iYL1228), and (**e**) *Saccharomyces cerevisiae* (iMM904). In-degree (denoted as a red square), out-degree (blue triangle), or total degree (black circle) was plotted against their probabilities on logarithmic scales

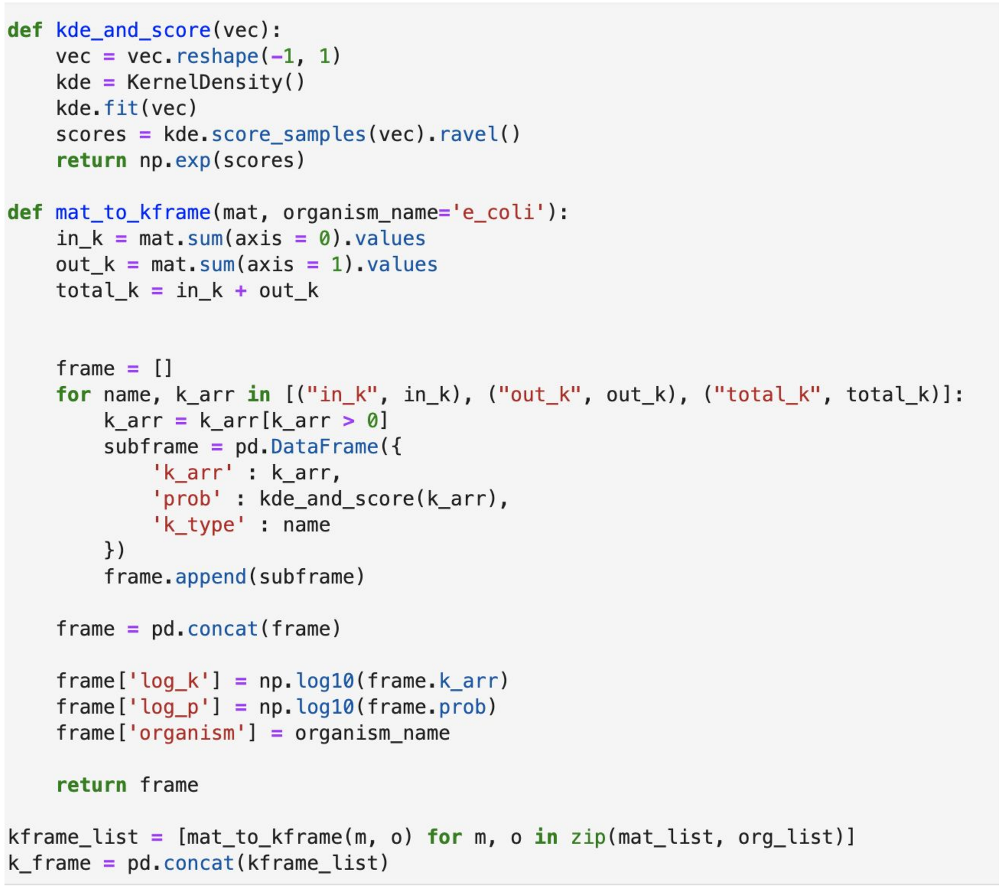
Recreating Fig 2.

Number of In degrees >>
k_in = row_sum()

Number of Out degrees >>
k_out = col_sum()

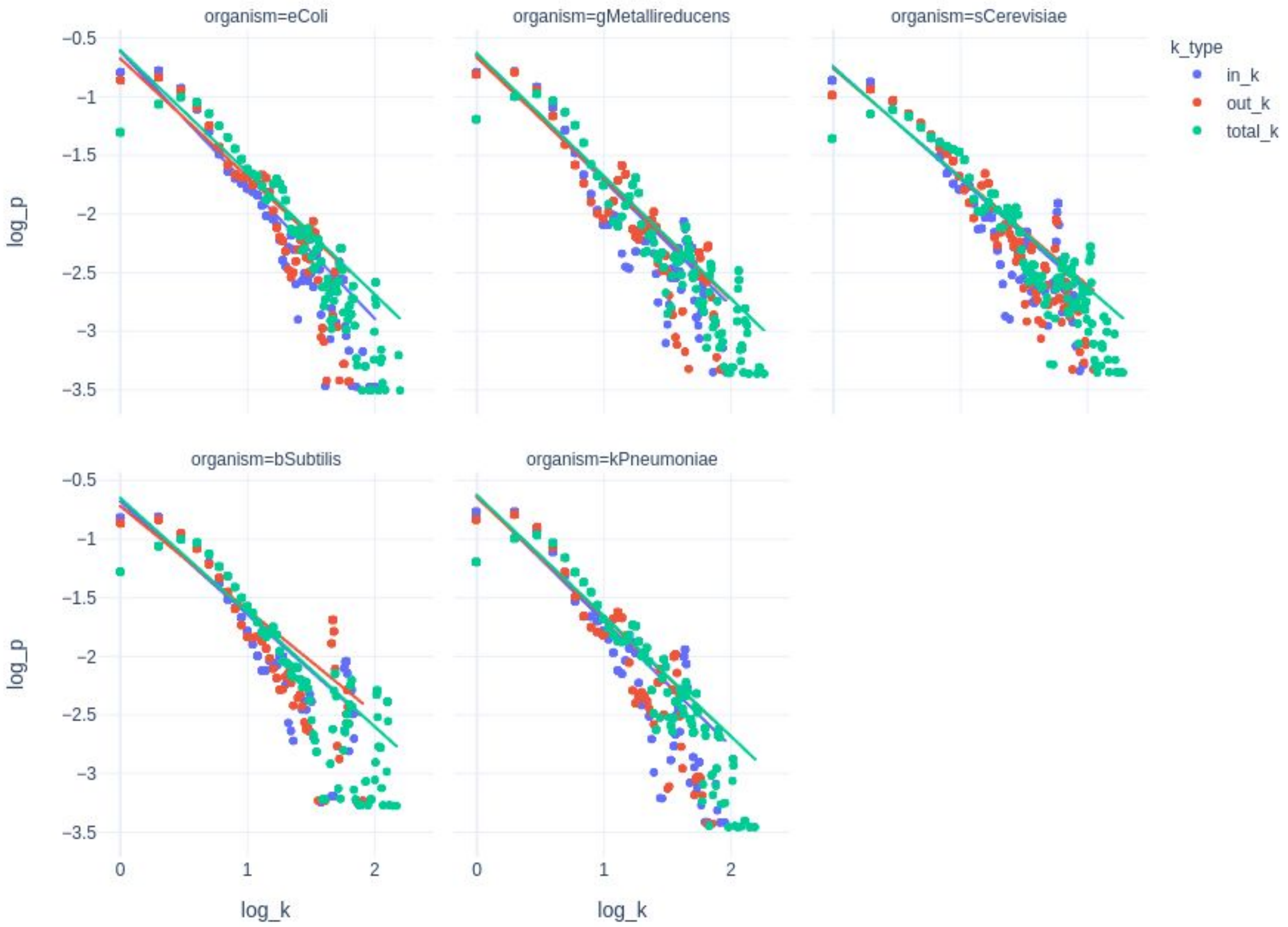
k_total = k_in + k_out

Pr(k) obtained from KDE



```python
def kde_and_score(vec):
    vec = vec.reshape(-1, 1)
    kde = KernelDensity()
    kde.fit(vec)
    scores = kde.score_samples(vec).ravel()
    return np.exp(scores)

def mat_to_kframe(mat, organism_name='e_coli'):
    in_k = mat.sum(axis = 0).values
    out_k = mat.sum(axis = 1).values
    total_k = in_k + out_k


    frame = []
    for name, k_arr in [("in_k", in_k), ("out_k", out_k), ("total_k", total_k)]:
        k_arr = k_arr[k_arr > 0]
        subframe = pd.DataFrame({
            'k_arr' : k_arr,
            'prob' : kde_and_score(k_arr),
            'k_type' : name
        })
        frame.append(subframe)

    frame = pd.concat(frame)

    frame['log_k'] = np.log10(frame.k_arr)
    frame['log_p'] = np.log10(frame.prob)
    frame['organism'] = organism_name

    return frame

kframe_list = [mat_to_kframe(m, o) for m, o in zip(mat_list, org_list)]
k_frame = pd.concat(kframe_list)
```

Our Fig 2.


organism=eColi
organism=gMetallireducens
organism=sCerevisiae
organism=bSubtilis
organism=kPneumoniae
k_type
in_k
out_k
total_k
log_p
log_k
−0.5
−1
−1.5
−2
−2.5
−3
−3.5
0
1
2

Regression Analysis of Node Degrees (k)

Organism	R-Squared	Intercept	Coefficient	P_val
sCerevisiae	0.8564	-0.73135	-0.93891	< 2e-16
gMetallireducens	0.8586	-0.61384	-1.04682	< 2e-16
KPneumonie	0.8515	-0.61567	-1.02967	< 2e-16
bSubtilis	0.8170	-0.64510	-0.97476	< 2e-16
eColi	0.8379	-0.59247	-1.04100	< 2e-16

$$y = mx + b$$

Probability Distribution

Coefficient(Degree)

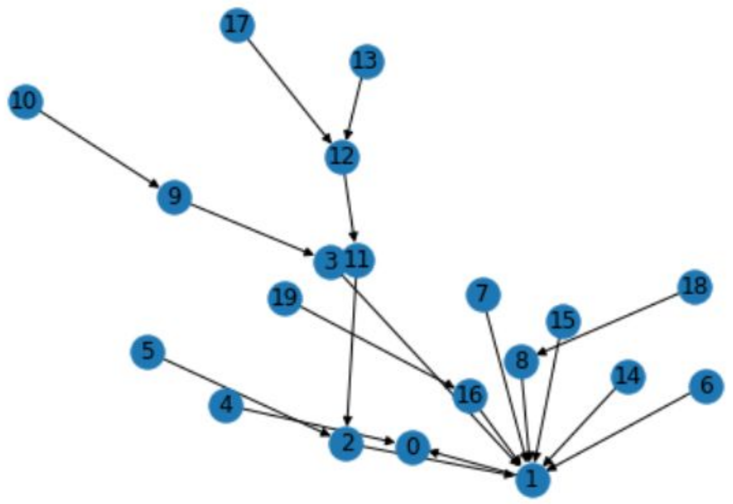
Intercept

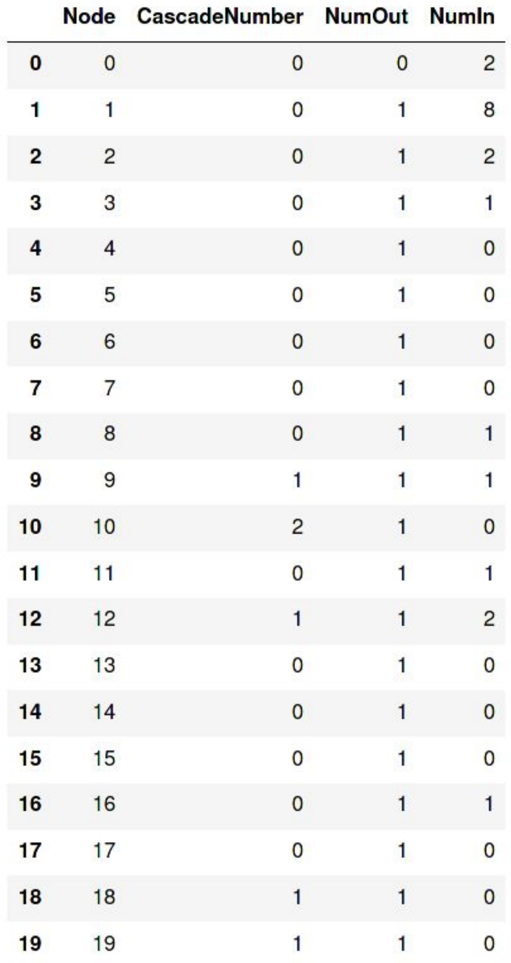
Paper’s Cascade Number Algorithm

Let a directional network be an ordered pair, (V, A), where V is the set of nodes and A is the set of arcs of the network. Then, the cascade set and cascade number are computed by the following algorithm:

```
Cascade(Node, Network(V, A))
    Initialize a queue with the Node
    Mark Node as cut off
        Add all nodes v for which (Node,v) in A to the queue
    Traverse queue
        The node at the front of the queue is q
        If q has no ancestors that are not marked as cut off
            Mark q as cut off
            Add all nodes v for which (q,v) in A to the queue
        End if
    End Traverse
    The cascade set of Node is the set of nodes marked as cut off.
    Cascade number = size of Cascade set
```

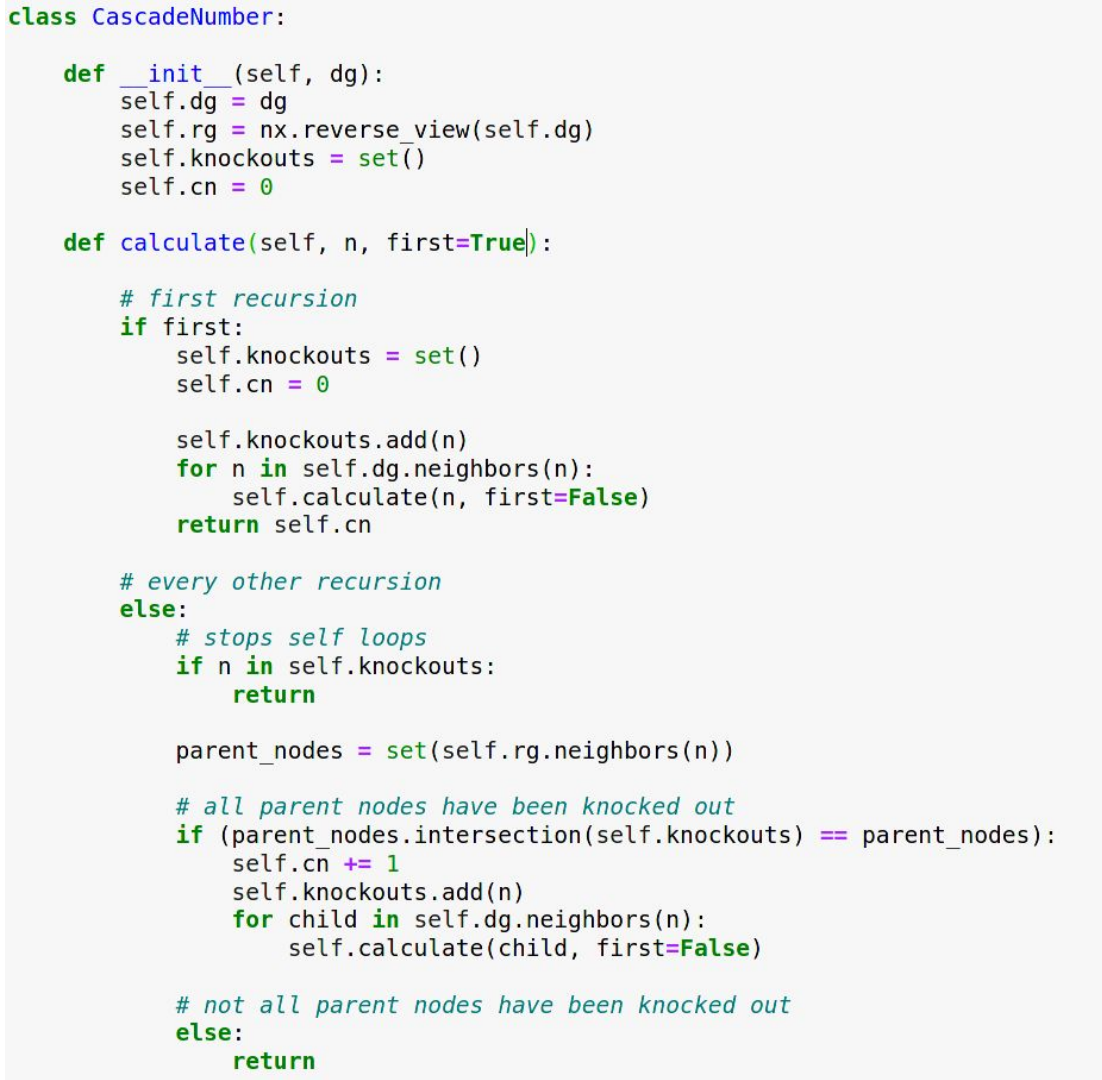
Cascade Number Algorithm





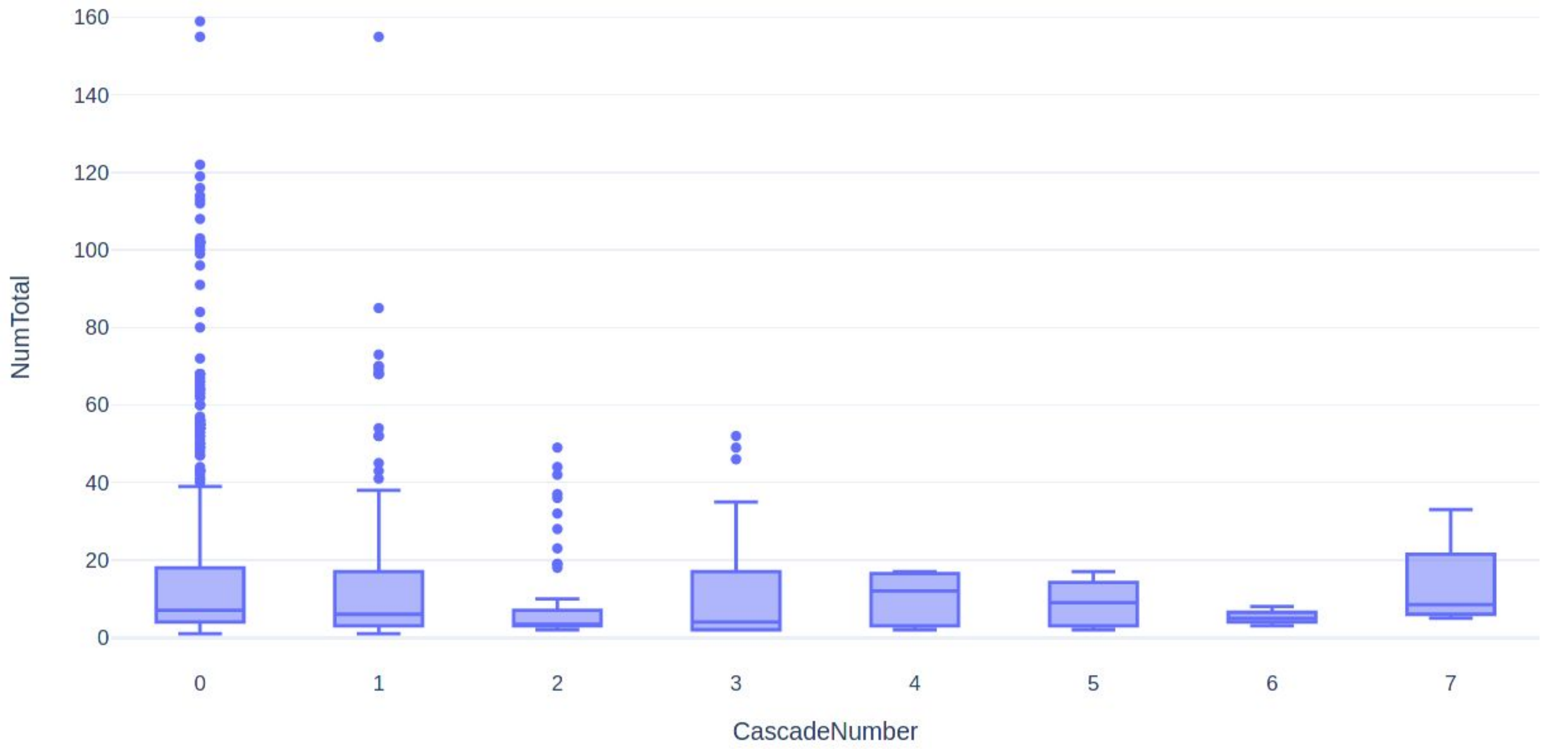
	Node	CascadeNumber	NumOut	NumIn
0	0	0	0	2
1	1	0	1	8
2	2	0	1	2
3	3	0	1	1
4	4	0	1	0
5	5	0	1	0
6	6	0	1	0
7	7	0	1	0
8	8	0	1	1
9	9	1	1	1
10	10	2	1	0
11	11	0	1	1
12	12	1	1	2
13	13	0	1	0
14	14	0	1	0
15	15	0	1	0
16	16	0	1	1
17	17	0	1	0
18	18	1	1	0
19	19	1	1	0

The Code



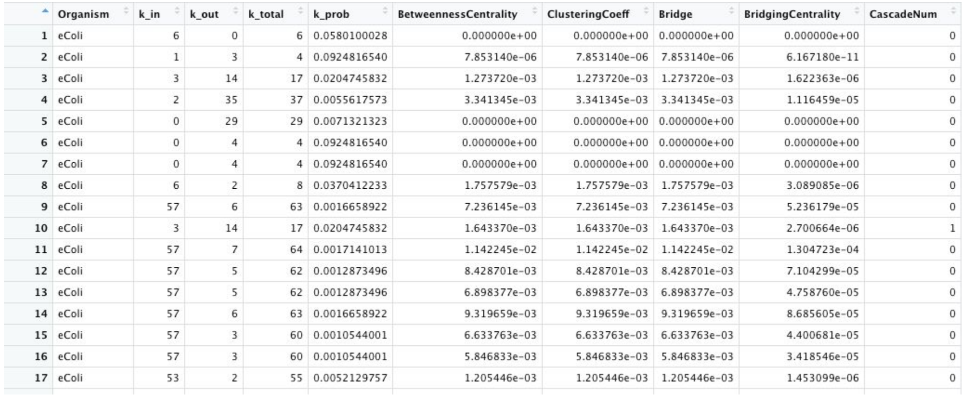
```python
class CascadeNumber:

    def __init__(self, dg):
        self.dg = dg
        self.rg = nx.reverse_view(self.dg)
        self.knockouts = set()
        self.cn = 0

    def calculate(self, n, first=True):

        # first recursion
        if first:
            self.knockouts = set()
            self.cn = 0

            self.knockouts.add(n)
            for n in self.dg.neighbors(n):
                self.calculate(n, first=False)
            return self.cn

        # every other recursion
        else:
            # stops self loops
            if n in self.knockouts:
                return

            parent_nodes = set(self.rg.neighbors(n))

            # all parent nodes have been knocked out
            if (parent_nodes.intersection(self.knockouts) == parent_nodes):
                self.cn += 1
                self.knockouts.add(n)
                for child in self.dg.neighbors(n):
                    self.calculate(child, first=False)

            # not all parent nodes have been knocked out
            else:
                return
```

Correlation of Cascade Number and Degrees



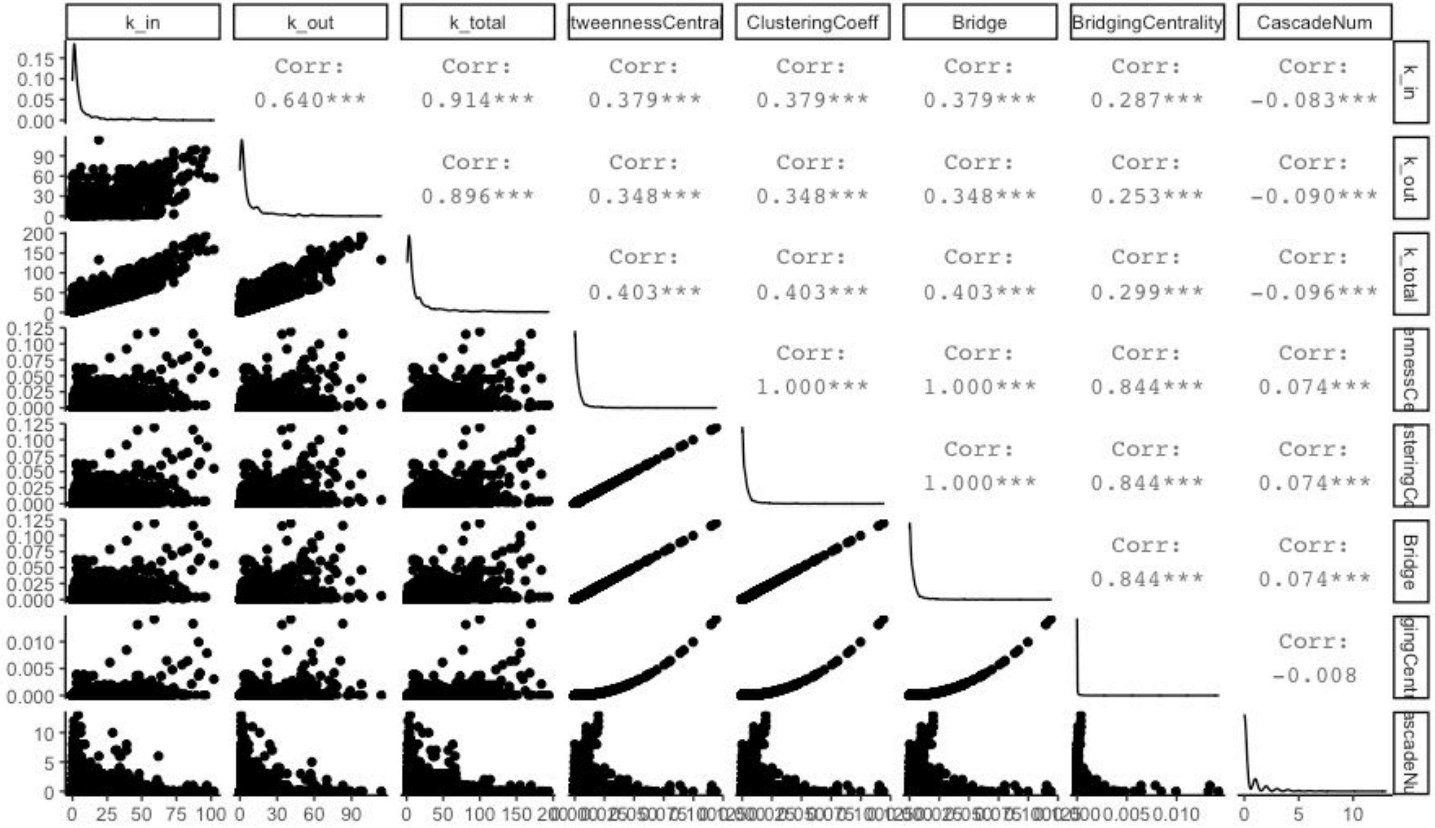
Additional Analysis: Analyzing Centrality Metrics

- 5 different organism
- 4917 nodes total



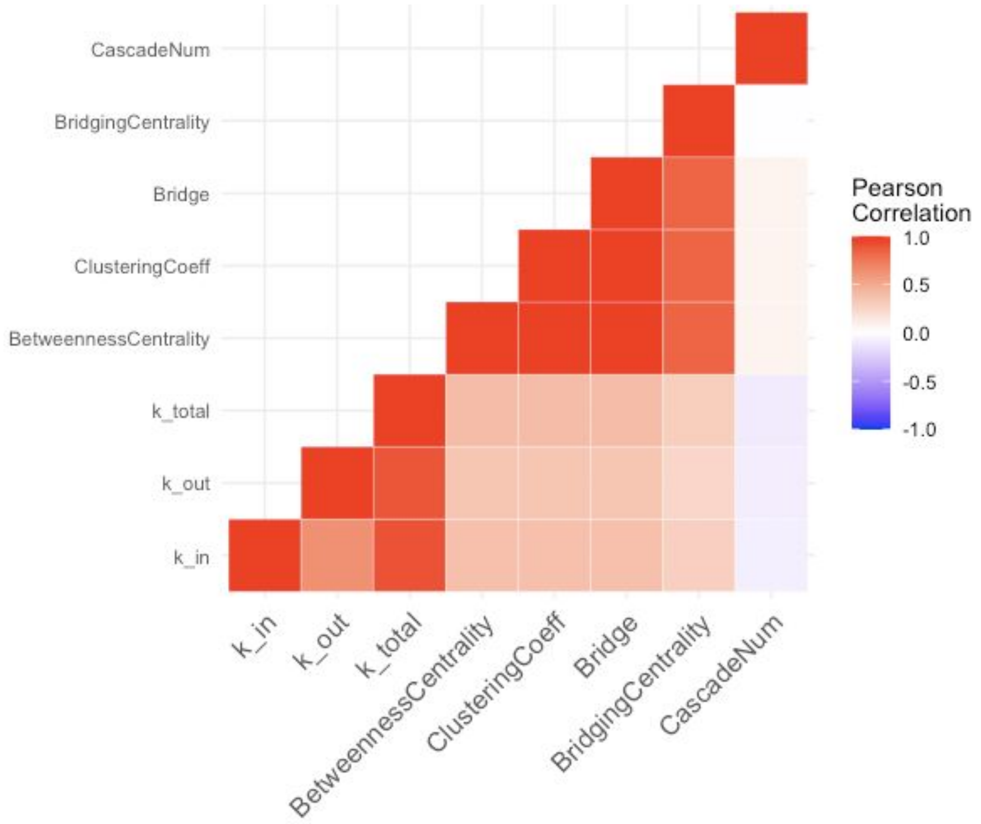
	Organism	k_in	k_out	k_total	k_prob	BetweennessCentrality	ClusteringCoeff	Bridge	BridgingCentrality	CascadeNum
1	eColi	6	0	6	0.0580100028	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
2	eColi	1	3	4	0.0924816540	7.853140e-06	7.853140e-06	7.853140e-06	6.167180e-11	0
3	eColi	3	14	17	0.0204745832	1.273720e-03	1.273720e-03	1.273720e-03	1.622363e-06	0
4	eColi	2	35	37	0.0055617573	3.341345e-03	3.341345e-03	3.341345e-03	1.116459e-05	0
5	eColi	0	29	29	0.0071321323	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
6	eColi	0	4	4	0.0924816540	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
7	eColi	0	4	4	0.0924816540	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0
8	eColi	6	2	8	0.0370412233	1.757579e-03	1.757579e-03	1.757579e-03	3.089085e-06	0
9	eColi	57	6	63	0.0016658922	7.236145e-03	7.236145e-03	7.236145e-03	5.236179e-05	0
10	eColi	3	14	17	0.0204745832	1.643370e-03	1.643370e-03	1.643370e-03	2.700664e-06	1
11	eColi	57	7	64	0.0017141013	1.142245e-02	1.142245e-02	1.142245e-02	1.304723e-04	0
12	eColi	57	5	62	0.0012873496	8.428701e-03	8.428701e-03	8.428701e-03	7.104299e-05	0
13	eColi	57	5	62	0.0012873496	6.898377e-03	6.898377e-03	6.898377e-03	4.758760e-05	0
14	eColi	57	6	63	0.0016658922	9.319659e-03	9.319659e-03	9.319659e-03	8.685605e-05	0
15	eColi	57	3	60	0.0010544001	6.633763e-03	6.633763e-03	6.633763e-03	4.400681e-05	0
16	eColi	57	3	60	0.0010544001	5.846833e-03	5.846833e-03	5.846833e-03	3.418546e-05	0
17	eColi	53	2	55	0.0052129757	1.205446e-03	1.205446e-03	1.205446e-03	1.453099e-06	0

- k_in= Number of In Degrees
- k_out = Number of out Degrees
- k_total=Total Number of Degrees
- BetweennessCentrality= number of times a node acts as the shortest path of the system
- ClusteringCoeff=measure the degree to which nodes will cluster together
- Bridge=measure the number of edges whose removal causes the number of connected components to increase
- BridgingCentrality=Measure the extent of bridging capability. CC*Br
- CascadeNumber=Controllability of the graph



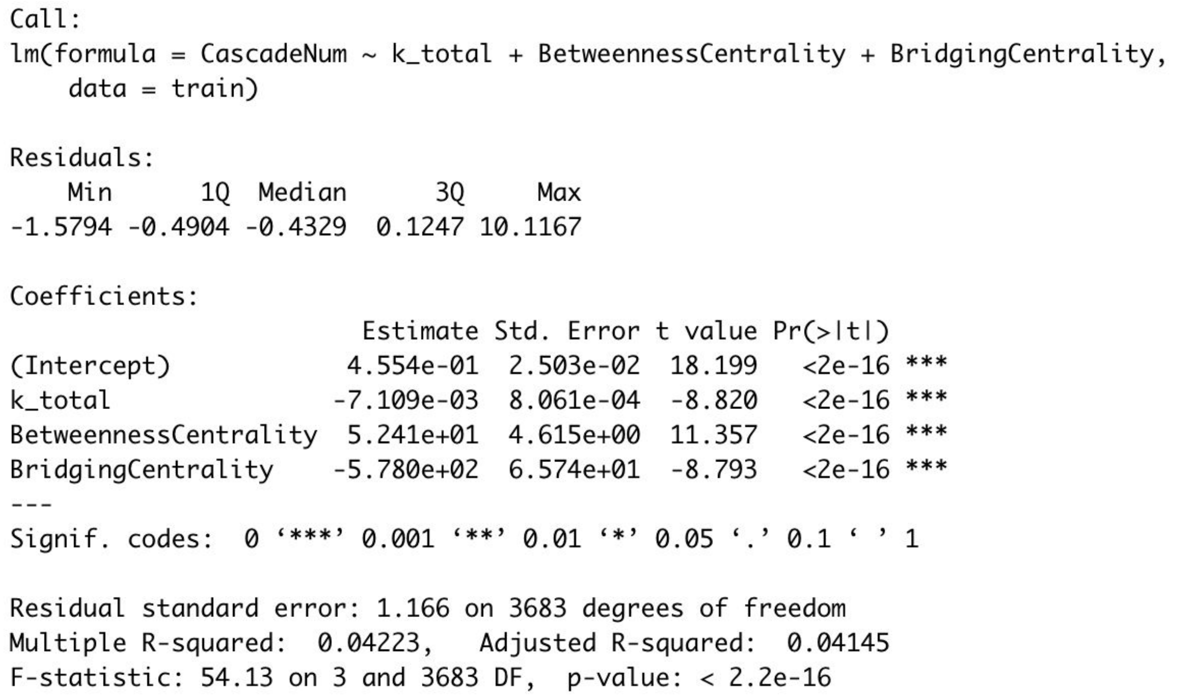
k_in
k_out
k_total
tweennessCentra
ClusteringCoeff
Bridge
BridgingCentrality
CascadeNum
Corr: 0.640***
Corr: 0.914***
Corr: 0.379***
Corr: 0.379***
Corr: 0.379***
Corr: 0.287***
Corr: -0.083***
Corr: 0.896***
Corr: 0.348***
Corr: 0.348***
Corr: 0.348***
Corr: 0.253***
Corr: -0.090***
Corr: 0.403***
Corr: 0.403***
Corr: 0.403***
Corr: 0.299***
Corr: -0.096***
Corr: 1.000***
Corr: 1.000***
Corr: 0.844***
Corr: 0.074***
Corr: 1.000***
Corr: 0.844***
Corr: 0.074***
Corr: 0.844***
Corr: 0.074***
Corr: -0.008

Correlation Matrix(pearson), of each centrality metric



Cascade number does not correlate well with any of the other statistics

Create a model to test sensitivity of Cascade Number



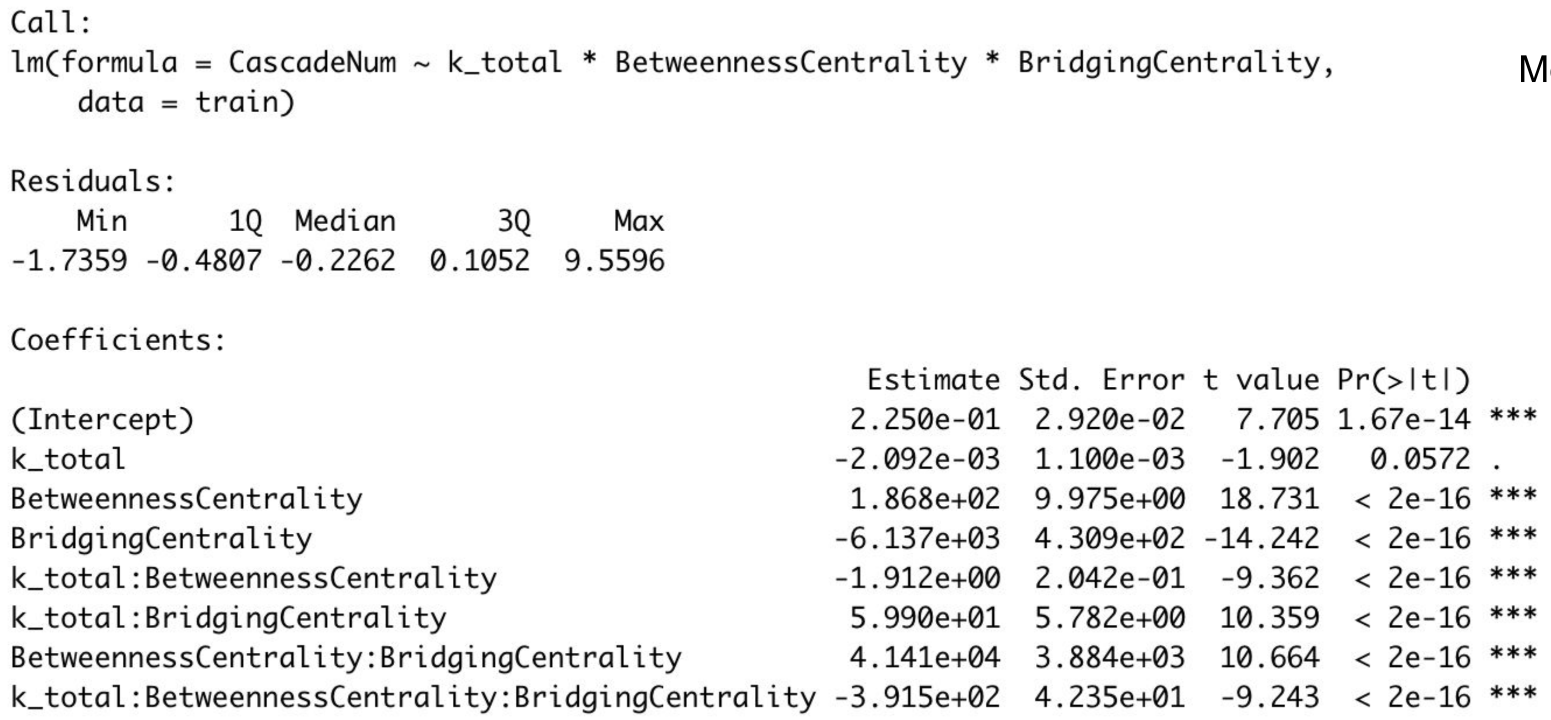
```
Call:
lm(formula = CascadeNum ~ k_total + BetweennessCentrality + BridgingCentrality,
    data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5794 -0.4904 -0.4329  0.1247 10.1167 

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)    
(Intercept)            4.554e-01  2.503e-02  18.199   <2e-16 ***
k_total               -7.109e-03  8.061e-04  -8.820   <2e-16 ***
BetweennessCentrality  5.241e+01  4.615e+00  11.357   <2e-16 ***
BridgingCentrality    -5.780e+02  6.574e+01  -8.793   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.166 on 3683 degrees of freedom
Multiple R-squared:  0.04223,	Adjusted R-squared:  0.04145 
F-statistic: 54.13 on 3 and 3683 DF,  p-value: < 2.2e-16
```

Model 1

Excluded Bridge,Clustering Coefficient, k_in, k_out due to singularities.

Model 2: Interaction of Variables

Model 2



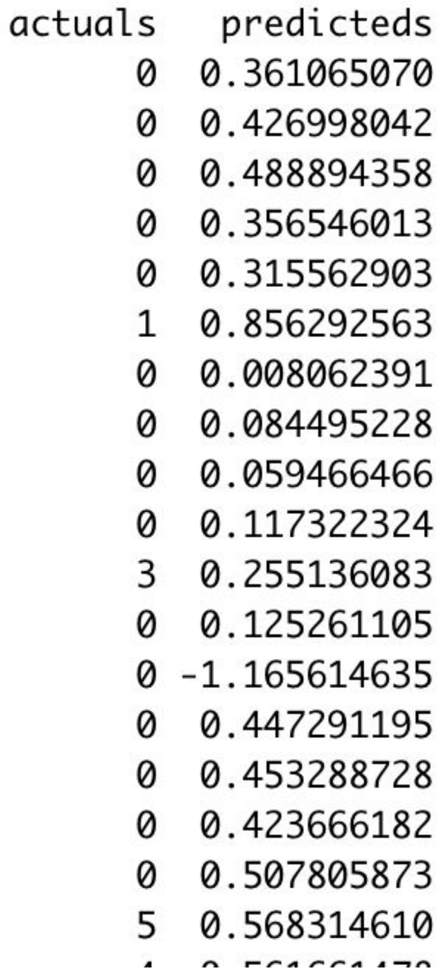
```
Call:
lm(formula = CascadeNum ~ k_total * BetweennessCentrality * BridgingCentrality,
    data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.7359 -0.4807 -0.2262  0.1052  9.5596 

Coefficients:
                                                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)                                     2.250e-01  2.920e-02   7.705 1.67e-14 ***
k_total                                        -2.092e-03  1.100e-03  -1.902   0.0572 .  
BetweennessCentrality                           1.868e+02  9.975e+00  18.731  < 2e-16 ***
BridgingCentrality                             -6.137e+03  4.309e+02 -14.242  < 2e-16 ***
k_total:BetweennessCentrality                  -1.912e+00  2.042e-01  -9.362  < 2e-16 ***
k_total:BridgingCentrality                      5.990e+01  5.782e+00  10.359  < 2e-16 ***
BetweennessCentrality:BridgingCentrality        4.141e+04  3.884e+03  10.664  < 2e-16 ***
k_total:BetweennessCentrality:BridgingCentrality -3.915e+02  4.235e+01  -9.243  < 2e-16 ***
```

Tested to Validate the data

Created 80/20 split of training to test data for these models.

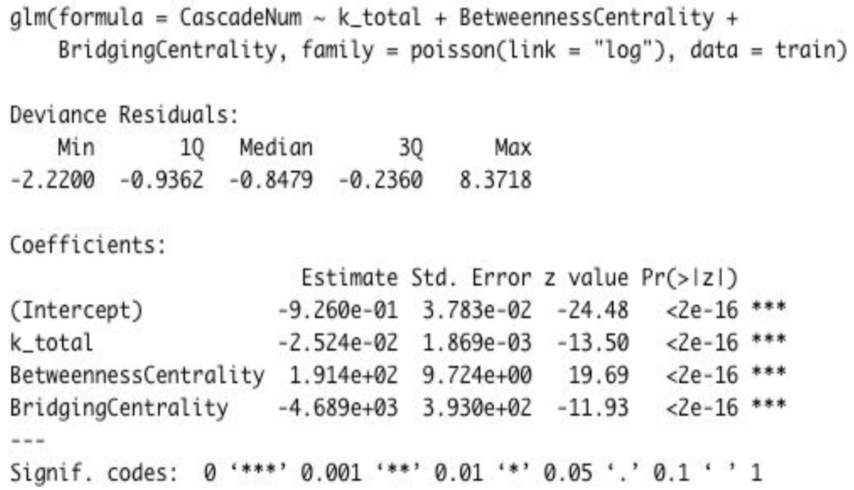
Model 1	58% accuracy
Model 2	59% accuracy

Cascade Number is Discrete!
I need to change my model to predict a discrete feature and not continous

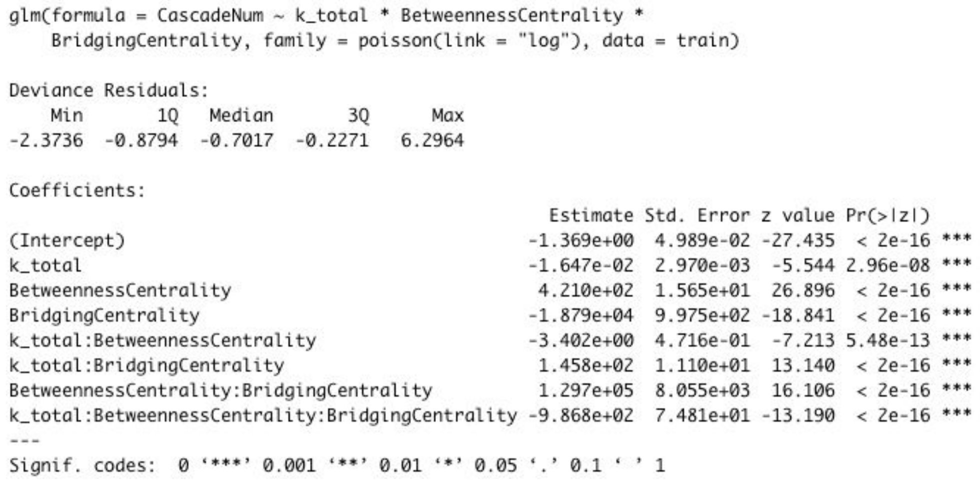


```
actuals   predicteds
      0  0.361065070
      0  0.426998042
      0  0.488894358
      0  0.356546013
      0  0.315562903
      1  0.856292563
      0  0.008062391
      0  0.084495228
      0  0.059466466
      0  0.117322324
      3  0.255136083
      0  0.125261105
      0 -1.165614635
      0  0.447291195
      0  0.453288728
      0  0.423666182
      0  0.507805873
      5  0.568314610
```

Creating a GLM under a poisson distribution



```
glm(formula = CascadeNum ~ k_total + BetweennessCentrality +
    BridgingCentrality, family = poisson(link = "log"), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2200  -0.9362  -0.8479  -0.2360   8.3718

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -9.260e-01  3.783e-02  -24.48   <2e-16 ***
k_total               -2.524e-02  1.869e-03  -13.50   <2e-16 ***
BetweennessCentrality  1.914e+02  9.724e+00   19.69   <2e-16 ***
BridgingCentrality    -4.689e+03  3.930e+02  -11.93   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



```
glm(formula = CascadeNum ~ k_total * BetweennessCentrality *
    BridgingCentrality, family = poisson(link = "log"), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3736  -0.8794  -0.7017  -0.2271   6.2964

Coefficients:
                                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                                     -1.369e+00  4.989e-02 -27.435  < 2e-16 ***
k_total                                         -1.647e-02  2.970e-03  -5.544 2.96e-08 ***
BetweennessCentrality                            4.210e+02  1.565e+01  26.896  < 2e-16 ***
BridgingCentrality                              -1.879e+04  9.975e+02 -18.841  < 2e-16 ***
k_total:BetweennessCentrality                   -3.402e+00  4.716e-01  -7.213 5.48e-13 ***
k_total:BridgingCentrality                       1.458e+02  1.110e+01  13.140  < 2e-16 ***
BetweennessCentrality:BridgingCentrality         1.297e+05  8.055e+03  16.106  < 2e-16 ***
k_total:BetweennessCentrality:BridgingCentrality -9.868e+02  7.481e+01 -13.190  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

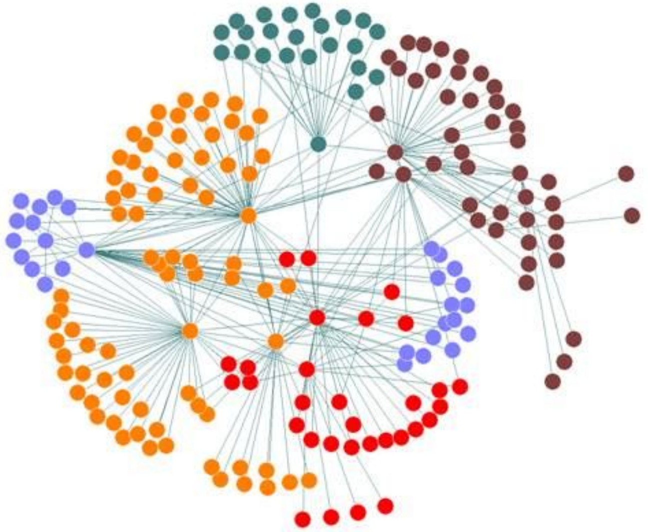
Model 3	64% Accuracy
Model 4	64% Accuracy

Lessons/Challenges

- Data was limited
  - Edge direction was not given in adjacency matrix
  - More network metrics would be helpful
- Translating cascade algorithm from pseudocode
- Discrete vs Continuous predictive modeling

Nice job on the presentation! Clear background and description of the network; good work fitting a linear model and reimplementing cascade number as a recursion. As we discussed in class, the high CV accuracy despite low R-square is a tip-off that there may be leakage between the training and test sets. Unbalanced classes also affect performance metrics.

Grade: 27/30



QUESTIONS