##
This is just a summary for a very rough and preliminary study about network detection theory advised by Prof. Santo. Raw data and block number was collected/calculated by Lucas.

Test john55 dataset.

John55.gml is a social network with about 5000 nodes and many edges.
Introduction:
Stochastic block model is a method to detect communities in a network.
Find the necessary information for stochastic block model in this paper:
https://arxiv.org/abs/1008.3926
We are going to test if the stochastic block model can work for most practical data in real world. Here is what we want to test and the procedure:

1. Find 2 communities A and B detected by stochastic block model.
2. For community A, find 2 nodes i and j. Calculate Ki and Kj, where K is number of the nodes connected to them (the degree of a vortex).
3. Find Ki_B and Kj_B, where K_B is the number of neighbors in community B.
4. Plot Ki_B/Kj_B ~ Ki/Kj, if it's a straight y~x line?

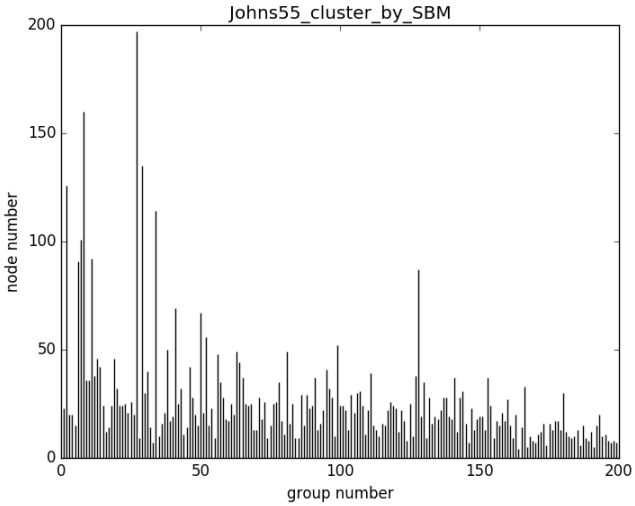
First, cluster the nextwork

1. **Clustered by graph_tool.minimized_blockmodel_ml()**

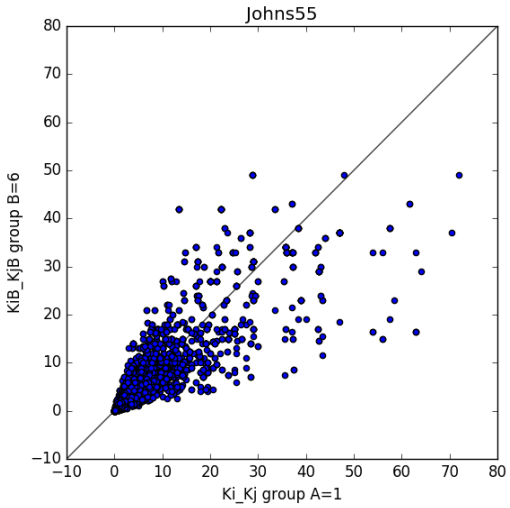
Instead of my naïve implementation of Newman's stochastic block model, we can just use igraph. I'll use 2 igraph methods and then show the Ki_B/Kj_B ~ Ki/Kj plots for each.
graph-tool python package is fast with help of some C/C++ library so the method is almost linear O(NlogNlogN) and it's the only available stochastic block model (SBM) method I can use for now instead of the code I wrote for SBM.
Document for the function.
https://graph-tool.skewed.de/static/doc/demos/inference/inference.html

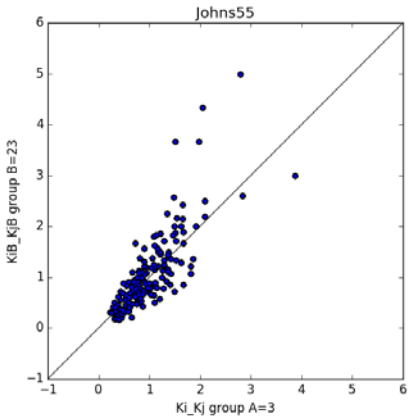
Idea is the same, maximizing likelihood P(G|{bi}), {bi} is the set of parameters for block, but the procedure is different from Newman's 2011 paper. It doesn't always pick the move with largest delta_L. It uses the method in this paper:
Tiago P. Peixoto, "Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models", Phys. Rev. E 89, 012804 (2014), DOI: 10.1103/PhysRevE.89.012804, arXiv: 1310.4378

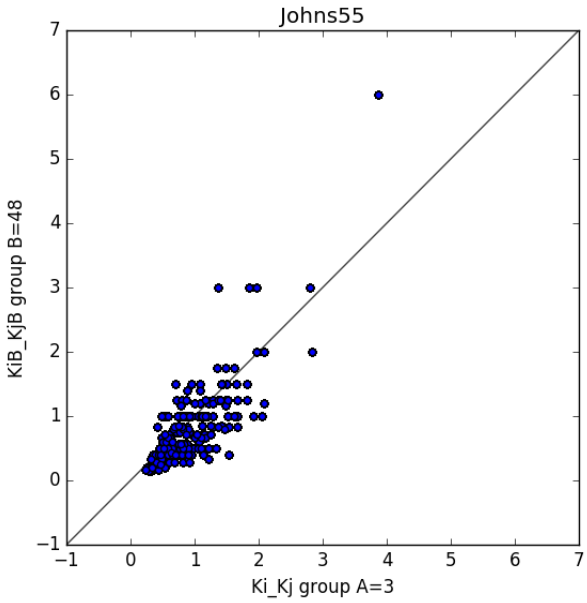
The method was tested in network with size from N=369 to N=654 782 in the paper.

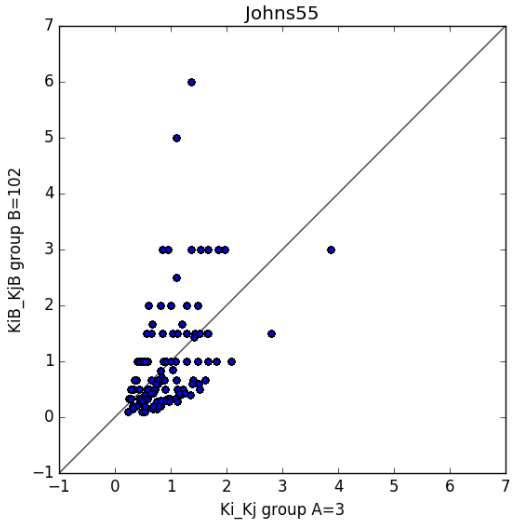


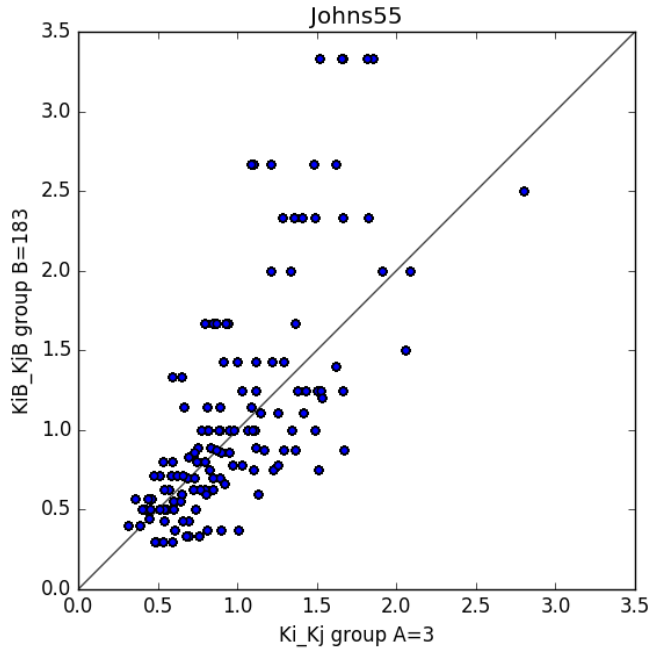
Then plot the Ki_B/Kj_B ~ Ki/Kj
list some significant plots:

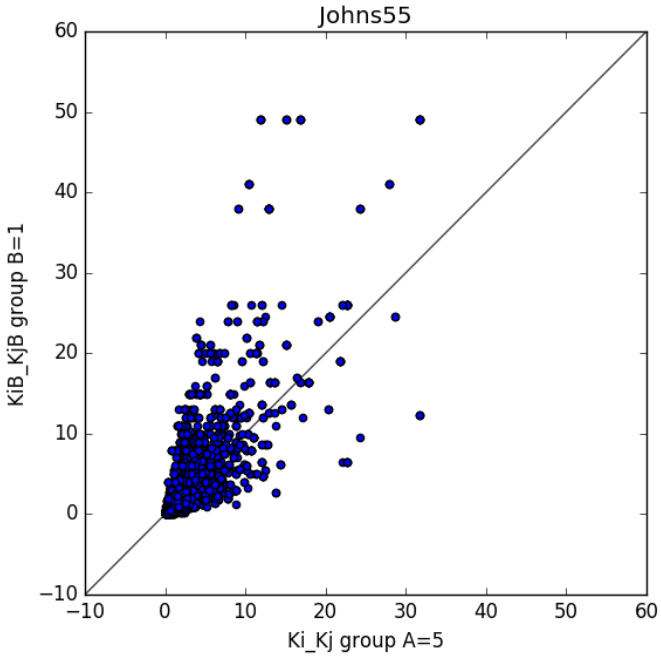


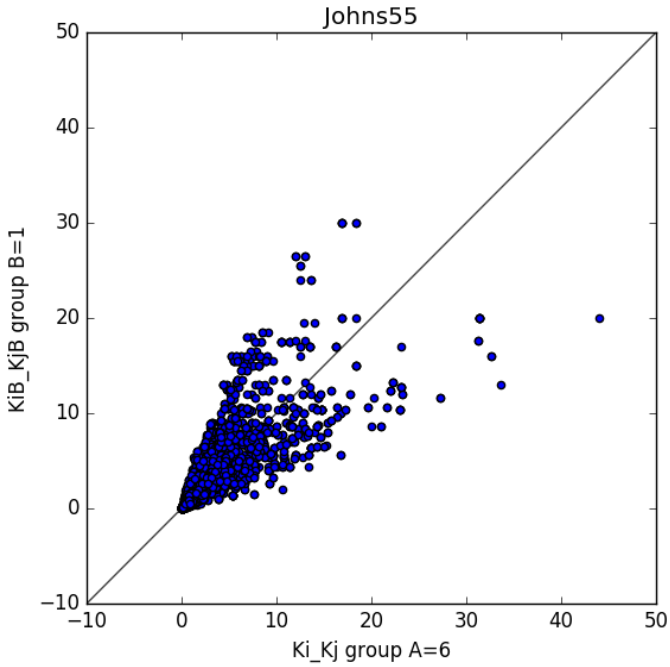


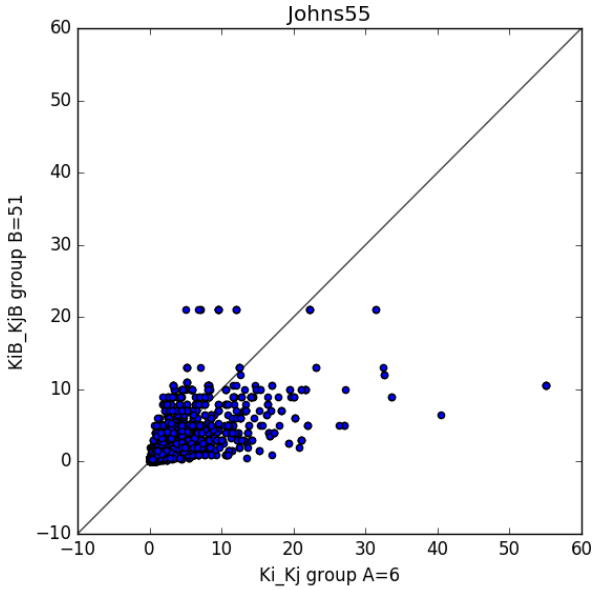

Johns55
KiB_KjB group B=48
Ki_Kj group A=3

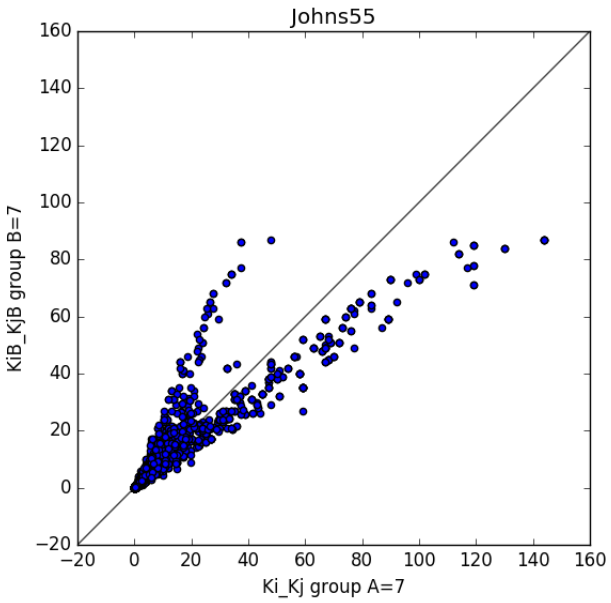

Johns55
KiB_KjB group B=102
Ki_Kj group A=3

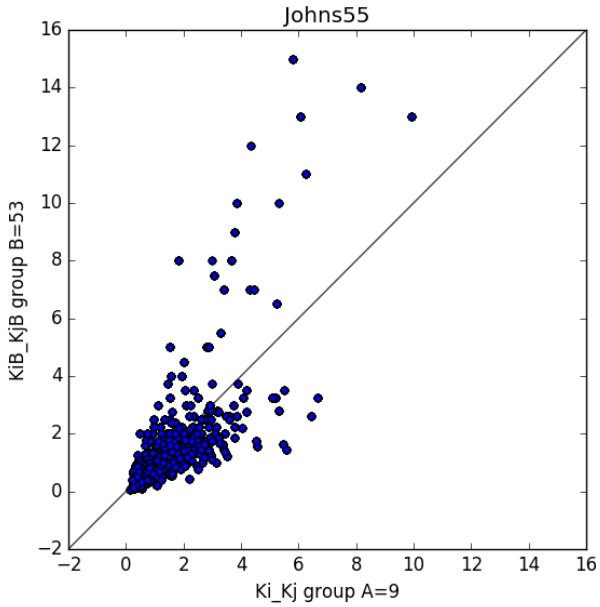

Johns55
KiB_KjB group B=183
Ki_Kj group A=3
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

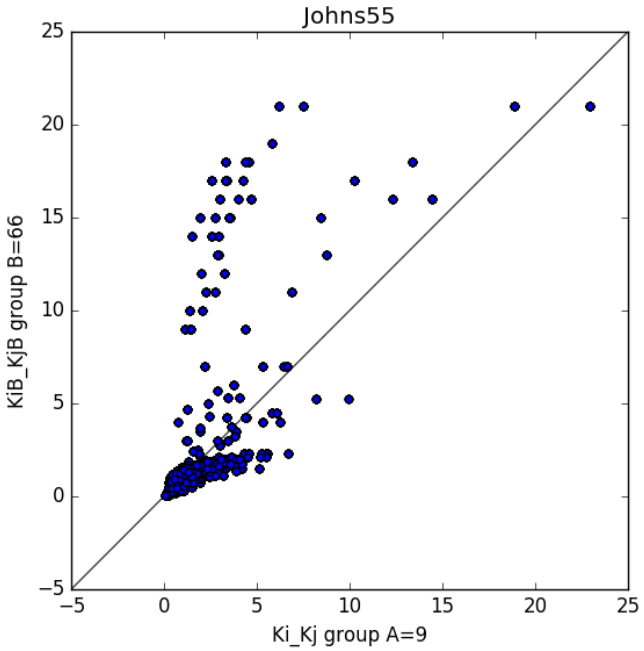

Johns55
KiB_KjB group B=1
Ki_Kj group A=5
−10
0
10
20
30
40
50
60

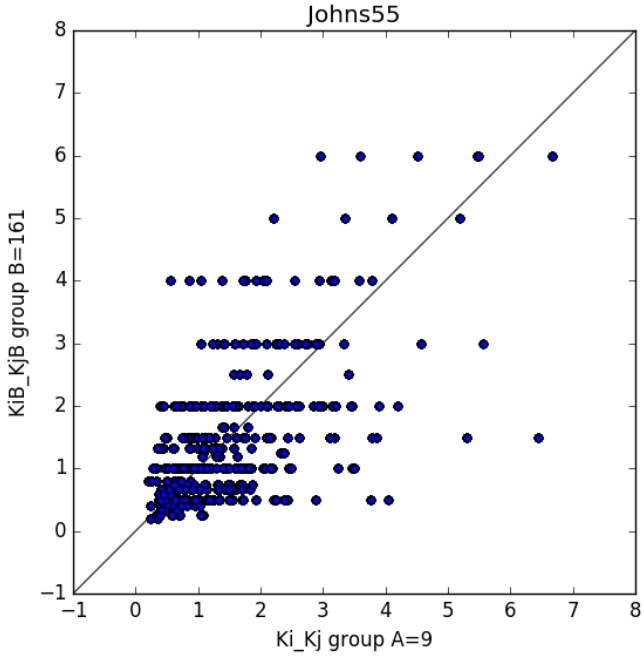

Johns55
KiB_KjB group B=1
Ki_Kj group A=6
50
40
30
20
10
0
−10
−10
0
10
20
30
40
50

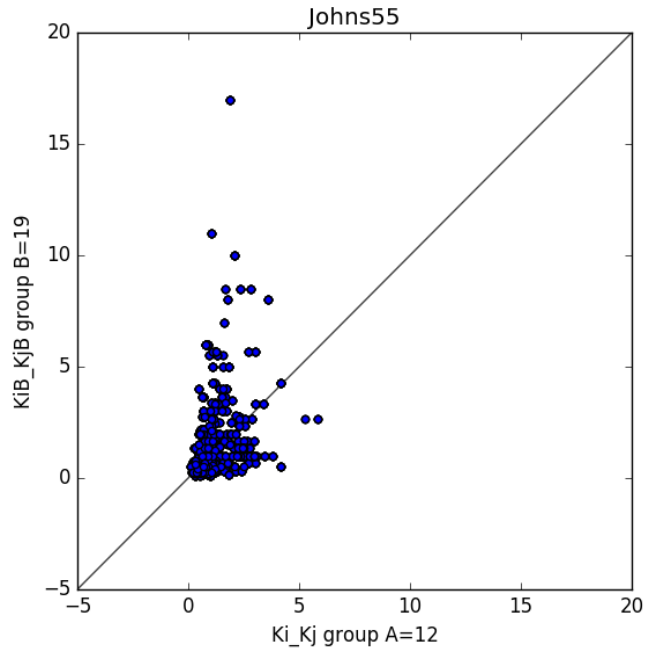

Johns55
KiB_KjB group B=51
Ki_Kj group A=6
60
50
40
30
20
10
0
−10
−10
0
10
20
30
40
50
60

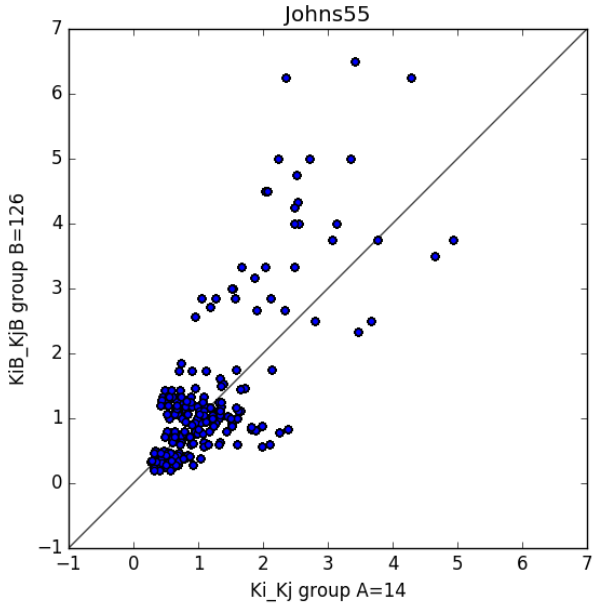

Johns55
KiB_KjB group B=7
Ki_Kj group A=7
160
140
120
100
80
60
40
20
0
−20
−20
0
20
40
60
80
100
120
140
160

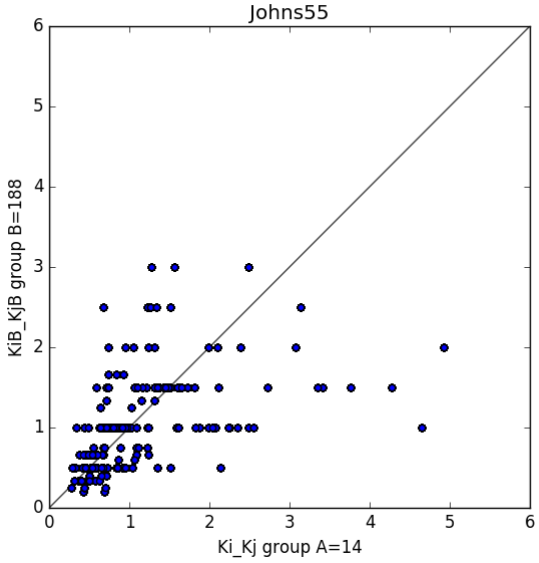

Johns55
KiB_KjB group B=53
Ki_Kj group A=9
16
14
12
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
12
14
16

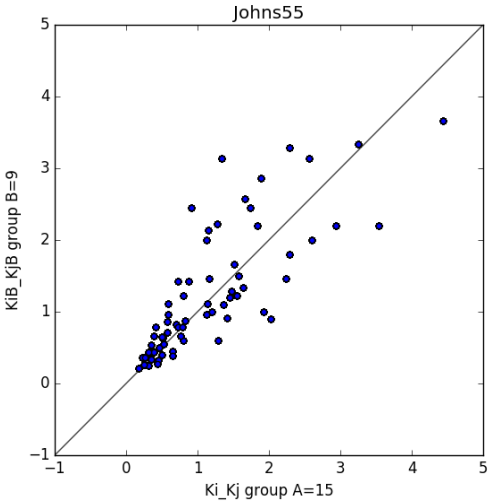

Johns55
KiB_KjB group B=66
Ki_Kj group A=9
25
20
15
10
5
0
−5
−5
0
5
10
15
20
25

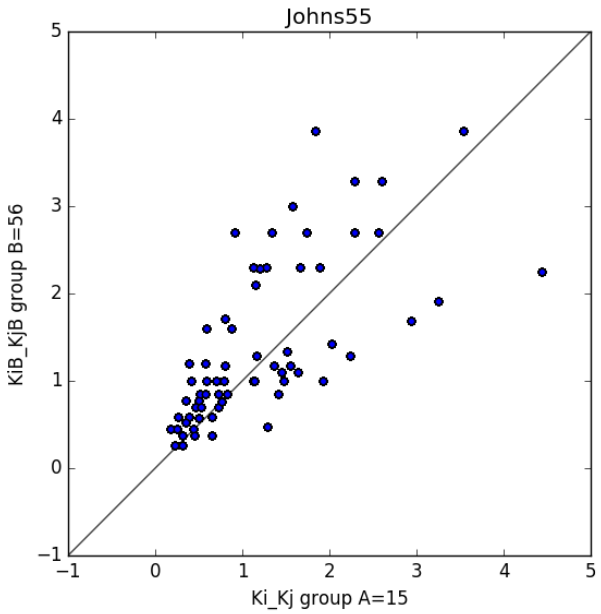

Johns55
KiB_KjB group B=161
Ki_Kj group A=9
8
7
6
5
4
3
2
1
0
−1
−1
0
1
2
3
4
5
6
7
8


Johns55
KiB_KjB group B=19
Ki_Kj group A=12
20
15
10
5
0
−5
−5
0
5
10
15
20


Johns55
KiB_KjB group B=126
Ki_Kj group A=14
7
6
5
4
3
2
1
0
−1
−1
0
1
2
3
4
5
6
7

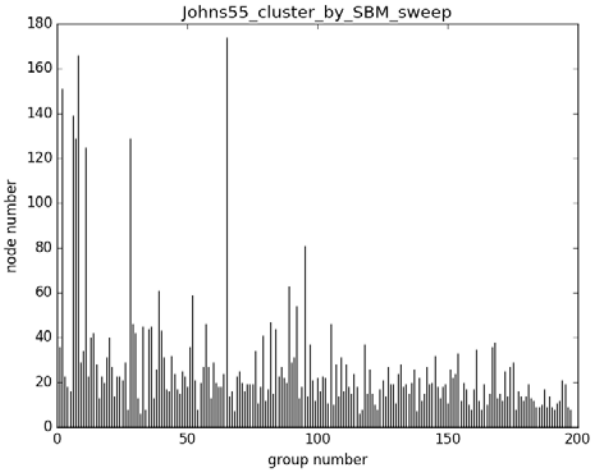

Johns55
KiB_KjB group B=188
Ki_Kj group A=14
0
1
2
3
4
5
6

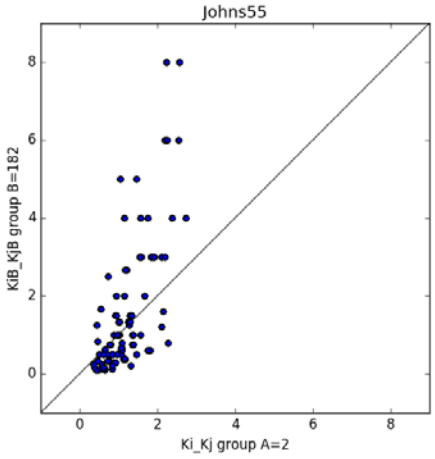

Johns55
KiB_KjB group B=9
Ki_Kj group A=15
−1
0
1
2
3
4
5

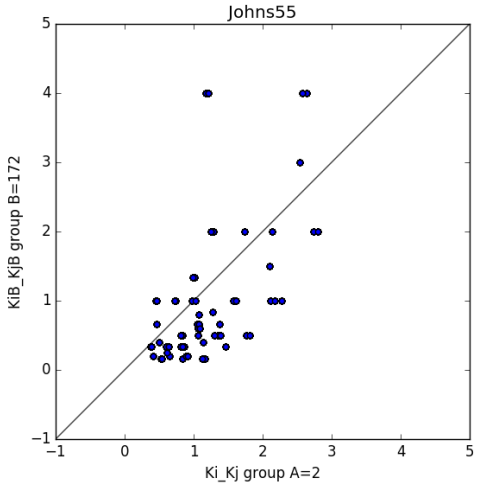


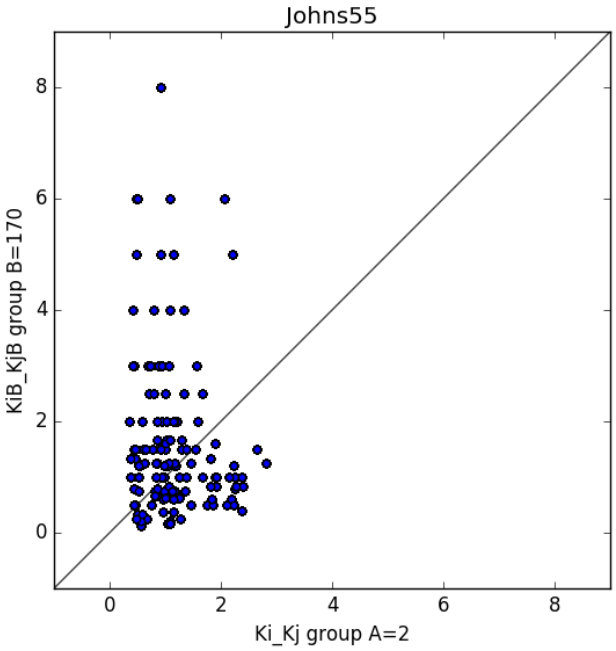
2. Clustered by graph_tool.minimized_blockmodel_ml() and mcmc_equilibate()

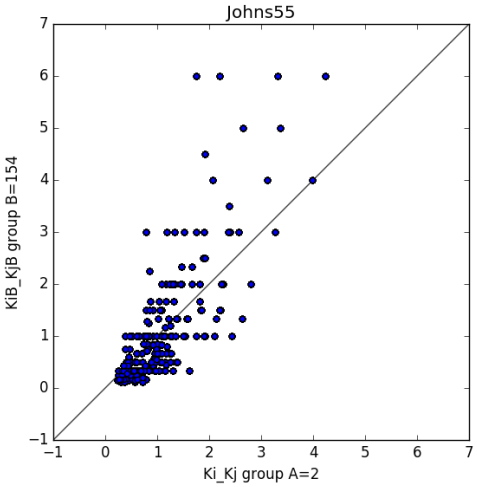
However graph_tool.minimized_blockmodel_ml() cannot correctly cluster karate.gml into 2 groups, it always results in one singe group. Maybe it thinks the whole thing is more likely to be a single community, which makes me suspicious on its correctness.
An improvement is mcmc_equilibrate(), which can avoid metastate, and use the state obtained by graph_tool.minimized_blockmodel_ml() as initial state.

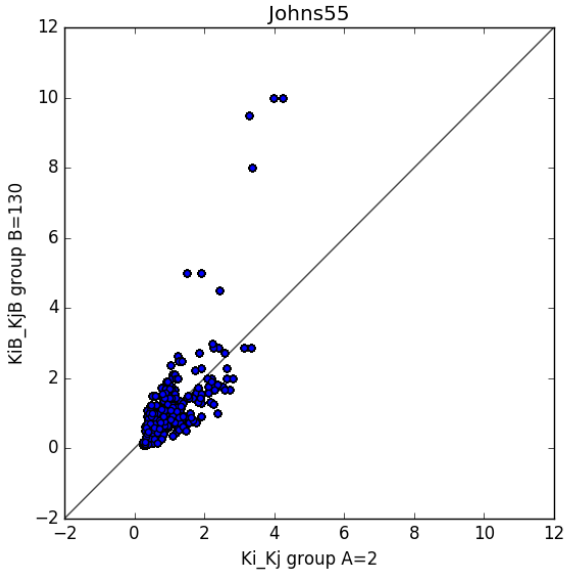

Johns55_cluster_by_SBM_sweep
node number
group number
0
20
40
60
80
100
120
140
160
180
0
50
100
150
200

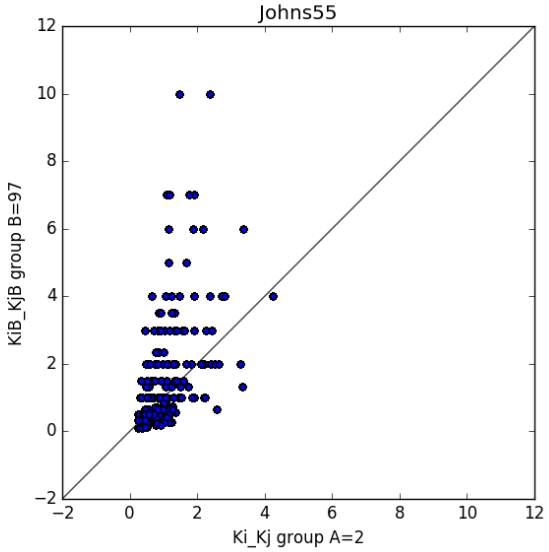

Johns55
KiB_KjB group B=182
Ki_Kj group A=2
0
2
4
6
8

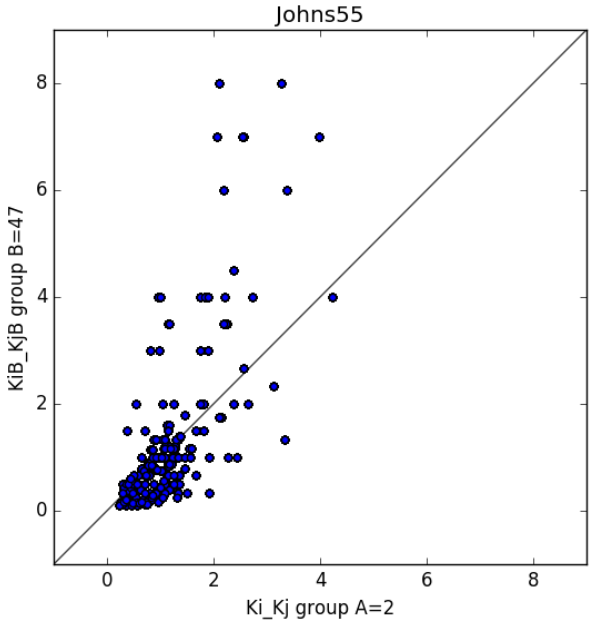

Johns55
KiB_KjB group B=172
Ki_Kj group A=2
−1
0
1
2
3
4
5

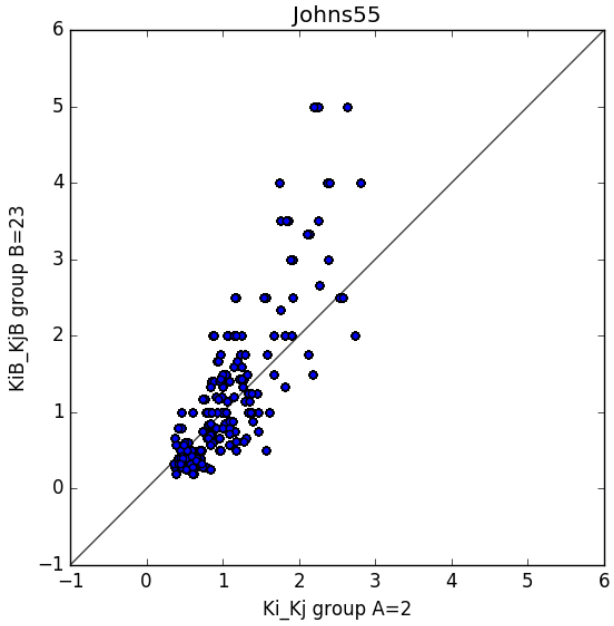

Johns55
KiB_KjB group B=170
Ki_Kj group A=2
0
2
4
6
8

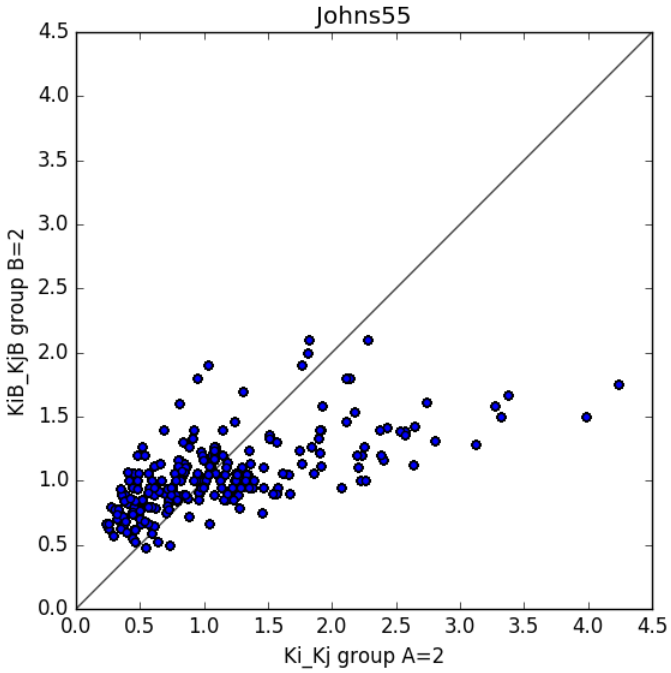

Johns55
KiB_KjB group B=154
Ki_Kj group A=2
−1
0
1
2
3
4
5
6
7

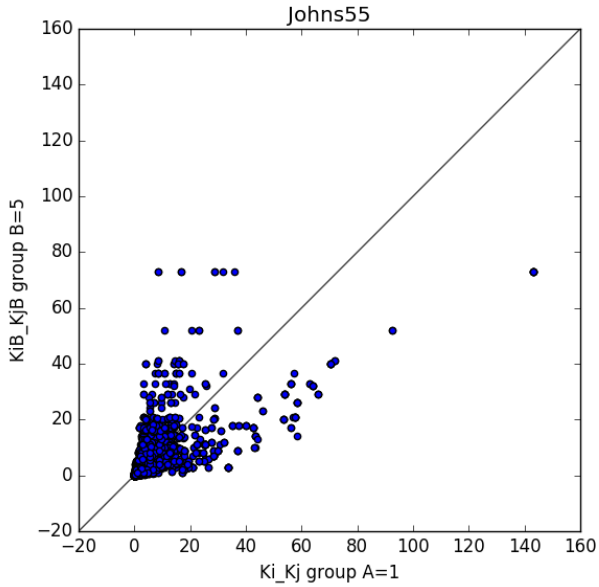

Johns55
KiB_KjB group B=130
Ki_Kj group A=2
12
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
12

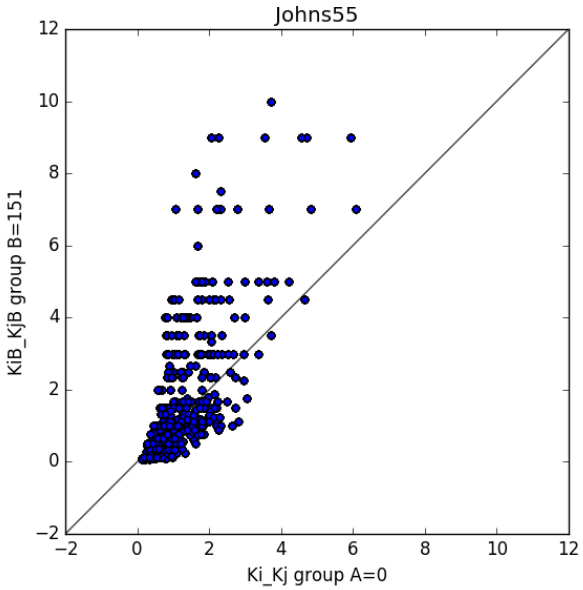

Johns55
KiB_KjB group B=97
Ki_Kj group A=2
12
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
12

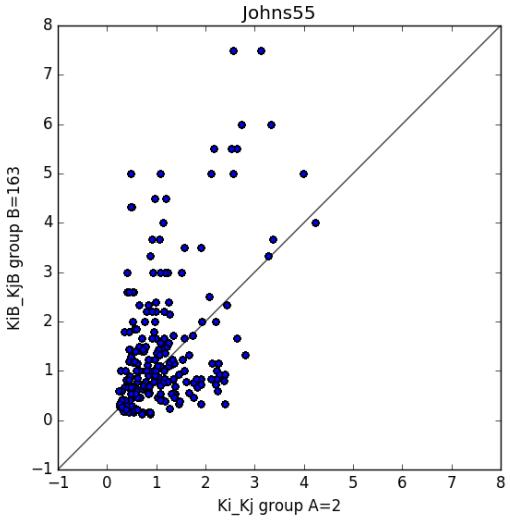

Johns55
KiB_KjB group B=47
Ki_Kj group A=2
0
2
4
6
8

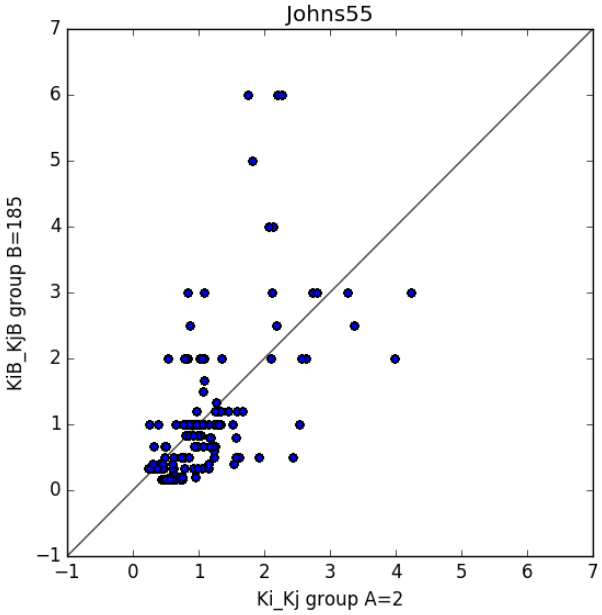

Johns55
KiB_KjB group B=23
Ki_Kj group A=2
−1
0
1
2
3
4
5
6


Johns55
KiB_KjB group B=2
Ki_Kj group A=2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5


Johns55
KiB_KjB group B=5
Ki_Kj group A=1
−20
0
20
40
60
80
100
120
140
160


Johns55
KiB_KjB group B=151
Ki_Kj group A=0


Johns55
KiB_KjB group B=163
Ki_Kj group A=2

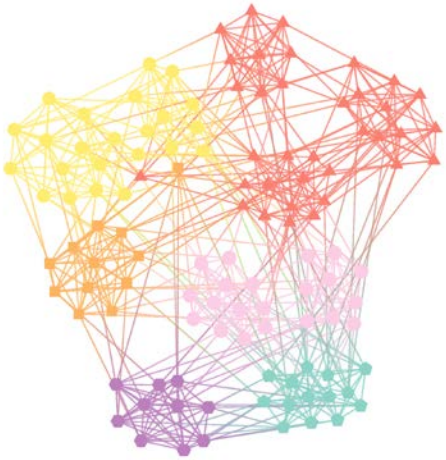


Test both methods using karate.gml and football.gml

Neither can cluster karate.gml correctly. Maybe it's because the size of network is too small.

The second method is better for football.gml. Divided it into 6 groups.

first method



second method