Reconstructing semantic networks - 11/1/2015

Overview

The following simulations reproduce a sample of 100 known graphs using 5 observed lists. Except when otherwise noted, the comparisons between reconstruction methods use exactly the same graphs and lists (Xs). For each simulation, all other parameters except the test parameter are being held constant. The known graphs were generated using a small-world process and contain 15 nodes and 30 edges.

The graphs were evaluated using 3 processes:

- **RW** simply connects all items in the observed lists by an edge;
- **w/o IRT** is identical to INVITE;
- **w/ IRT** is our extension which includes IRTs.

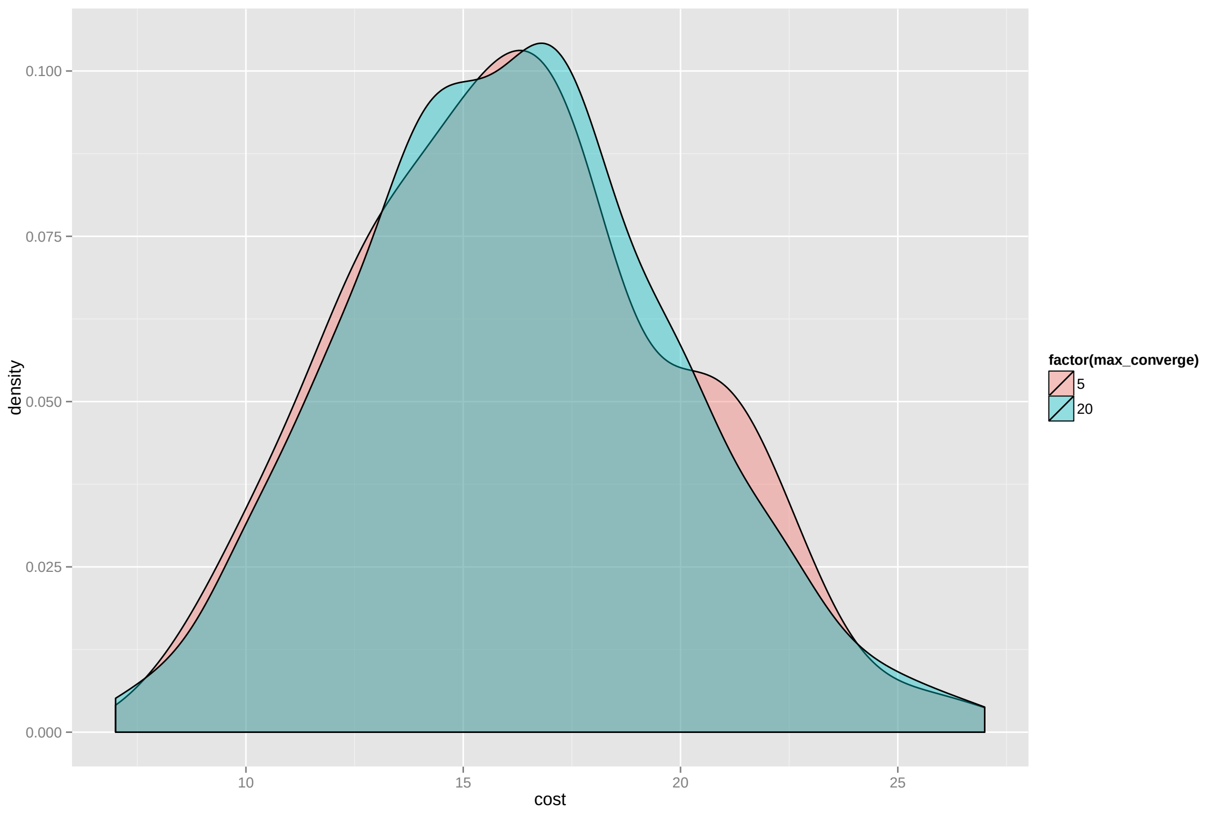
The cost of each method reflects the number of edge changes (additions or deletions) required to obtain the true graph from the best fitting graph.

Preliminary: convergence parameter

As a preliminary step, I want to make sure the graph search process is converging, so I tried two tolerance parameters. The process converged reasonably well with a small tolerance parameter ("max converge"), though the fitting is still fairly slow, and is considerably slower with larger graphs and more lists. It's also likely that if the graph size were increased, the search parameters will need to be altered to ensure that it explores the space appropriately. Currently this process is still very ad hoc.

max_converge	RW cost	w/ IRT	w/o IRT
5	23.77	16.97	15.44
20	23.77	17.08	15.30

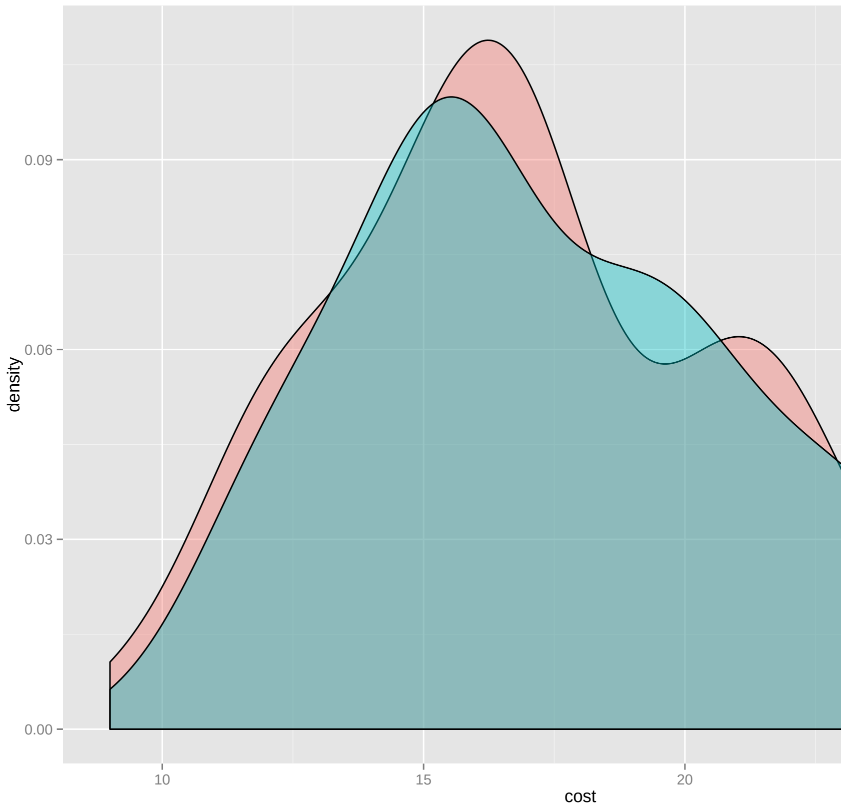
The process seems fairly stable using a low tolerance, and for whatever reason the fit when using IRTs actually got a bit worse when using a higher tolerance. Note that our methods perform better than RW, but using IRTs actually makes the graph fit worse; I'll get to that later. Here is the distribution of graph costs, excluding the "RW" method but collapsing over IRT and no-IRT graphs. Note there are 105 possible edges in the network, so the maximum possible cost is 105 A cost of zero would be identical to reproducing the original graph.



Preliminary: beta parameter for IRTs

To fit the IRTs we need to specify two parameters: a beta parameter used in the Gamma function, and an offset value to prevent log(0) errors. From previous simulations, I had a rough idea of what this parameter should be; here I test its resilience with two values.

beta	RW cost	w/ IRT	w/o IRT
0.9	23.77	16.97	15.44
1.4	23.77	17.54	15.44



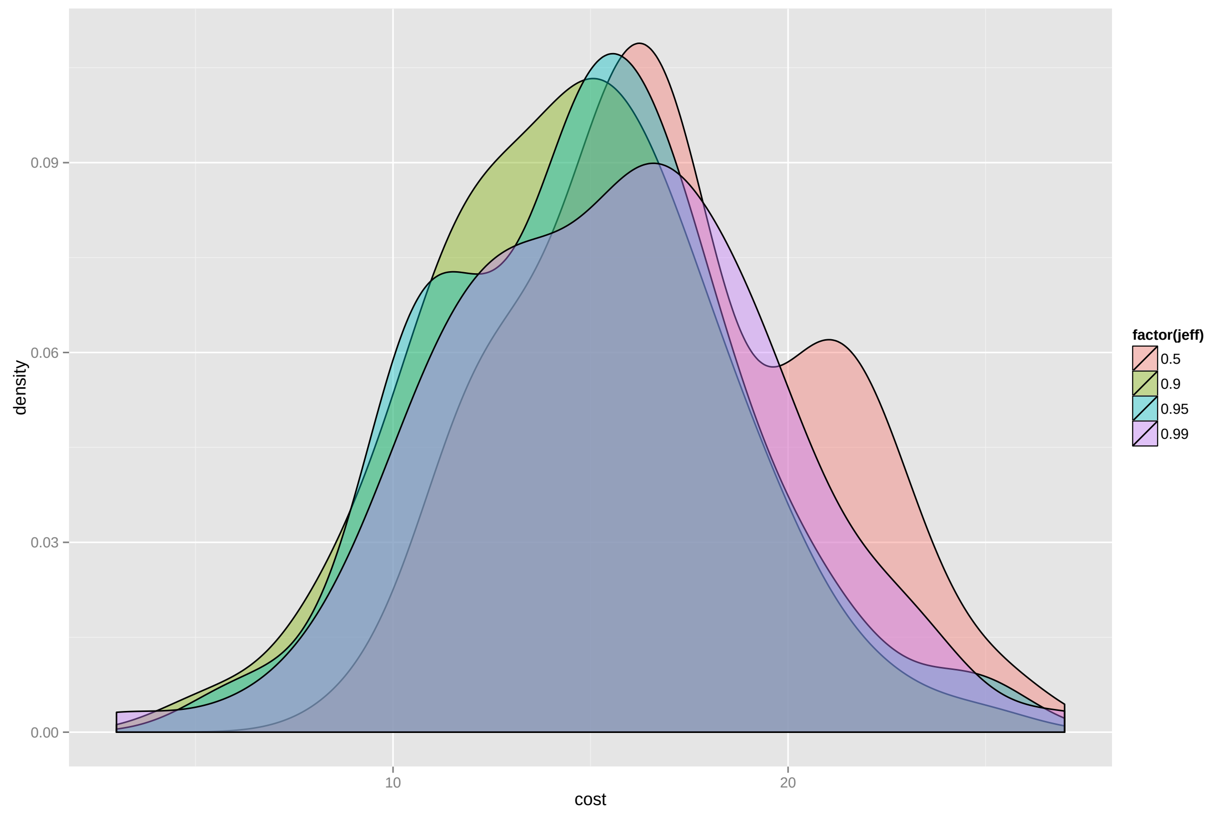
And the distribution of costs for the two beta values:

IRT weighting

One problem is that our method uses a weighting parameter to combine IRT and non-IRT information. Colloquially, this is the “jeff” parameter. A Bayesian solution should have this parameter set to 0.5, as was done in the above simulations. Higher values give more weight to non-IRT components; a weight of 1 would be identical to exlcuding IRTs.

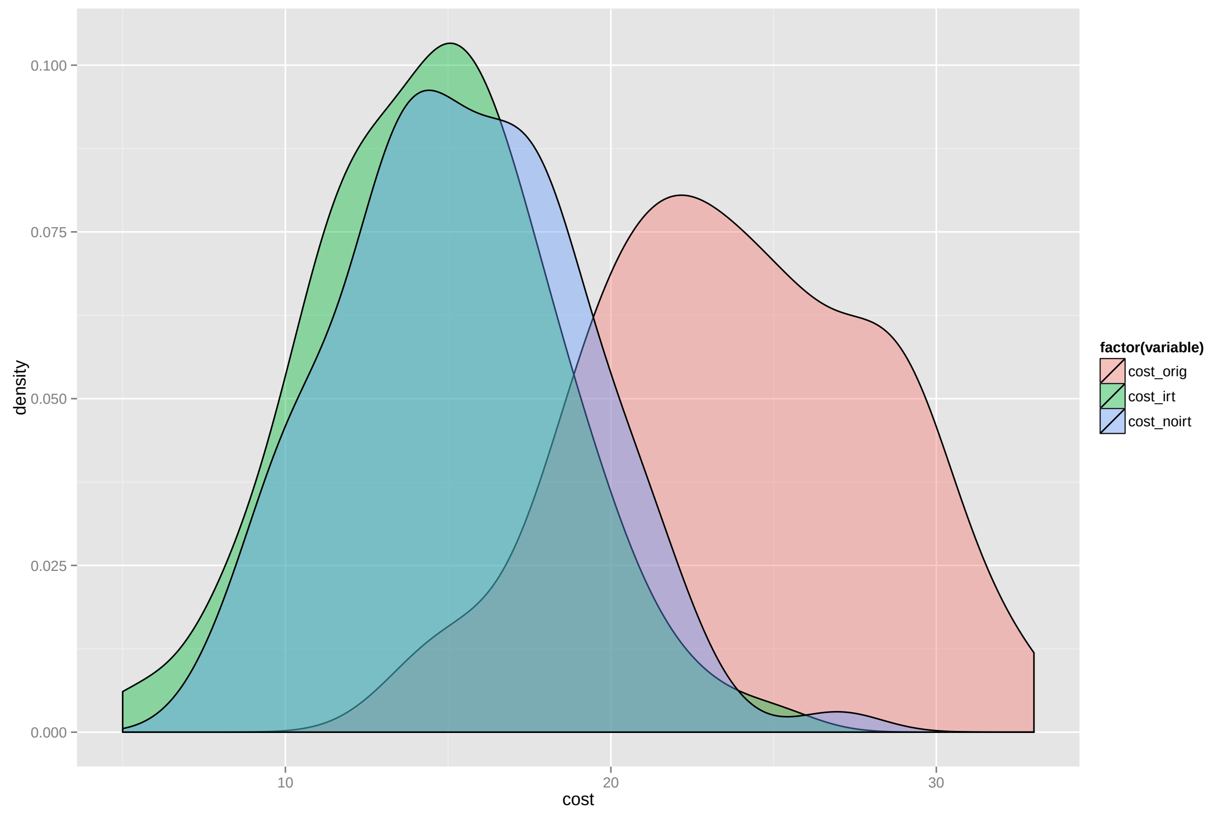
jeff	RW cost	w/ IRT	w/o IRT
0.50	23.77	16.97	15.44
0.90	23.77	14.53	15.44
0.95	23.77	14.91	15.44
0.99	23.77	15.51	15.44

By setting a higher value for this parameter, we start to do better when we include IRTs (yay!) though it would be nice if we could eventually do away with this parameter. Of the values tested, a weight of 0.9 does the best.



Benefit of IRTs

Sow how much does using IRTs buy us? Using the best parameters from the above analyses:



The answer is that it buys you a little, but not a lot, at least with these parameters. But it's nice to know our method does something.

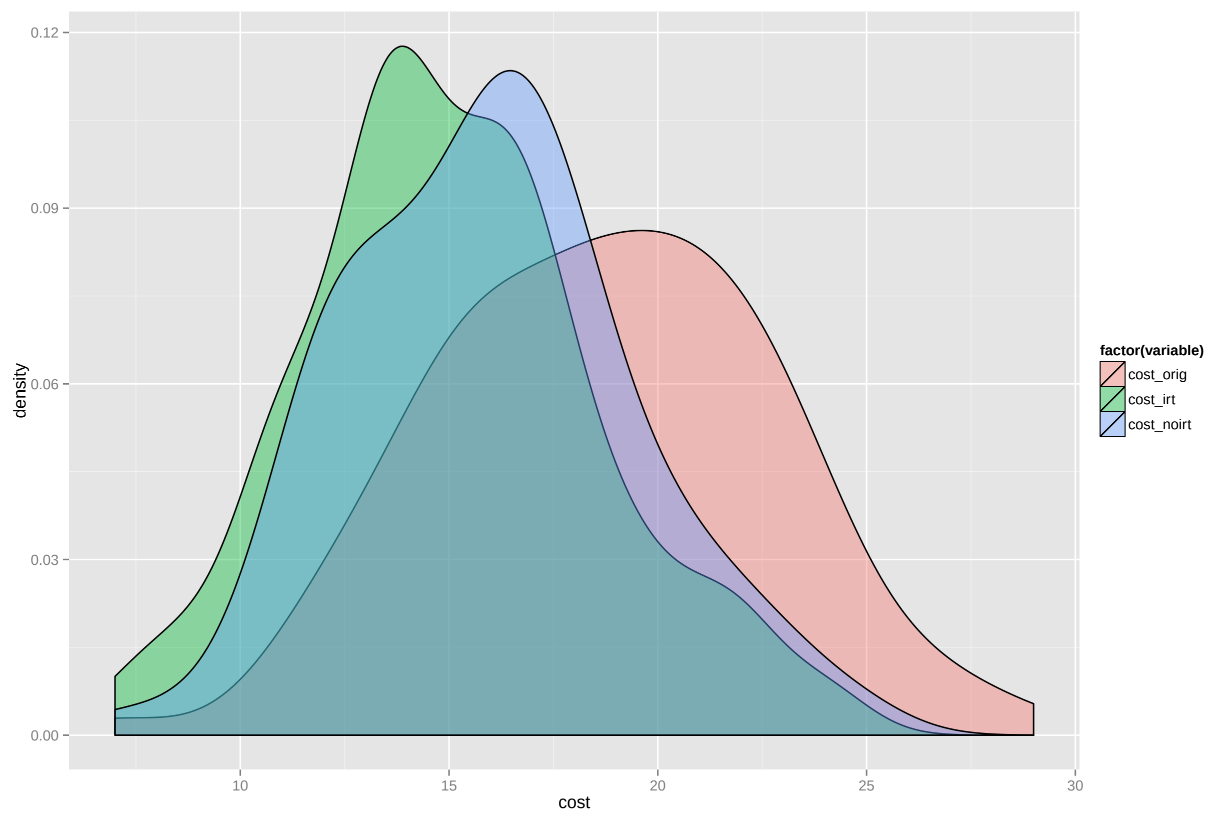
With or without IRTs, the method clearly does better than the RW method. As the number of lists increases, RW should converge to a maximally connected graph, whereas the other methods should converge to the true graph. I haven't test the First-Edge estimator method, though it's a trivial comparison here. With 5 lists, the FE method will have at most 5 correct edges, and so cannot have a cost lower than 25. In contrast to RW, the FE method will converge to the true graph as well as the number of lists increase.

Truncated lists

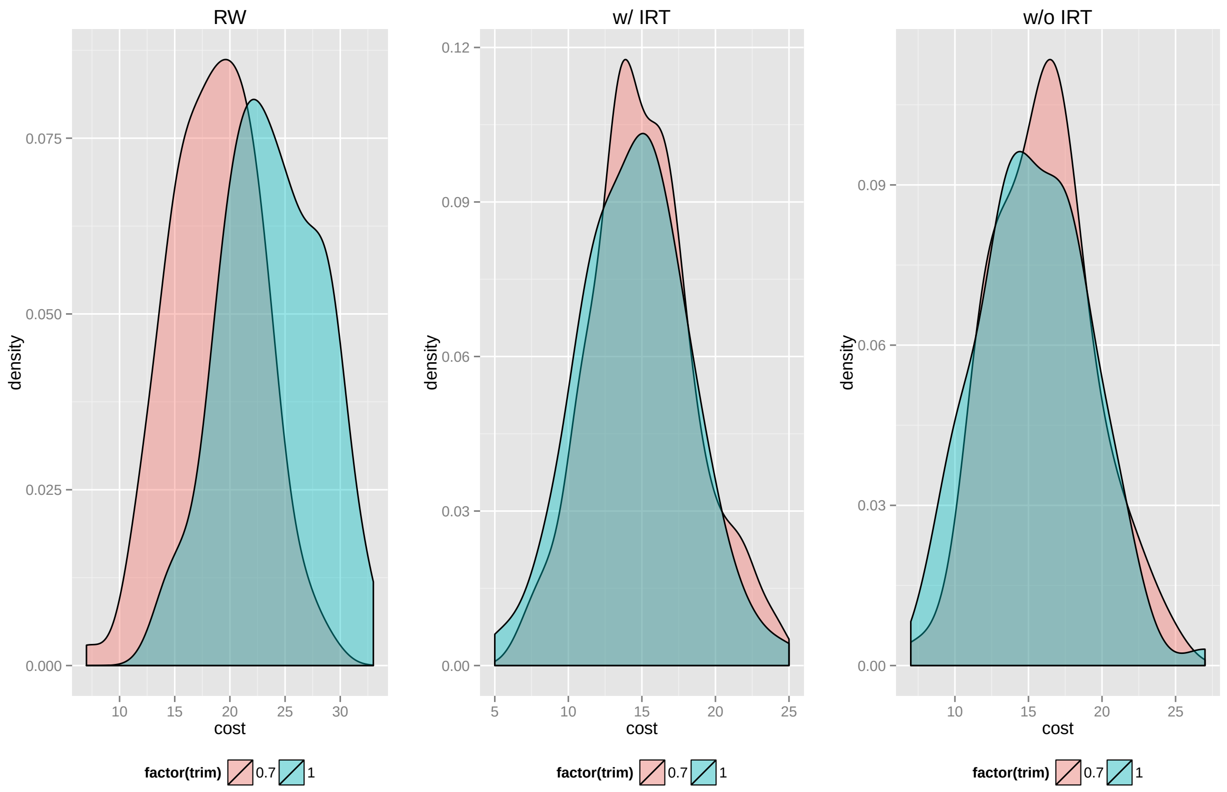
Unlike simulated data, a semantic fluency list will not cover the entire graph space. In our experiments, each list a participant generated covered about 70% of the total animals lists. These simulations show how the results change when we alter the lists to cover only 70% of the space.

trim	RW cost	w/ IRT	w/o IRT
1.0	23.77	14.53	15.44
0.7	18.70	15.07	15.96

Unsurprsingly, the IRT and no-IRT methods do slightly worse. Counterintuitively, the RW method does better. Presumably this is for the same reason increasing the number of lists is bad for RW. Additionally, the items towards the beginning of a list are much more likely to be directly connected. You can see when using lists truncated to 70%, the RW distribution has been pushed closer to the other distributions:



These distributions compare each fitting method before and after truncating the lists:

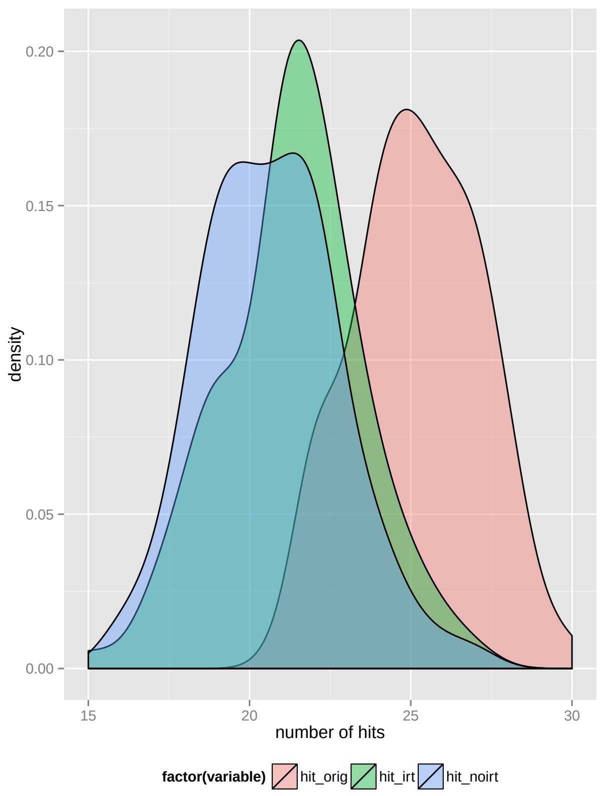


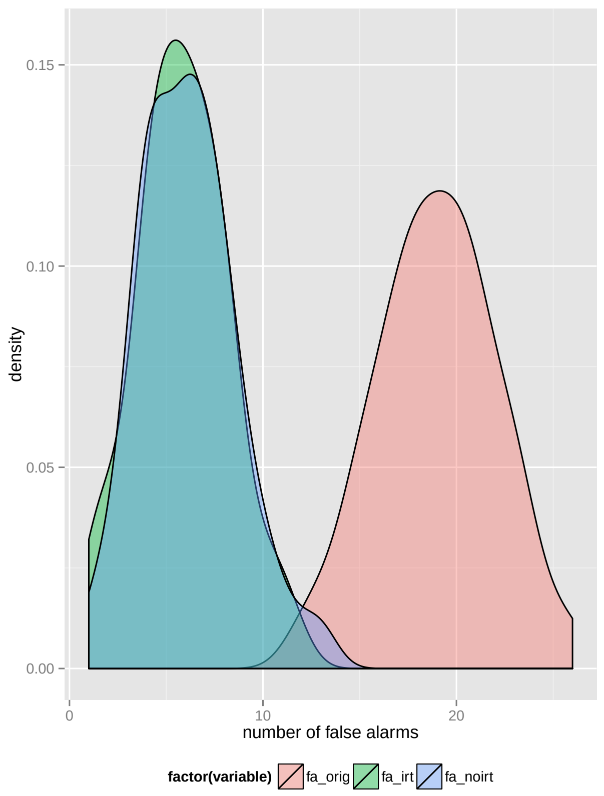
For the most part, truncating the lists had little effect on the other methods besides RW.

Signal Detection

Using the same toy graphs as above, there are 30 edges in the true graph and 75 non-edges. The RW method results in a substantial number of false alarms, but also has the highest hit rate. Our method substantially reduces the number of false alarms while also reducing the hit rate. Using IRTs results in both more hits and fewer false alarms than not using IRTs.

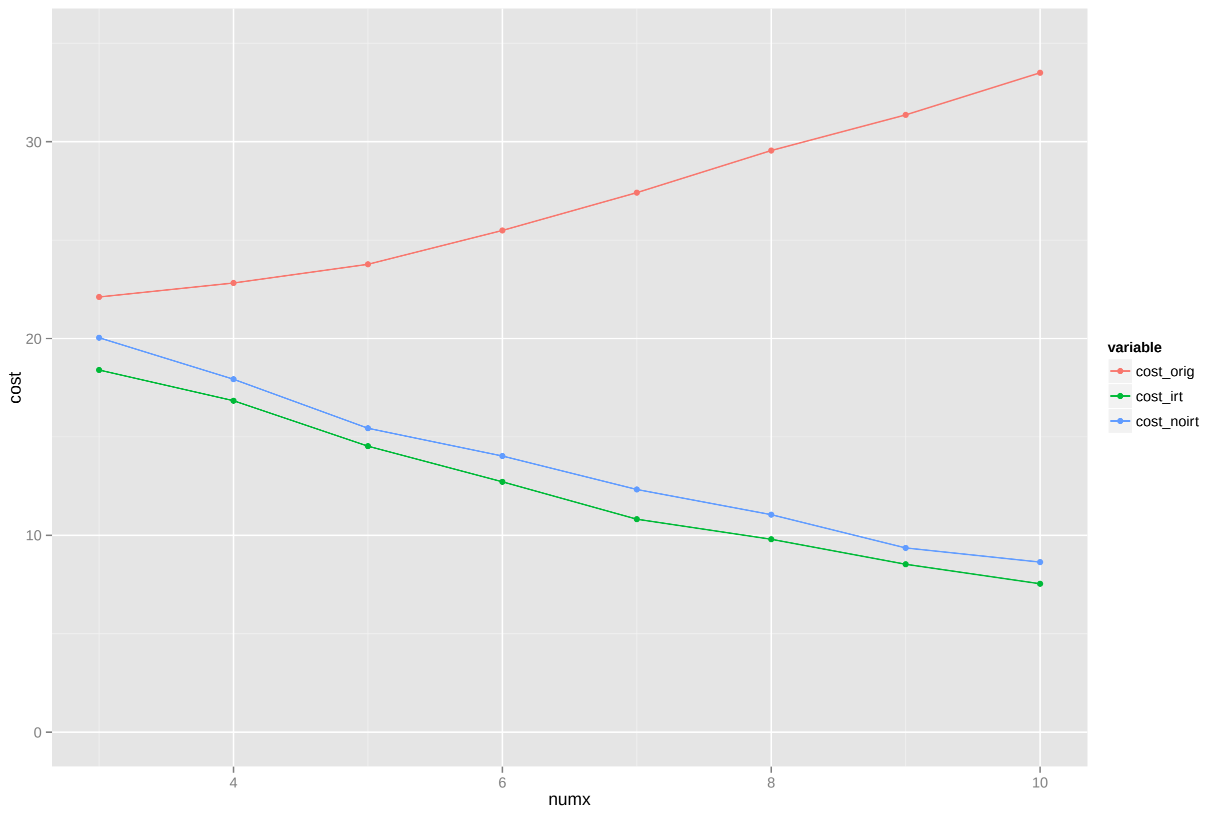
sdt	irt	noirt	orig
fa	5.87	6.12	18.97
hit	21.34	20.68	25.20



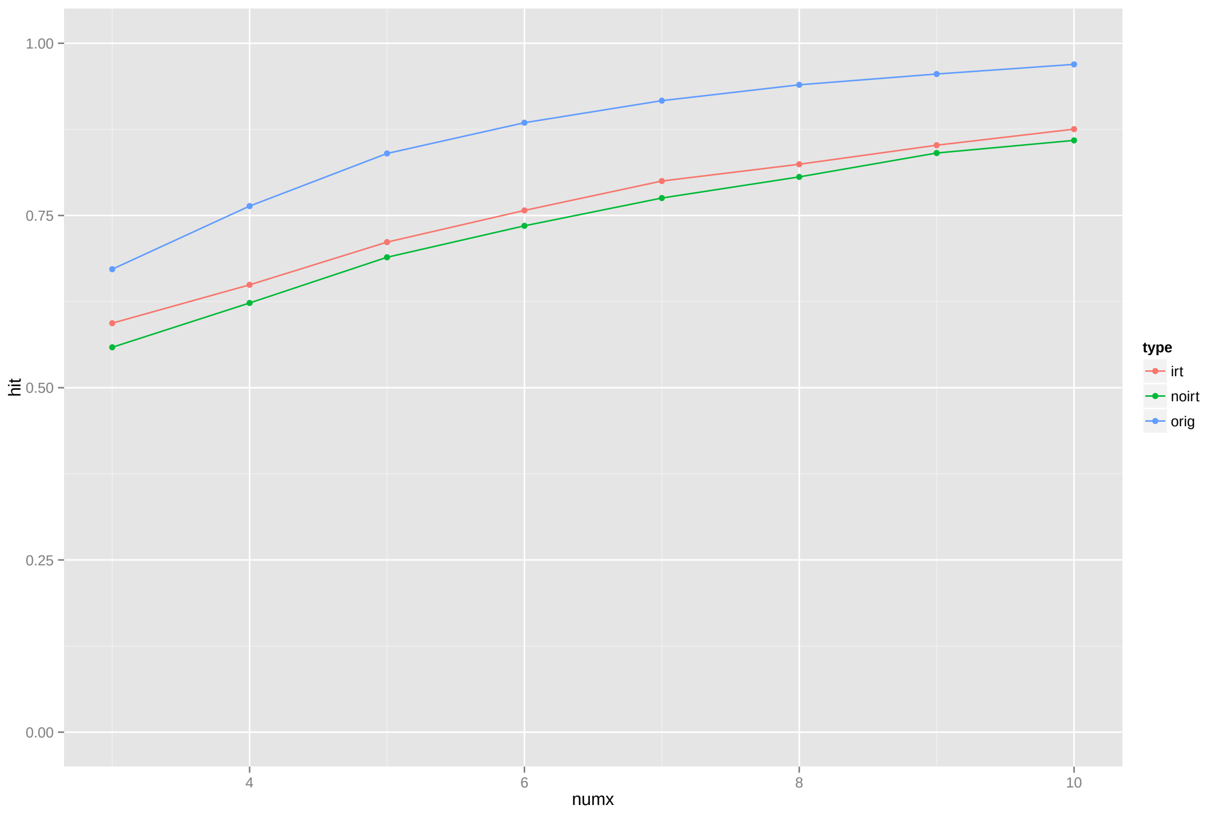


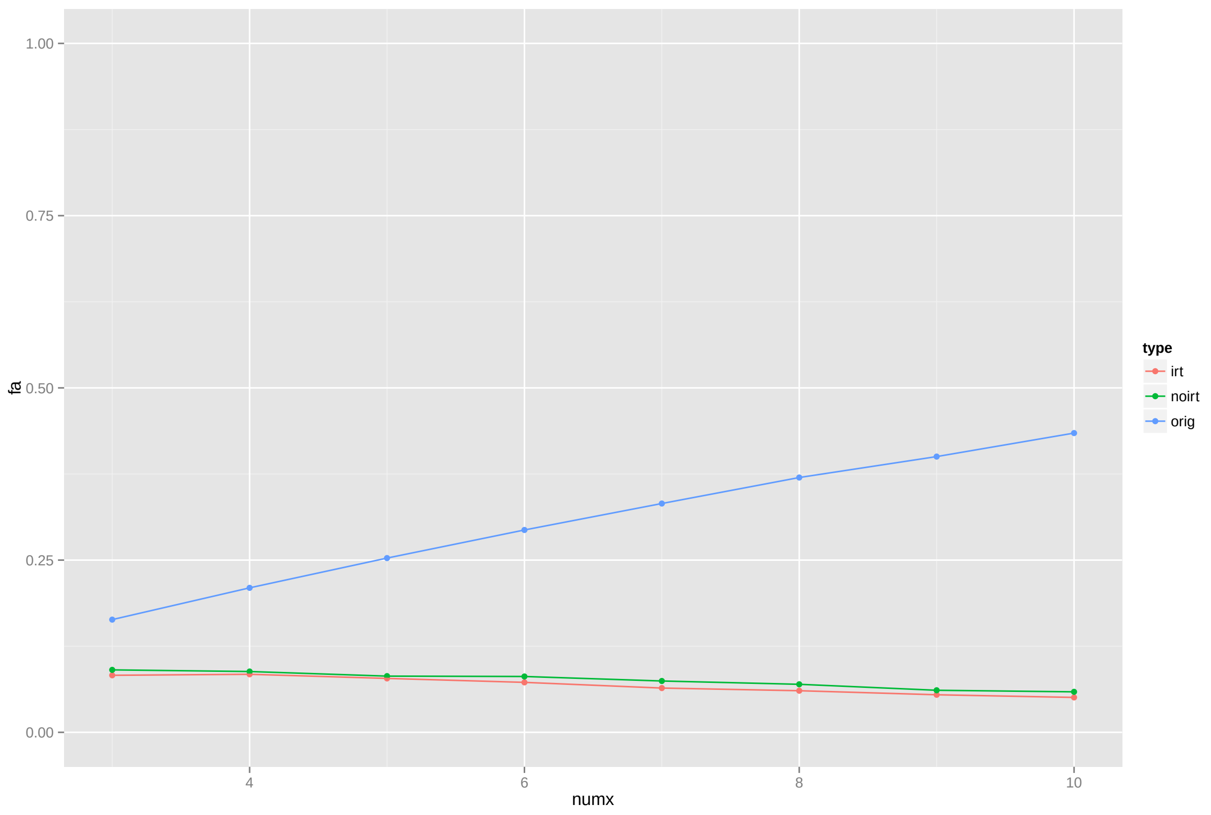
Number of lists

As the number of lists increases, the method improves roughly linearly; the RW method gets worse.



Here is the data split into hits and false alarms. The y-axis in the first graph represents the proportion of true edges correctly identified; the y-axis in the second graph represents the proportion of non-edges correctly identified.


1.00
0.75
0.50
0.25
0.00
hit
4
6
8
10
numx
type
irt
noirt
orig


1.00
0.75
0.50
0.25
0.00
fa
4
6
8
10
numx
type
irt
noirt
orig