



## Detecting Activity Changes in Graphs

David J. Marchette

May 24, 2008

---

Interface 2008

# Outline

Motivation

Definitions and Model

Change Detection

Conclusions



# Outline

Motivation

Definitions and Model

Change Detection

Conclusions



# Social Networks

This builds on work presented at QMDNS on Wednesday.  
Many of the slides will be repeats from that talk.

- ▶ A model of the relationships between entities.
- ▶ Also used to study insurgent groups, terrorist cells, etc.
- ▶ Relates actors (nodes in the network) through relationships (edges in the network).
- ▶ Typically used for small groups, with full knowledge of all links.



# Covert Networks

- ▶ Actors have a vested interest in not being observed.
- ▶ Networks may be very large.
- ▶ The networks change in time.
- ▶ An actor may try to hide (change email address, change phone number, start calling themselves Colonel Guapa).
- ▶ In this work we are interested in cases where the actor changes their activity profile. Can we detect that they have made a significant change?



# Methodology

- ▶ Assume the existence of a “social space”  $\mathcal{S}$  which controls the structure of the network.
- ▶ The probability of an edge in the network is a function of the “closeness” of the nodes in  $\mathcal{S}$ .
- ▶ The social space provides a framework from which inference can be performed.



# Social Space

- ▶ Early work reported by Hoff et al in JASA.
- ▶ Model based on location:
  - ▶ Probability of an edge between  $v_i$  and  $v_j$  a function of their distance in social space.
  - ▶ Several variations proposed.
- ▶ Versions of the Exponential Random Graph Models (ERGMs) (Hunter et al, JASA 2008) can be thought of in terms of a “social space”.
- ▶ We will discuss a “social space” model that has a simple least squares algorithm for fitting the parameters, which can be used on large graphs (thousands to tens of thousands of nodes or more).



# Outline

Motivation

**Definitions and Model**

Change Detection

Conclusions



# Graph Definitions

- ▶ A graph is a pair  $(V, E)$  where  $V$  is a set (vertices) and  $E$  is a collection of unordered pairs of vertices (edges).
- ▶ We can consider directed graphs  $(V, A)$  where  $A$  (arcs or arrows) are ordered pairs.
- ▶ The order of the graph is  $|V|$  and the size of the graph is  $|E|$  (or  $|A|$  in the case of directed graphs (digraphs)).
- ▶ Vertices are sometimes called “nodes” or “actors”.
- ▶ Edges are sometimes called “links” or “relations”.
- ▶ The adjacency matrix  $A = (a_{ij})$  is the  $|V| \times |V|$  binary matrix with a 1 in those places where an edge occurs in the graph.



# Probabilistic Framework

- ▶ We place a probability structure on the network.
- ▶ This means we fit a **generative** model to the graph.
- ▶ This allows us to estimate the probability of a missing (unknown) link.
- ▶ We can bring node attributes into the model.
- ▶ We are essentially choosing the “most likely” graph given the model assumption and the observed edges.



# Random Dot Product Graphs

- ▶ Each vertex  $v_i$  has associated with it a vector  $x_i$ .
- ▶ Place an edge  $v_i v_j$  between vertices  $v_i$  and  $v_j$  with probability proportional to  $x_i x_j$ , the dot product of  $x_i$  and  $x_j$ .
- ▶ Thus  $p_{ij} = f(x_i x_j)$ . We'll use the threshold function for  $f$ :

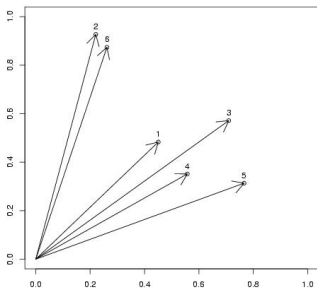
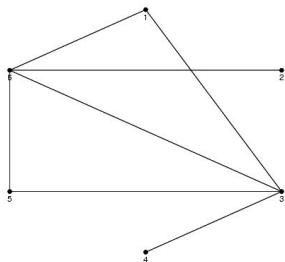
$$f(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$$

- ▶ The edges in the random graph are no longer independent.
- ▶ We need to estimate the  $x_i$  from the observed graph.
- ▶ We can extend the model to directed graphs by having in- and out-vectors  $x_i^I$  and  $x_i^O$  with  $p_{ij}$  proportional to  $x_i^O x_j^I$ .



$\mathcal{S}$ 

- ▶ Each vertex  $v_i$  has associated with it a vector  $x_i \in \mathcal{S}$ .
- ▶ The proximity (as measured by the dot product) of two vectors controls the probability of an edge.
- ▶ Thus  $\mathcal{S}$  is the space which defines the random graph that we observe.

 $\mathcal{S}$  $\mathcal{G}$ 

# Linear Algebra (Least Squares)

Note that if we want to find the vectors  $U$  which best “match” the adjacency matrix  $A$  (best in Frobenius norm), then the singular value decomposition:  $A = UDV'$  almost works (the problem is the diagonal).

1. Set  $D = \text{diag}(0)$ .
  - 1.1  $s = \text{svd}(A + D)$ .
  - 1.2  $X = U$ , scaled by the singular values.
  - 1.3  $D = \text{diag}(XX')$ .
2. Repeat 1–3 until convergence.
3. Return  $X$ .



# Outline

Motivation

Definitions and Model

**Change Detection**

Conclusions



# Aliases

- ▶ Given two graphs  $G_t$  and  $G_{t+1}$ .
- ▶ Suppose we know some of the vertices are shared by these graphs (and which ones they are).
- ▶ There is one vertex in  $G_{t+1}$  that we have not seen before.
- ▶ Assuming that this vertex appeared in  $G_t$  with a different label, can we determine this vertex?
- ▶ For change detection, we want to compare each vertex at time  $t$  with itself at time  $t - 1$ , and all other vertices at this time.
- ▶ If the vertex is less like itself than it is like the others, it has changed.



# Aliases

- ▶ Setup:
  - ▶ Two graphs,  $G_t = (V \cup U_t, E_t)$  and  $G_{t+1} = (V \cup U_{t+1}, E_{t+1})$ .
  - ▶ All vertices are labeled (email addresses).
  - ▶ Vertices in  $V$  are named (individual associated with the address).
  - ▶ Vertices in  $U_i$  are not named.
- ▶ Want to associate the names to the vertices in  $U_{t+1}$ .



# Methodology

- ▶ Assign the name to vertex  $u$  whose vector  $x_v$  is closest to the vector  $x_u$ .
- ▶ Optimize:

$$(X, Y_1, Y_2) = \arg \min_{X, Y_1, Y_2} \left\| \left( \begin{pmatrix} X \\ Y_1 \end{pmatrix} \begin{pmatrix} X \\ Y_1 \end{pmatrix}^T \right)_0 - A_1 \right\|_F + \left\| \left( \begin{pmatrix} X \\ Y_2 \end{pmatrix} \begin{pmatrix} X \\ Y_2 \end{pmatrix}^T \right)_0 - A_2 \right\|_F,$$

- ▶  $M_0$  means  $M$  with the diagonal replaced with zeros.
- ▶ Thus, we are attempting to fit a set of vectors to the known and a set each for the unknown in the two graphs. Fitting to the knowns constrains the  $Y_i$  to lie in the same space.



## The Setup

- ▶ Input  $A_1, A_2$ , the adjacency matrices of the graphs corresponding to the vertices  $(V, U_j)$ .
- ▶ Set  $B$  to be the average of  $A_1[V]$  and  $A_2[V]$ , the blocks corresponding to  $V$ .
- ▶ Set  $N = n + n_1 + n_2$ .
- ▶ Set  $A$  to be the  $N \times N$  matrix with first  $n \times n$  block equal to  $B$ , and blocks  $A[V, U_j] = A_j, A[U_j, V] = A_j'$ .

$$A = \begin{pmatrix} \frac{A_1[V, V] + A_2[V, V]}{2} & A_1[V, U_1] & A_2[V, U_2] \\ A_1[U_1, V] & A_1[U_1, U_1] & Y' \\ A_2[U_2, V] & Y & A_2[U_2, U_2] \end{pmatrix}$$

where  $Y$  is the dot product of vectors derived from  $U_1$  and  $U_2$ .



# Fitting the Alias

1. Setup as described previously.
2. Set  $D = 0_{N \times N}$ .
3. Set the first  $n \times n$  block of  $D$  equal to the the dot product of the result of running the least squares Algorithm on  $B$ .
  - 3.1 While(Not Converged)
  - 3.2  $Y = g_d(A + D)$  (scaled  $U$  from the svd).
  - 3.3 Set the unknown entries of  $D$  (such as those corresponding to  $U_1 \times U_2$ ) to the dot products of the appropriate parts of  $Y$ .
4. Output  $Y$ 
  - ▶ Use the vectors to find the alias: closest named vector to the one associated with the alias.



## Change Detection

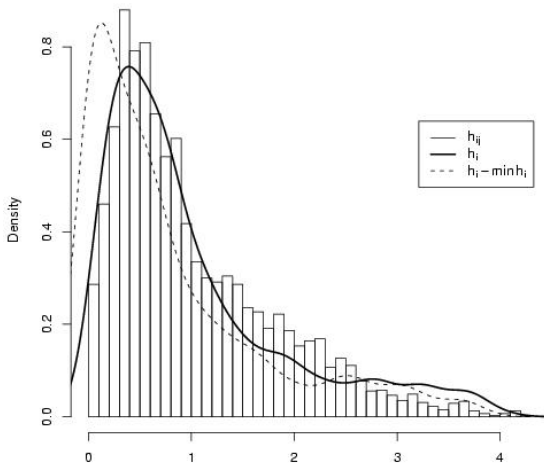
- ▶ Run the alias detection code with  $U_t = U_{t-1} = \{v_i\}$  and with  $U_t = \{v_i\}$ ,  $U_{t-1} = \{v_j\}, j \neq i$ .
- ▶ Use the distance to the correct vertex compared to those with other vectors to determine whether a change has occurred.
- ▶ We will look at some simulations illustrating this:
  1. Consider Erdős-Renyí graphs: the distribution of the distances for the correct vertex should match those for the others.
  2. Consider RDPG graphs: the distribution of the distances for the correct vertex should be stochastically smaller than those for the others.
- ▶ The simulations will bear this out.
- ▶ Note that one need not compare all  $\binom{n}{2}$  pairs in order to obtain an estimate of the alternative: a subsample is sufficient for larger graphs, or one could compute this once off-line if one knew the basic model for the graphs.



# Change Detection Simulation: ER

Erdős-Renyí graphs (no change):  $|V| = 10, p = 0.5$ .

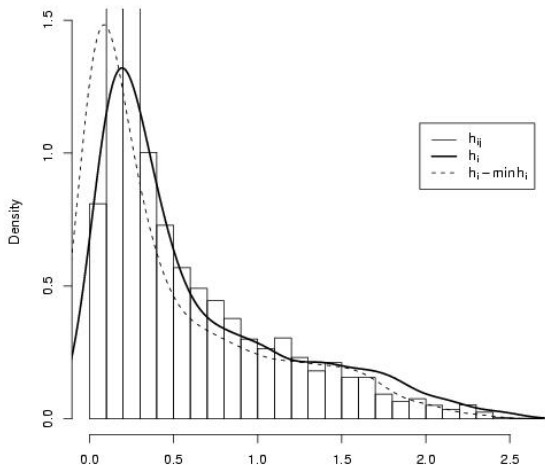
1 graph before the purported change point, 1 graph after.



# Change Detection Simulation: ER

Erdős-Renyí graphs (no change):  $|V| = 10, p = 0.5$ .

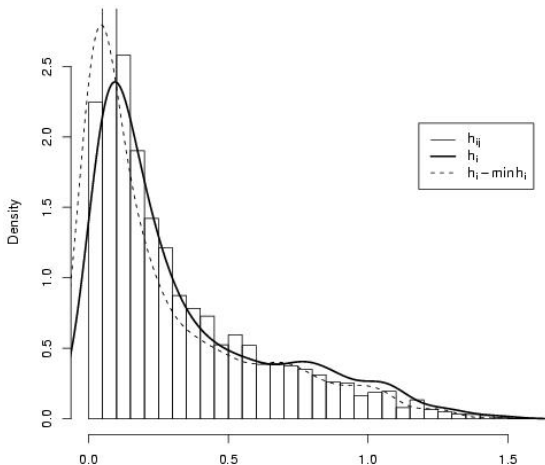
10 graphs before the purported change point, 10 graphs after.



# Change Detection Simulation: ER

Erdős-Renyí graphs (no change):  $|V| = 10, p = 0.5$ .

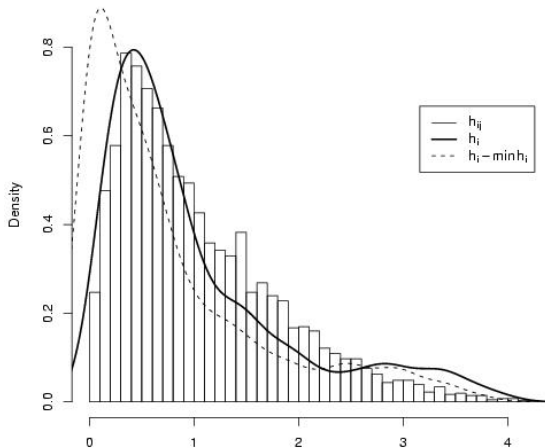
100 graphs before the purported change point, 100 graphs after.



# Change Detection Simulation: ER

Erdős-Renyí graphs (no change):  $|V| = 100, p = 0.5$ .

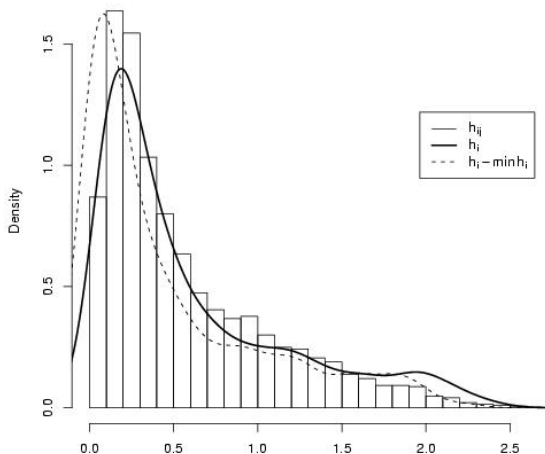
1 graph before the purported change point, 1 graph after.



# Change Detection Simulation: ER

Erdős-Renyí graphs (no change):  $|V| = 100, p = 0.5$ .

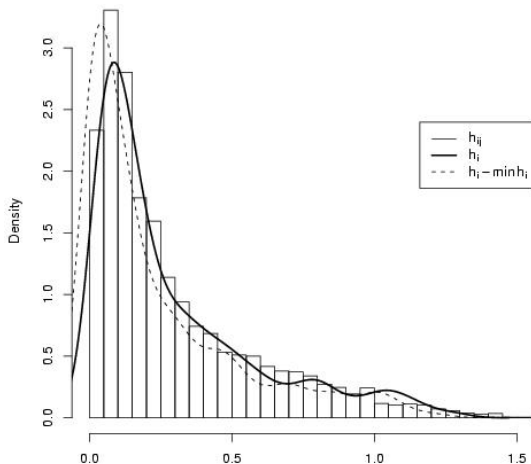
10 graphs before the purported change point, 10 graphs after.



# Change Detection Simulation: ER

Erdős-Renyí graphs (no change):  $|V| = 100, p = 0.5$ .

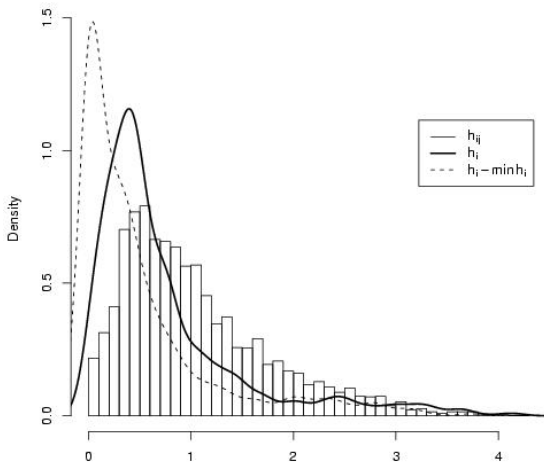
100 graphs before the purported change point, 100 graphs after.



# Change Detection Simulation: RDPG

Random Dot Product graphs (no change):  $|V| = 10$ .

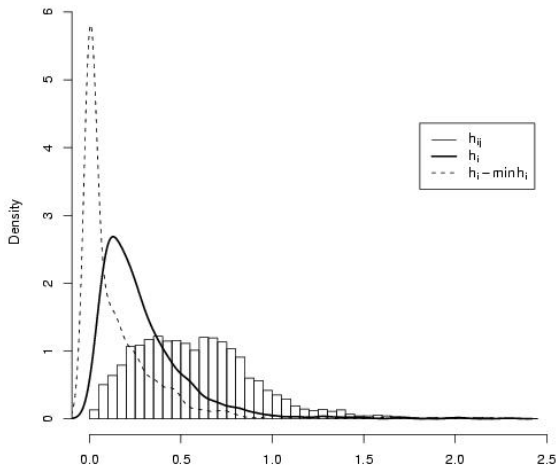
1 graph before the purported change point, 1 graph after.



# Change Detection Simulation: RDPG

Random Dot Product graphs (no change):  $|V| = 10$ .

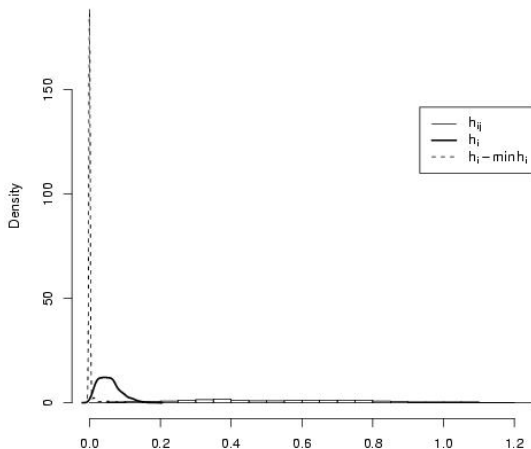
10 graphs before the purported change point, 10 graphs after.



# Change Detection Simulation: RDPG

Random Dot Product graphs (no change):  $|V| = 10$ .

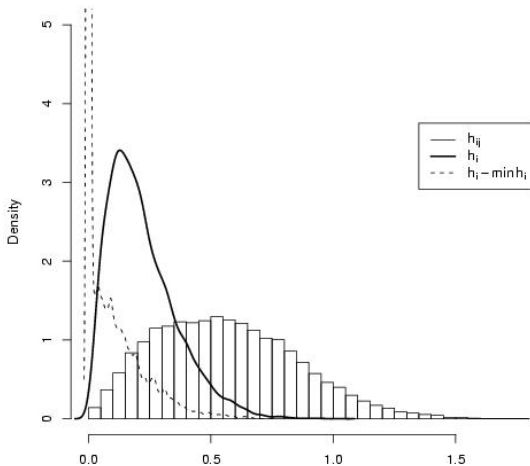
100 graphs before the purported change point, 100 graphs after.



# Change Detection Simulation: RDPG

Random Dot Product graphs (no change):  $|V| = 100$ .

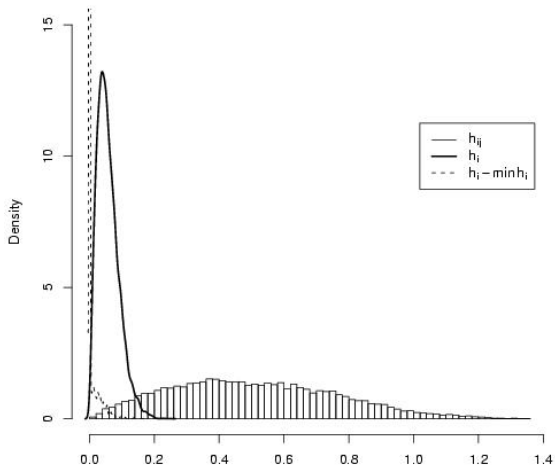
1 graph before the purported change point, 1 graph after.



# Change Detection Simulation: RDPG

Random Dot Product graphs (no change):  $|V| = 100$ .

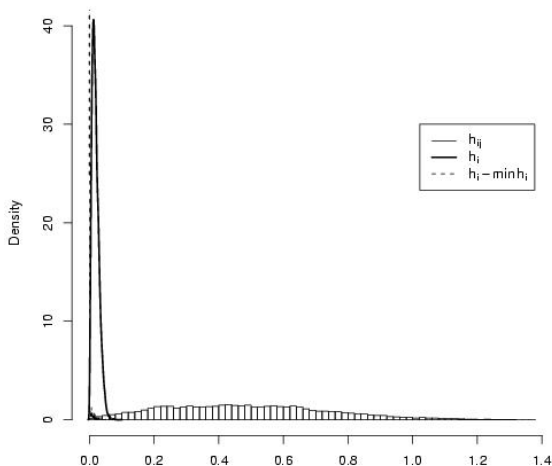
10 graphs before the purported change point, 10 graphs after.



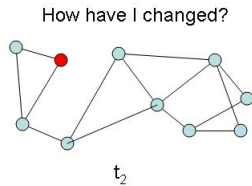
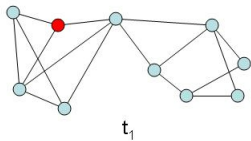
# Change Detection Simulation: RDPG

Random Dot Product graphs (no change):  $|V| = 100$ .

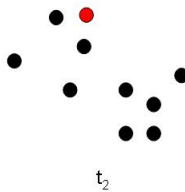
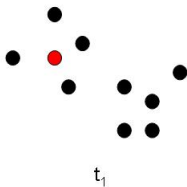
100 graphs before the purported change point, 100 graphs after.



# Cartoon



Social Space



# Outline

Motivation

Definitions and Model

Change Detection

Conclusions



# Conclusions

- ▶ Social space provides a mechanism for modeling and inference on graphs and time series of graphs.
- ▶ Dot product graph model is simple, but easy to fit using linear algebra.
- ▶ Sparse matrix approaches can make this efficient.
- ▶ It is possible to add covariates (measurements at the nodes) into the model and still use the linear algebra approach, but this work is preliminary.
- ▶ Changes in the graphs can be detected at the vertex level, and we have a natural estimate of the distribution under the alternative (comparison to the  $v_i$  vs  $v_j$  distances).



# Questions?

Contact Information: [dmarchette@gmail.com](mailto:dmarchette@gmail.com)

