“Perfect” Power Law Graphs: Generation, Sampling, Construction, and Fitting

Jeremy Kepner

SIAM Annual Meeting, Minneapolis, July 9, 2012

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

This work is sponsored by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government.

Outline

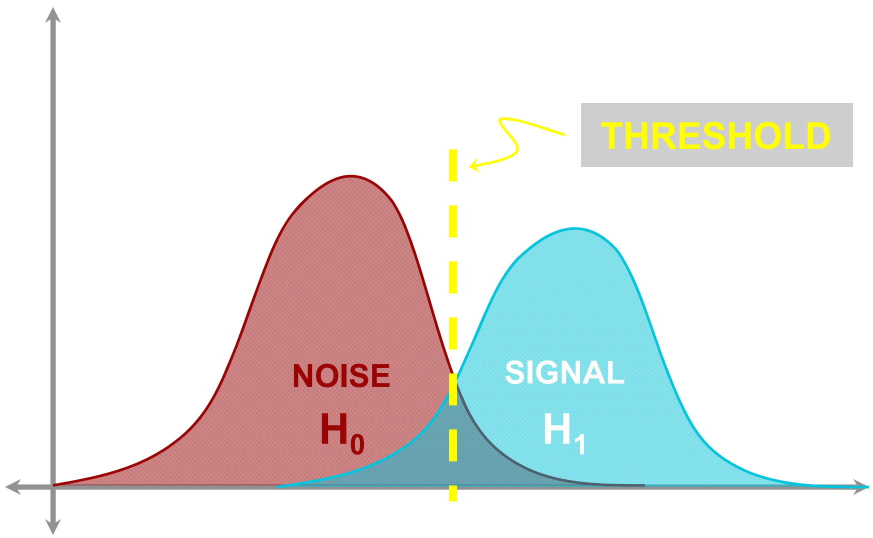
- Introduction
- Sampling
- Sub-sampling
- Reuter's Data
- Summary

Goals

- **Develop a background model for graphs based on "perfect" power law**
- **Examine effects of sampling such a power law**
- **Develop techniques for comparing real data with a power law model**

Detection Theory

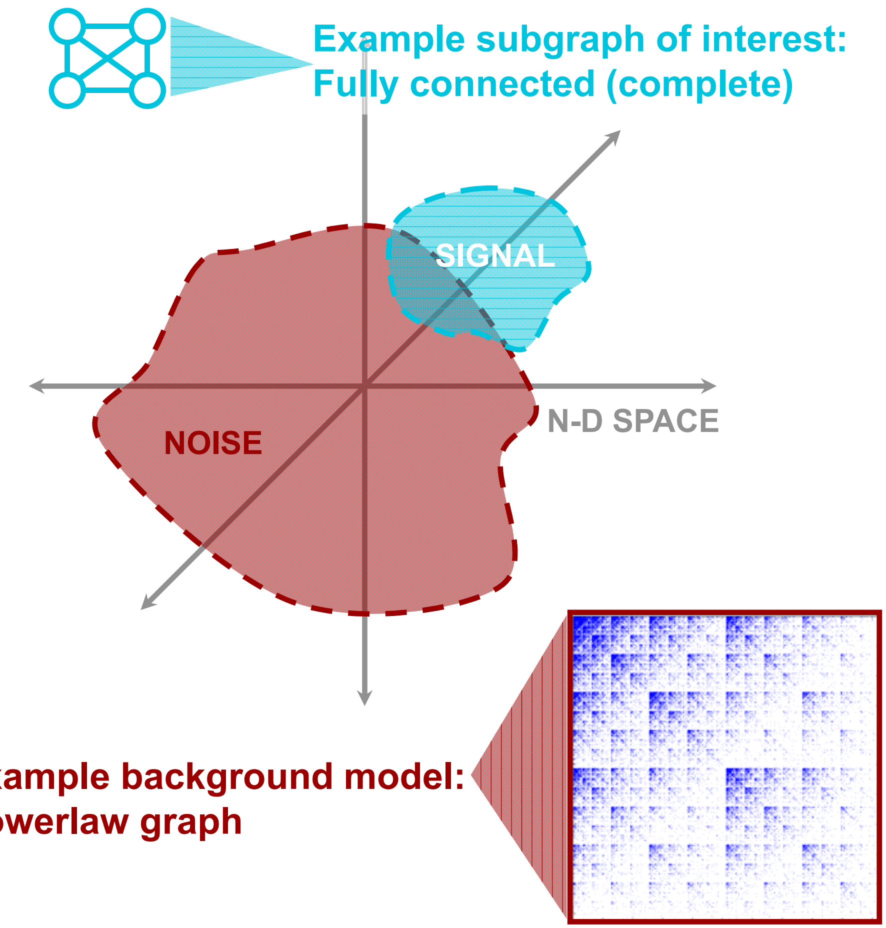
DETECTION OF SIGNAL IN NOISE



ASSUMPTIONS

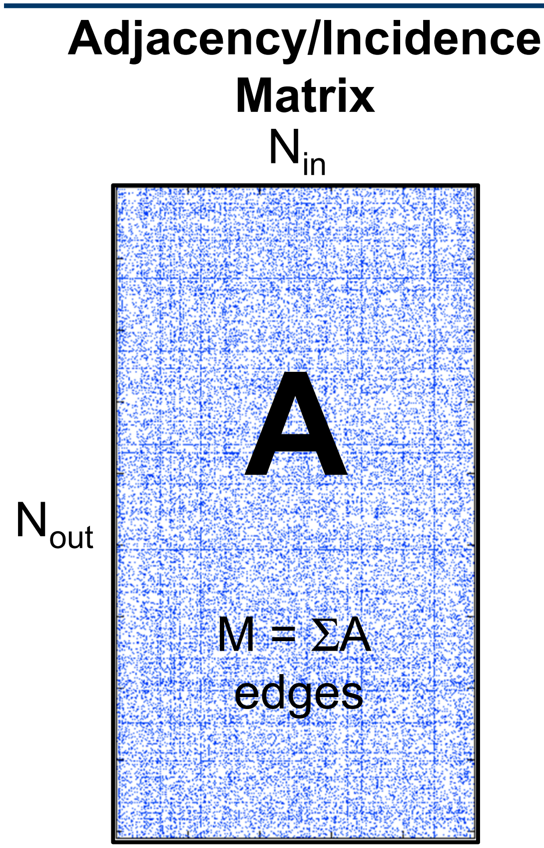
- Background (noise) statistics
- Foreground (signal) statistics
- Foreground/background separation
- Model ≈ reality

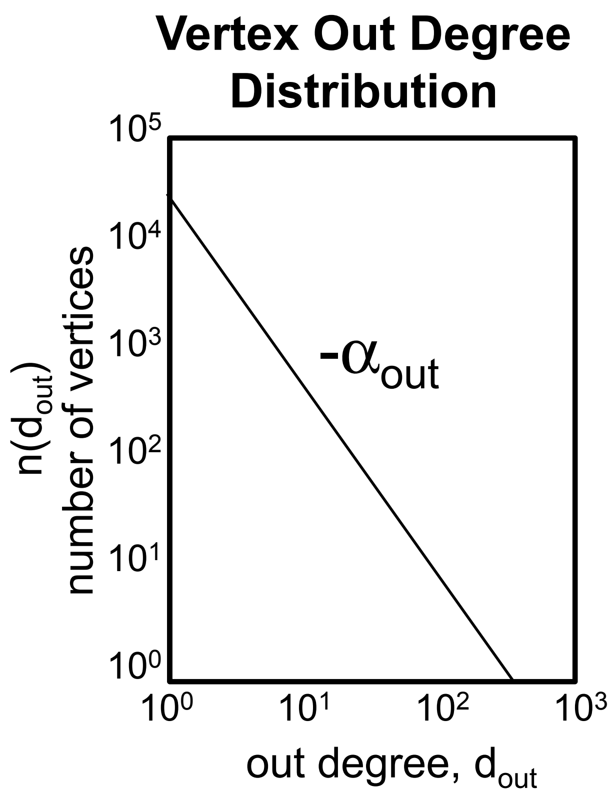
DETECTION OF SUBGRAPHS IN GRAPHS

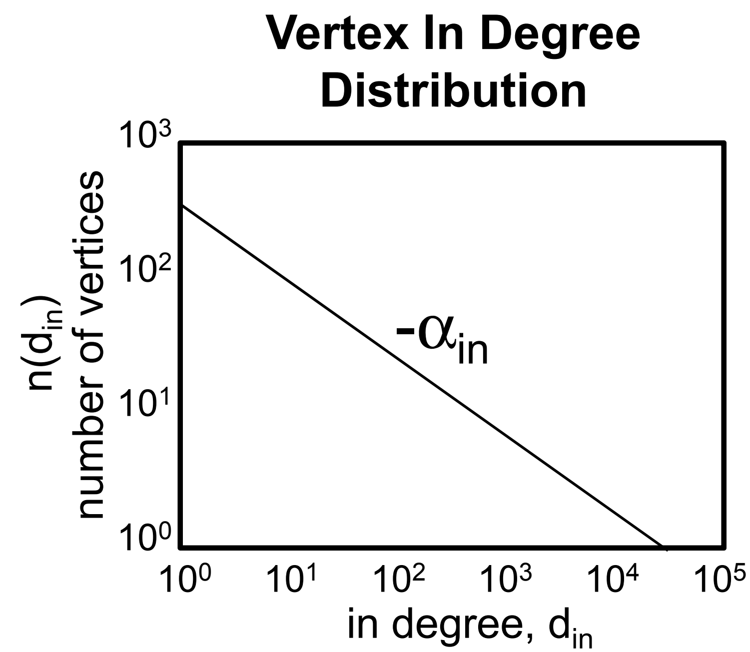


Can we construct a background model based on power law degree distribution?

"Perfect" Power Law Matrix Definition







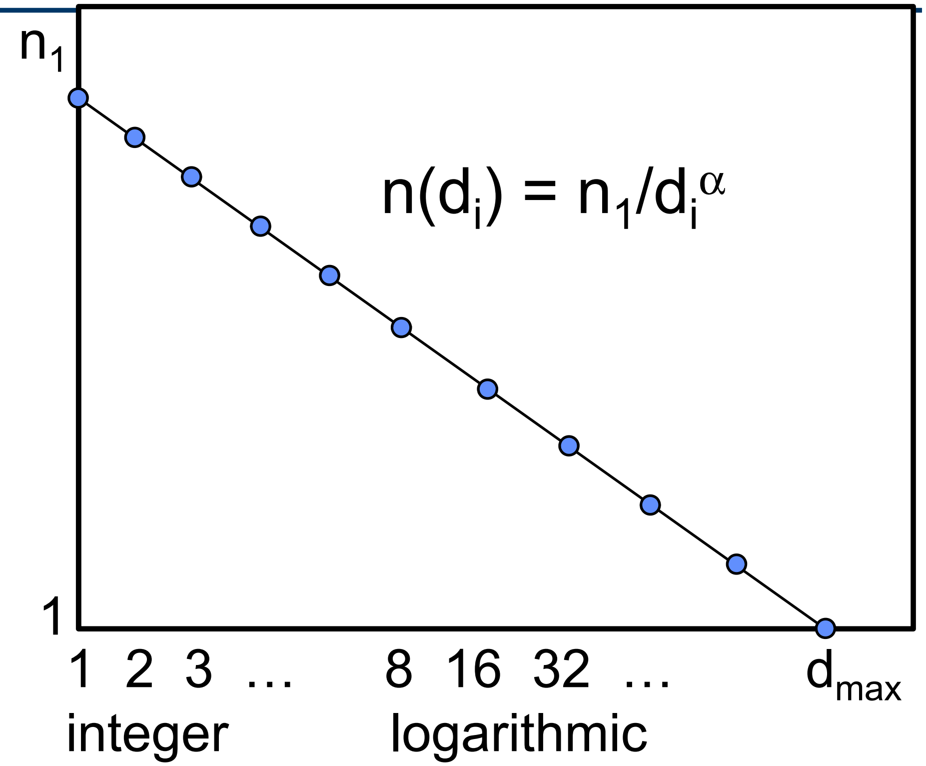
- **Graph represented as a rectangular sparse matrix**
  - **Can be undirected, multi-edged, self-loops, disconnected, hyper edges, …**
- **Out/in degree distributions are *independent* first order statistics**
  - **Only constraint:** $\Sigma\, n(d_{out})\, d_{out} = \Sigma\, n(d_{in})\, d_{in} = M$

Power Law Distribution Construction

- **Perfect power law matlab code**

```
function [di ni] = PPL(alpha,dmax,Nd)
logdi = (0:Nd) * log(dmax) / Nd;
di    = unique(round(exp(logdi)));
logni = alpha * (log(dmax) - log(di));
ni    = round(exp(logni));
```

- **Parameters**
  - `alpha` **= slope**
  - `dmax` **= largest degree vertex**
  - `Nd` **= number of bins (before unique)**

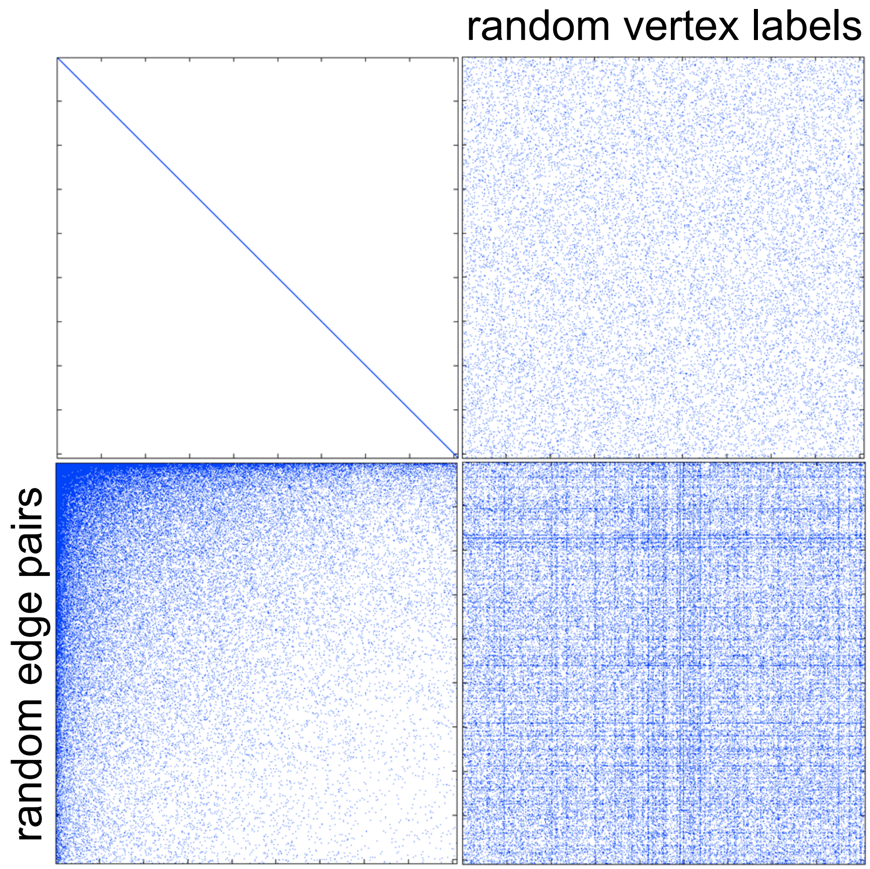


- **Simple algorithm naturally generates perfect power law**
- **Smooth transition from integer to logarithmic bins**
- **"Poor man's" slope estimator:** $\alpha = \log(n_1)/\log(d_{max})$

Power Law Edge Construction

- **Power law vertex list matlab code**

```
function v = PowerLawEdges(di,ni);
A1 = sparse(1:numel(di),ni,di);
A2 = fliplr(cumsum(fliplr(A1),2));
[tmp tmp d] = find(A2);
A3 = sparse(1:numel(d),d,1);
A4 = fliplr(cumsum(fliplr(A3),2));
[v tmp tmp] = find(A4);
```



- **Degree distribution independent of**
  - **Vertex labels**
  - **Edge pairing**
  - **Edge order**

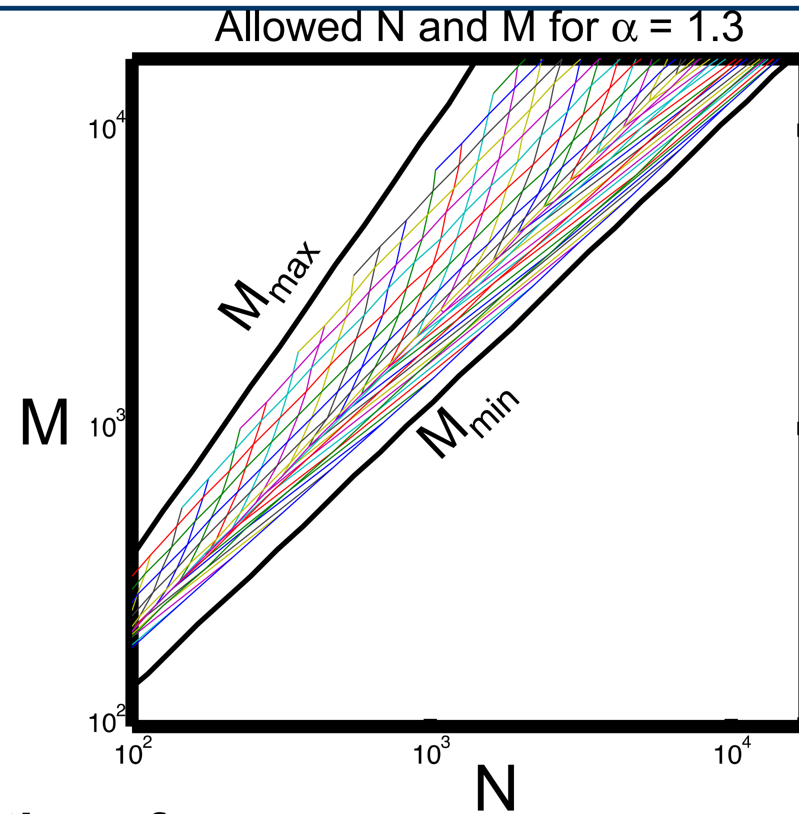
- **Algorithm generates list of vertices corresponding to any distribution**
- **All other aspects of graph can be set based on desired properties**

Fitting α, N, M

- **Power law model works for any**
  $\alpha > 0$, $d_{max} > 1$, $N_d > 1$

- **Desire distribution that fits**
  α, N, M

- **Can invert formulas**
  - $N = \Sigma_i\, n(d_i)$
  - $M = \Sigma_i\, n(d_i)\, d_i$



- **Highly non-linear; requires a combination of**
  - **Exhaustive search, simulated annealing, and Broyden's algorithm**

> - **Given** α, N, M **can solve for** N_d **and** d_{max}
> - **Not all combinations of** α, N, M **are consistent with power law**

Outline

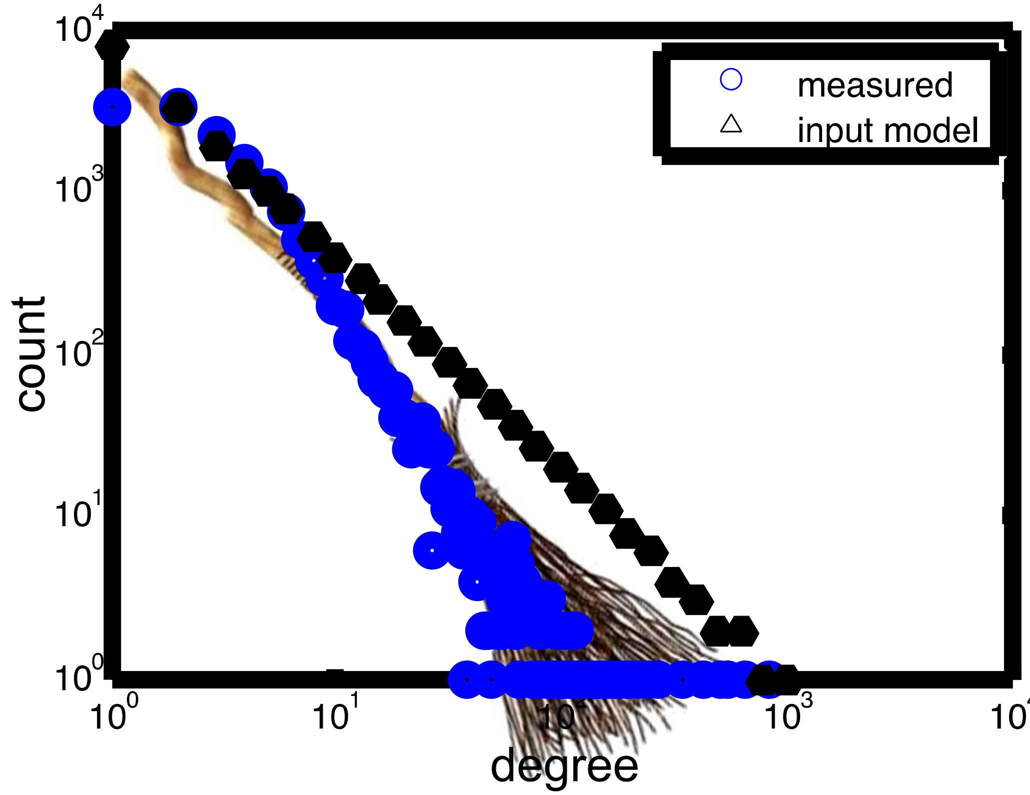
- Introduction
- **Sampling**
- Sub-sampling
- Reuter's Data
- Summary

Graph Construction Effects

- **Generate a perfect power law NxN randomize adjacency matrix A**
  - α = 1.3, d_{max} = 1000, N_d = 50
  - N = 18K, M = 84K

- **Make undirected, unweighted, with no self-loops**

```
A = triu(A + A');
A = double(logical(A));
A = A - diag(diag(A));
```

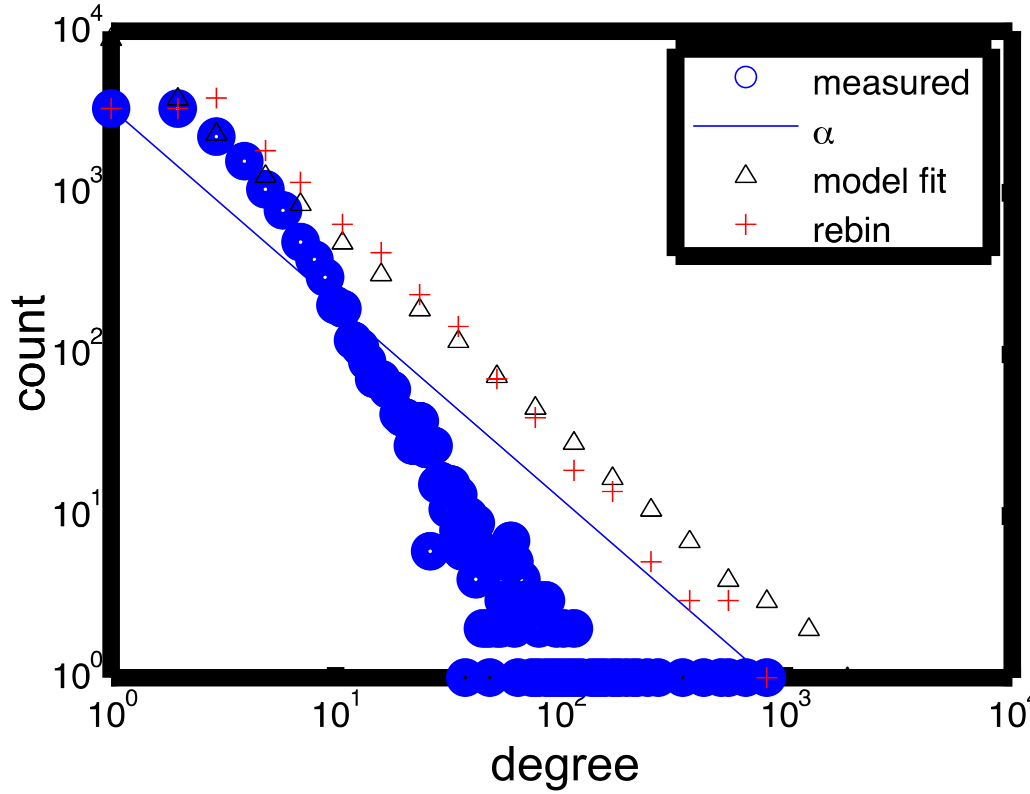


- **Graph theory best for undirected, unweighted graphs with no self-loops**
- **Often "clean up" real data to apply graph theory results**
- **Process mimics "bent broom" distribution seen in real data sets**

Power Law Recovery

Procedure

- **Compute** α, N, M **from measured**
- **Fit perfect power law to these parameters**
- **Rebin measured data using perfect power law degree bins**



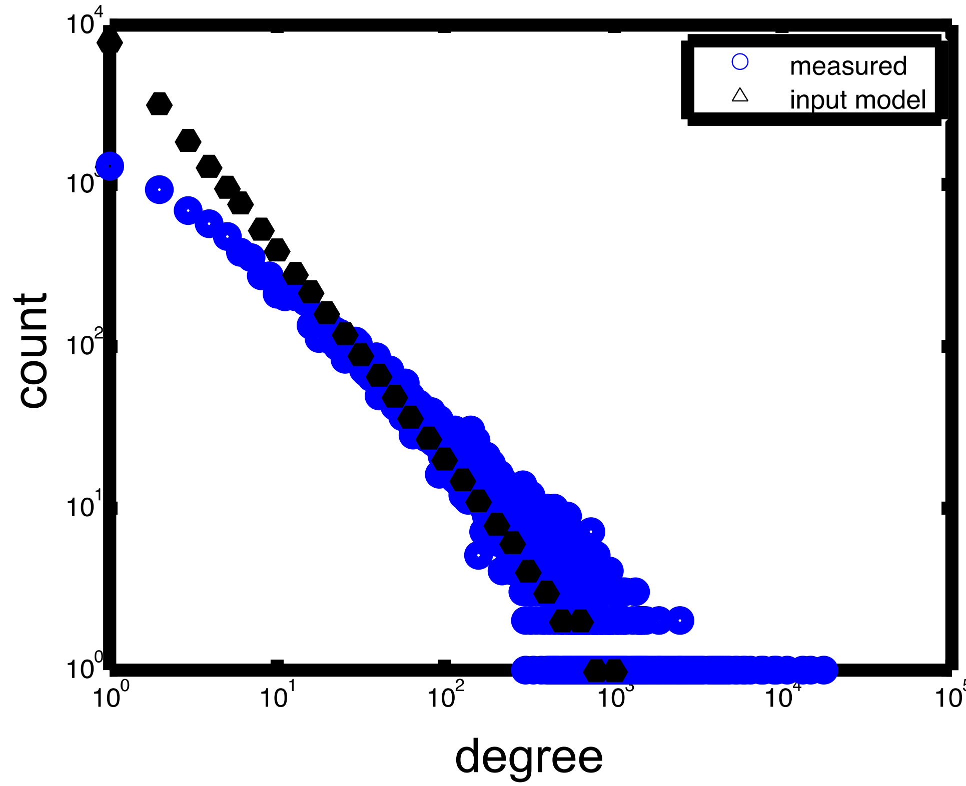
- **Perfect power law fit to "cleaned up" graph can recover much of the shape of the original distribution**

Correlation Construction Effects

- **Generate a perfect power law NxN randomize incidence matrix E**
  - $\alpha = 1.3$, $d_{max} = 1000$, $N_d = 50$
  - N = 18K, M = 84K

- **Make unweighted and use to form correlation matrix A with no self-loops**

```
E = double(logical(E));
A = triu(E' * E);
A = A - diag(diag(A));
```

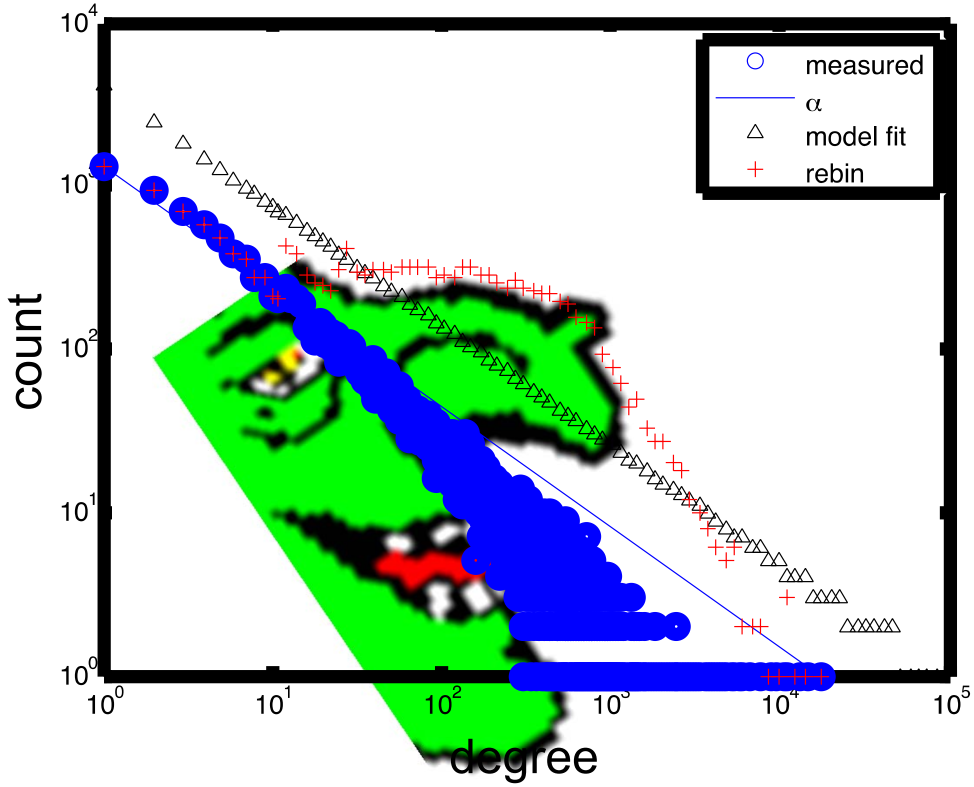


- **Correlation graph construction from incidence matrix results in a "bent broom" distribution that strongly resembles a power law**

Power Law Lost

Procedure

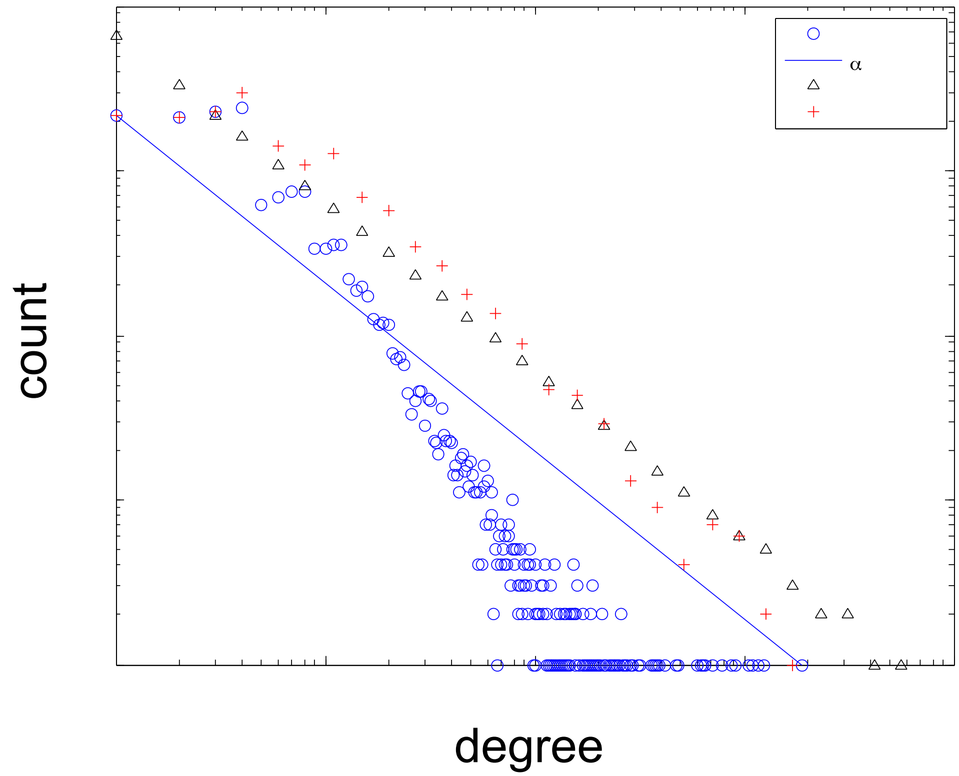
- **Compute** α, N, M **from measured**
- **Fit perfect power law to these parameters**
- **Rebin measured data using perfect power law degree bins**



- **Perfect power law fit to correlation shows non-power law shape**
- **Reveals "witches nose" distribution**

Power Law Preserved

- **In degree is power law**
  - α = 1.3, d_{max} = 1000, N_d = 50
  - N = 18K, M = 84K
- **Out degree is constant**
  - N = 16K, M = 84K
  - Edges/row = 5 (exactly)

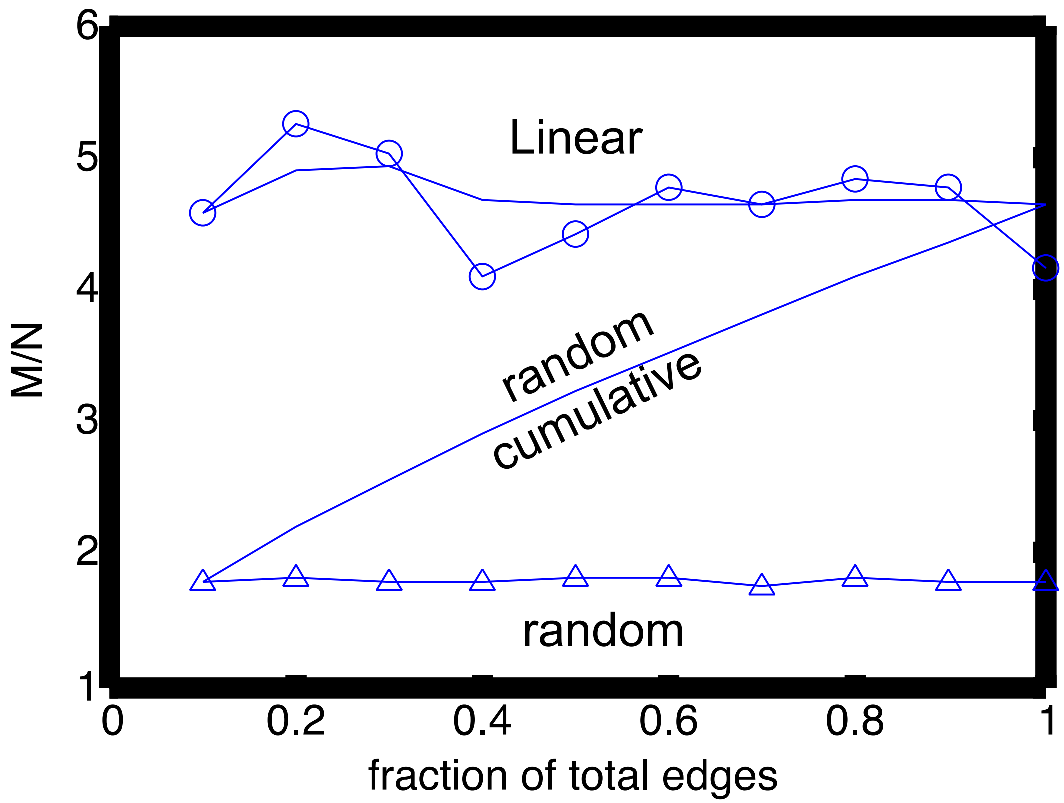


- **Make unweighted and use to form correlation matrix A with no self-loops**

- **Uniform distribution on correlated dimension preserves power law shape**

Edge Ordering: Densification

- **Compute M/N cumulatively and piecewise for 2 orderings**
  - **Linear**
  - **Random**
- **By definition M/N goes from 1 to infinity for finite N**
- **Elimination of multi-edges reduces M and causes M/N to grow more slowly**

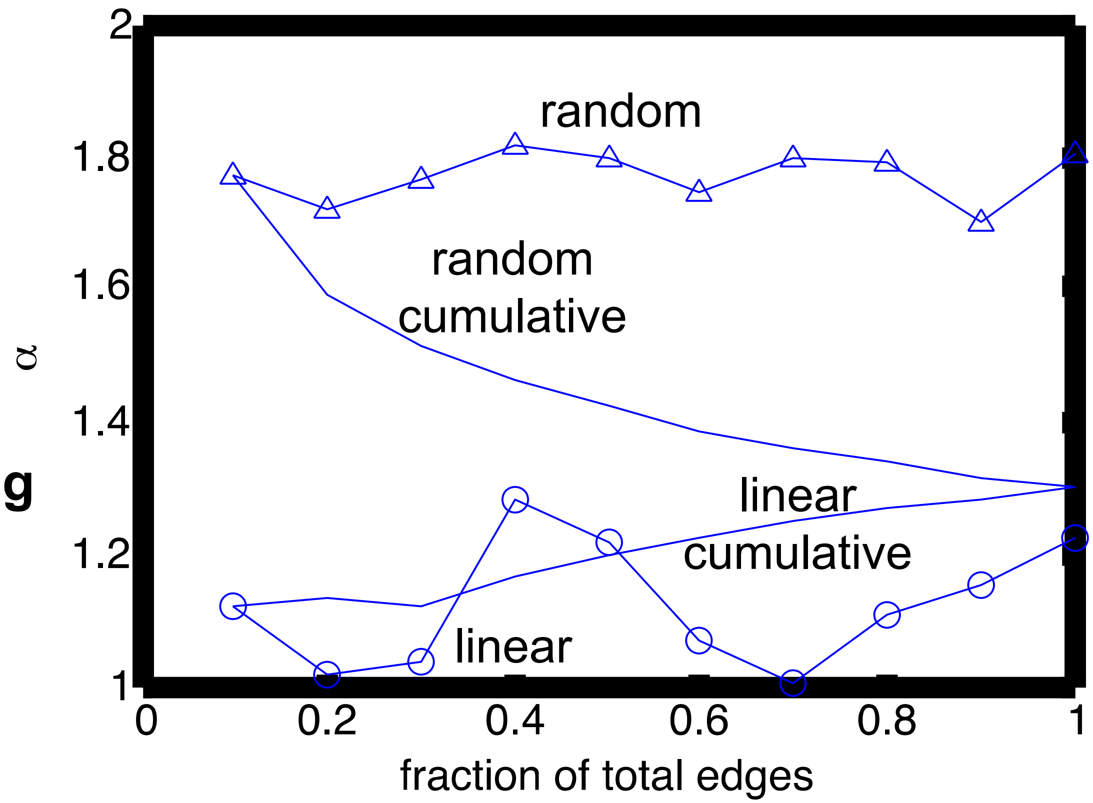


- **"Densification" is the observation that M/N increases with N**
- **Densification is a natural byproduct of randomly drawing edges from a power law distribution**
- **Linear ordering has constant M/N**

Edge Ordering: Power Law Exponent (α)

- **Compute α cumulatively and piecewise for 2 orderings**
  - **Linear**
  - **Random**

- **Edge ordering and sampling have large effect on the power law exponent**



- **Power law exponent is fundamental to distribution**
- **Strongly dependent on edge ordering and sample size**

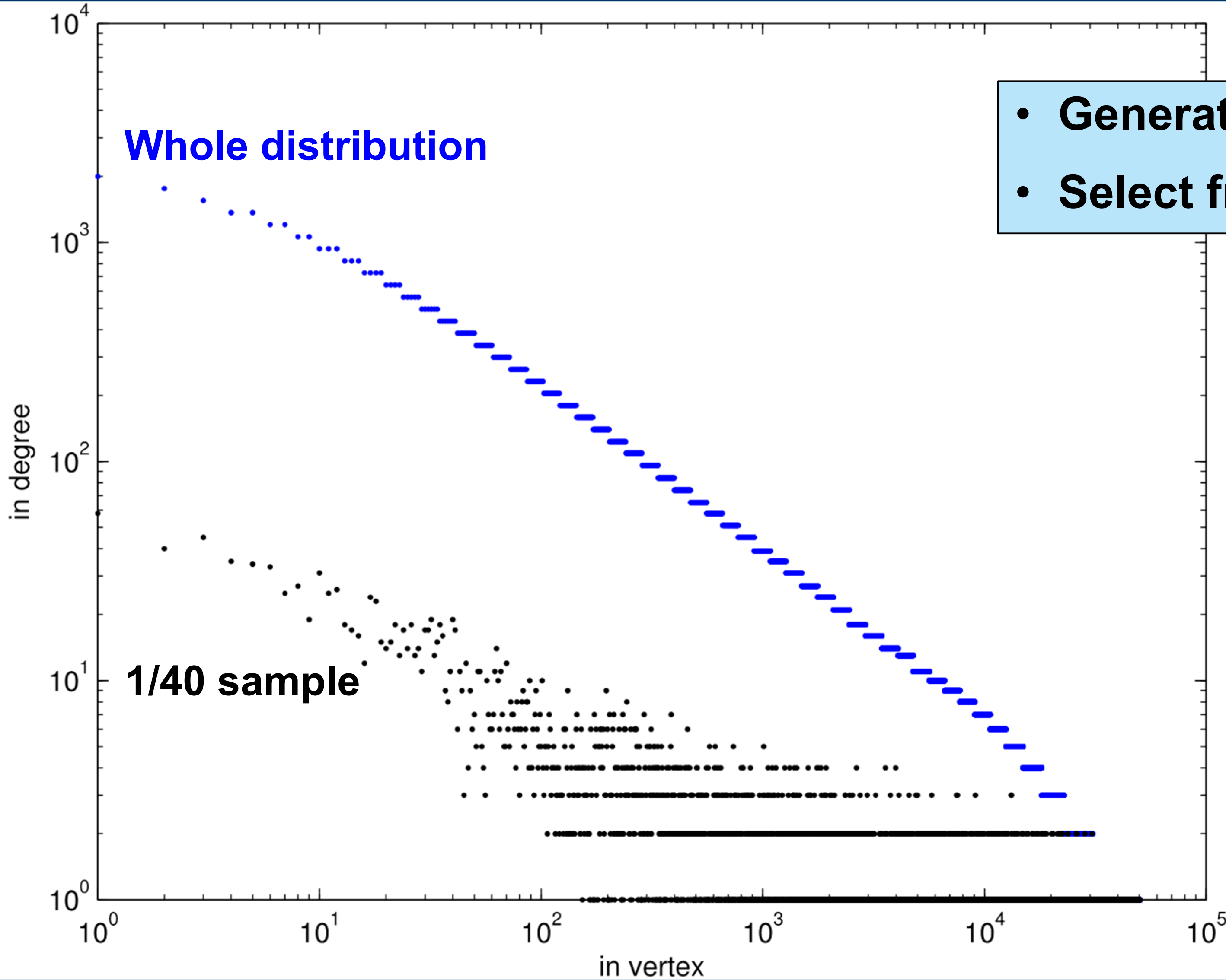
Outline

- Introduction
- Sampling
- **Sub-sampling**
- Reuter's Data
- Summary

Sub-Sampling Challenge

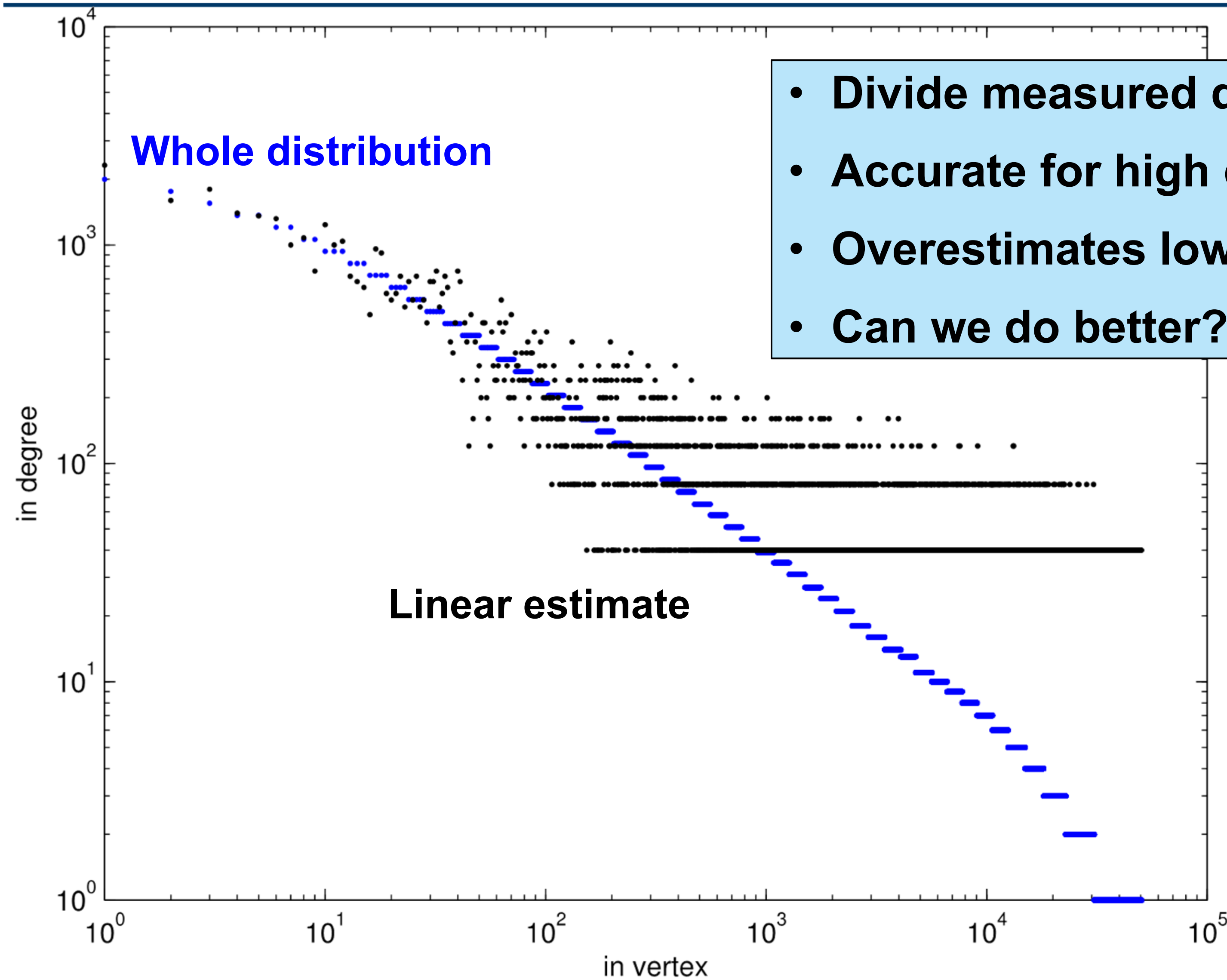
- **Anomaly detection requires good estimates of background**
- **Traversing entire data sets to compute background counts is increasingly prohibitive**
  - **Can be done at ingest, but often is not**
- **Can background be accurately estimated from a sub-sample of the entire data set?**

Sampling a Power Law



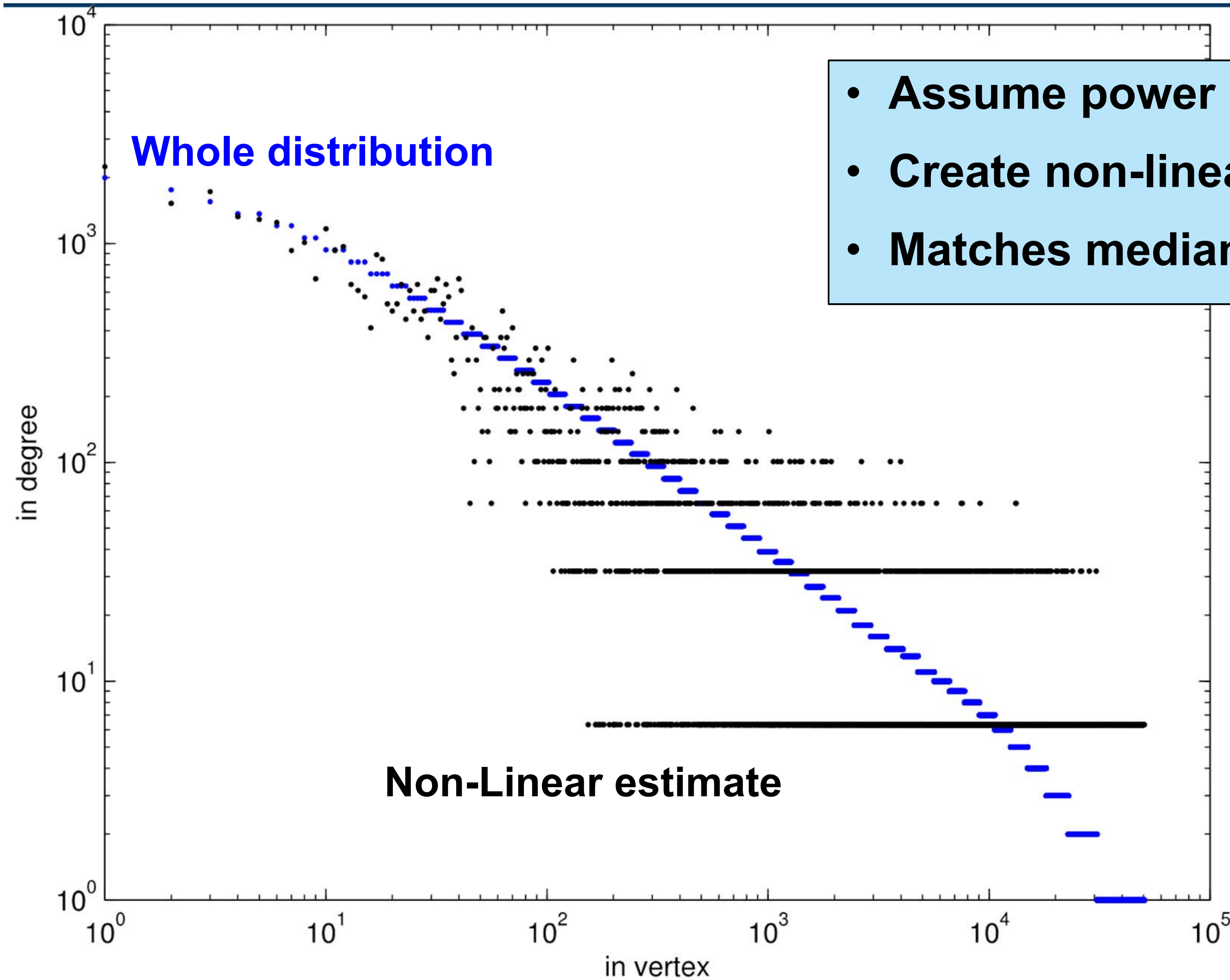
- Generate power law
- Select fraction of edges

Linear Degree Estimate



- Divide measured degree by fraction
- Accurate for high degree
- Overestimates low degree
- Can we do better?

Non-Linear Degree Estimate



- Assume power law input
- Create non-linear estimate
- Matches median degree

Outline

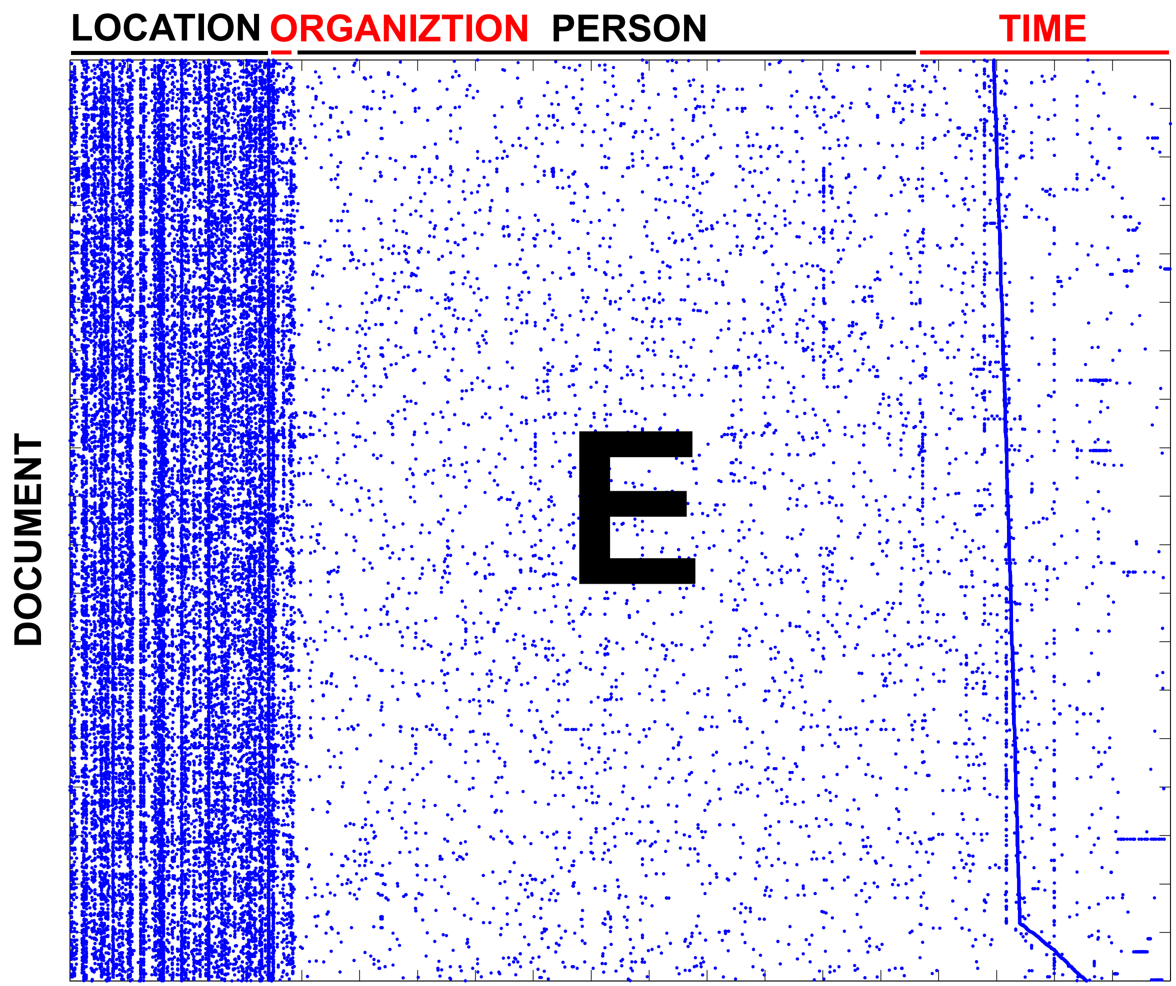
- Introduction
- Sampling
- Sub-sampling
- Reuter's Data
- Summary

Reuter's Incidence Matrix

- **Entities extracted from Reuter's Corpus**
- **E(i,j) = # times entity appeared in document**

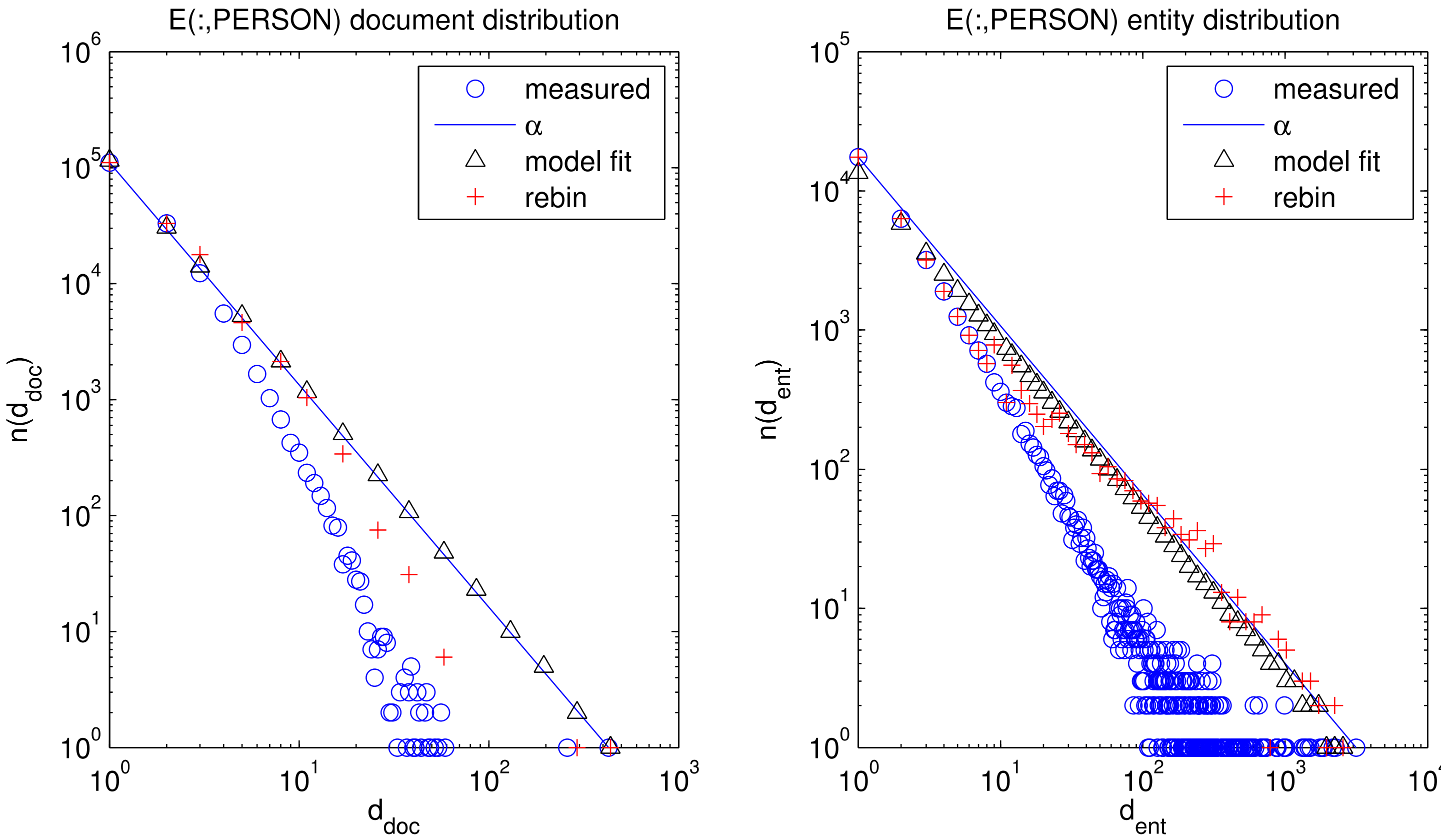
- N_{doc} = 797677
- N_{ent} = 47576
- M = 6132286

- **Four entity classes with different statistics**
  - **LOCATION**
  - **ORGANZATION**
  - **PERSON**
  - **TIME**



- **Fit power law model to each entity class**

E(:,PERSON) Degree Distribution



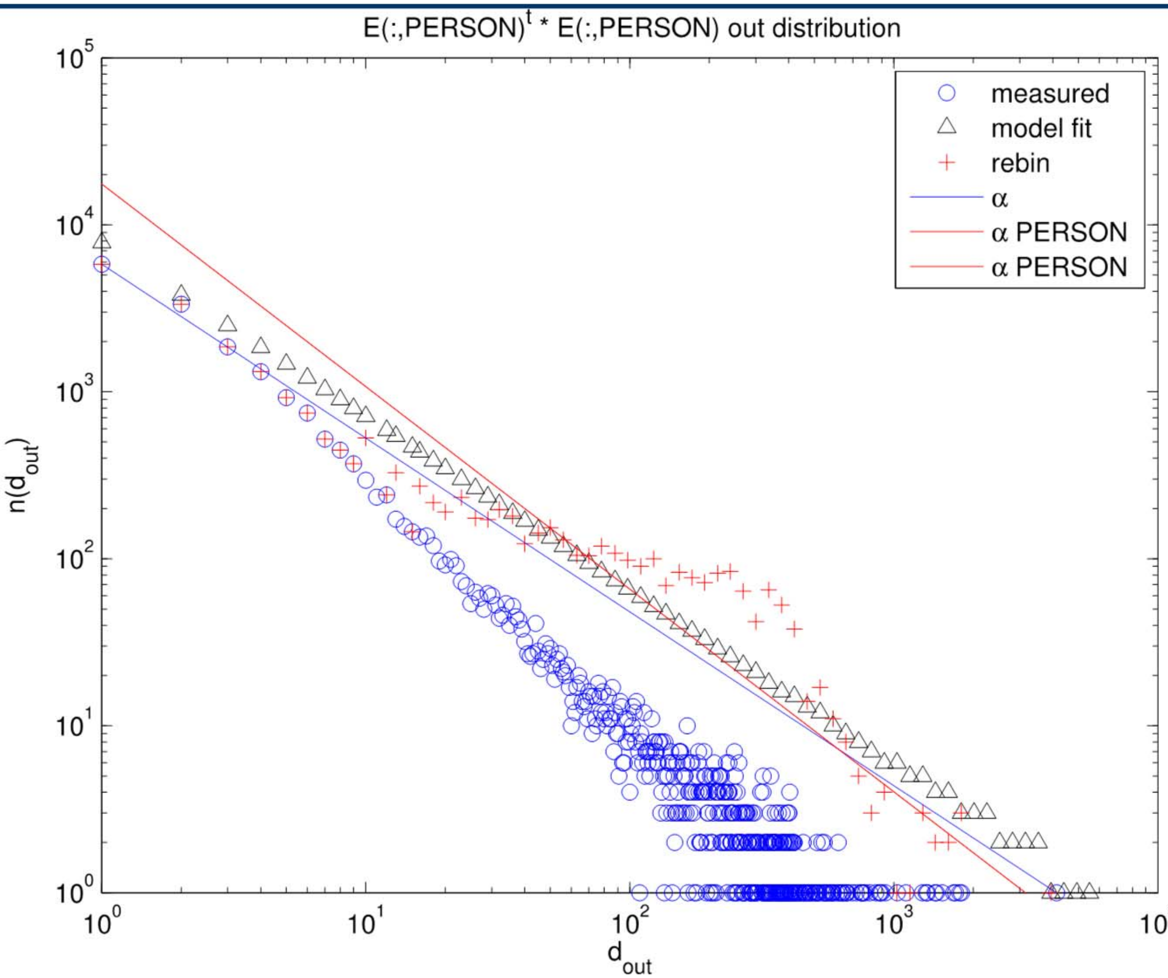
	M	N	M/N	α	M_{fit}	N_{fit}	M_{fit}/N_{fit}
Document	299333	170069	1.76	1.92	302478	170066	1.78
Entity	299333	37191	8.05	1.21	299748	37449	8.00

E(:,PERSON)^t x E(:,PERSON)

Procedure

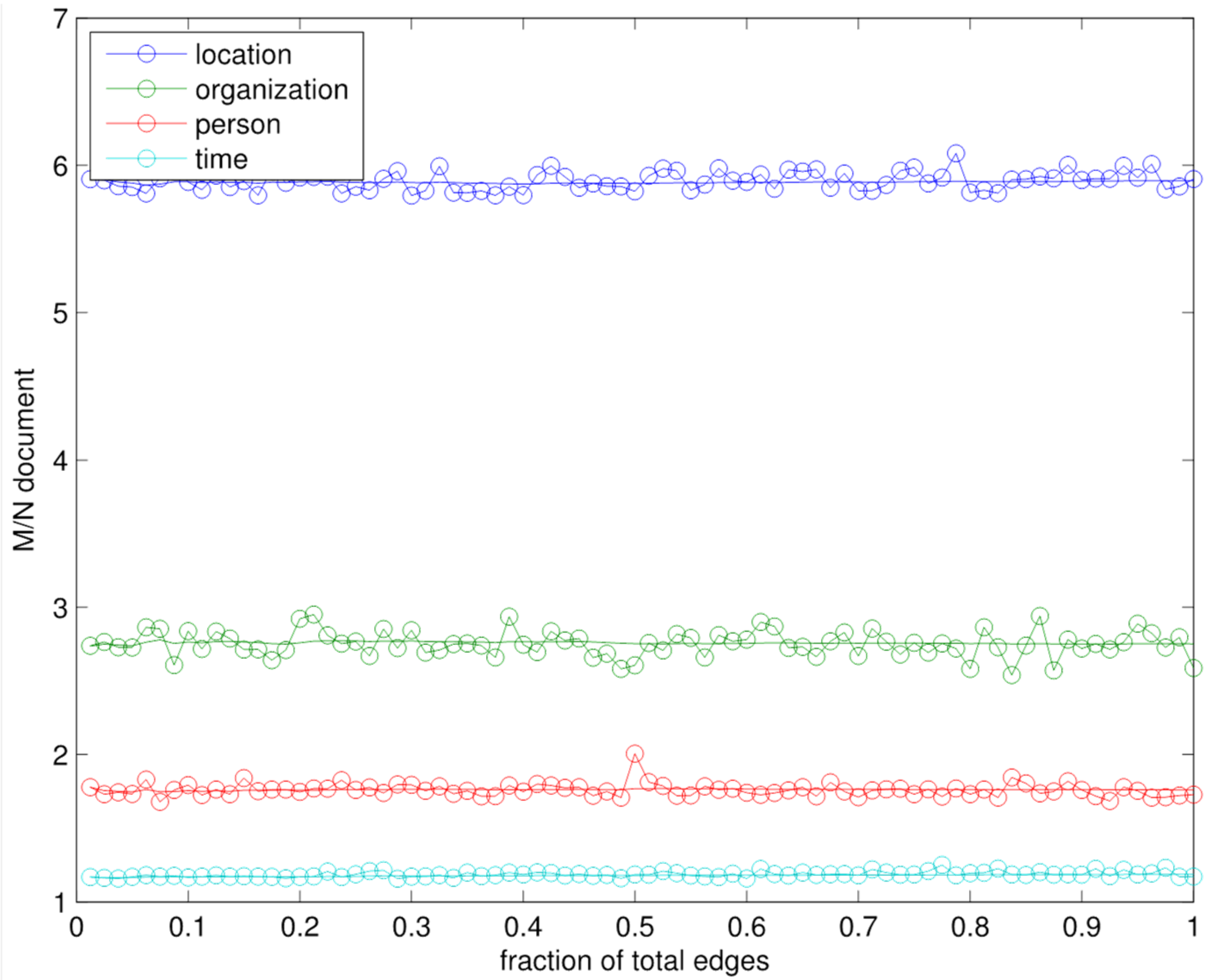
- **Make unweighted and use to form correlation matrix A with no self-loops**

```
E = double(logical(E));
A = triu(E' * E);
A = A - diag(diag(A));
```



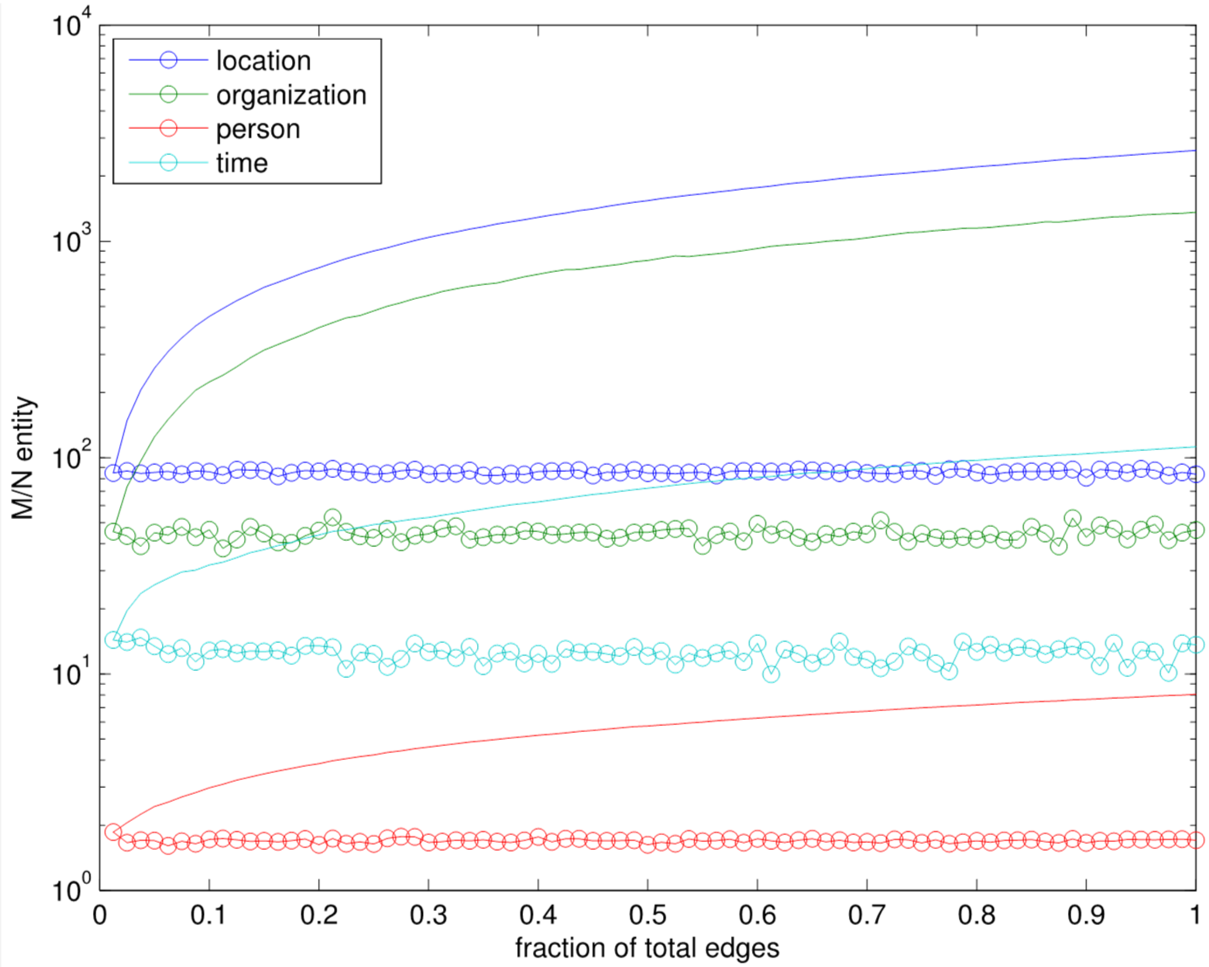
- **Perfect power law fit to correlation shows non-power law shape**
- **Reveals "witches nose" distribution**

Document Densification



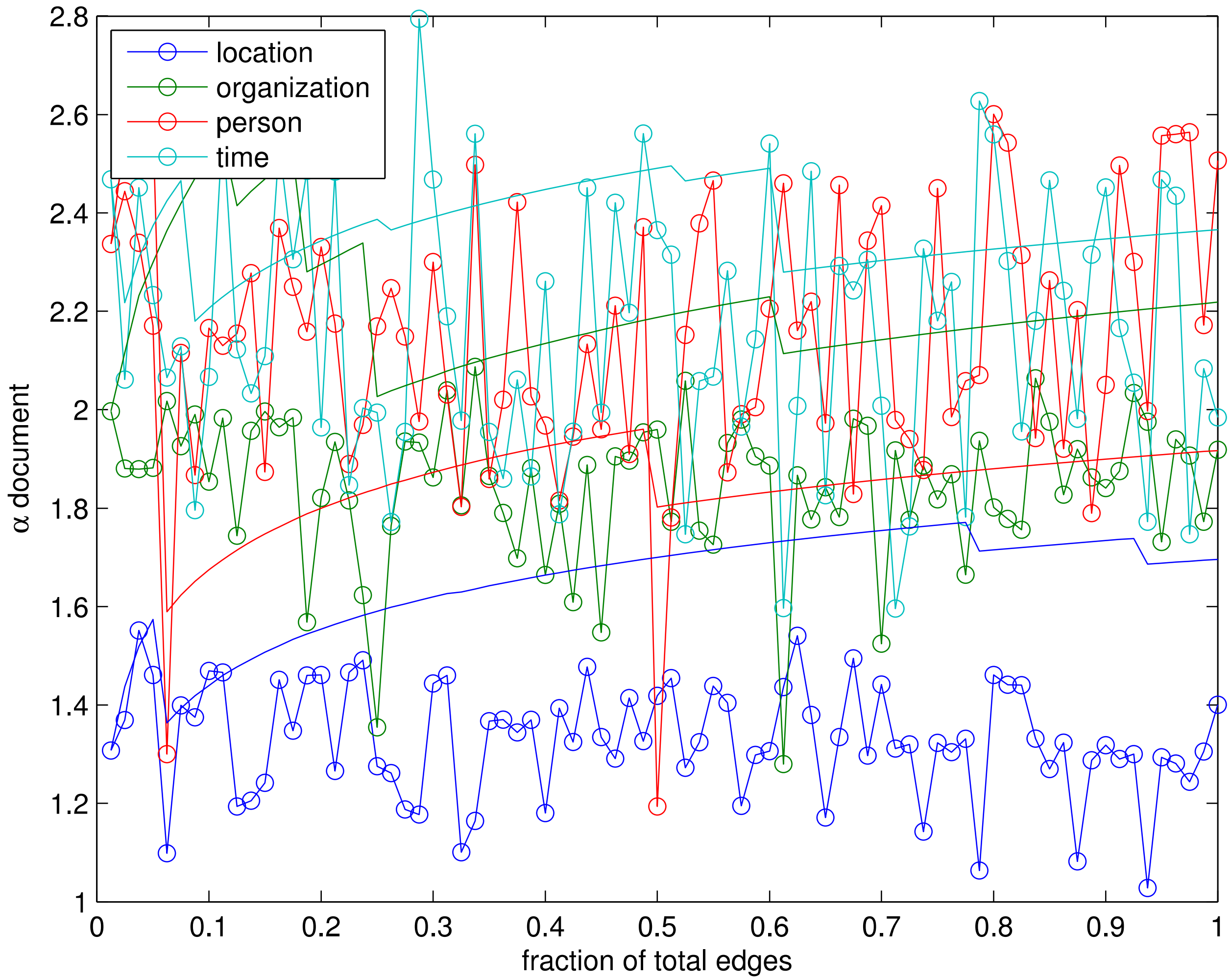
- **Constant M/N consistent with sequential ordering of documents**

Entity Densification



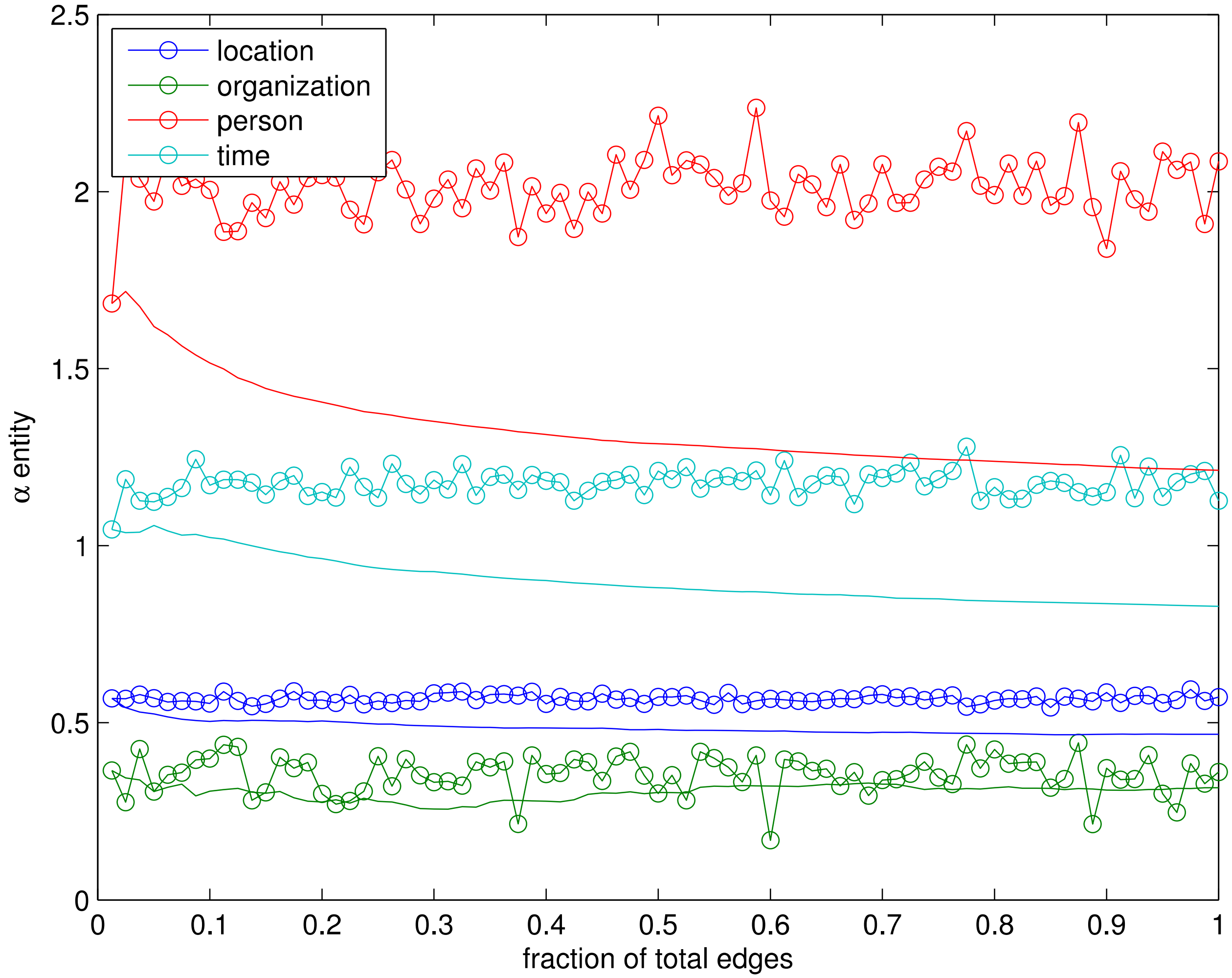
- **Increasing M/N consistent with random ordering of entities**

Document Power Law Exponent (α)



- **Increasing α consistent with sequential ordering of documents**

Entity Power Law Exponent (α)



- **Decreasing α consistent with random ordering of entities**

Summary

- Developed a background model for graphs based on "perfect" power law
  - Can be done via simple heuristic
  - Reproduces much of observed phenomena
- Examine effects of sampling such a power law
  - Lossy, non-linear transformation of graph construction mirrors many observed phenomena
- Traditional sampling approaches significantly overestimate the probability of low degree vertices
  - Assuming a power law distribution it is possible to construct a simple non-linear estimate that is more accurate
- Develop techniques for comparing real data with a power law model
  - Can fit perfect power-law to observed data
  - Provided binning for statistical tests

Acknowledgements

- Nicholas Arcolano
- Michelle Beard
- Nadya Bliss
- Bob Bond
- Matthew Schmidt
- Ben Miller
- Bill Arcand
- Bill Bergeron
- David Bestor
- Chansup Byun,
- Matt Hubbell
- Pete Michaleas
- Julie Mullen
- Andy Prout
- Albert Reuther
- Tony Rosa
- Charles Yee

Appendix

Sub-Sampling Formula

- f = fraction of total edges sampled
- $\underline{n}_1$ = # of vertices of degree 1
- $\underline{d}_{max}$ = maximum degree
- Allowed slope: $\ln(\underline{n}_1)/\ln(\underline{d}_{max}/f) < \alpha < \ln(\underline{n}_1)/\ln(\underline{d}_{max})$

- Cumulative distribution

$$P(\alpha,d) = (f^{1-\alpha}\, \underline{d}_{max}{}^{\alpha} / \underline{n}_1)\, \Sigma_{i<d}\, i^{1-\alpha}\, e^{-fi}$$

- Find α^* such that $P(\alpha^*,\infty) = 1$
- Find $d_{50\%}$ such that $P(\alpha^*,d_{50\%}) = ½$
- Compute $K = 1/(1 + \ln(d_{50\%})/\ln(f))$

- Non-linear estimate of true degree of vertex v from sample $\underline{d}(v)$

$$d(v) = \underline{d}(v) / f^{1-1/(K\, \underline{d}(v))}$$