# Counting Triangles in Large Graphs using Randomized Matrix Trace Estimation: an Overview

Steven Haussmann, Louis Hyde, Peter Wood

Dec. 3, 2018

# Background

A **triangle** is a group of three nodes in a graph that are pairwise connected

- Alice, Bob, and Charlie are all friends with one another
- Three cities are connected by freeways

- The number of triangles in a graph provides interesting information:
  - Clustering coefficient - how likely are nodes to be linked?
- Iterating over nodes or edges takes $O(\sum_{v \in V} deg(v)^2) \approx O(|E|^2)$ time and is hard to parallelize
- Monte Carlo simulation to check if random nodes/edges form a triangle only works for dense graphs

We examined a 2010 paper on the subject, *Counting Triangles in Large Graphs using Randomized Matrix Trace Estimation*

- Authored by Haim Avron of Tel-Aviv University

Avron presents a *randomized approach* to triangle counting.
We examine two of his proposed algorithms.

# Algorithm

# Randomized Trace Estimation

- $A$ is the adjacency matrix
- $x$ is a vector whose entries are independent random variables with expectation 0 and variance 1
- Calculate $y = Ax$
- Then $y^T Ay = (x^T A)A(Ax) = x^T A^3 x$ approximates $tr(A^3)$

# Randomized Trace Estimation, cont'd

- $\mathbb{E}(x^T Ax) = \sum_{i=0}^{n} \sum_{j=0}^{n} \mathbb{E}(x_i A_{ij} x_j)$
- Since $\mathbb{E}x_i = 0$, $\mathbb{E}(x_i x_j) = 0$ for $i \neq j$, so $\sum_{i \neq j} \mathbb{E}(x_i A_{ij} x_j) = 0$
- Since $\mathbb{E}(x_i^2) = \mathbb{E}(x_i^2) + Var(x_i) = 1$, $\mathbb{E}(x_i A_{ii} x_i) = A_{ii}$
- Thus $\mathbb{E}(x^T Ax) = \sum_{i=0}^{n} A_{ii} = Tr(A)$

Two variants of the same algorithm were presented by the author:

- `TraceTriangleN` (normal RVs)
- `TraceTriangleR` (Rademacher RVs)

`TraceTriangleN`

---

**Result:** $\Delta$, the estimated number of triangles

Form an adjacency matrix $A \in R^{n \times n}$;

$M \leftarrow \lceil \gamma \ln^2 n \rceil$;

$i \leftarrow 0$;

**while** $i < M$ **do**

  $x \leftarrow (x_k)$ where $x_i \sim N(0,1)$ ($x_i$ are i.i.d.);

  $y \leftarrow Ax$;

  $T_i \leftarrow (y^T A y)/6$;

  $i \leftarrow i + 1$;

**end**

$\Delta \leftarrow \frac{1}{M} \sum_{i=1}^{M} T_i$;

---

Variants of the algorithm:

- Multiple ways to sample $x$
- Author notes that $N(0, 1)$ gives the best bound
- Rademacher approach is better for implementation, however
- Also theoretically has a better estimator of the trace

In summary:

- Compute the average of $M$ samples
- Samples can be taken in parallel
- Experimental results suggest $M = \lceil \gamma \ln^2 |V| \rceil$

# Analysis

# Speedup

- Any exact algorithm has to involve matrix multiplication, which is naïvely $O(|V|^3)$
- Better multiplication algorithms live somewhere above $O(|V|^2)$
- `TraceTriangle` requires $O(|E|)$ time for each sample
  - Requires $A \times x$ for each iteration, which would make it $O(|V|^2)$ but if $A$ is sparse, this could be made $O(|E|)$
- $\lceil \gamma \log^2(|V|) \rceil$ samples are needed
- Total runtime: $O(\gamma |E| \log^2 |V|)$

Avron describes $(\varepsilon, \delta)$ bounds on error.

- Depends on $\rho(G)$ - the *eigenvalue triangle sparsity* of the graph.
- $\rho(G) = \frac{\sum_{i=1} n|\lambda_i^3|}{6\Delta(G)}$
- Very small triangle counts and very large eigenvalues cause large error bounds
- $\gamma$ required for $\varepsilon = 0.05$, $\delta = 0.05$ is enormous
- Author notes that small $\gamma$ works well *in practice* for commonly encountered graphs.

# Random Bits

- `TraceTriangleN` requires $n$ random numbers per step
- $O(\gamma \log^2 |V|)$ total bits
- `TraceTriangleR` is similar, requiring $n$ random bits
- Avron proposed a third algorithm requiring significantly fewer; not examined here

Random number generation may seem trivial, but becomes difficult in massively parallel situations

# Scalability

`TraceTriangleN` is *embarrassingly parallel*

- Each iteration is independent of all other iterations
- Can easily farm out to many processors

The algorithm can also be streamed.

- One pass requires $O(|V| \log^2 |V|)$ memory
- Less memory is consumed if several passes are made

We evaluated the performance of `TraceTriangleN` and `TraceTriangleR` in three situations:

- Runtime on graphs of varying size
- Runtime and variance of results with varying $\gamma$
- Variance and error of results with varying triangle sparsity

Notes:

- The exact algorithm directly computes $\mathrm{Tr}(A^3)$
- `TraceTriangleN` and `TraceTriangleR` are not parallelized
- $\gamma = 1$ unless otherwise noted
- Sparse matrix multiplication was not performed (increasing runtime)

# Runtime

We evaluated runtime for graphs of varying size.

- $|V| = 1000, 2000, \ldots, 10000$
- $\gamma = 1$
- 25 trials

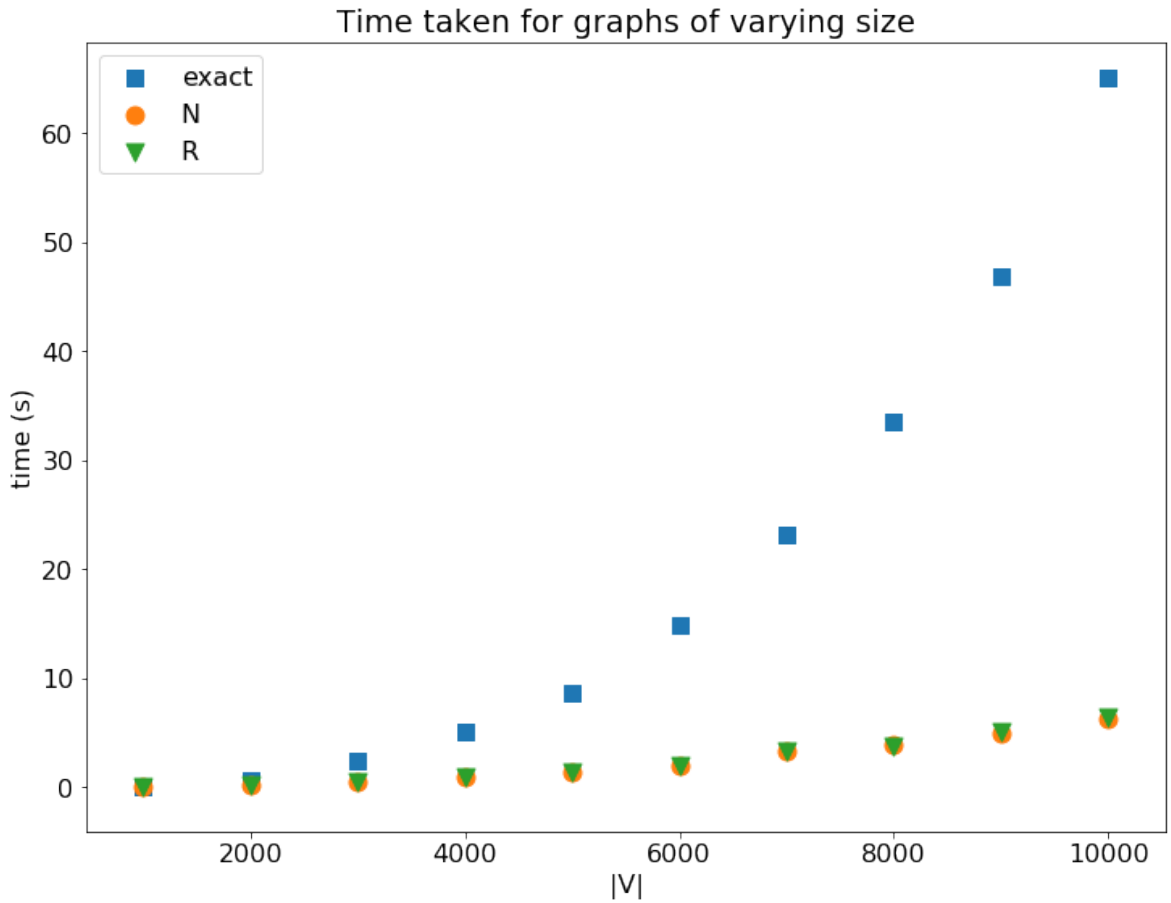


Figure: Exactly computing $\mathrm{Tr}(A^3)$ proves to be far more expensive than the randomized approach.

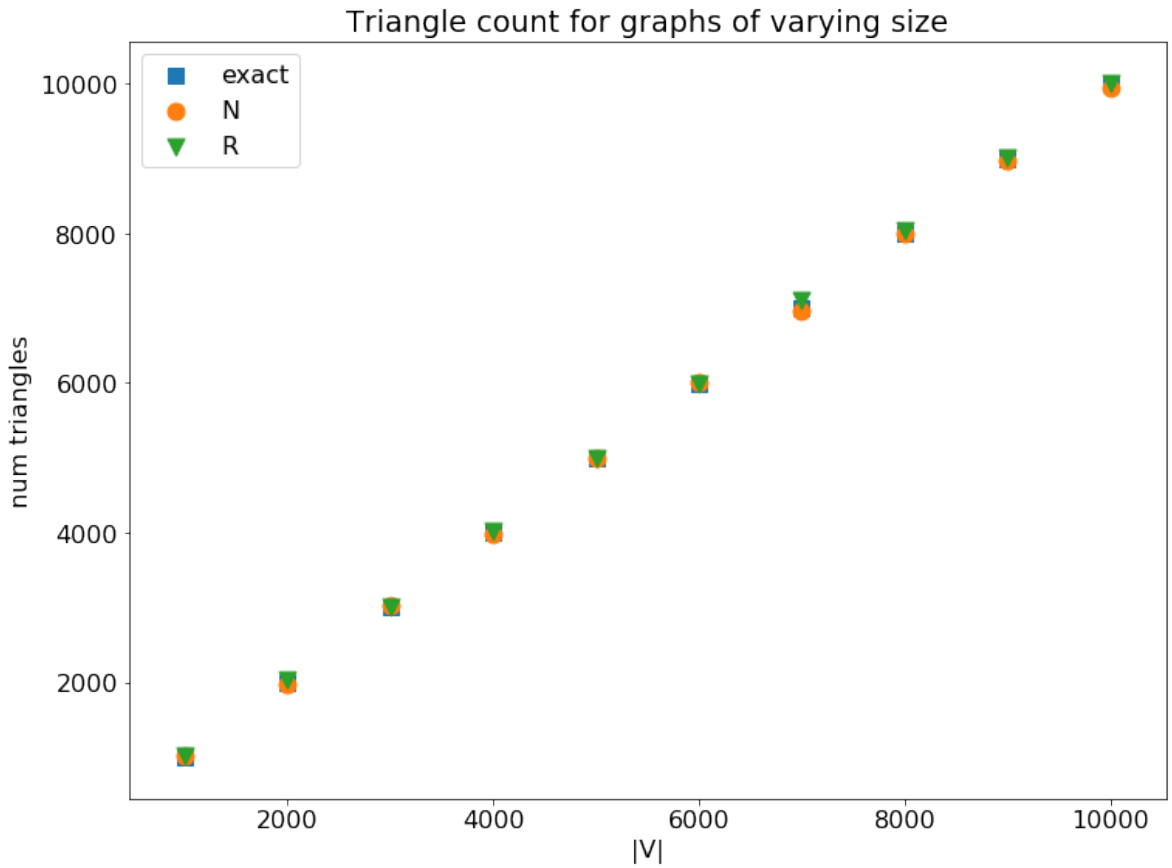


Figure: $\gamma = 1$, but values presented are the average of 25 trials (similar to letting $\gamma = 25$). We examine accuracy more closely next.

# Gamma

We examined the variance and average absolute percent error, where:

- $|V| = 1000$
- $A$ is tridiagonal
- $\gamma = 1, 2, \ldots, 10$
- 100 trials were performed for each value of $\gamma$

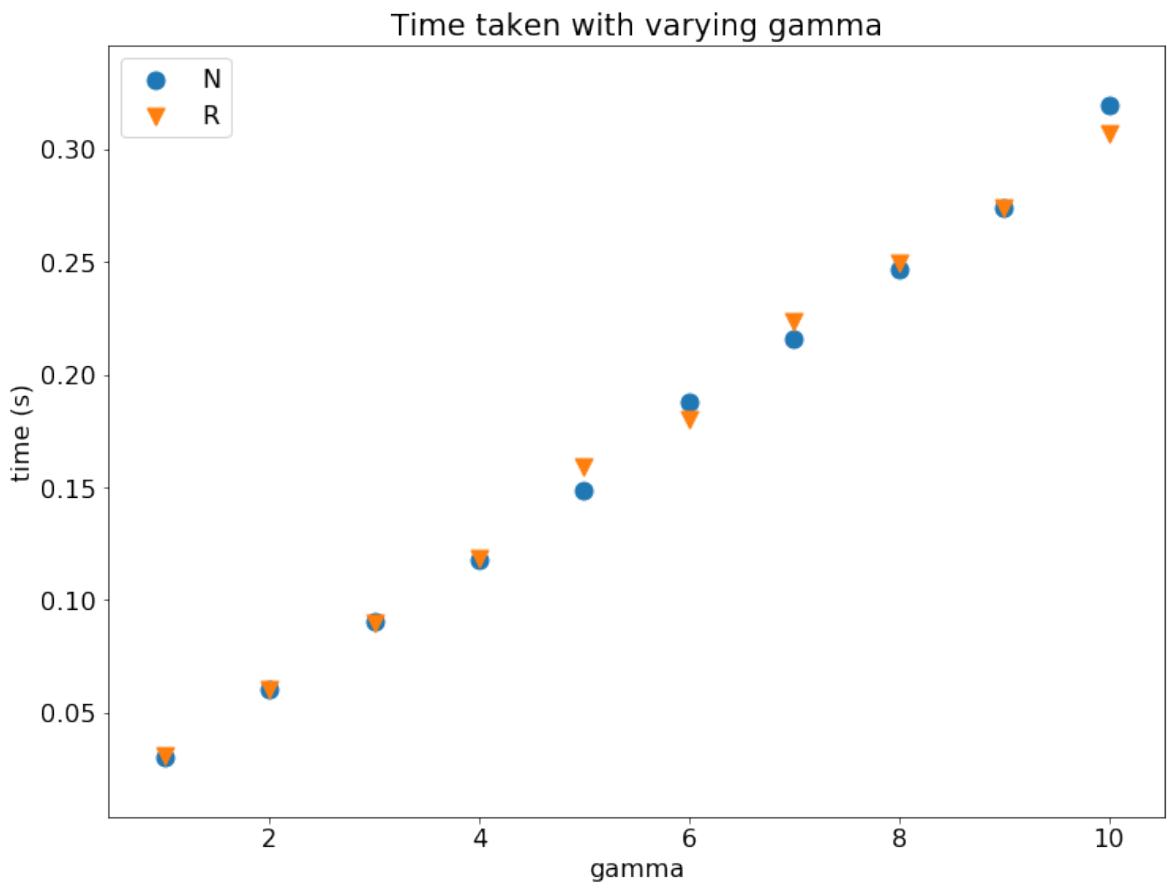


Figure: As expected, time taken scales linearly with $\gamma$

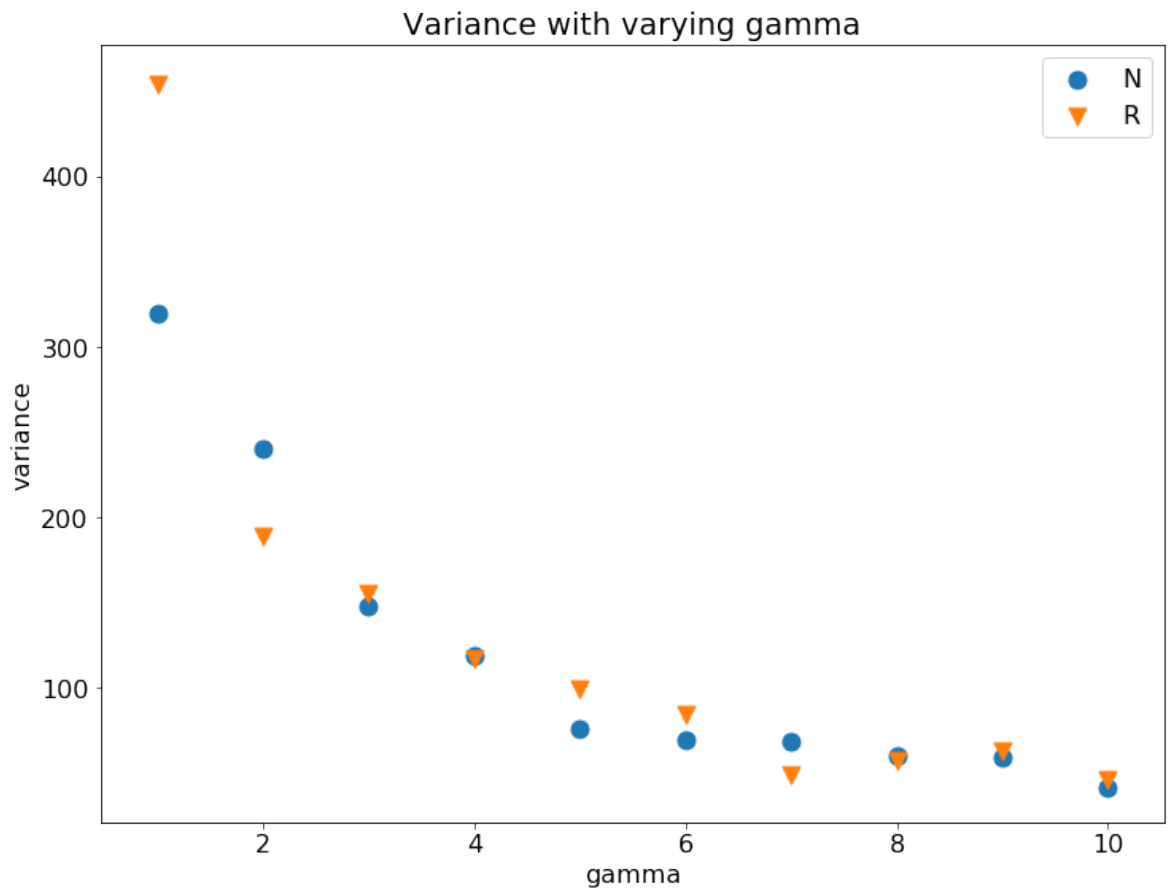


Figure: Variance declines as $\gamma$ increases. Note the apparent noise, even with 100 trials.

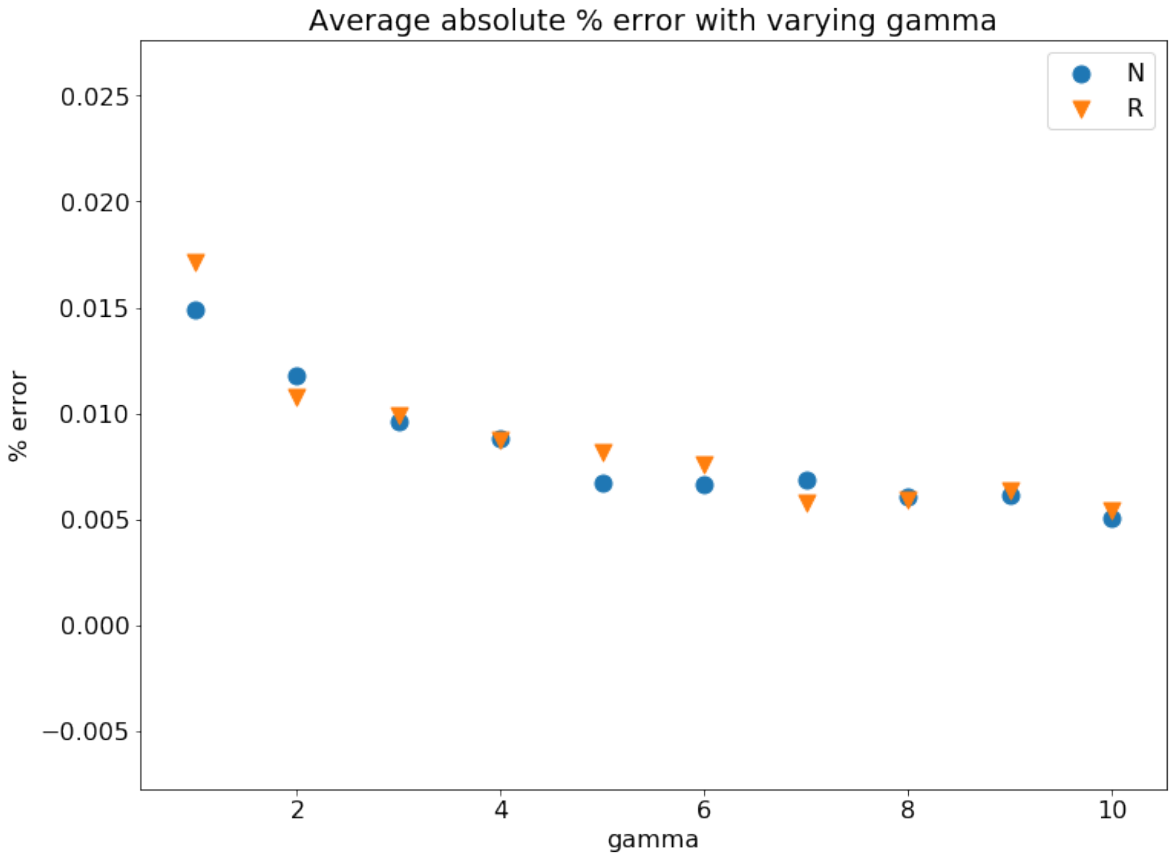


Figure: Average absolute percent error also declines.

# Sparsity

We examined the variance and average absolute percent error, where:

- $|V| = 100$
- $\max(\Delta(G)) = 100 \times 99 \times 98/6 = 161700$
- $A$ has $n$ number of triangles added randomly:
  - Select random distinct $i, j, k$
  - Connect vertices $i, j, k$ together
- $n = [2^0, 2^{\frac{1}{2}}, 2^1, \ldots, 2^{20}]$
- $\gamma = 1$
- 25 trials were performed for each value of $n$

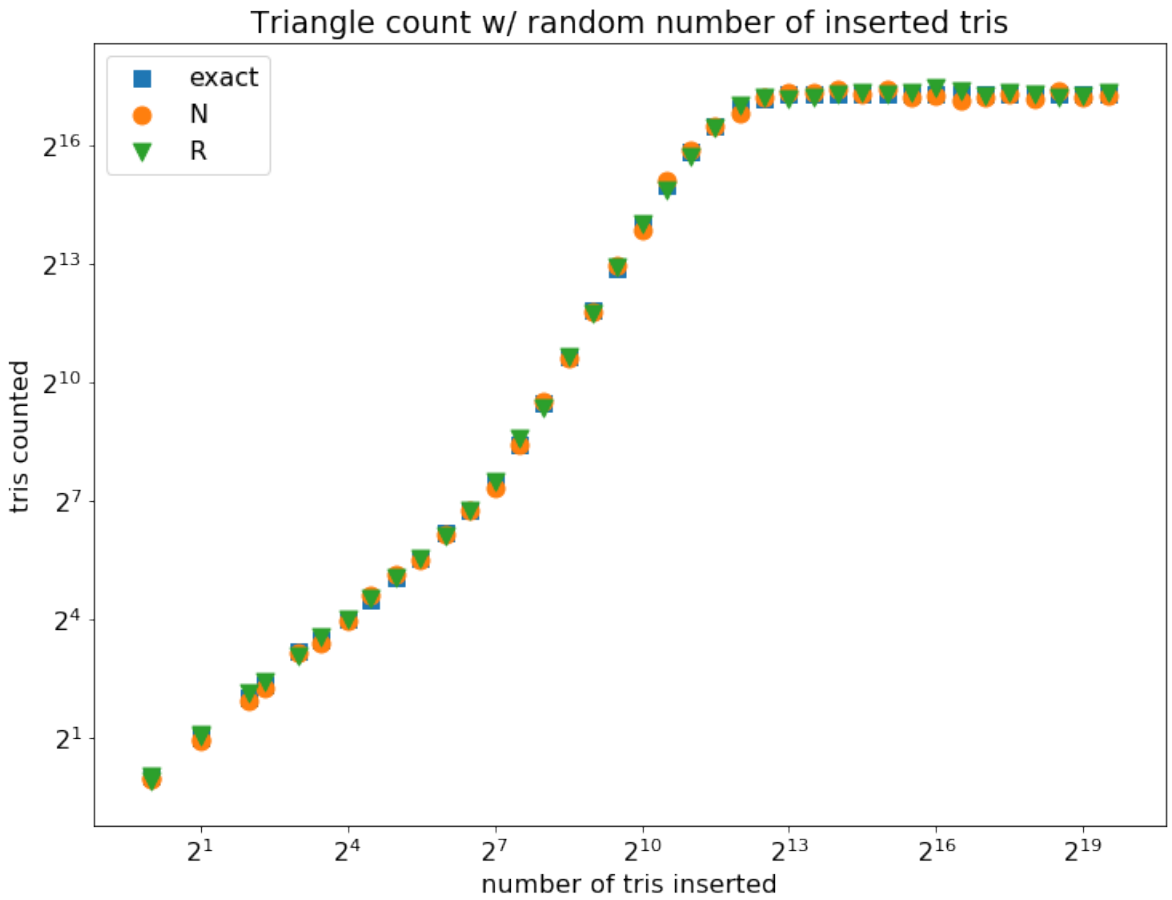


Figure: Note how each added triangle may actually create more than one triangle, and how the graph becomes "saturated"

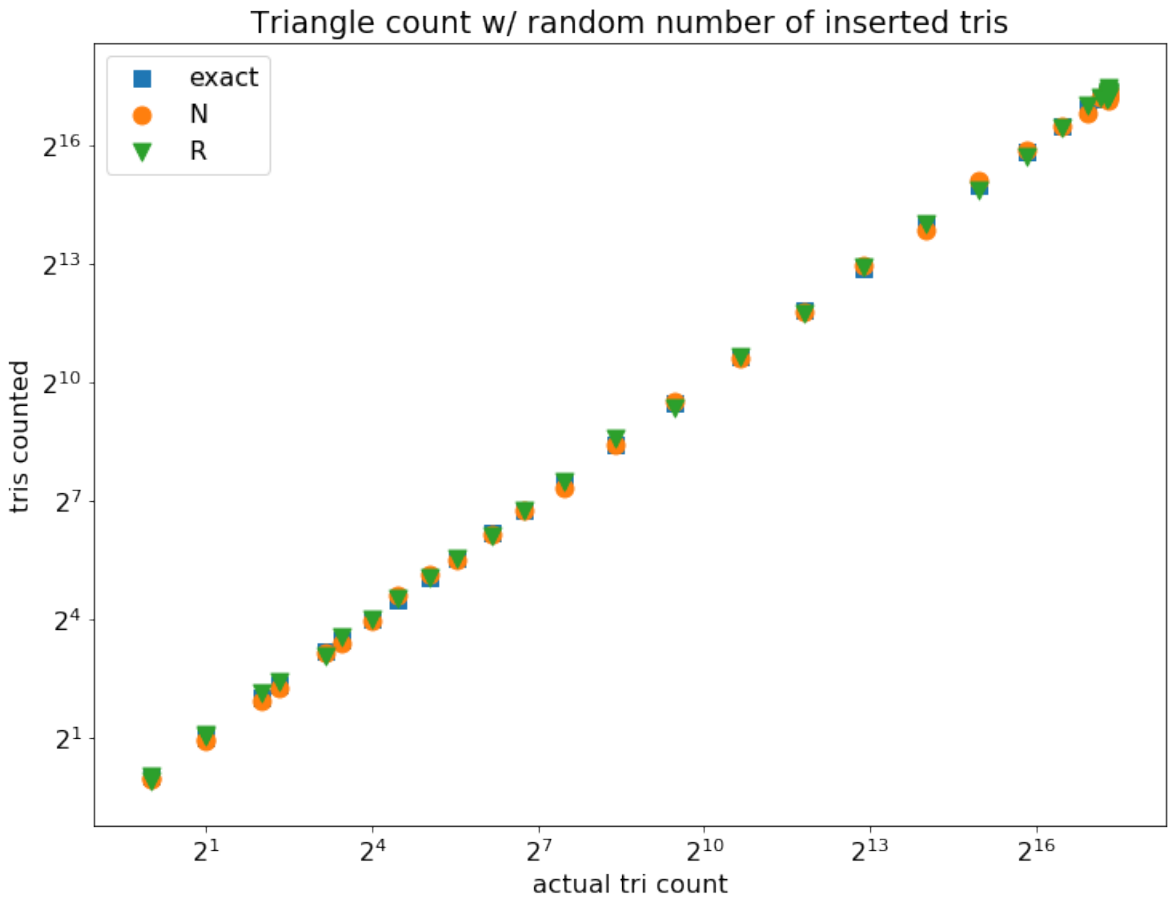


Figure: Replacing the x-axis with the *actual* number of triangles gives a straigther line.

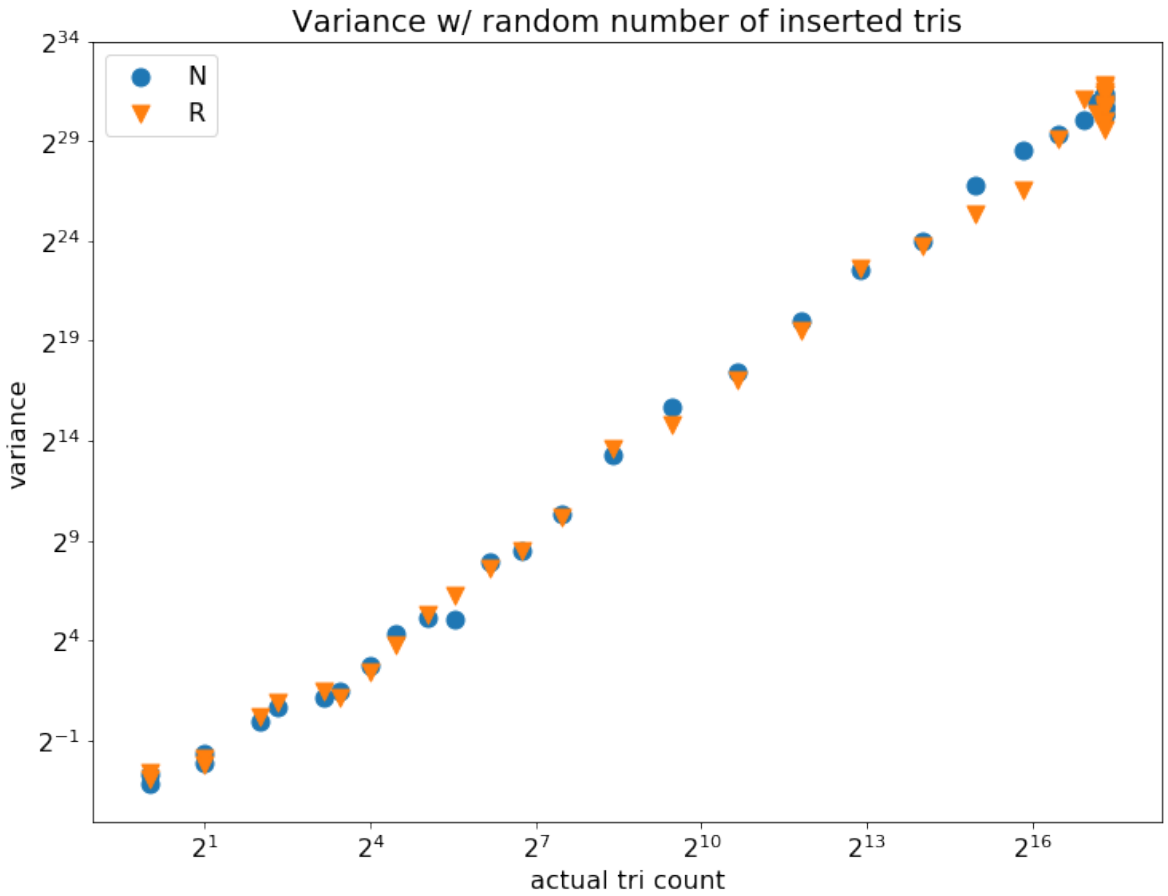


Figure: The variance grows with the number of triangles, as expected. A slight dip is observed in the sparser regions.

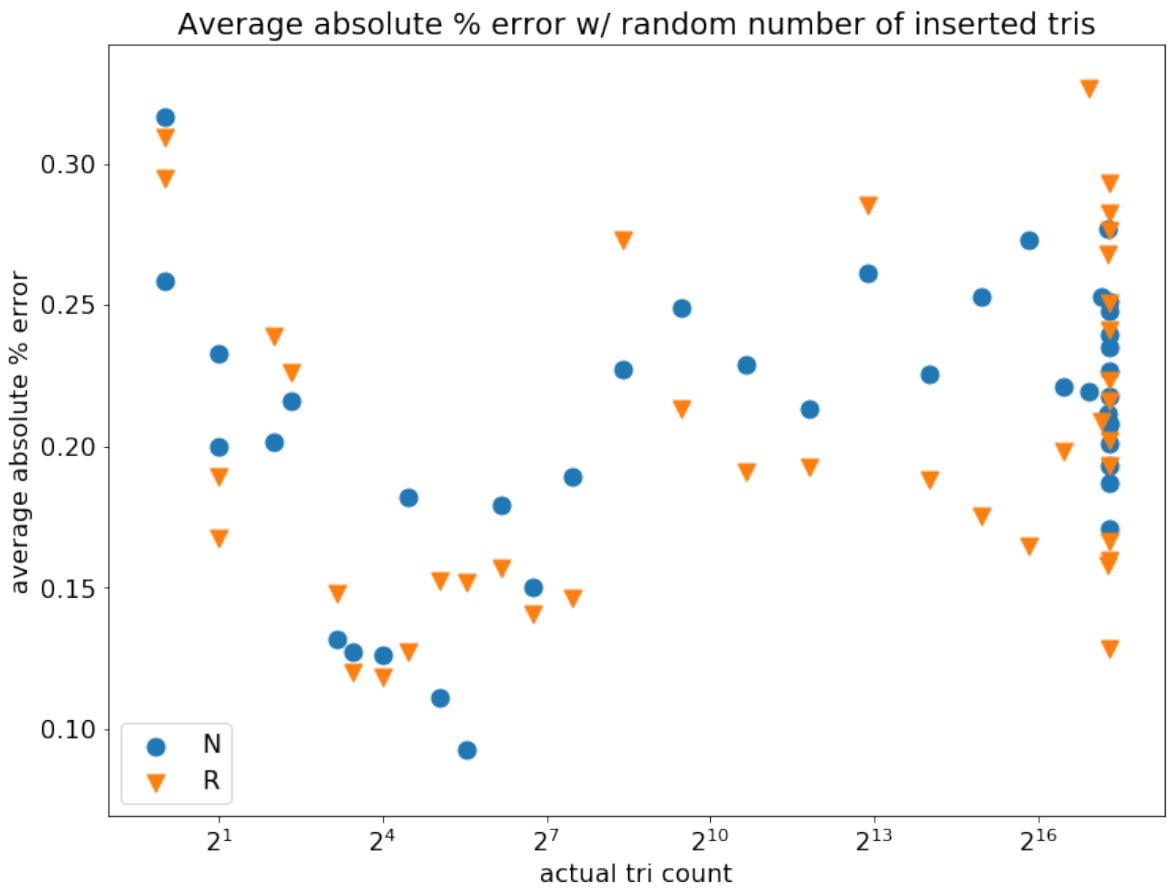


Figure: The average absolute percent error is lowest when the graph is sparse (but not *too* sparse). Note that the repeated trials for a completely full graph gave widely varying results.

# Conclusions

Objective

- `TraceTriangle` offers significant speedup over the naïve approach
- Parallelization is straightforward, although bandwidth is a concern.
- Accuracy is reasonable
  - Being an unbiased estimator means many trials can be averaged
- Somewhat questionable theoretical basis - low values of $\gamma$ don't provide strong guarantees, but do work on real data.

Subjective

- *Counting Triangles* has over 50 citations
- Builds on both deterministic and randomized work
- 2010 paper, so work has been done since publication