Experimental Asymptotic Analysis of Algorithms

NISS

Catherine C. McGeoch

March 2008

Algorithm = Abstraction

Algorithm

Program/Code

Running Process

Quicksort A:

Select element x from array A: constant cost

Partition A around x: linear cost

Recur to left of x

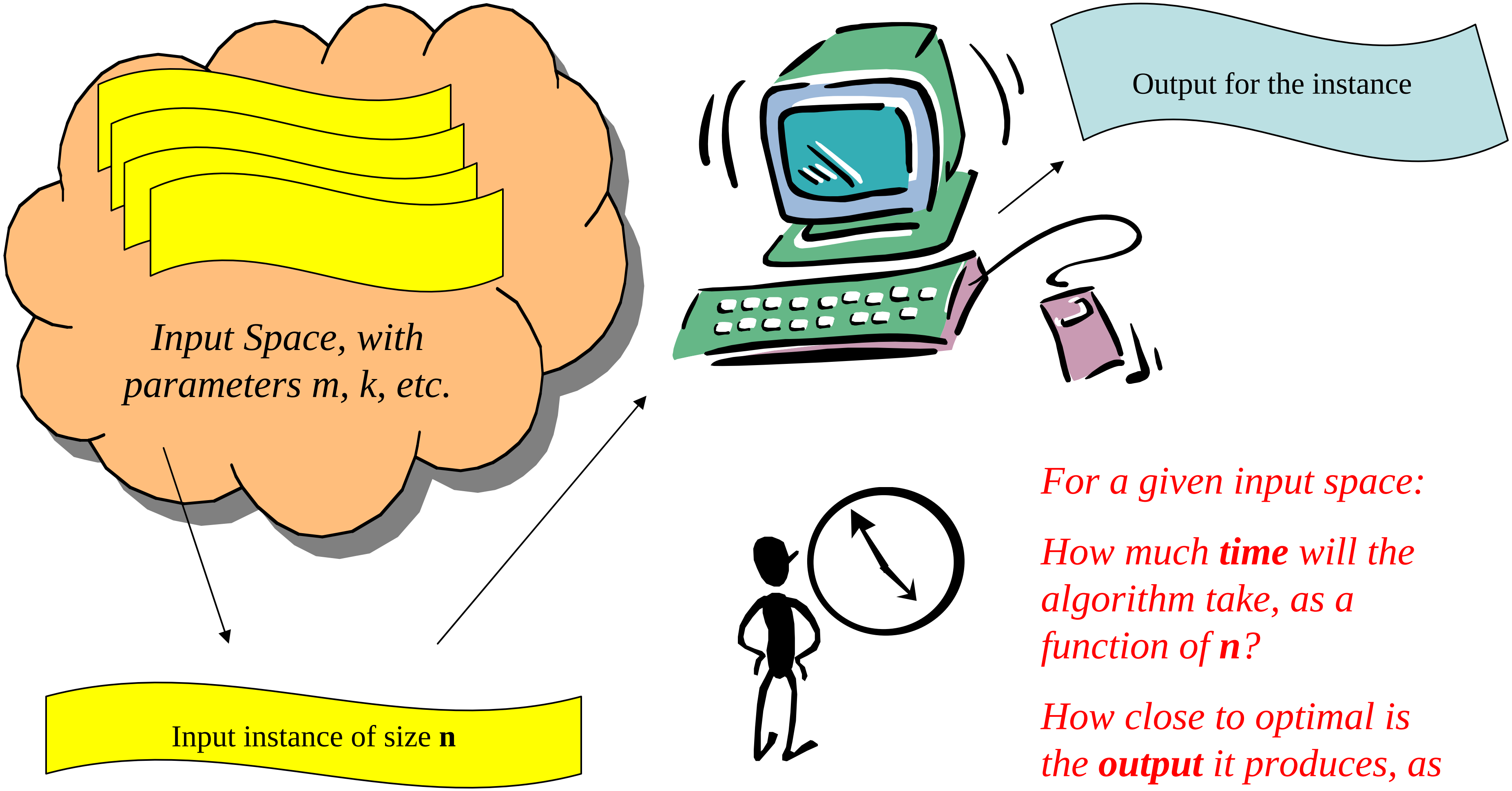
Recur to right of x

```
void Qsort(A, l, h) {
        if (l >= h) return;
        int p = Partition(A);
        Qsort(A, l, p-1);
        Qsort(A, p+1, h);
}
```

Measure this

Draw conclusions about this

Analyzing an Algorithm



Input Space, with parameters m, k, etc.

Output for the instance

Input instance of size **n**

For a given input space:

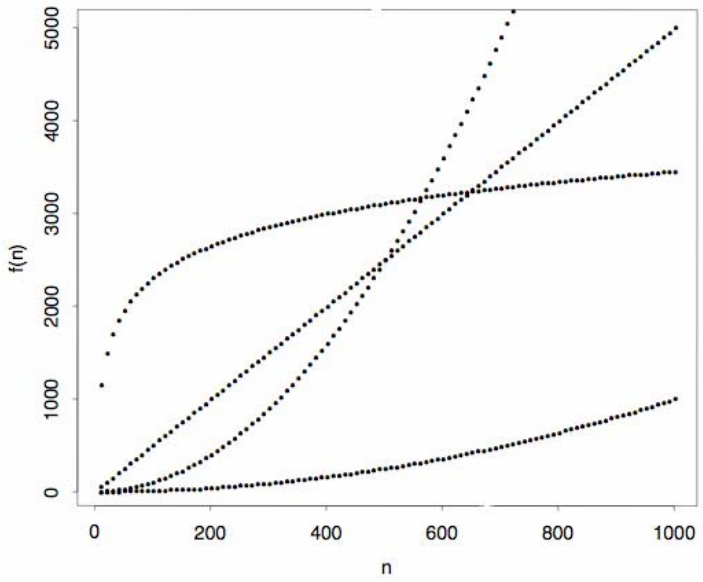
*How much **time** will the algorithm take, as a function of **n**?*

*How close to optimal is the **output** it produces, as a function of **n**?*

Asymptotic Analysis

Definition: A function $f(n)$ is in the set $O(g(n))$ if there exist constants $c > 0$ and $n_0 > 0$ such that

$$0 \leq f(n) \leq c \cdot g(n) \quad \forall n > n_0.$$



What is the *order of the leading term* of the function? What is an *upper (lower) bound* on the order of the leading term?

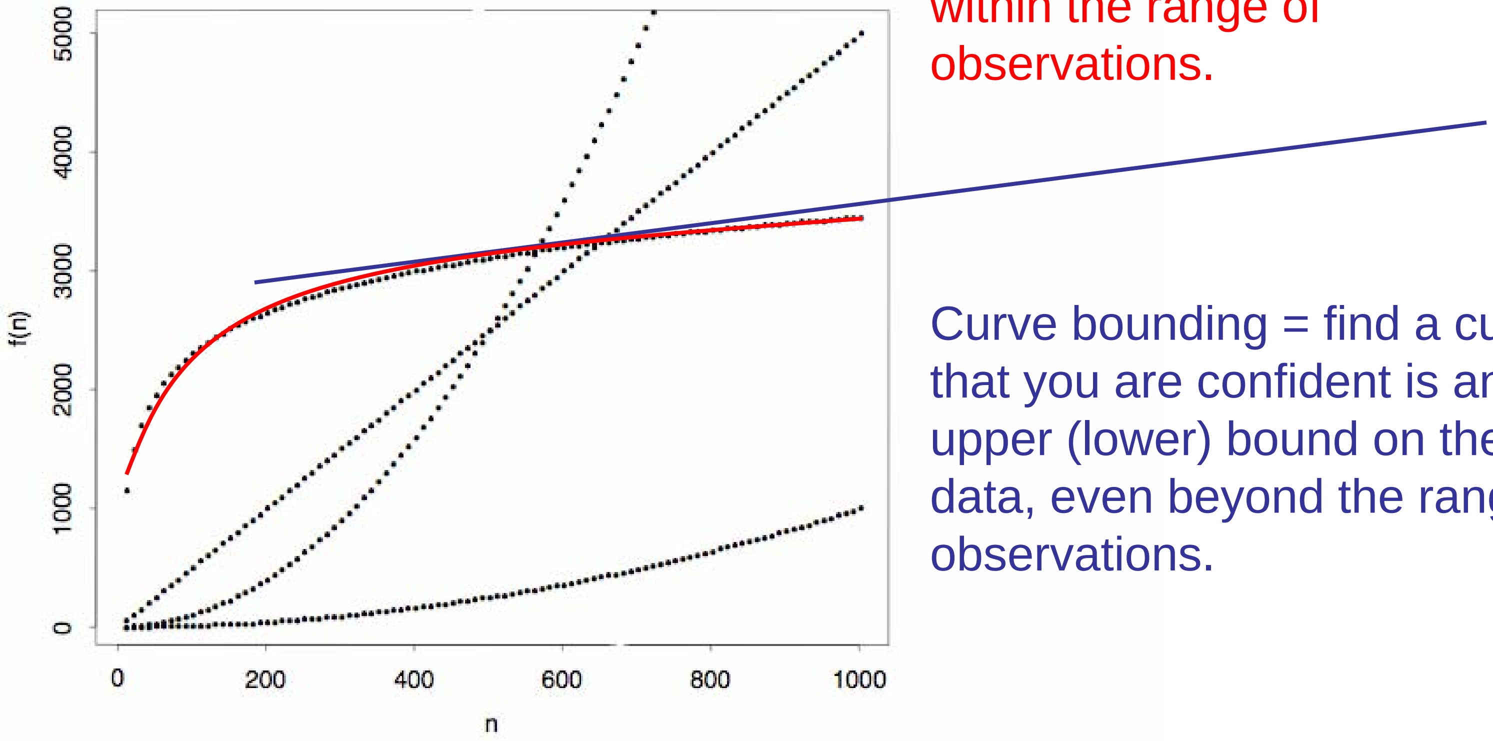
$f(n) = 3n^2 - 6n + 12$ is $O(n^2)$

$f(n) = 0.5n + \log_2 n$ is $O(n)$

$f(n) = 20 \log_2 n + 4$ is $O(\log n)$

$f(n) = 3500 - 7/n$ is $O(1)$

Asymptotic Curve Bounding



Curve fitting = find a curve that is close to the observed data within the range of observations.

Curve bounding = find a curve that you are confident is an upper (lower) bound on the data, even beyond the range of observations.

Why Asymptotic Algorithm Analysis?

- Dominant cost model explains / predicts performance best when *n* is large.
- We care more about cost when *n* is large.
- Death, taxes, problem sizes: *n* will be larger in the future.
- Asymptotic properties are universal, fundamental, and independent of transient technology (platforms, programming languages, coding skills).

Average-Case Analysis

- Input: Draw instances of size *n* at random from parameterized space *S(m, k, ...).*
- Experiment: Measure algorithm performance in several independent trials for varying *n, m, k*...
- Goal: Find an asymptotic function $C_{m,k}(n)$ that bounds the mean cost (Time or Solution Quality).

Experiments on Algorithms

Good news	*Bad news*
Nearly total control over the experiment. Algorithms are easy to probe. Simple mechanisms, models (compared to living things). Lots of data points, usually. Model validation not much of a problem.	Unusual data: skewed, bounded, nonmonotonic, stepped. Unusual questions: Asymptotic analysis. Unusual questions: Curve bounding vs curve fitting. Unusually precise questions: is it *O(n)* or *O(n log n)*?

Outline

- Three Case Studies in Algorithm Research
  - *FF Rule for Bin Packing*
  - *All Pairs Shortest Paths with Essential Subgraph*
  - *Sampling Graph Colorings*
- Some Data Analysis "Techniques" I've Tried
  - *Power Law*
  - *Guessing*
  - *Data Transformation*
  - *Others*
- My Questions, Your Questions

Three Case Studies, Many Questions

- FF Rule for Bin Packing
- All Pairs Shortest Paths with the Essential Subgraph
- Sampling Graph Colorings with Jerrum's algorithm

•*How do I analyze the data to find asymptotic bounds?*

•*How do I assess the quality of my analysis? How confident am I in the results?*

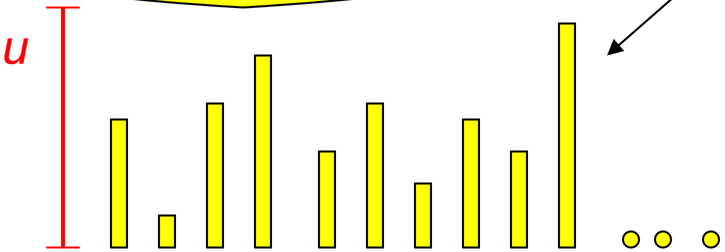
•*Where do I place sample points? How many random trials?*

•*How do I design my second experiment?*

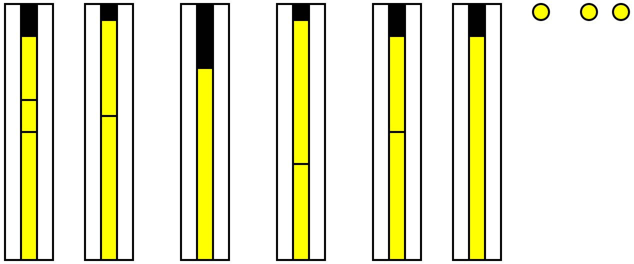
•*Which performance measures are easier to analyze? How can I tell in advance?*

First Fit (FF) Bin Packing

Input: List of *n* item sizes drawn uniformly iid from *(0,u), 0 < u <= 1.*



First Fit Algorithm: Pack items into unit-sized bins



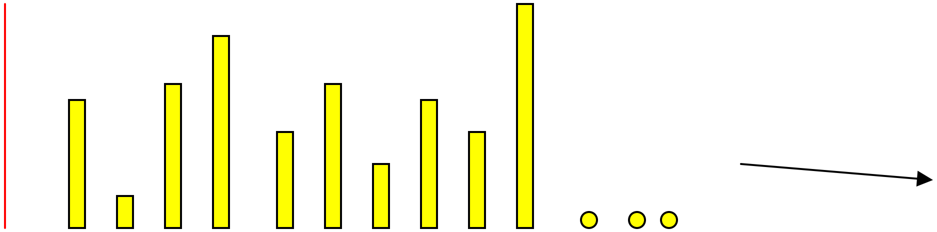
Solution Quality:

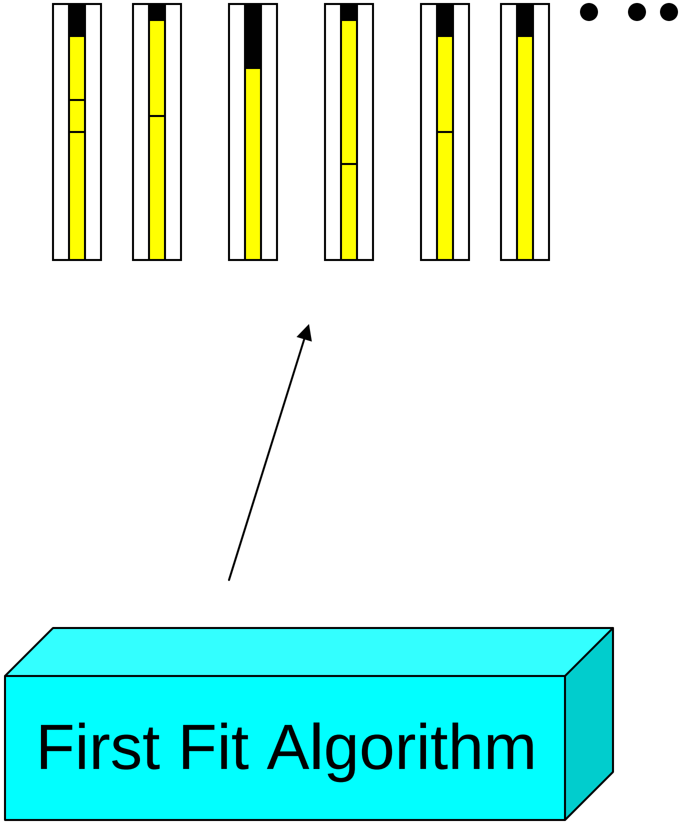
How much empty space in the packing?

First Fit (FF)

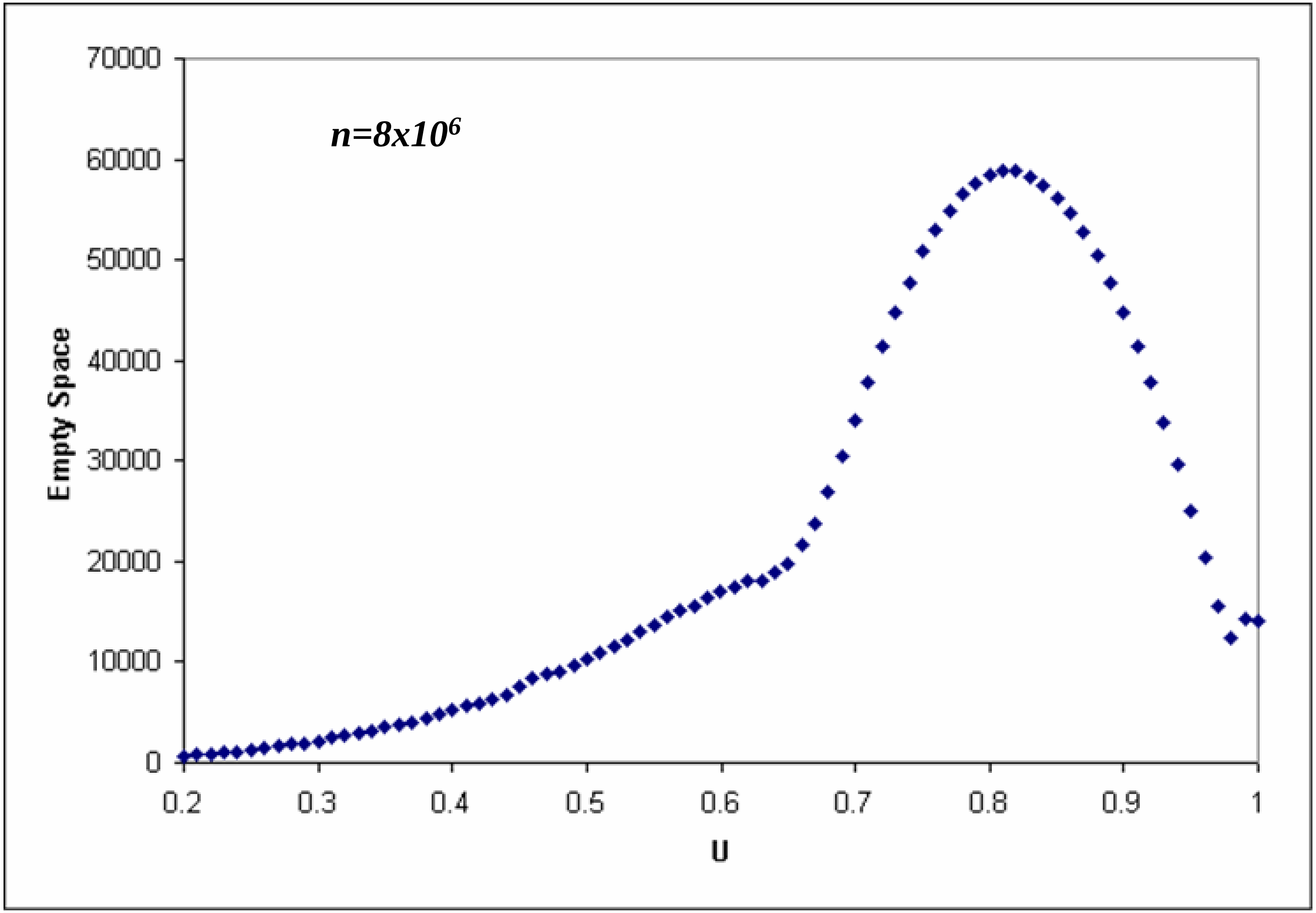
For given u, mean empty space $e_u(n)$ is either asymptotically linear or strictly sublinear in n. Sublinearity implies optimality.

For which values of u is $e_u(n)$ optimal?





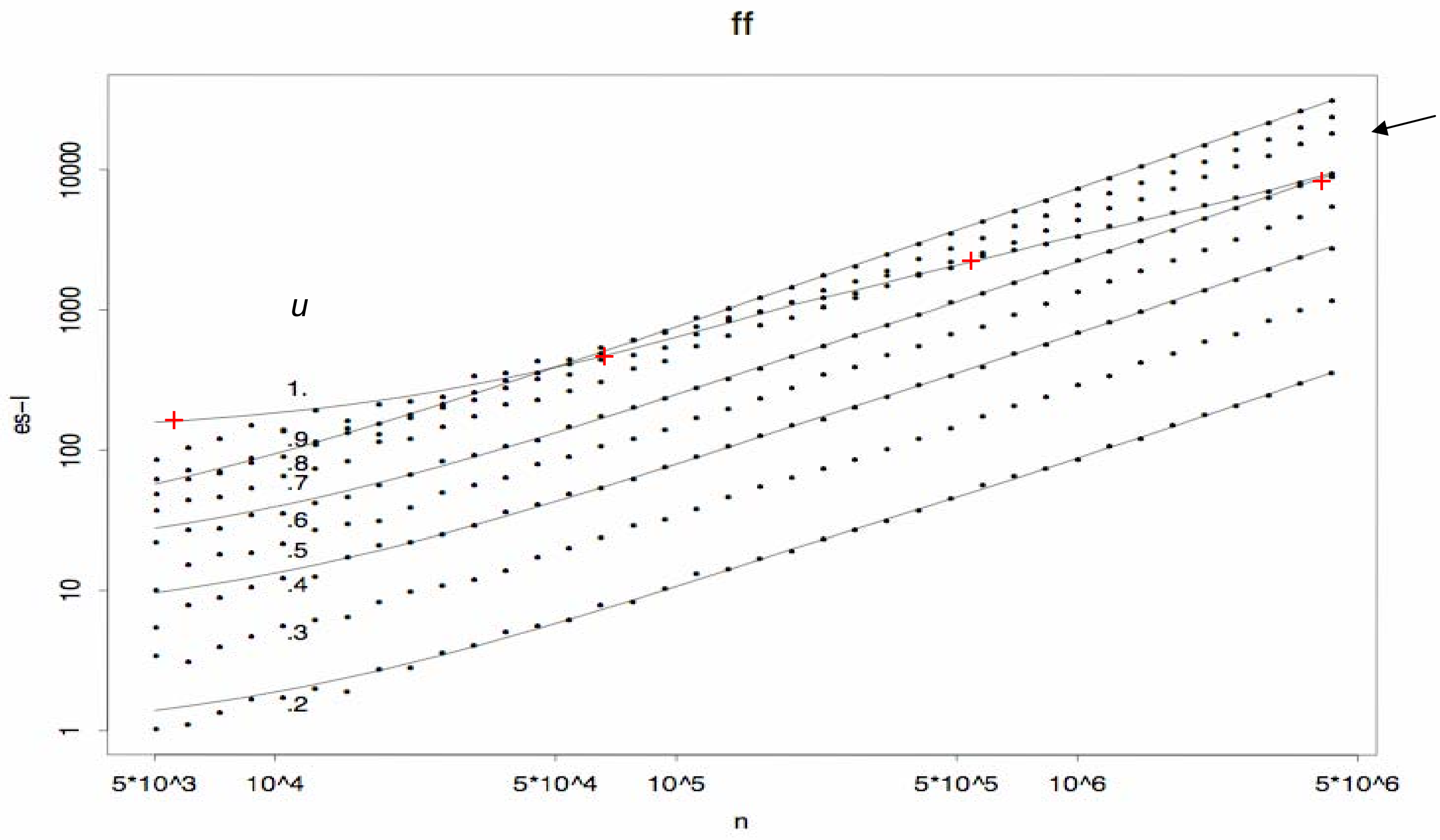
Empty Space at N = 8 million



Some values of u produce bad FF packings. How bad? Which values of u?

Empty Space growth in n

Power law: Linear regression on log-log scale. Analyze slope: If $e = an^b$ then $\log e = b \log n + \log a$

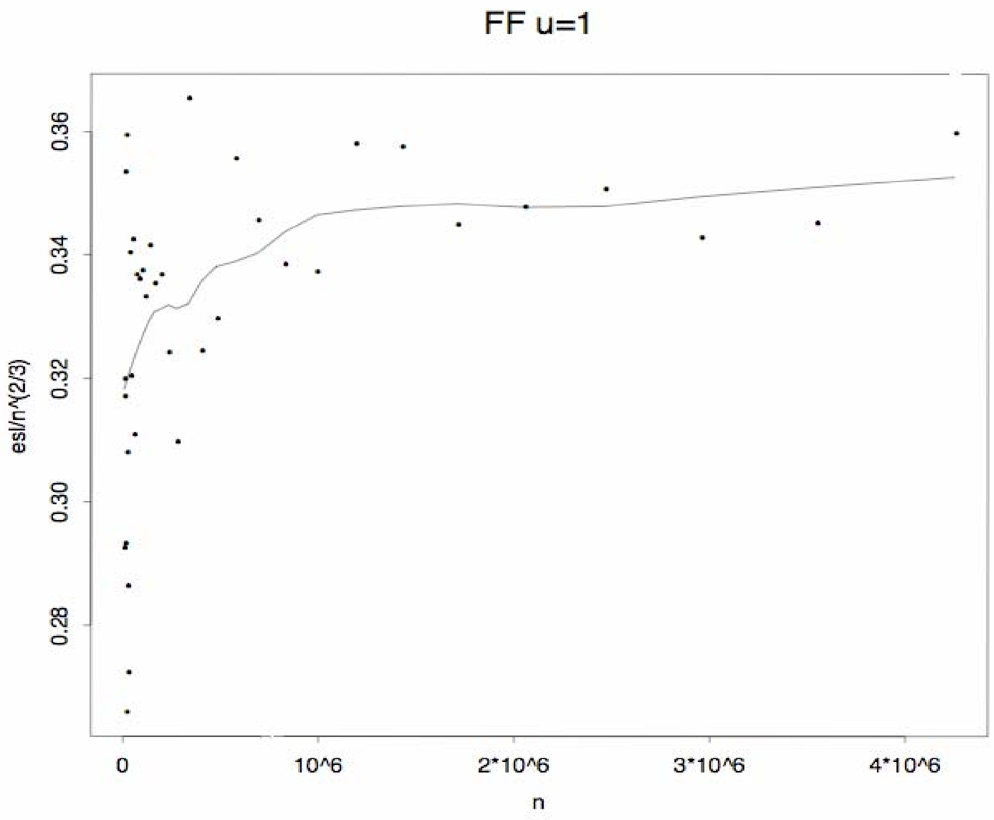


u=1 appears to be sublinear, slope near 0.68.

Others have slopes in (0.974 ... 0.998). Are they asymptotically 1 (linear)?

Sublinear when u=1. What function?

Guess the leading term is of the form $cn^{2/3}$, plot $e/n^{2/3}$, assess convergence to a constant.



Is e asymptotically $O(n^{2/3})$ or $O(n^{2/3} \log n)$?

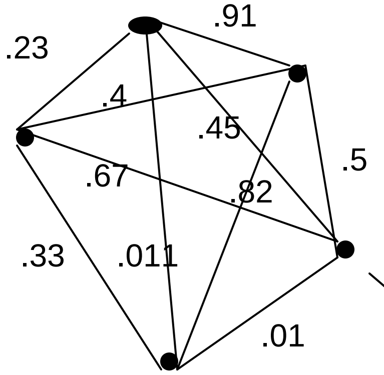
Is this function bounded above by a constant?

All Pairs Shortest Paths (APSP)

Input: Complete graphs G, on n vertices, with weights on edges iid uniform from (0,1).

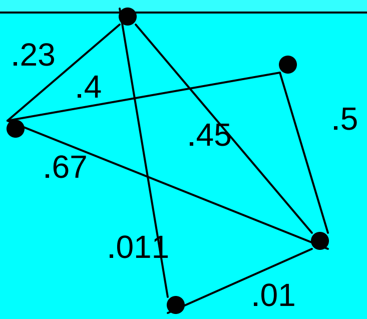
Time depends on H: How many edges in H?

APSP: all vertex-pair distances



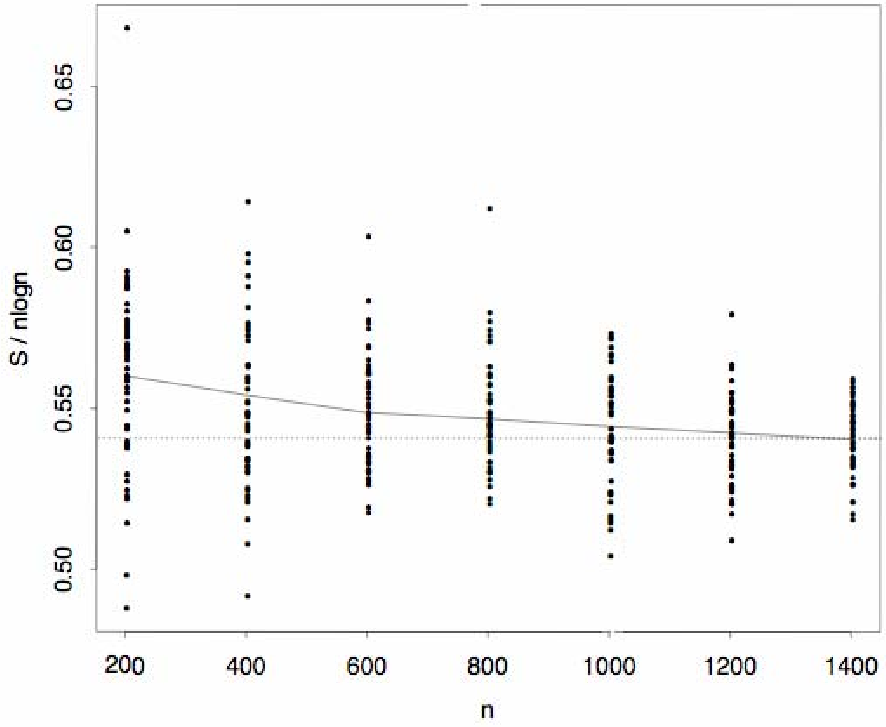
Input *G*, *n=5*

Algorithm computes APSP using subgraph ***H***



S edges in H: O(n) or O(n log n)?

Known: n-2 < s and E[s] < 13.5 n log $_e$ n

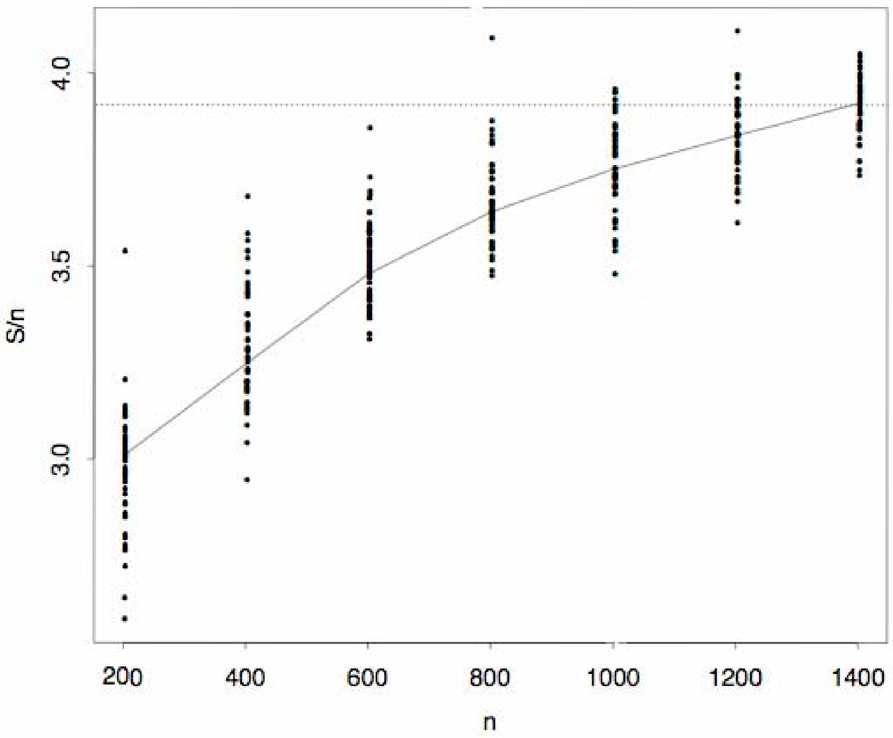


Plot: *S/n* log *n.*

Does this converge to 0 or to *c*>0?

What is the asyptotic lower bound on c?

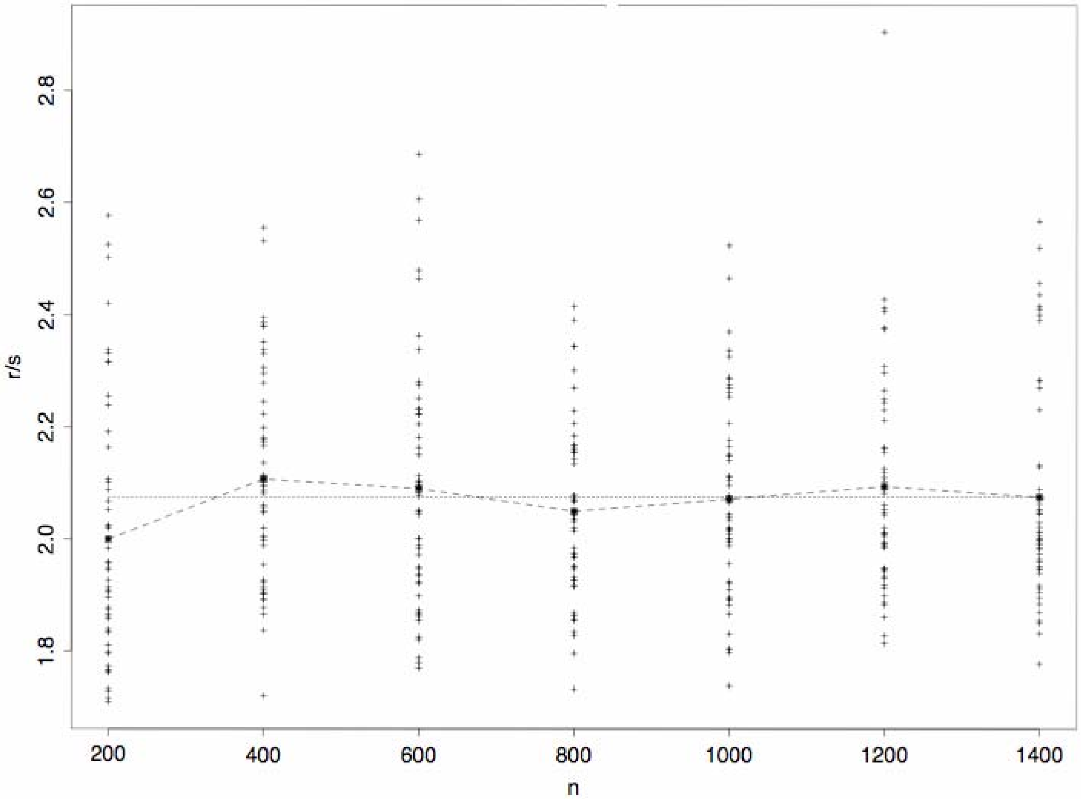
S: O(n) or O(n log n)?



Plot: S/n. Does this converge to c > 0? Or does it grow unbounded by a constant?

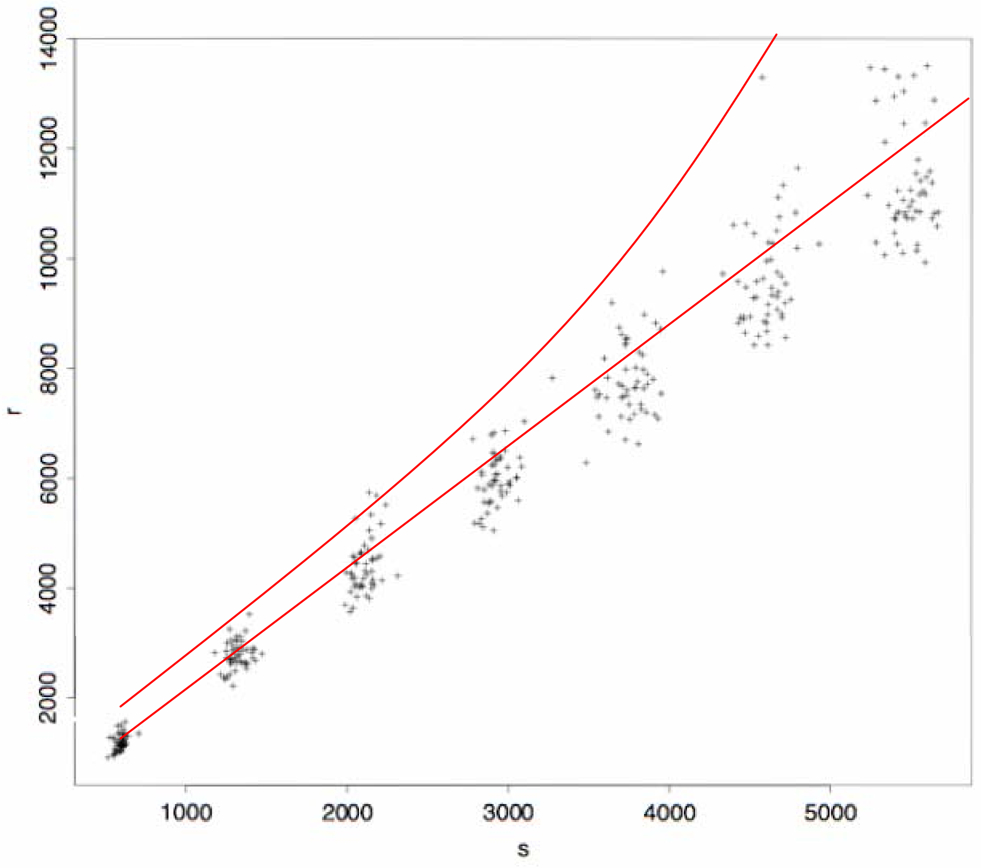
What is the rank *R* of the largest edge *in H* among the *n(n-1)/2* edges in *G*?

Known: S <= R and n log $_e$ n < E[R]



Plot of n vs R/S. Does this converge to a constant c? What is an upper bound on c?

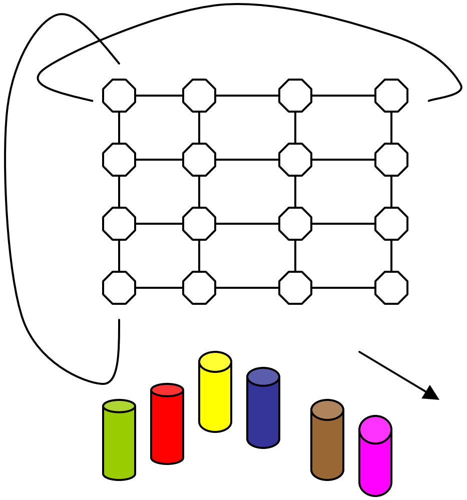
Size vs Rank



Plot of S vs R. How can I bound asymptotically the mean and the expected max value of of R?

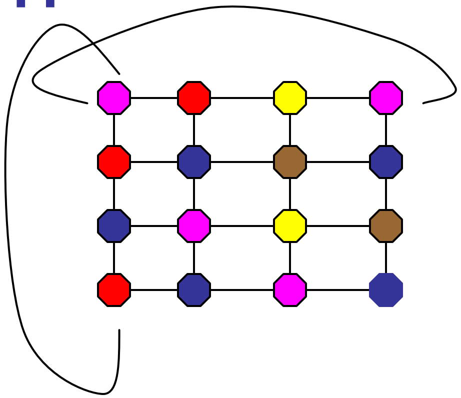
Jerrum’s Graph Coloring Sampling Algorithm

Input: Grid graph G of *n* vertices, degree *d* in (4,6,8), and a color count *k*.



d=4, k=6

Jerrum's Algorithm: random walk in space of colorings



Output C: A *valid* coloring of G, drawn uniformly from the space of valid k-colorings.

Time: How quickly does the distribution of the random walk converge to (within ε of) uniform?

Jerrum's Algorithm

Theorem: For any graph G, *n* nodes, maximum degree *d*, color set *k*:

•*If k >= 2d the algorithm converges to Uniform in polynomial time.*

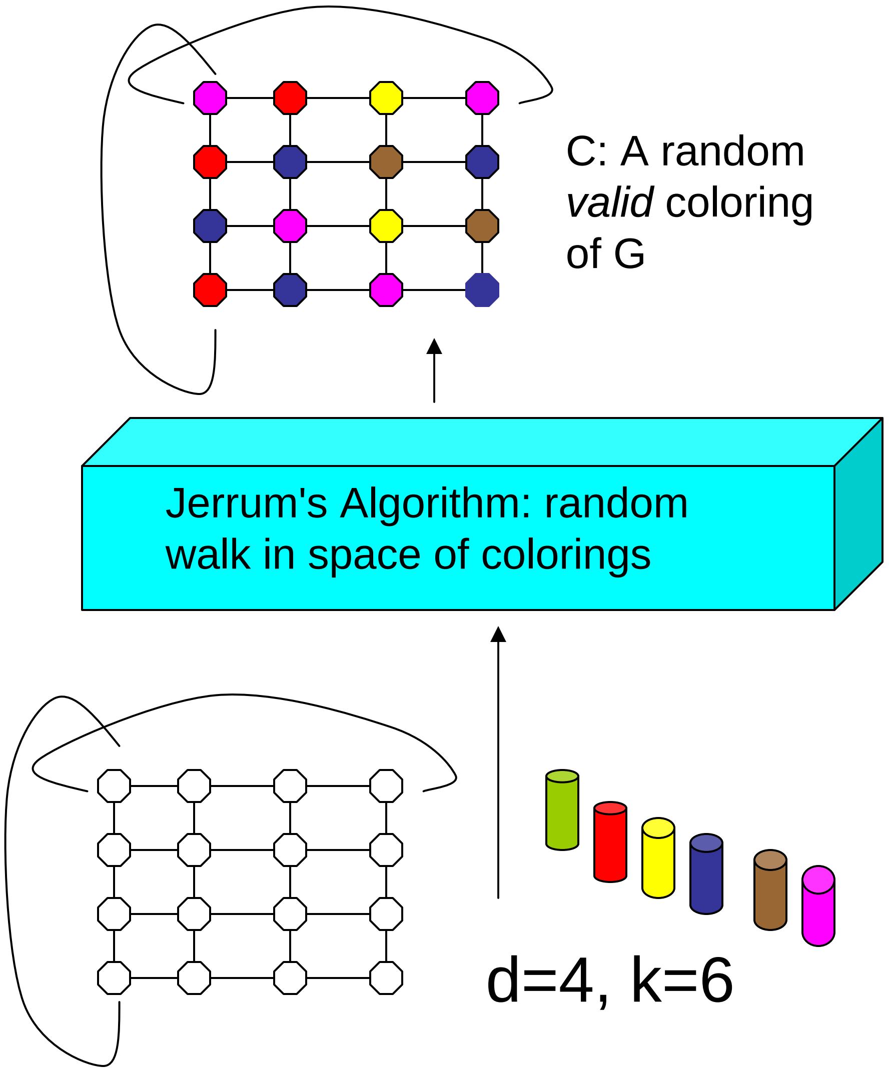
•*If k = d+1 the algorithm takes exponential time to converge.*

•*If k <= d the algorithm does not converge.*

What about *k* in the range (*d+2, 2d-1*)?

Conjecture: exponential throughout.

Time to couple is an upper bound on convergence rate. Proofs are especially difficult for *grid graphs*.....

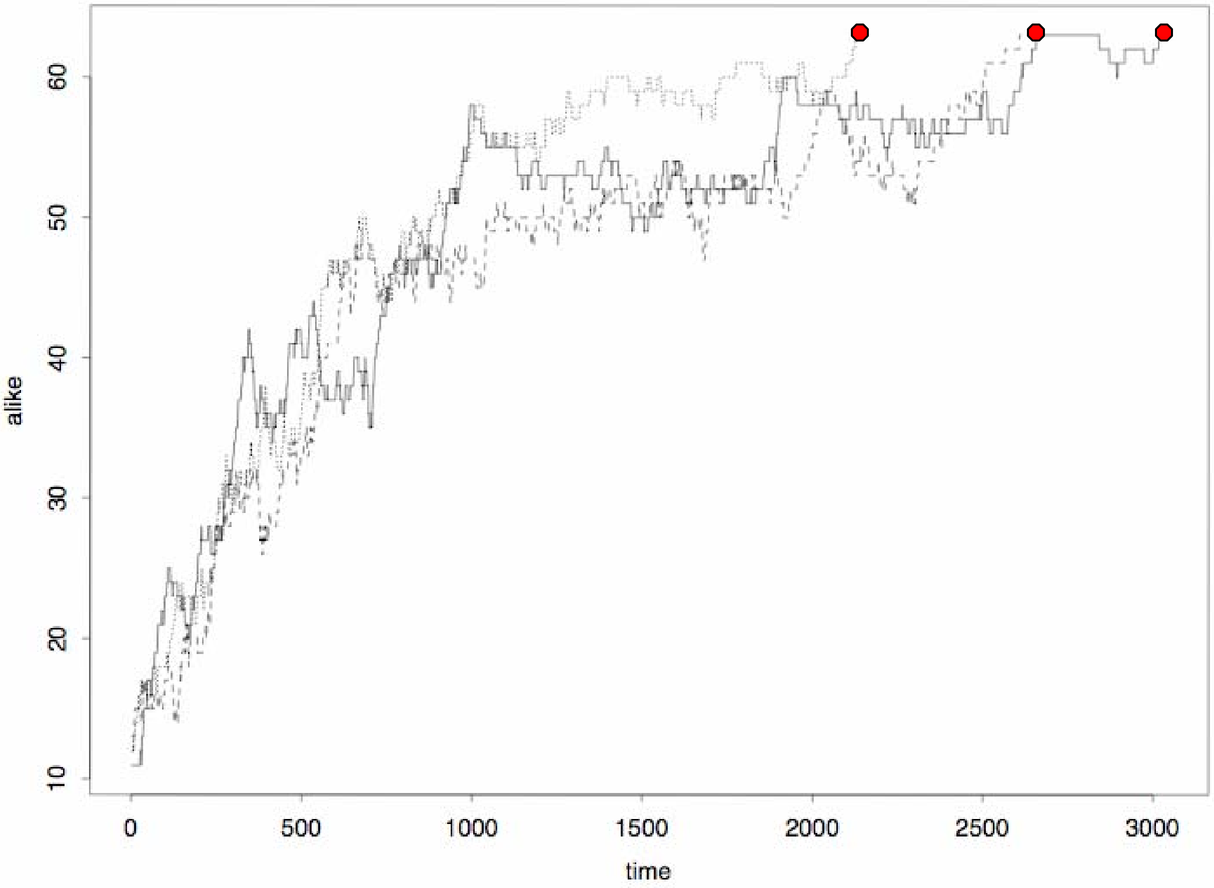


C: A random *valid* coloring of G

Jerrum's Algorithm: random walk in space of colorings

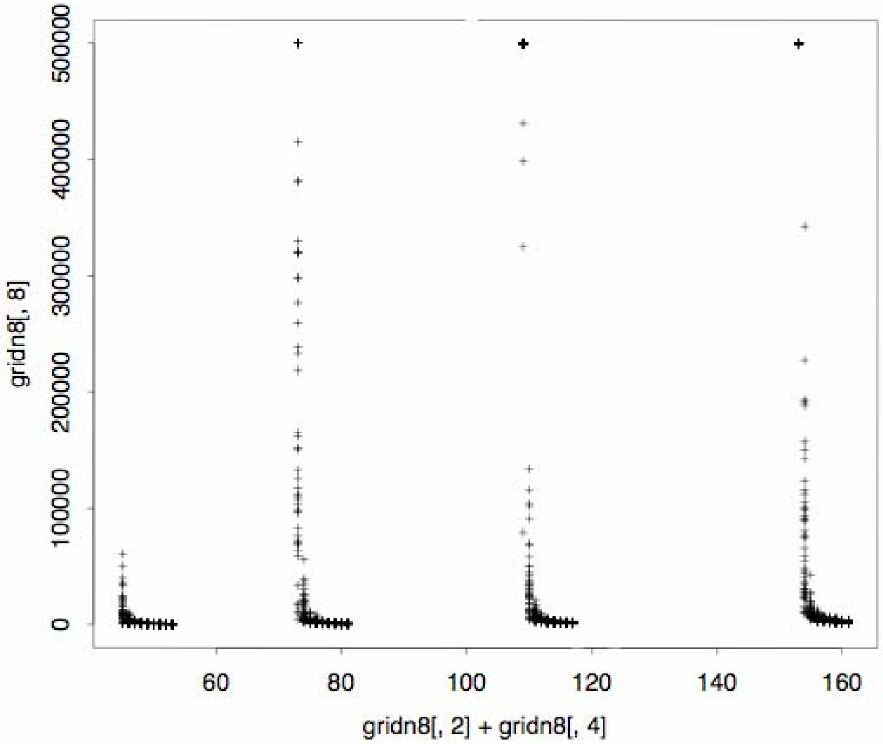
d=4, k=6

Jerrum's Algorithm: Coupling Time



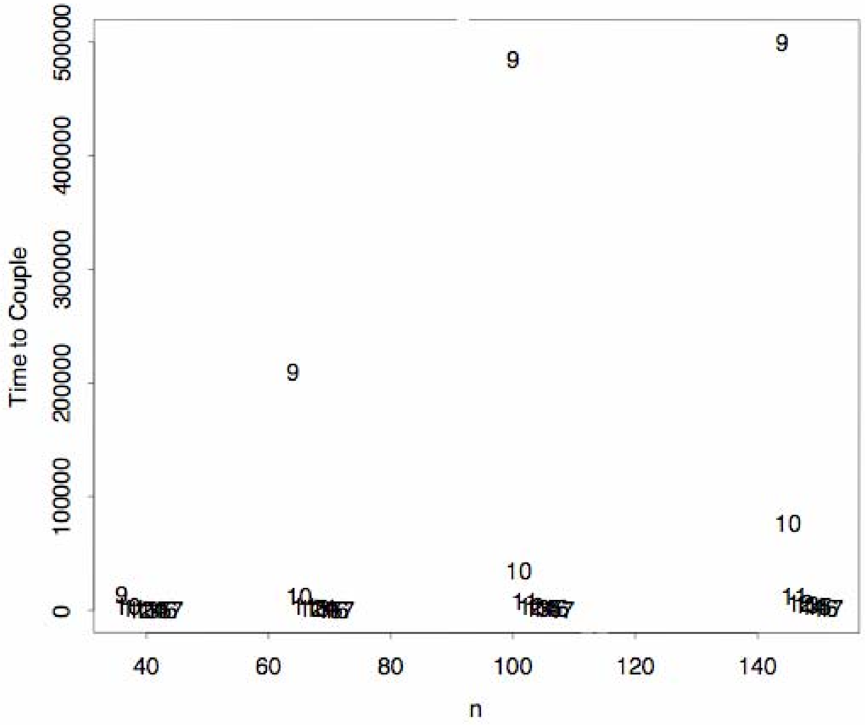
Time to Couple, T, is an upper bound on time to converge. Three trials, n=64, d=8, k=12.

Coupling Time



Grid graph d=8, k=(9..17), n=(36, 64, ... 144), 50 trials; note cutoff at 500000.

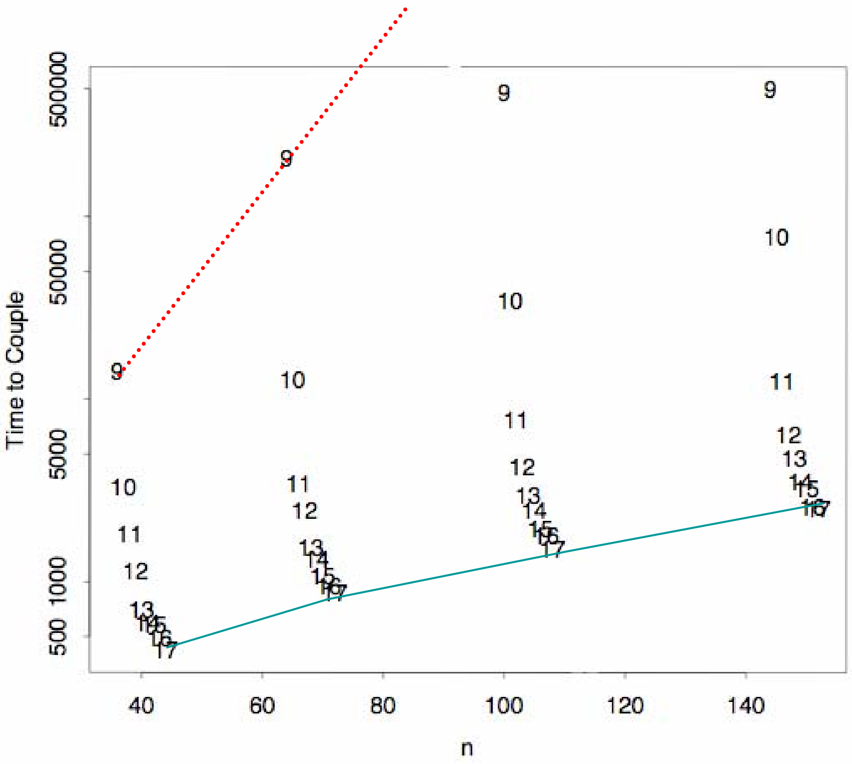
Coupling Time



Grid graph d=8, k=(9..17), n=(36, 64, ... 144). Means of 50 trials; note cutoff.

For which *k* does *T* show exponential growth in *n*?

Coupling Times for Grid Graphs



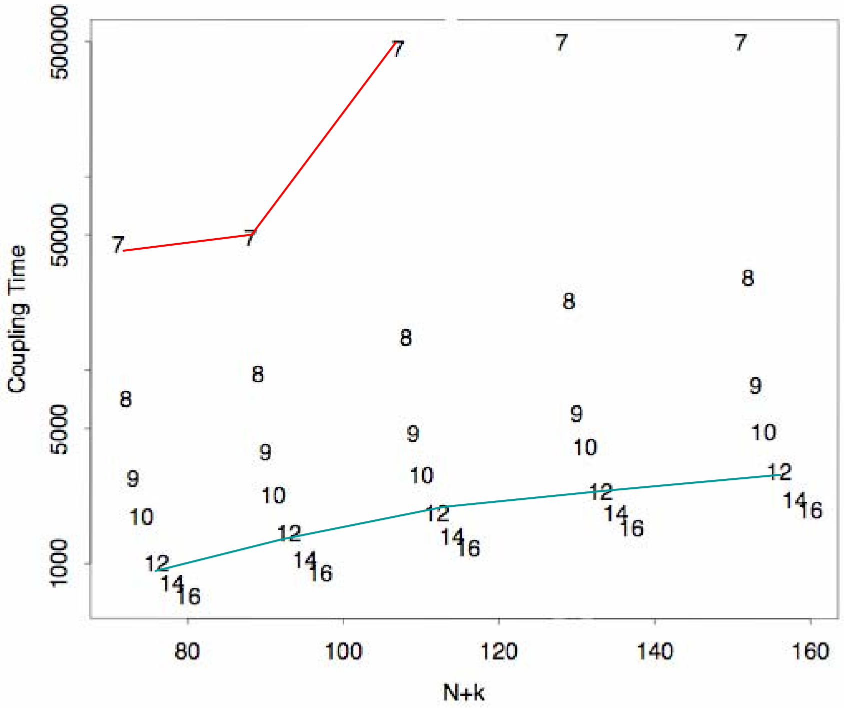
Log coupling time means of 50 trials.

d=8, k=9: known exponential.

d=8, k=17: known polynomial.

How do I classify the others? Where is the critical point?

Coupling Times for Grid Graphs



Log coupling times, means of 50 trials.

d=6, k=7: known exponential.

d=6, k >=12: known polynomial.

How do I classify the others? Where is the critical point?

Questions

- *Bin Packing: Convergence of empty space (a difference) is easier to evaluate than convergence of bin counts (a ratio). Why?*

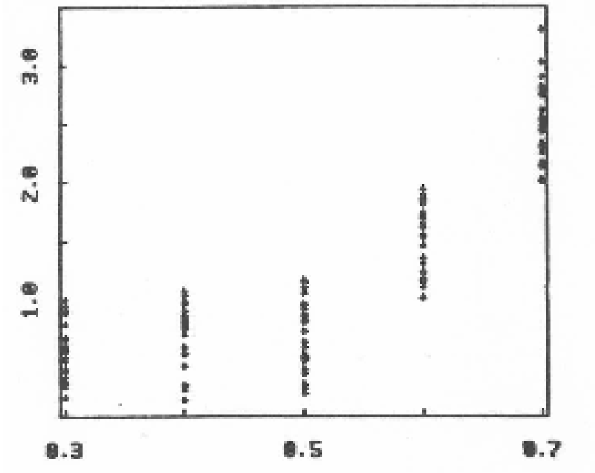
- *Is R (rank of largest edge) easier to analyze than S (number of edges)? How to find an asymptotic upper bound on the expected maximum?*

- *Jerrum's algorithm: How to distinguish polynomial from exponential functions?*

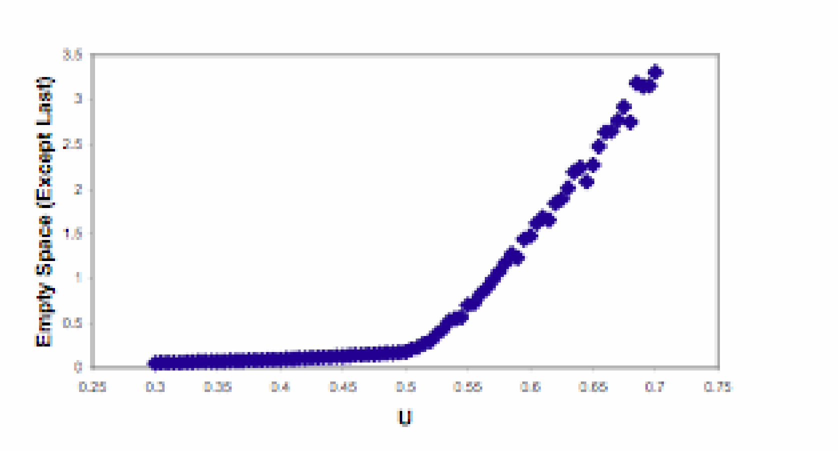
- *Sampling: Is an experiment with 1000 N values evenly spaced between 1 and N_{max} easier to evaluate than one with 10 points each at N, N/2, N/4, N/8 ...? Why?*

Where to place sample points?

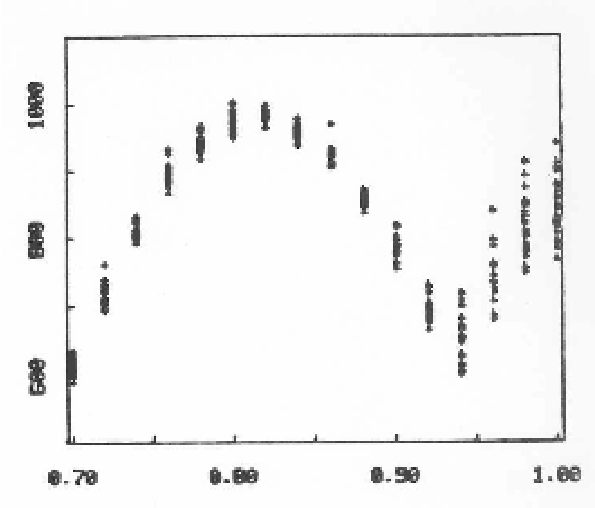
Empty space as f(u)



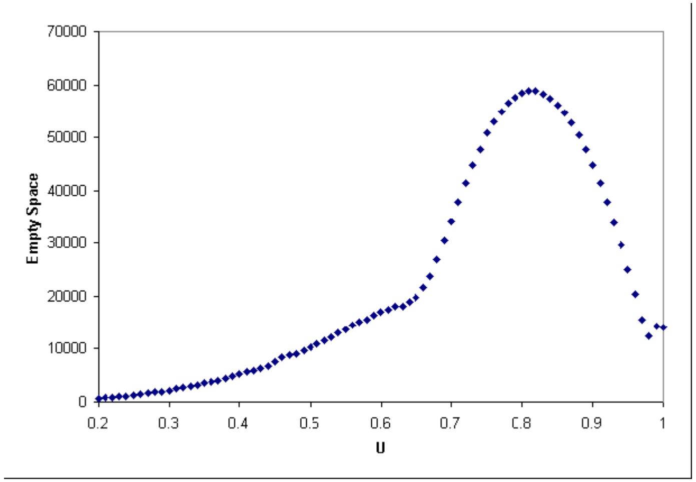
ESL as f(u)



n=128k



n=8m



Three Case Studies, Many Questions

- FF Rule for Bin Packing
- All Pairs Shortest Paths with the Essential Subgraph
- Sampling Graph Colorings with Jerrum's algorithm

•How do I analyze the data to find asymptotic bounds?

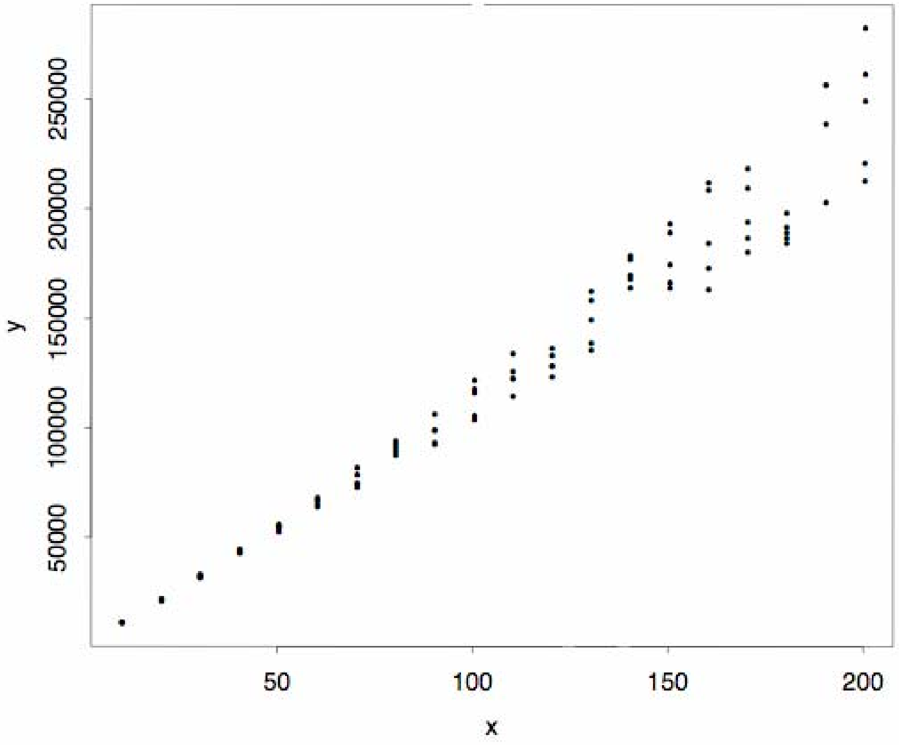
•How do I assess the quality of my analysis? How confident am I in the results?

•Where do I place sample points? How many random trials?

•How do I design my second experiment?

•Which performance measures are easier to analyze? How can I tell in advance?

Asymptotic Curve Bounding

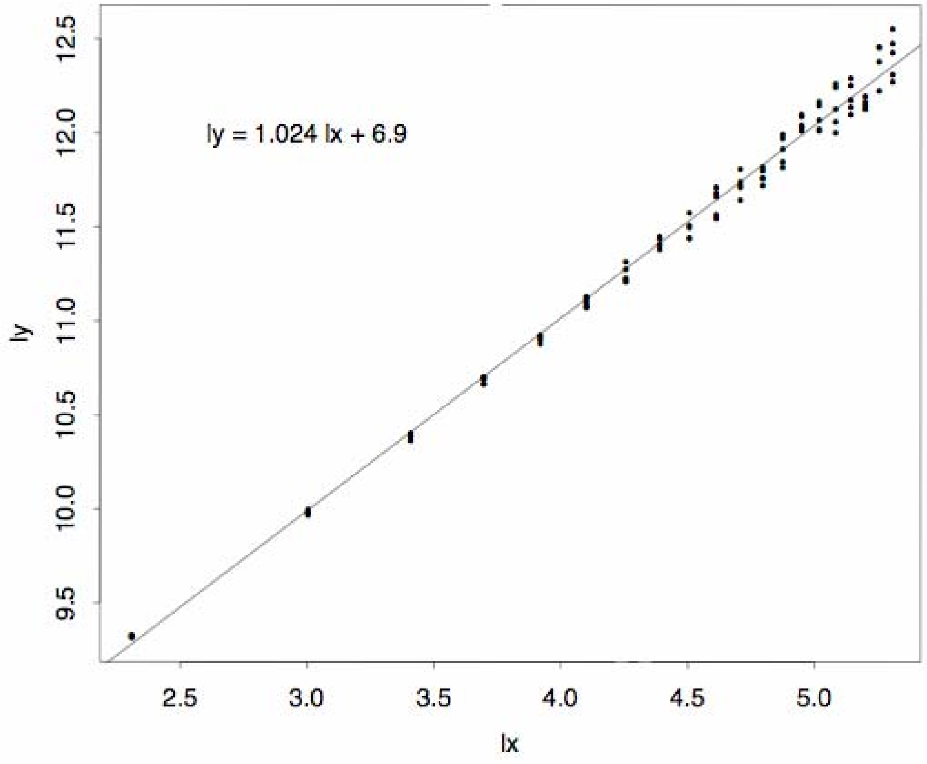


Generated data: Is y growing linearly, quadratically, or somewhere in between? Find an upper or lower bound.

Some Asymptotic Curve Bounding Techniques

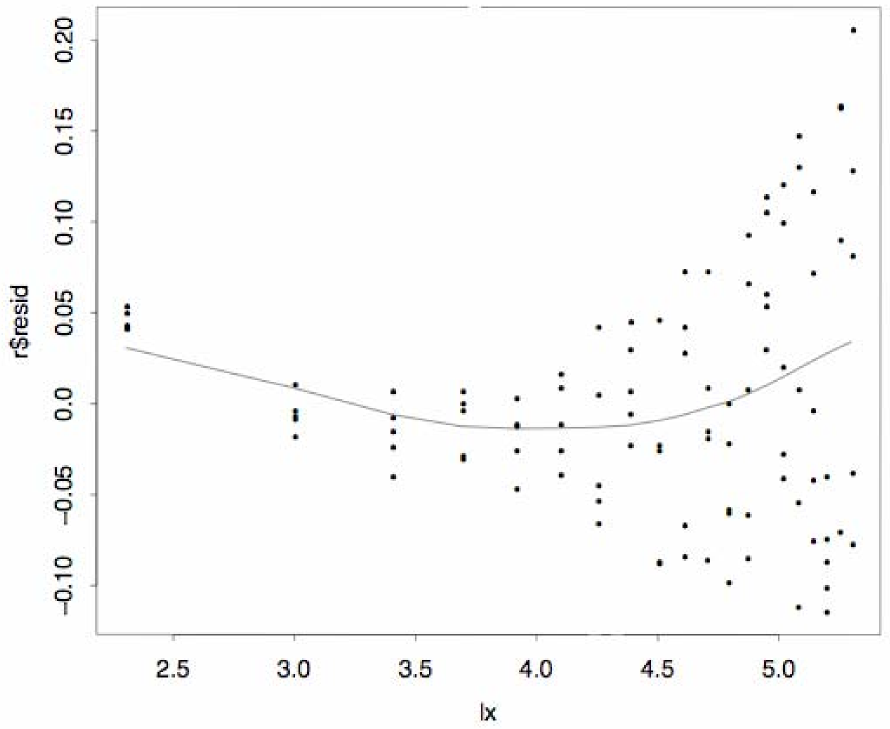
- *Power Law*
- *Guess - Ratio*
- *Guess - Difference*
- *Box - Cox transformation*
- *Newton's method of differences*
- *Generalized regression*
- *Tukey's ladder of transformations*

Power Law



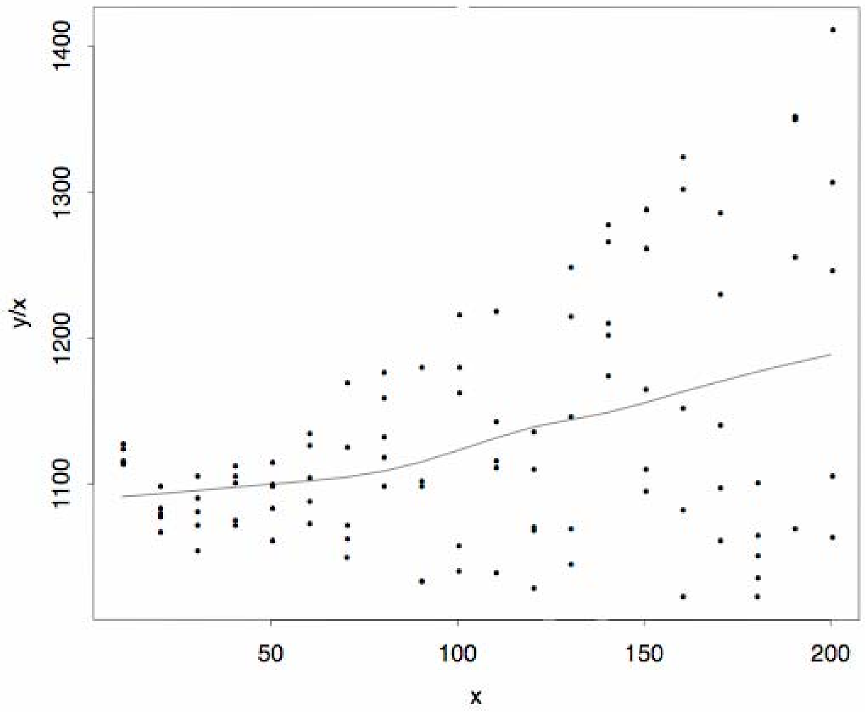
1. *Plot log-log data.*
2. *Fit a line.*
3. *Check slope.*
4. *Check residuals*.

Residuals from Power Law Fit



Conclusion: y grows faster than $x^{1.02}$

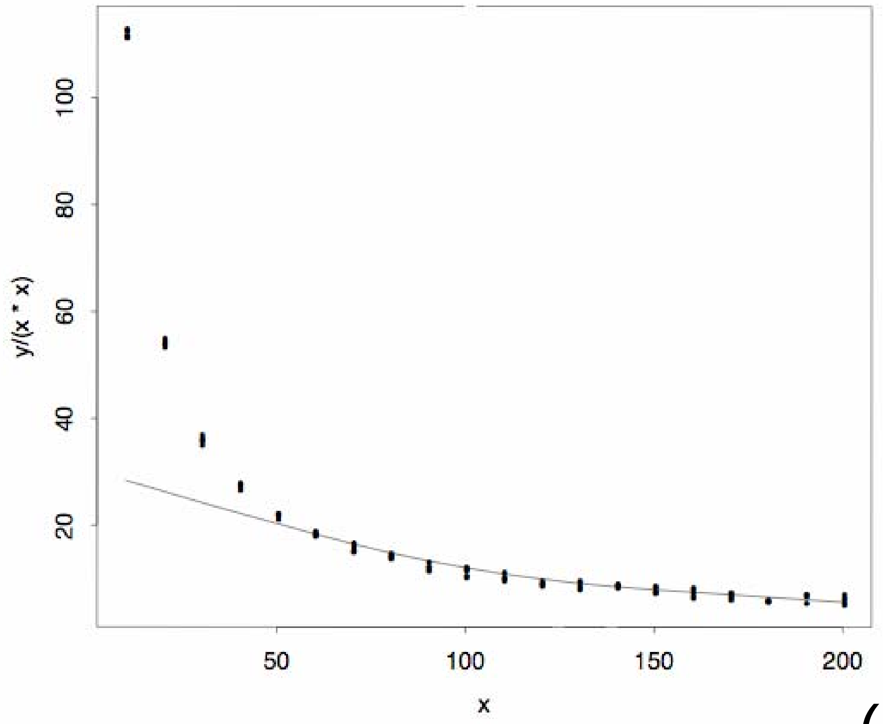
Guess - Ratio



(Faster than x)

1. *Guess a function g(x).*
2. *Plot y/g(x).*
3. *If increasing: y grows faster than g(x).*
4. *If decreasing to 0: y grows slower than x*
5. *If converging to constant > 0: y grows as x.*

Guess - Ratio



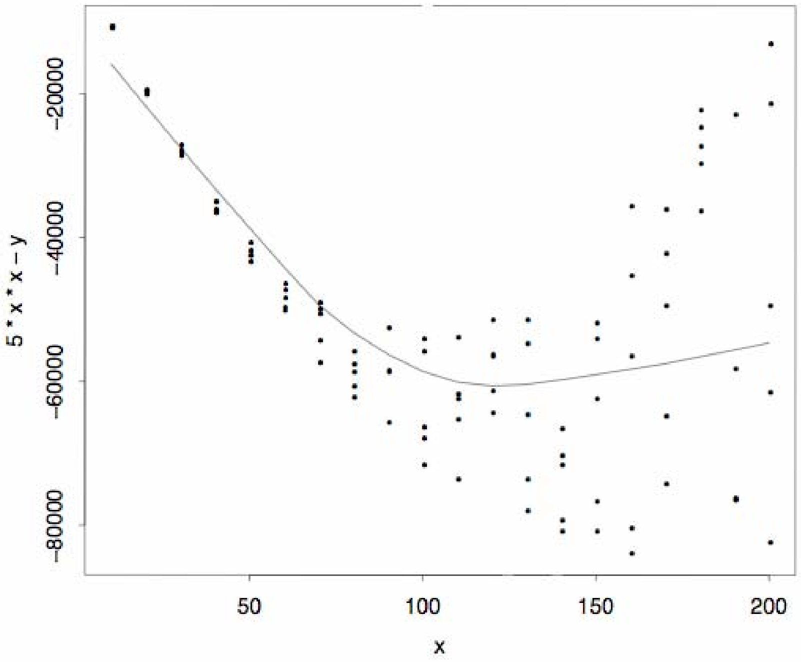
Conclusion (from iterated guesses): y grows faster than x $^{1.1}$.

(Slower than x ^{2} ?)

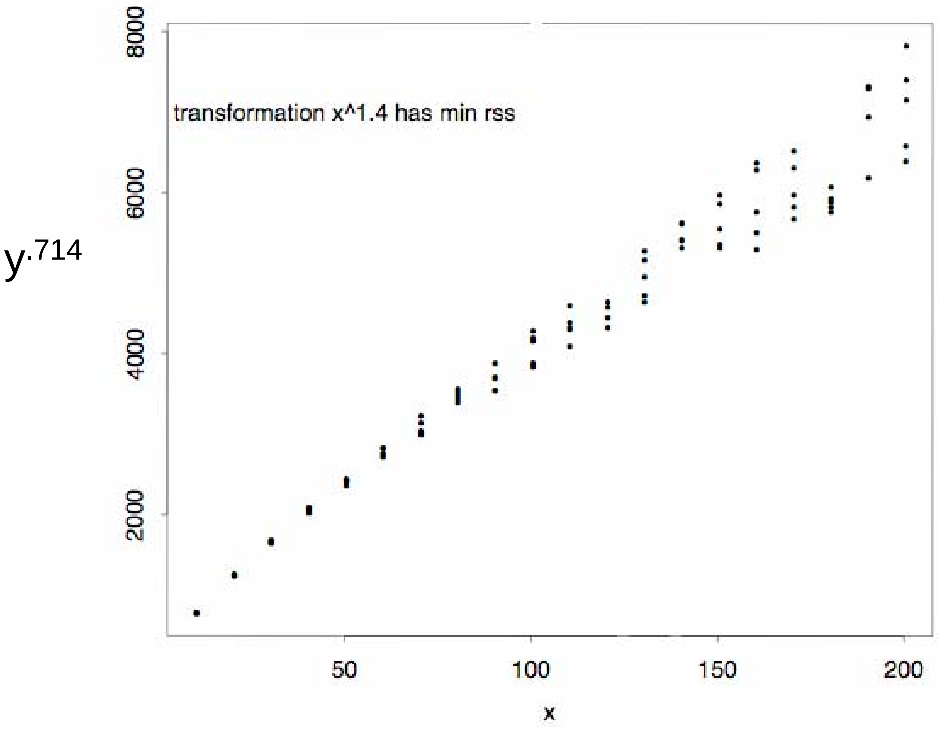
Guess - Difference

1. *Guess the first term g(n) = an b.*
2. *Plot g(n) - Y: If down-up, g(n) is an upper bound.*
3. *Iterate guess to find a tigher upper bound g(n).*

Conclusion: y grows more slowly than x ^{2} .

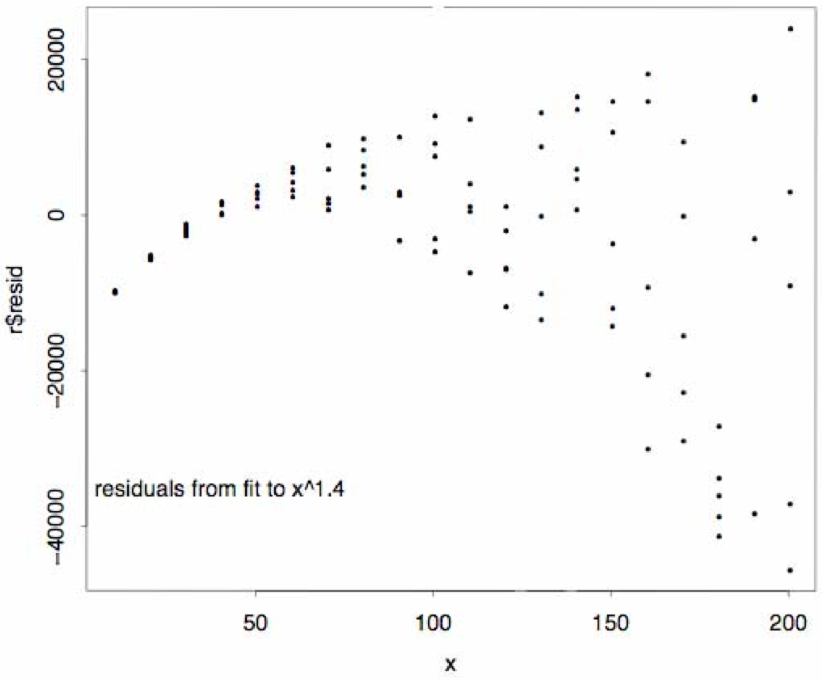


Box-Cox Rule



1. *Transform y using y^t (with scaling function).*
2. *Compare transformed data to a straight line.*
3. *Use scaled RSS to assess fit to line.*
4. *Repeat, find t with min scaled RSS.*
5. *Invert t to find y as f(x)*

Residuals from Box Cox Fit



Conclusion: y grows more slowly than x $^{1.4}$

Newton's Method of Differences

1. *Evaluate polynomial f(x) at evenly spaced x_1, x_2, x_3, ... x_n*
2. *Find differences in adjacent evaluations.*
3. *Repeat until differences are constant.*
4. *Number of repetitions = degree of polynomial.*

43 123 243 403 603

80 120 160 200

40 40 40

quadratic!

Problems:

- *Only works on integer degree polynomials.*
- *Requires evenly spaced x values*
- *Can't cope with random data. No answer for this problem.*

Generalized Regression

1. Guess a multi-term function g(n).
2. Iterate: add a term, delete a term ...
3. Use residuals, RSS to evaluate fit.
4. Find best fit, look at the leading term.

Problems:

- Best fit to the curve does not imply best choice of leading term.
- Different iteration methods (insert/delete paths) give different ``best'' fits. No sense of convergence to an optimal fit -- need an alternative to RSS.
- Residuals analysis can give contradictory results: growing faster than x^a and also growing slower than $x^{a.}$
- *It doesn't work.*

Digression

- *Can computer science help build a better generalized regression method? Current practice seems to be hill climbing with bad neighborhood rule and sketchy objective function.*

Tukey's Transformation Ladder

1 Transform y according to a scale (ladder) of choices:

- y^{2}
- $y^{1/2}$
- $\log y$
- $1/y$
- $1/y^{2}$

2 Look for a straight line. If sqrt(y) is straightest, conclude $y = x^{2}$.

3 Or transform x , or transform both.

Problems:

• Transforming *x* can give answers that contradict transforming *y:* *y* is faster than x^{a}, and *y* is slower than x^{a}.

• Low order terms have different importance in the transformed space.

•*It doesn't work.*

Asymptotic Curve Bounding

The answer: $y = 3\,x^{1.8} + 1000\,x + 1000 + noise$

- ✓ *Power law*: *y faster than* $x^{1.02}$
- ✓ *Guess - Ratio:* *y faster than* $x^{1.1}$
- ✓ *Guess - Difference:* *y slower than* x^{2}
- ✗ *Box - Cox*: *y slower than* $x^{1.4}$
- *(no answer) Newton's method of differences:*

Tests on Generated and Real Data

- *PW: Power Law*
- *PW3: Power Law high 3 data points*
- *PWD: Power Law with differencing*
- *GR: Guess - Ratio*
- *GD: Guess - Difference with up/down heuristic*
- *BC: Box Cox*
- *DF: Newton's Differencing with ``almost flat'' heuristic*

Functions $y = ax^b + cx^d$ varying *a, b, c, d.* Find a bound on b.

Functions $y = ax^b + cx^d + r$ with noise variate *r*.

Functions from algorithm research (some ranges known).

How much does increasing *x* help?

How much does random noise hurt?

Can humans do better?

Nonrandom Functions

$3x^{.2} + 1$	bc	.127 2	pwd
$3x^{.2} + 10^{2}$	pwd	.2 ...... .24	gd
$3x^{.2} + 10^{4}$	pwd	.2 ...... .24	gd
$3x^{.8} + 10^{4}$	pwd	.8 1	*gd,df
$3x^{.8} + x^{.2}$	pwd	.793 ... 1	*gd,df
$3x^{.8} - x^{.2}$		x .... .807	pwd
$3x^{.8} + x^{.6}$	pwd, bc	.778 ... 1	*gd, df
$3x^{.8} - x^{.6}$		x .... .829	pwd
$3x^{.8} +10^{4} x^{.6}$	gr,pw,pw3,pwd,bc	.6 1	*gd, df
$3x^{.8} - 10^{4} x^{.6} + 10^{6}$		x 1	*gd
$3x^{1.2} + 10^{4}$	pwd	1.2 .... 1.22	gd
$3x^{1.2} + x^{.2}$	pwd	1.19 ....1.2	bc
$3x^{1.2} + 10^{4} x^{.2}$	pwd	0.263 ... x	
$3x^{1.2} + x$	pwd	1.175 ... 1.21	gd
$3x^{1.2} - x$		x 1.233	pwd
$3x^{1.2} + 10^{4} x$	gr,pw,pw3, pwd,bc	1 2	*gd

Tightest bounds found.

x = 8,16,32, 64,128

Nonrandom Functions

$3x^{.2} + 1$

$3x^{.2} + 10^{2}$

$3x^{.2} + 10^{4}$ bc NA

$3x^{.8} + 10^{4}$ bc NA

$3x^{.8} + x^{.2}$

$3x^{.8} - x^{.2}$ gr .825 lb

$3x^{.8} + x^{.6}$

$3x^{.8} - x^{.6}$ gr .838 lb, bc .819 lb

$3x^{.8} +10^{4} x^{.6}$

$3x^{.8} - 10^{4} x^{.6} + 10^{6}$ pw, pw3, df negative/zero ub; pwd, bc NA

$3x^{1.2} + 10^{4}$ bc NA

$3x^{1.2} + x^{.2}$

$3x^{1.2} + 10^{4} x^{.2}$ gd NA, df 1 ub

$3x^{1.2} + x$

$3x^{1.2} - x$ gr 1.238 lb, bc 1.228 lb

$3x^{1.2} + 10^{4} x$ df 1ub

Wrong answers (bad bounds shown) and no answers (NA).

BC fails on nearly constant data (transformation $y^{1/b}$ is undefined if b=0).

GR fails on negative second order terms

DF ``almost flat'' rule can be fooled

All can fail on decreasing data, large second terms

Data From Algorithms Research

What is known:	wrong/NA	lower ... upper bounds
$y = (x+1)(2H_{x+2} - 2)$	gr, pwd	x ... 1.18 pw3
$y = (x^2 - x) / 4$	pwd	gr 2 ... 3.001 pw3
$E[y] = x/2 + O(1/x^2)$		gr,pw .99 ... x
$E[y] = \text{Theta}(x^{1/2})$	gr	x ... .5716 pw3
$E[y] = O(x^{2/3} (\log x)^{1/2})$	gr	x ... .695 pw3
$= \text{Omega}(x^{2/3})$		
$E[y] <= 0.68\, x$	pwd	pw .954 ... 1 gd,df
$x-1 <= y <= 13.5\, x \log_e x$	gr, pw3, pwd	x ... 1.142 pw
$x \log_e x < y < 1.2\, x^2$	pwd	gr 1.3 ... 1.31 pw

Note: Many rules failed to decide if the bound was upper or lower: returned ``close''. A close fit is bad in this context.

Some Conclusions

- *Power Law*
- *Power Law Top 3*
- *Power Law with differencing*
- *Guess - Ratio*
- *Guess - Difference*
- *Box Cox*
- *Newton's Differencing*
- *Generalized regression*
- *Tukey's Ladder*

Every rule sometimes fails.

Generalized regression & Tukey's Ladder are not internally consistent. Contradictory answers are artifact of application.

Doubling the largest problem size is less effective than expected: no rule ``became correct,'' and only a few have slightly tighter bounds.

Randomness in data makes curves in residuals harder to find; more ``close'' answers, fewer ``upper/lower bound'' answers.

Humans do about as well as automated rules, but much more slowly.

More Questions

- *Power Law*
- *Power Law Top 3*
- *Power Law with differencing*
- *Guess - Ratio*
- *Guess - Difference*
- *Box Cox*
- *Newton's Differencing*
- *Generalized regression*
- *Tukey's Ladder*

How to cope with logarithms in terms?

When/why should I trust the answer returned by the rule?

Can generalized regression & Tukey's Ladder be fixed?

I can't always choose whether the rule returns an upper bound or lower bound. Is there a way to control this?

I prefer a clear upper / lower bound to a close fit. How can I tune the rules?

How can I design my second experiment to get better results?

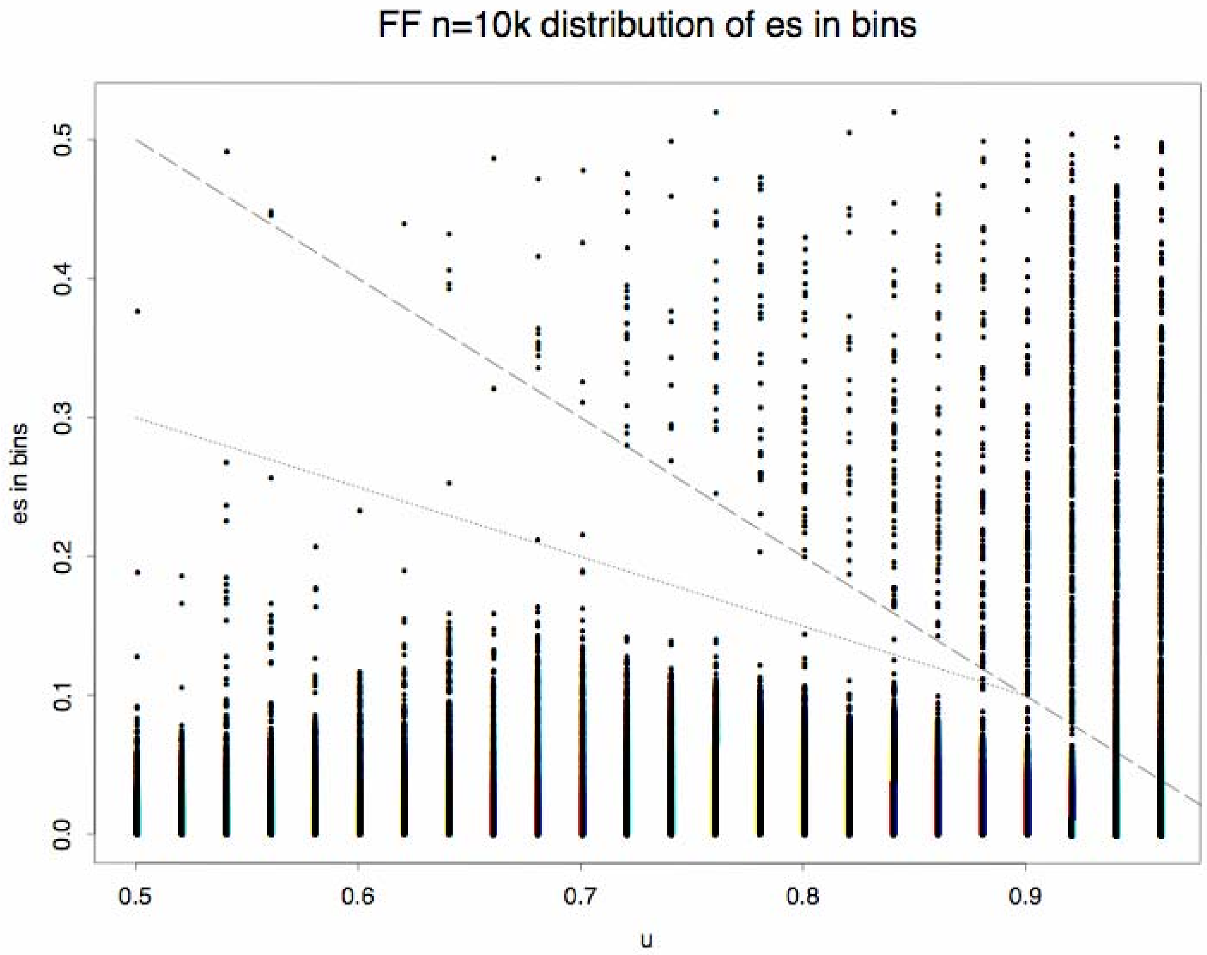
More Questions?

Top = 1-u

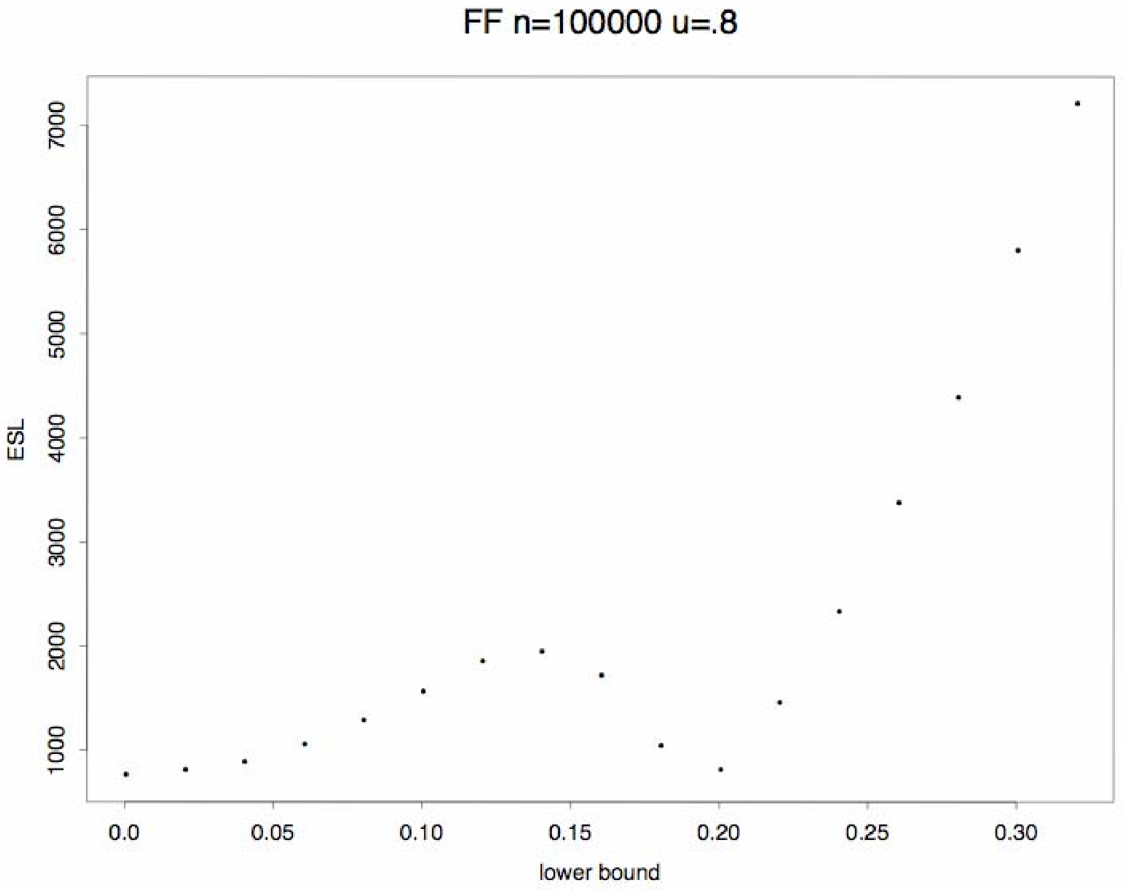
Bottom =
.55 - u/2

Describe the the `gap' where:

prob(x) < ε(u,n)



Unusual Functions



SYMBOL FONT

αβχδεϕγηιφκλμνοπθρστυϖωξψζ

1234567890−=[]∴;∍,./

!≅#∃%⊥&∗()_+{}|:∀<>?

ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ

Theory and Practice