Statistical Methods in Algorithm Evaluation

Master Class on Experimental Study of Algorithms and Benchmarking
CPAIOR 2010, Bologna

Catherine C. McGeoch
Amherst College

Overview

- 0. Introduction (history lesson).
- I. How not to do it: avoiding common mistakes.
- II. Before the experiment: making a plan.
- III. After the experiment: statistics and data analysis.

Overview

- **0. Introduction (history lesson).**
- I. How not to do it: avoiding common mistakes.
- II. Before the experiment: making a plan.
- III. After the experiment: statistics and data analysis.

Theory vs Practice

Theory + Practice + Experiment

Three approaches to algorithmic research - cooperation reaps benefits!

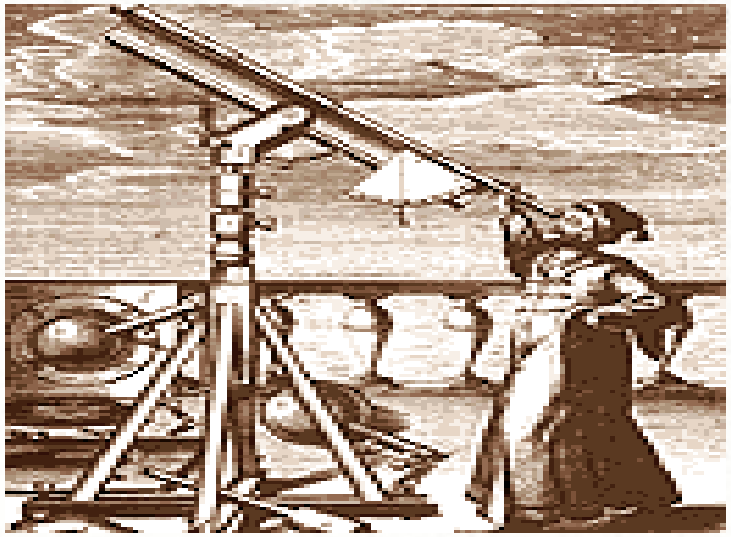
Two Types of Experiments

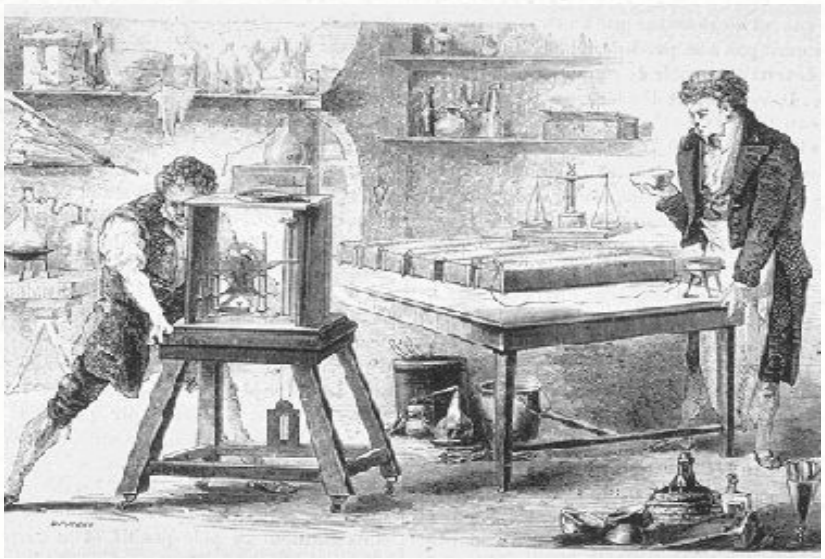
Field experiments:

- Observe phenomena
- Record observations
- Classify outcomes

Laboratory experiments:

- Isolate components
- Manipulate parameters
- Observe effects

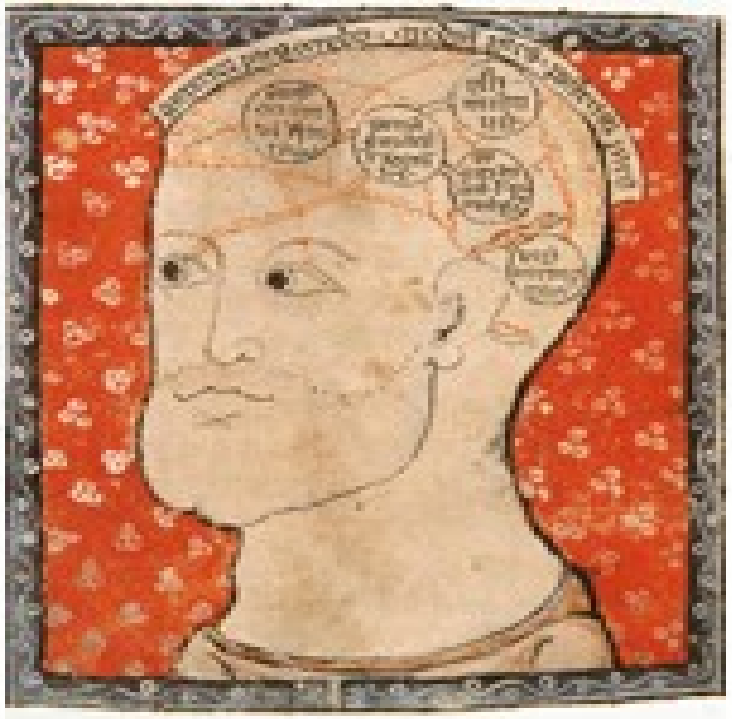




Practice vs Science

Practioner: What is the best way to do it?

Scientist: How does it work? What are the ranges of behavior?

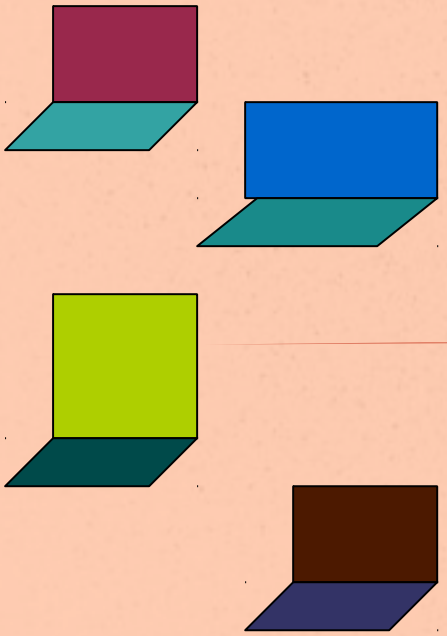


Modern Research on Algorithms

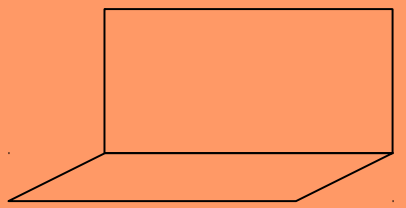
- Practice:
- Design = Implementation for a specific application
- Analysis = runtimes
- Observational (field) experiments
- Experimental algorithimcs
- Design = Engineering for broad application
- Analysis = boundaries, trends
- Controlled (laboratory) experiments

Theory + Experiment + Practice

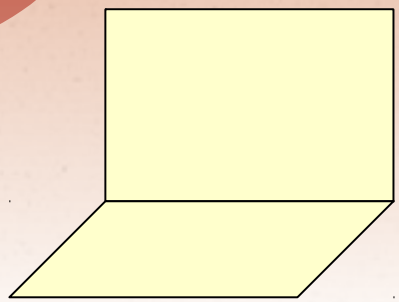
Practice: Implement solutions, solve problems.



Theory: build a model to describe practice.



Laboratory: Implement the model; simulate practice.



Algorithmic Experiments: The Good News

- Near total control of the experiment.
- Fairly simple models (the algorithm).
- Strong theoretical foundations.
- Clear understanding of cause and effect.
- Tons of data points.

Algorithmic Experiments: The Bad News

- High Expectations
- Unusual and extremely detailed questions = no known analyses methods.
- Lots of data = computational issues.

Three Lessons of Algorithmic Experiments

I. **Control = Responsibility:** *Avoiding naive but common mistakes.*

II. **Control = Power:** *Exploiting the unusual features of computational (simulation) experiments.*

III. **Control = Choice:** *Choosing the right tools for unusual jobs.*

Overview

- 0. Introduction (history lesson).
- **I. Control = Responsibility. How not to do it.**
- II. Control = Power. Before the experiment.
- III. Control = Choice. After the experiment.

I. Control = Responsibility

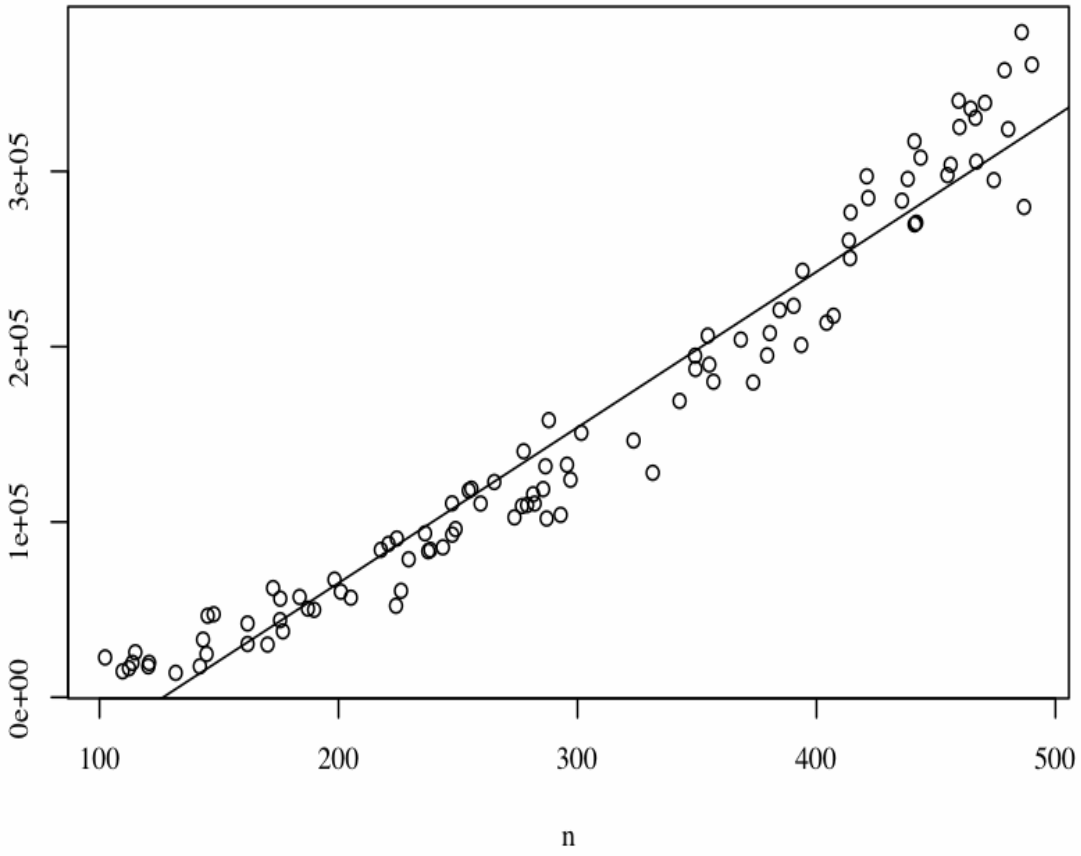
Four examples of methodology errors, from (anonymized) experiments on:

- an algorithm for hidden line elimination.
- an iterated greedy algorithm for graph coloring.
- a one-dimensional bin packing algorithm.
- a matching algorithm.

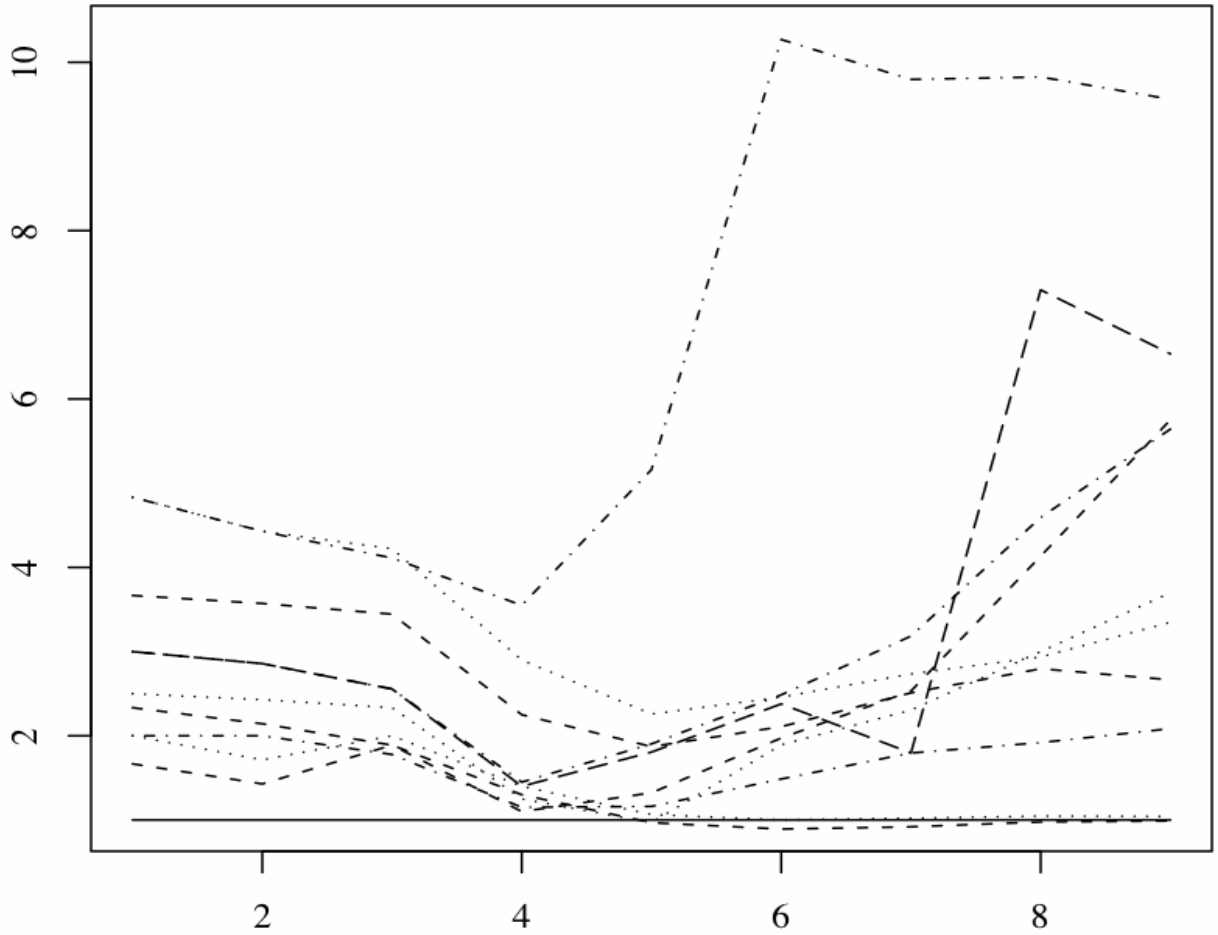
How Not to Do It: 1

``As conjectured, cost is asymptotically linear in n, except for a slight increase at large n.[1]"

1: CPU time increase is probably due to paging at large problem sizes.



CPU Times Are Not Accurate (within a factor of 10)



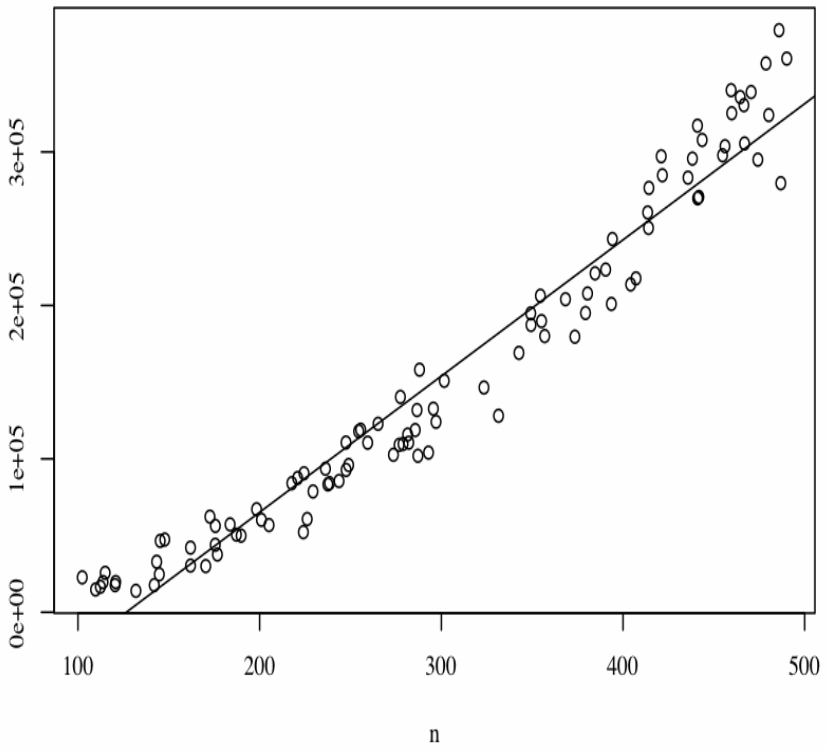
One program, 9 *identical* instances (ordered by size), run on 12 *common* platforms.

(11 other platforms couldn't run all instances.)

src: DIMACS TSP Challenge

- Don't choose the wrong cost measure: CPU time ≠ dominant opeation.
- Don't extrapolate from regression lines.
- Don't naively extrapolate CPU times.

- Don't choose the wrong cost measure: CPU time ≠ dominant opeation.
- Don't extrapolate from regression lines.
- Don't naively extrapolate CPU times.



How Not to Do It 2

My experiments using Culberson's Iterated Greedy Algorithm for Graph Coloring.

Look at:
Variations in Vertex rule

Benchmark application instances from the 2nd DIMACS Challenge (1993).

```
G = Input Graph
C = Initial Coloring

Loop

     Reorder colors (Color rule)
     Reorder vertices (Vertex
rule)

     C = GreedyColor(G) //save
best

     C = Kempe reduction (k
iterations)
     C = Revert (r iterations)
     stop = Stopping  rule (r,
max)

until (stop)

Report C* best coloring found.
```

DIMACS Challenge Benchmark Instances: Graph Coloring

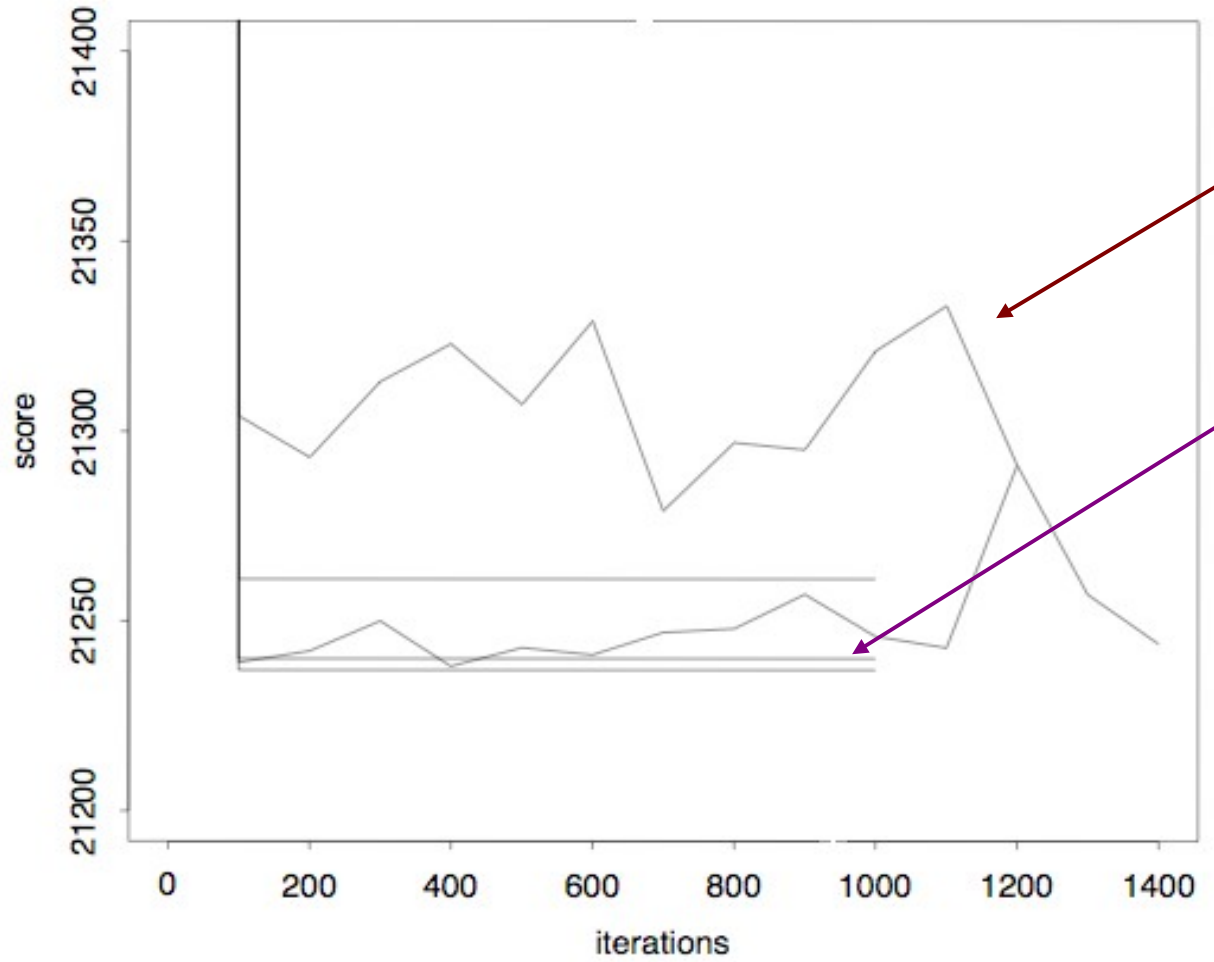
(Reg) = Register interference graphs from a compiler application. Optimal Colorings not known.

A test using this input, five variations on policy 4.

DIMACS COLORING BENCHMARKS				
File	Code	Nodes	Edges	Colors Needed
mulsol.i.1.col	(Reg)	197	3925	
mulsol.i.2.col	(Reg)	188	3885	
mulsol.i.3.col	(Reg)	184	3916	
mulsol.i.4.col	(Reg)	185	3946	
mulsol.i.5.col	(Reg)	186	3973	
zeroin.i.1.col	(Reg)	211	4100	
zeroin.i.2.col	(Reg)	211	3541	
zeroin.i.3.col	(Reg)	206	3540	
fpsol2.i.1.col	(Reg)	496	11654	
fpsol2.i.2.col	(Reg)	451	8691	
fpsol2.i.3.col	(Reg)	425	8688	
inithx.i.1.col	(Reg)	864	18707	
inithx.i.2.col	(Reg)	645	13979	
inithx.i.3.col	(Reg)	621	13969	
le450_15a.col	(Lei)	450	8168	15
le450_15b.col	(Lei)	450	8169	15
le450_15c.col	(Lei)	450	16680	15
le450_15d.col	(Lei)	450	16750	15
le450_25a.col	(Lei)	450	8260	25
le450_25b.col	(Lei)	450	8263	25
le450_25c.col	(Lei)	450	17343	25
le450_25d.col	(Lei)	450	17425	25
le450_5a.col	(Lei)	450	5714	5
le450_5b.col	(Lei)	450	5734	5
le450_5c.col	(Lei)	450	9803	5
le450_5d.col	(Lei)	450	9757	5
flat1000_50_0.col.b	(Cul)	1000	245000	50
flat1000_60_0.col.b	(Cul)	1000	245830	60
flat1000_76_0.col.b	(Cul)	1000	246708	76
flat300_20_0.col.b	(Cul)	300	21375	20
flat300_26_0.col.b	(Cul)	300	21633	26

Continued on next page

Culberson's Iterated Greedy Algorithm, scores for one input.



Report scores after every 100 iterations.

My Tests of Culberson's Algorithm, five variations on policy 4.

Score = number of. colors + ``niceness factor"

Two policies look like random walks.

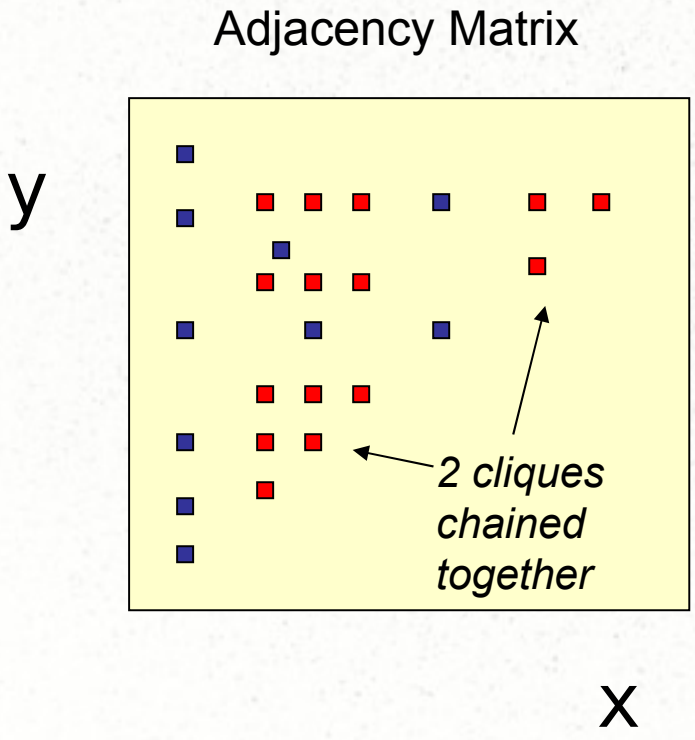
Three policies converge after 1 iteration.

(So 2 iteratiosn is as good as 1400.)

Why do the three converge so quickly?

...

Why do they converge so quickly?

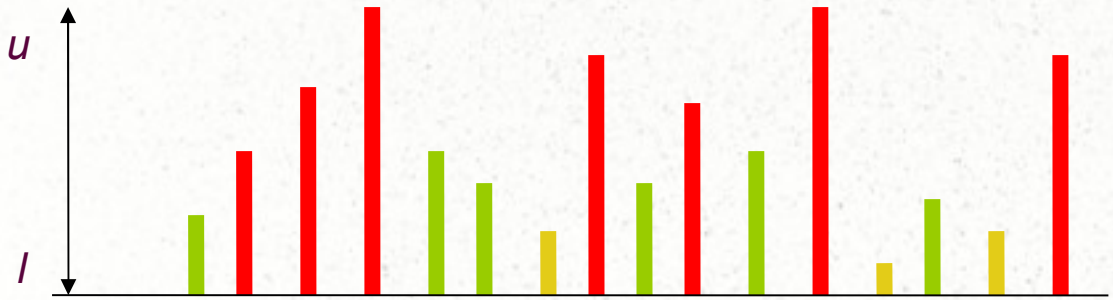


- Because the (Reg) graphs are trivial to solve!
- Big fat cliques connected by chains.
- Optimality of 1 pass of *simple* greedy on input order can be verified by hand!

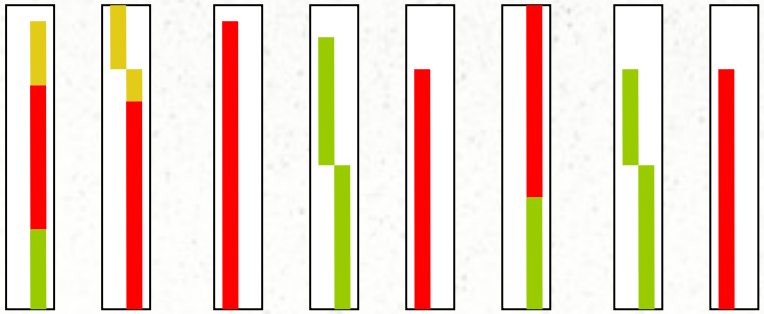
How Not to Do It 2

- Don't pull algorithm parameters (iteration count) out of your hat. My runtimes were 700 times bigger than necessary.
- Don't assume that good performance is due to your wonderful algorithm: it could be due to trivial instances.
- Don't just look at the final cost – look at the process.

How Not to Do It: 3



Consider weights in order of arrival; pack each into the first (leftmost) bin that can contain it.



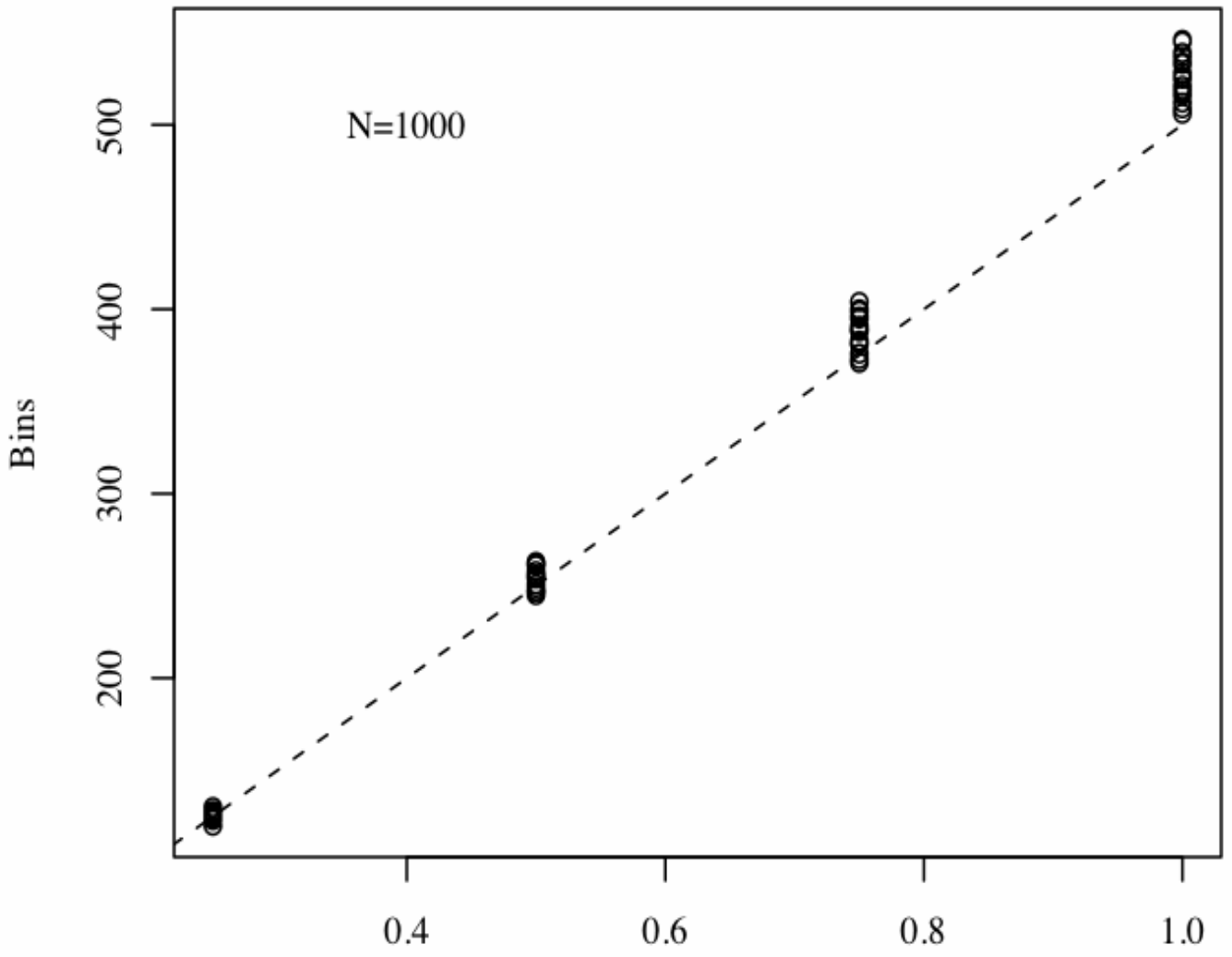
The First Fit Heuristic for Bin Packing.

(Bin packing is NP-Hard.)

L = list of *n* weights uniformly distributed on *(0, u)*.

How well does FF pack them into unit-capacity bins, as a function of *u,* asymptotically in *n*?

First Fit Bin Packing



Experiments circa 1978: Pack *n*=1000 weights, *u*=.25, .5, .75, 1, 20 trials each.

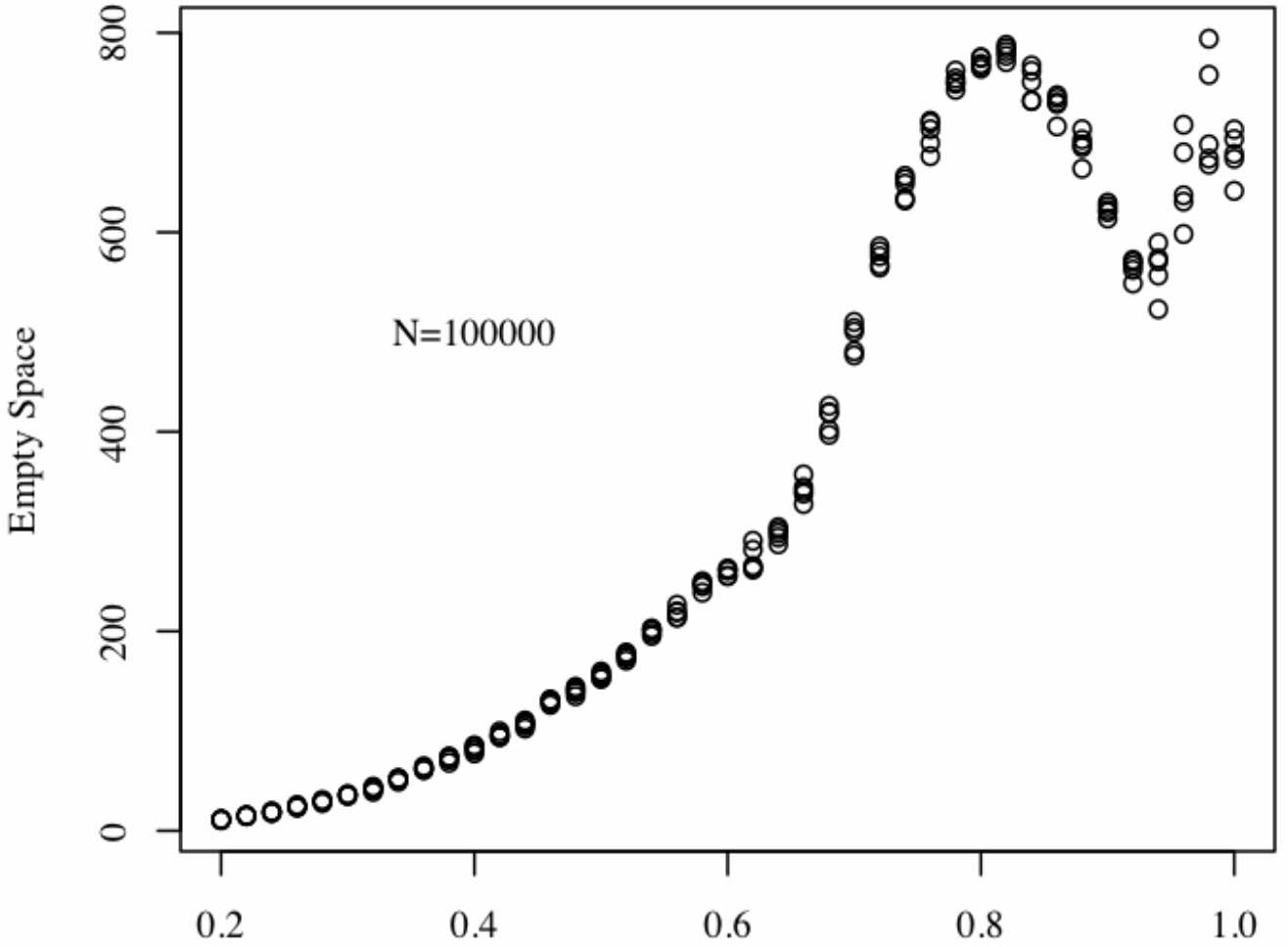
Measure *B* = bins used in the packing.

Sight line = average-case lower bound on B, based on sum of weights.

Known: FF is optimal at *u*=1.

Conjecture: FF is optimal for all *u*.

First Fit Bin Packing



Ten years later:

Pack n=100000 weights, u = .2, .22, ... .98 1.0.

Measure Empty Space: *E = B – Weightsum.*

Observe = Empty space is not linear in u!

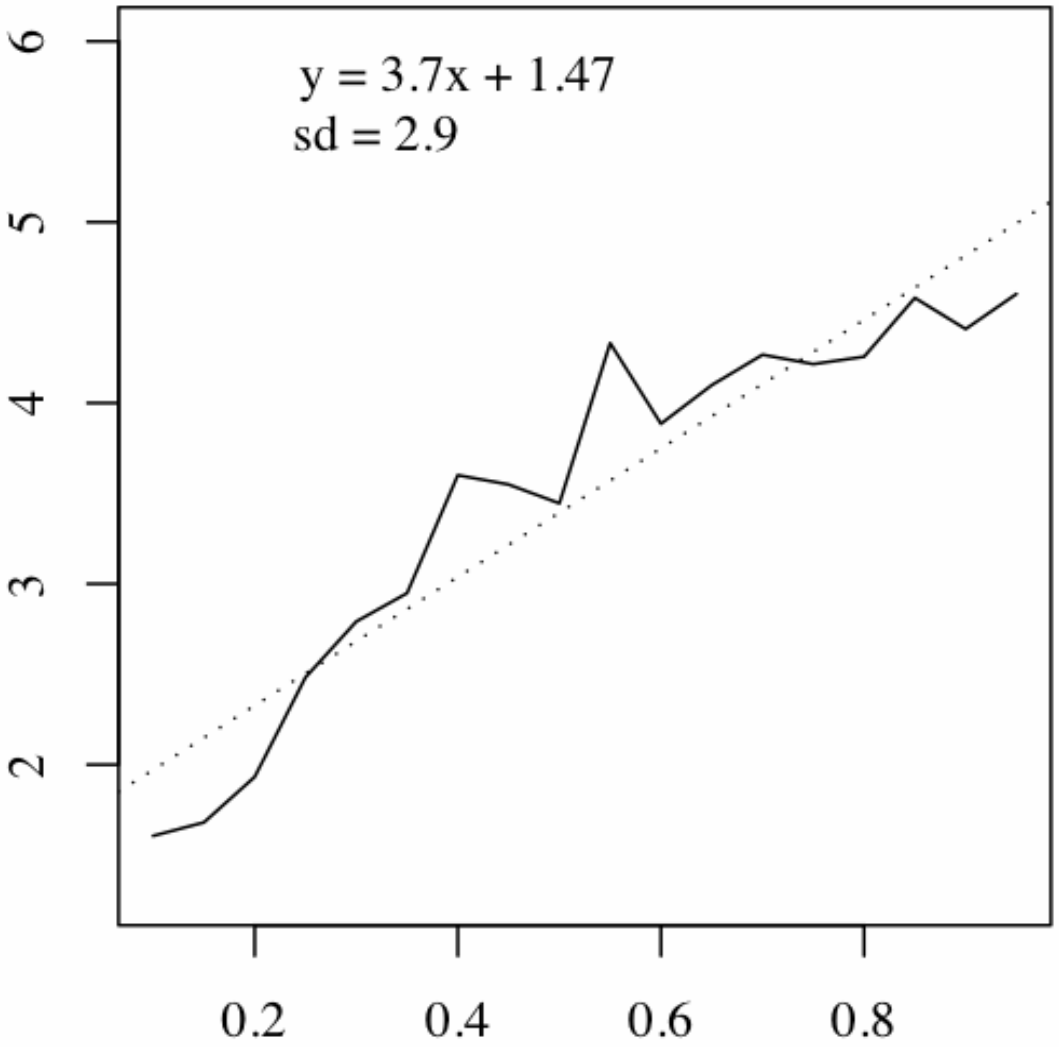
The ``peak'' grows as n increases; the valley disappears.

New Conjecture: FF is optimal *u=1,* but nowhere else.

How Not To Do It 3

- Do not assume you are looking at asymptopia.
- Do not use a cost measure that is too coarse to see the interesting behavior. Bins = WeightSum + EmptySpace is 99% weights, 1% empty space.
- Do not (naively) interpolate a line when the model (the underlying function) is not linear.

How Not To Do It 4

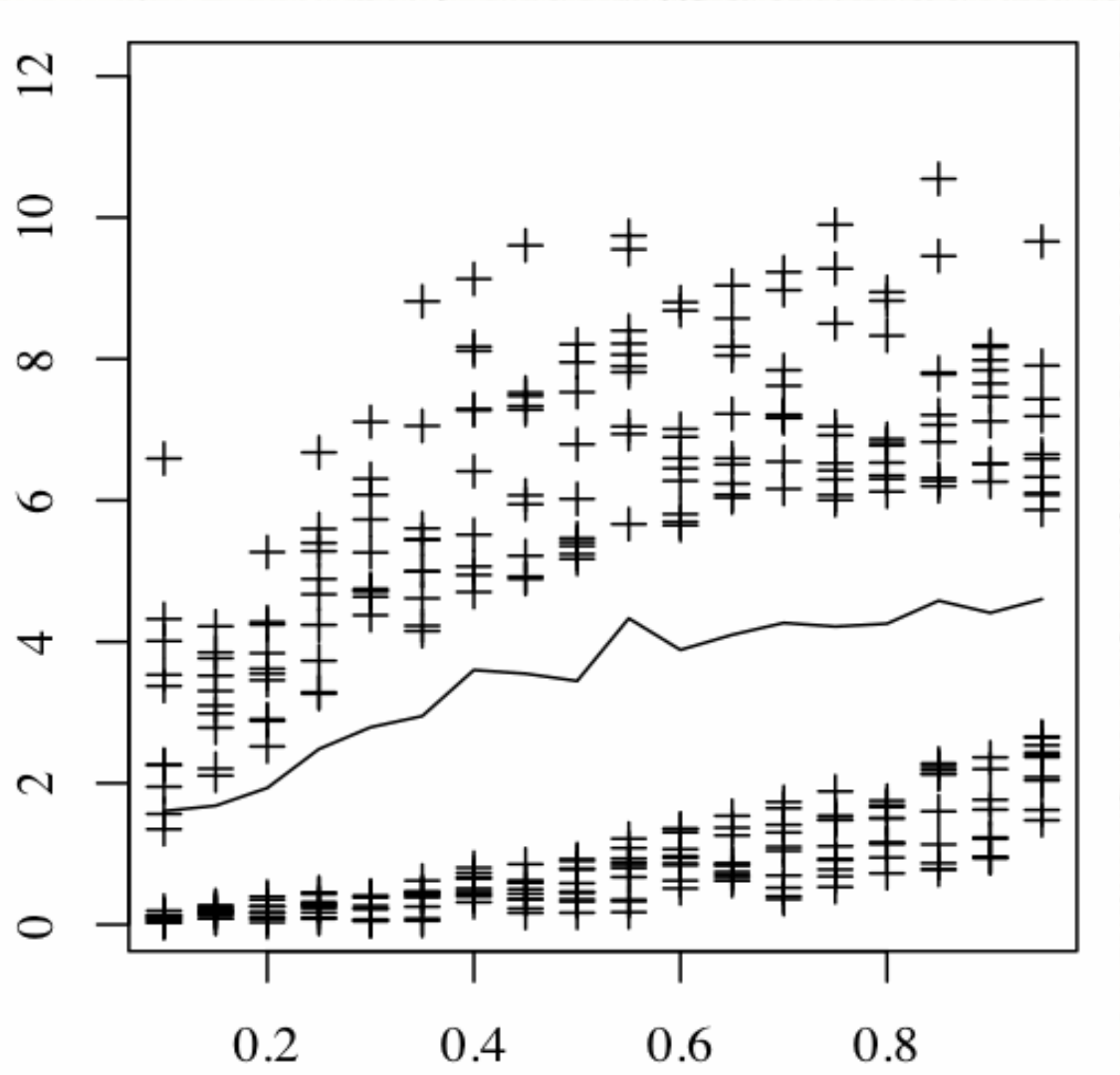


Analysis from Regression Fit and calculation of Standard Deviation:

Mean cost is approx *3.7u*

Standard deviation: most of the data is within +/- 3 of the line.

How Not To Do It 4



Avoid Premature Summarization:

Don't let the program summarize the data before you get a chance to look at it!

You never know what you might discover.

How To Avoid These Types of Errors

- Learn from your mistakes. (Like I did.)
- Learn what statisticians have to say about experimental design and data analysis. (Surveyed in these lectures.)
- Learn to adopt a philosophical attitude. (You can't avoid every difficulty.)

Control = Responsiblity
Any Questions?

Statistical Methods for Algorithm Evaluation – Three Lessons

I. Control = Responsibility: How not to do it.

II. Control = Power: Before the experiment. Choosing factors, what to measure, formal design of experiments, variance reduction techniques.

III. Control = Choice: Statistics and data analysis.

II. Control = Power: Before the Experiment

- ***2. Planning Experiments:*** Informally, choosing the right cost measures, which parameters to look at.
- ***1. Design of Experiments (DOE)***: Formally, how to maximize the power (information gained per unit of effort) in the experiment.
- ***3. Variance Reduction Techniques:*** Improving the quality of data produced by the experiment.

Some Terminology

- The ***performance indicator*** is the ``outcome'' you measure (usually an indicator of time performance or solution quality).
- A ***factor*** is something that affects performance.
- A factor can be set to certain ***levels***, like *n=10, 20, 30*, or *InitPolicy = (random, greedy)*.

More Terminology

- Factors can be ***manipulated*** (you choose the levels), ***fixed*** (level stays constant) ***randomized*** (generated at random), or merely ***reported*** as found.
- A ***design point*** is a tuple assigning levels to (manipulated) factors: e.g. (*n=20, IP=greedy*).
- A ***trial*** is one measurement of performance indicator(s) at a specific design point.
- An ***experiment*** produces a collection of measurements, over multiple trials at several design points.

Notation

$$X = \{x_1, x_2, \ldots x_n\}$$

A sample of *n* measurements.

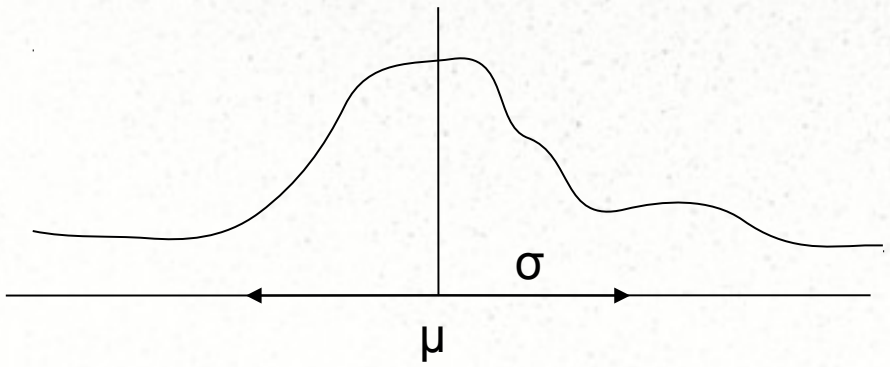
$$\overline{X} = \left(\frac{1}{n}\right)\sum_{i=1}^{n} x_i$$

The *sample mean.*

$$Var(X) = \left(\frac{1}{n-1}\right)\sum_{i=1}^{n} \left(x_i - \overline{X}\right)^2$$

The *sample variance*.

$sd(X) = \sqrt{Var(X)}$



Parameters of the *underlying mode*l: μ, σ

Mean of *X* is an *estimator* of *μ*.

Design of Experiments (DOE)

- *After planning (deciding which factors to manipulate in the experiment, and what performance indicator to use):*
- Which levels should I assign to (manipulated) factors?
- Which design points should I test?
- How many trials at each design point?

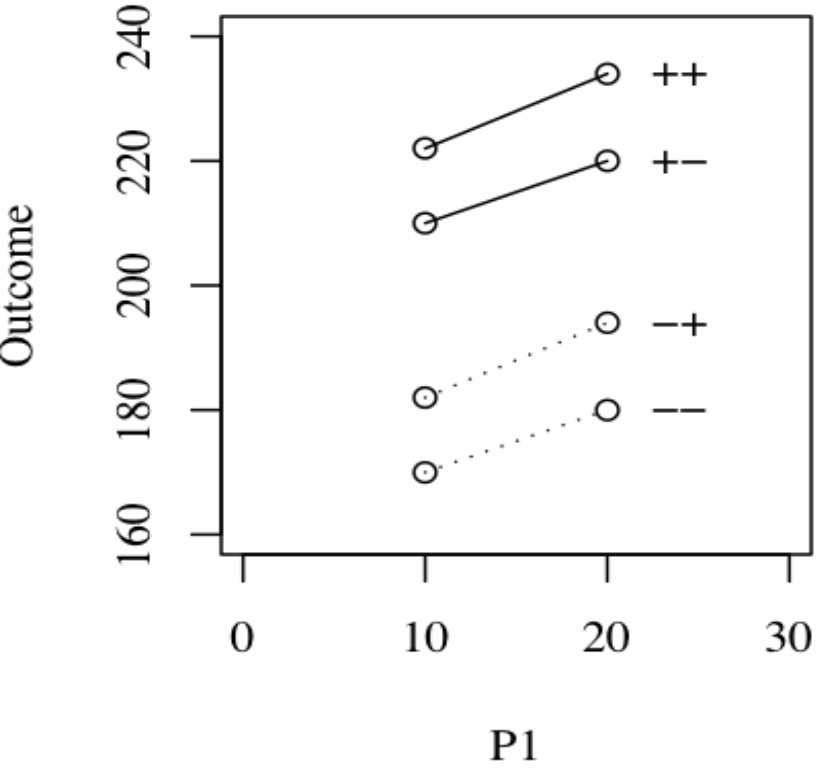
Basics of DOE

- Given *k* (manipulated) factors,
- Assign each a ``high'' and a ``low'' level, labelled + and -.
- A *full factorial design* is an experiment using all combinations of all levels.

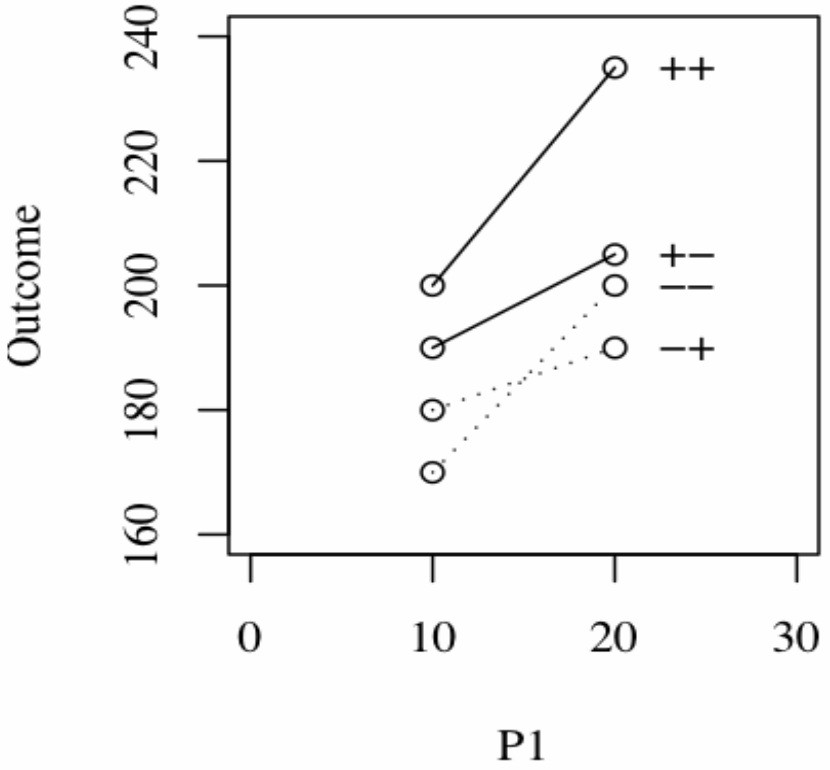
P1	P2	P3
-	-	-
-	-	+
-	+	-
-	+	+
+	-	-
+	-	+
+	+	-
+	+	+

Factorial Designs

Main effects: Increasing P1 always increases cost (positive slopes); Increasing P2 always increases cost (solid vs dash); P3 also has (small) positive effect (+- vs ++).

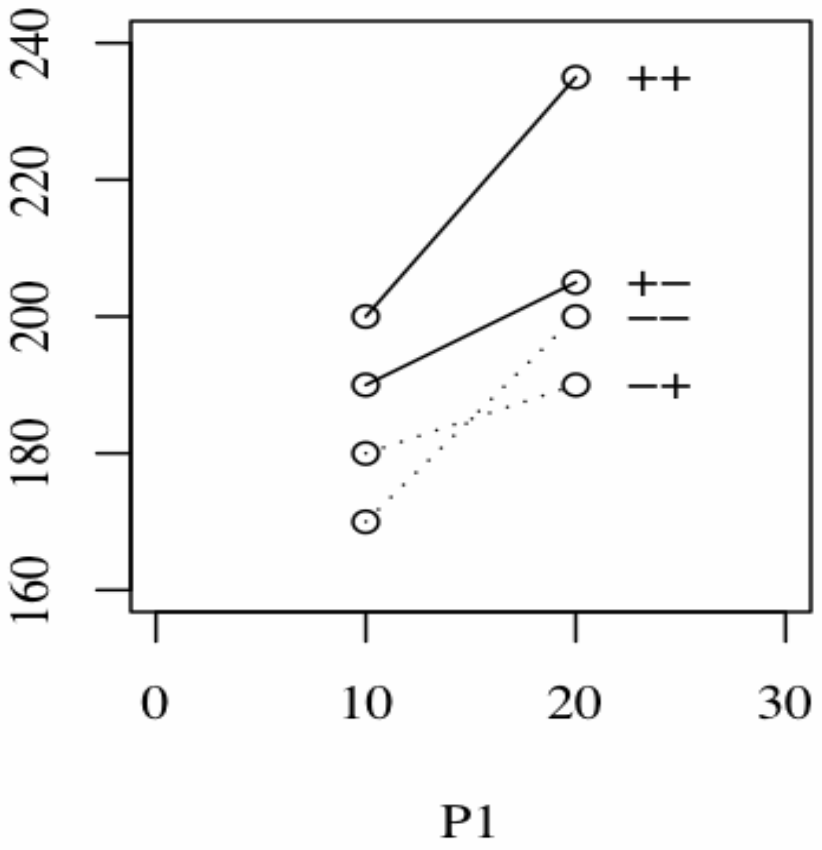


Interaction effects: The magnitude of the effect of P1 depends on P2==P3. (This is a three-way interaction.)



Factorial Designs

- Full 2^k factorial designs capture *all* main and interaction effects.
- Partial designs omit some effects.
- Fixed P3 = omit some lines.
- Randomize P3 = average some lines (if randomized properly).

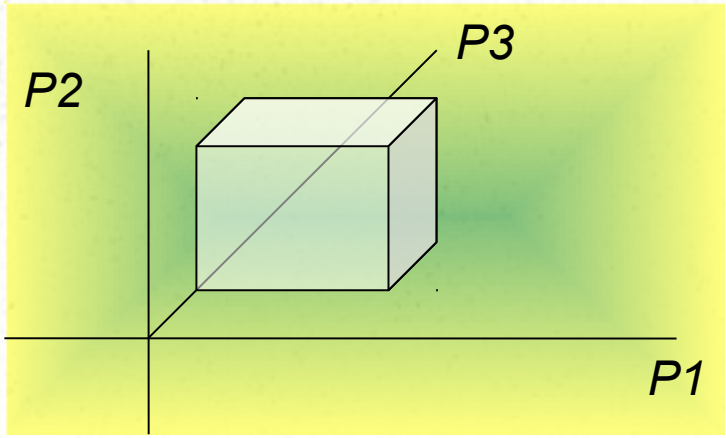


Analyses of Full Factorial Designs

- Assume linear model, normal error term.
- Estimate coefficients *a* … *h*.
- Which factors are most (least) important to cost?
- Identify interactions.
- Estimate ε.

$$C = a\,P1 + b\,P2 + c\,P3 + d\,P1\,P2 + e\,P1\,P3 + f\,P2\,P3 + g\,P1\,P2\,P3 + h + \varepsilon.$$

Coarse approximatio may be ok for preliminary surveys of factors and costs.



Full Factorial Designs for Algorithms

- Choose the important factors to manipulate,
- Try all combinations of factors,
- Build a simple (but often sufficient and effective) model of outcomes.
- Publish!

- But it's usually not that simple ...

DOE for Algorithms

- k factors x 2 levels per sample = 2^k design points. Exponential explosion!
- Formal DOE is intended to be used with, say, $\leq$ 3 factors and $\leq$ 3 levels per factor.
- Algorithms (especially heuristics) involve way too many factors.
- Often need way more than two levels.

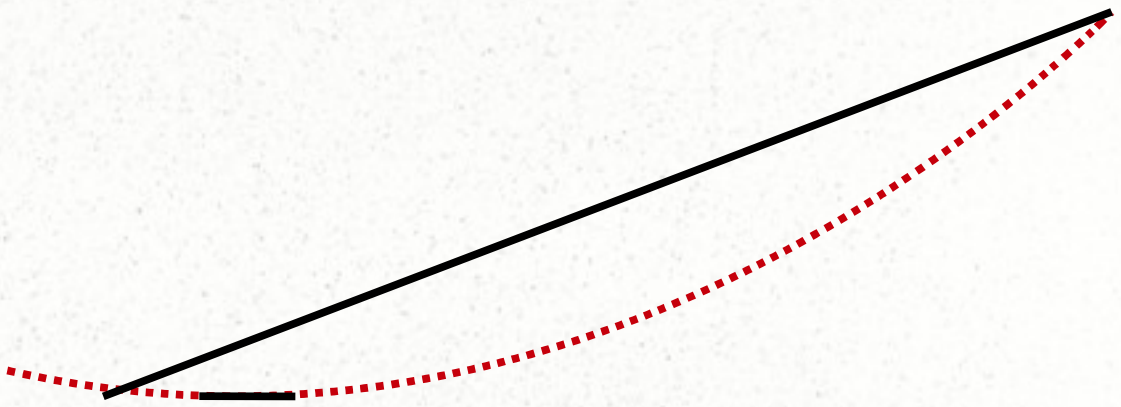
Too Many Factors

- Iterated Greedy has 7 algorithm factors ...
- Plus several input factors ...
- And don't forget environment, code factors ...
- Which ones to select?

```
G = Input Graph (Input factors)
C = Initial Coloring (factor 1)

Loop

     Reorder colors (factor 2 )
     Reorder vertices (factor 3)

     C = GreedyColor(G) //save
best

     C = Kempe reduction (factor
4)
     C =  Retry (factor 5)
     stop = Stopping  rule
(factors 6, 7)

until (stop)

Report C* best coloring found.
```

How Many Levels (Quantitative)?

- Factors 4,5,6 (Kempe, retry, max iterations) are quantitative:
- Where to place lo, hi?
- Scaled or constant?
- How far apart?
- How many levels are needed?



Nearby = closer to the truth; far = range of performance.

Interaction effects depend on scale.

Nonlinear models: Need *b+1* levels to fit a *b*-degree polynomial.

Unknown model need more levels. (Recall bad example 3).

Which Levels (Qualitative)?

What is the meaning of ``hi'' and ``lo'' levels in this context?

What is the meaning of ``linear model''?

Can split into 6 factors, but ...

- **Factors 1,2,3,8 (Initial coloring, Color, Vertex rules, target) are qualitative.**
- **`Color Rule = W.`**

$$W = (w_1, w_2, w_3, w_4, w_5, w_6)$$

a 6-tuple of weights. Rule r is selected at each iteration with probability $\sim w_r$

Algorithmic Design Questions

- *How to reduce the number of factors?*
- *How to choose appropriate levels?*
- *How to reduce the number of design points?*
- *How many trials per design point?*

- *Some answers from formal DOE...*
- *Some answers from informal experience...*

Formal DOE

- ***Fractional factorial designs***: formal designs that omit some design points, but preserve main and low degree effects.
- ***Factor analysis (ANOVA and etc.)***: to decide which factors are least important to outcomes, and can be omitted.
- *See textbooks on design of experiments for more information.*

Informal Planning of Experiments

- Use situational knowledge to decide what to prioritize and what to eliminate.
- Build situational knowledge using a Pilot Study.
- Remember that experiments are iterative: use outcomes of the previous experiment to plan the next one.
- *More about this in the next section...*

Any Questions About ...

- Design of Experiments?
- Factorial designs, other types of designs?
- Formal models?
- Levels and design points?

Planning Experiments

- What should I measure?
- What do I do about all these factors?
- How to choose reasonable levels?
- How many trials?

What Should I Measure?

- ***Theory = Accuracy:*** Asymptotic bound on growth of the dominant operation.
- ***Practice = Precision***: Measure CPU time or Wall Clock time.

Accurate = Always true, no matter where you are.

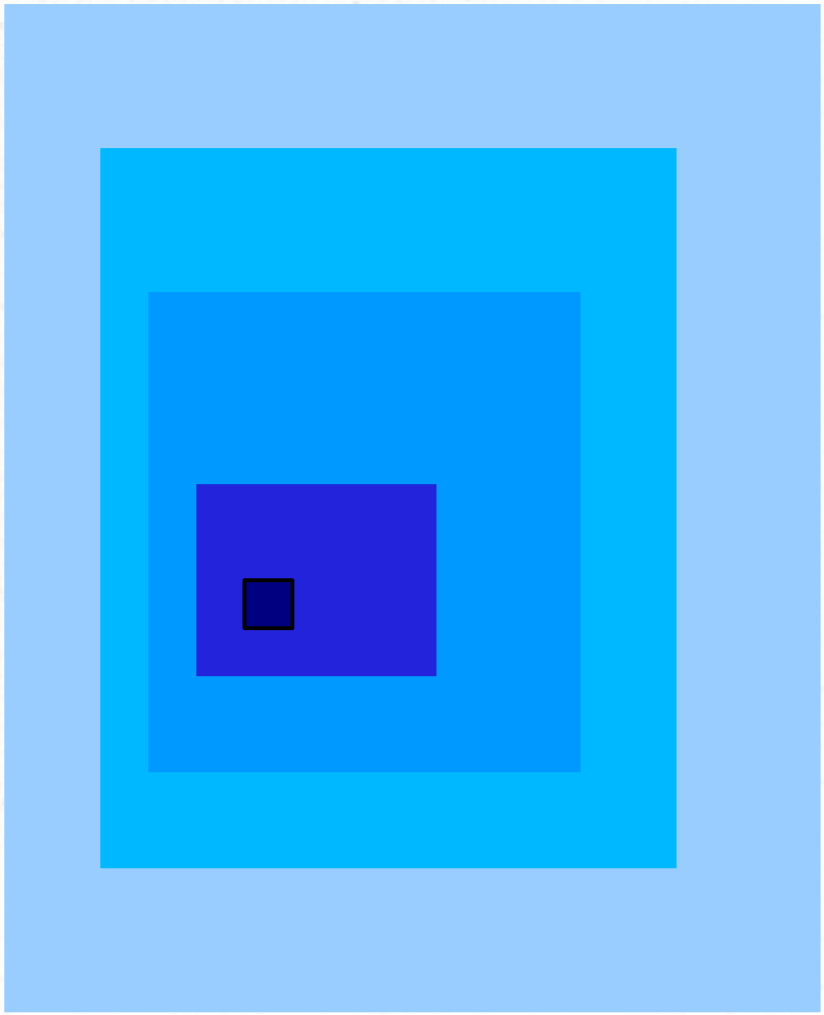
Precise = Many significant digits.

How Deep is the Ocean?

- ***Theory:*** The ocean is at most O(*n*^2) feet deep when *n* feet from shore.
- ***Practice:*** New York Harbor is 24.1532 feet deep. The Atlantic Ocean is 12,881.828 feet deep in one spot; and 102.039 feet deep in another ....

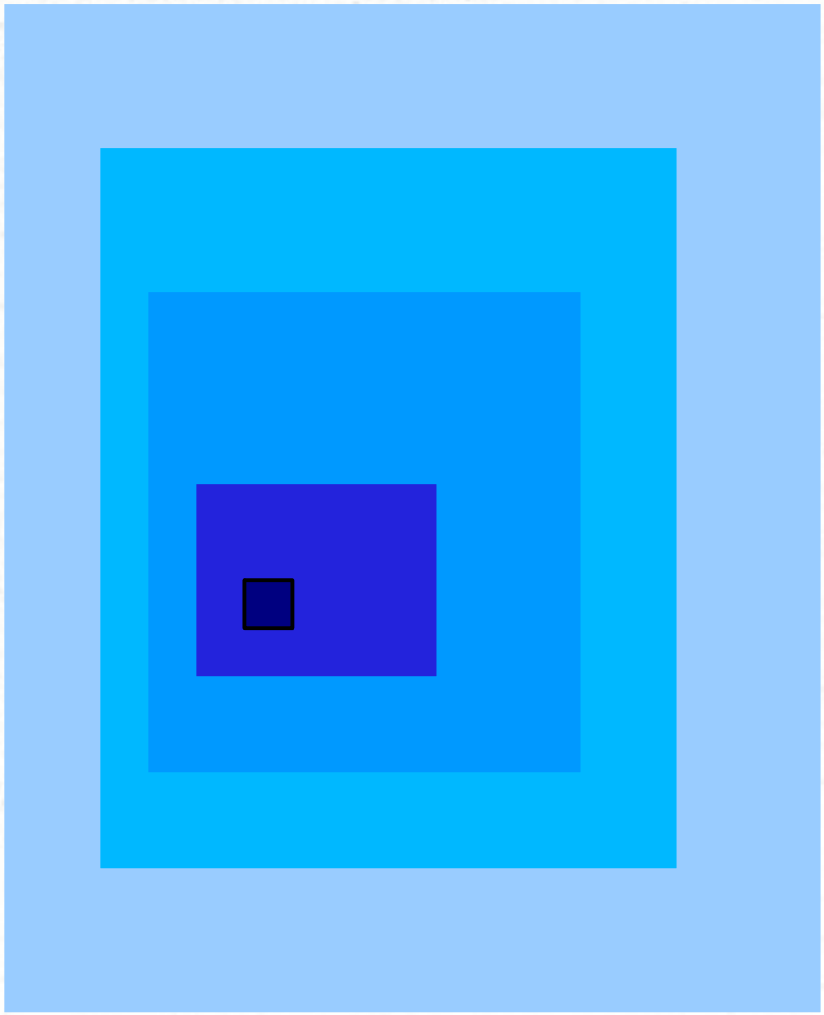
What Should I Measure?

- Theory's dominant op.
- Data structure ops.
- Cost per operation
- Main loop iterations
- Bottleneck operations
- Code blocks
- Instruction counts
- CPU time
- Memory accesses
- Cache & page misses
- Wall clock time
- *... experimenters have more choices.*



How to Measure Performance

- Match the performance indicator to the research question, on a scale from abstract (most accurate) to instantiated (most precise).



Instantiations of the Algorithm

Algorithm

Quicksort A[lo,hi]:

x is an element of A

Partition A around x

Recur to left of x

Recur to right of x

Program

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

Process

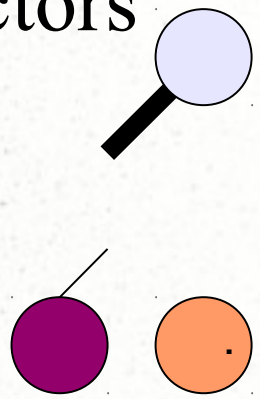
... Paradigm ... Algorithm ... Data Structures ... Source ... Object ... Process ...

Most Accurate

Most precise

How to Decide

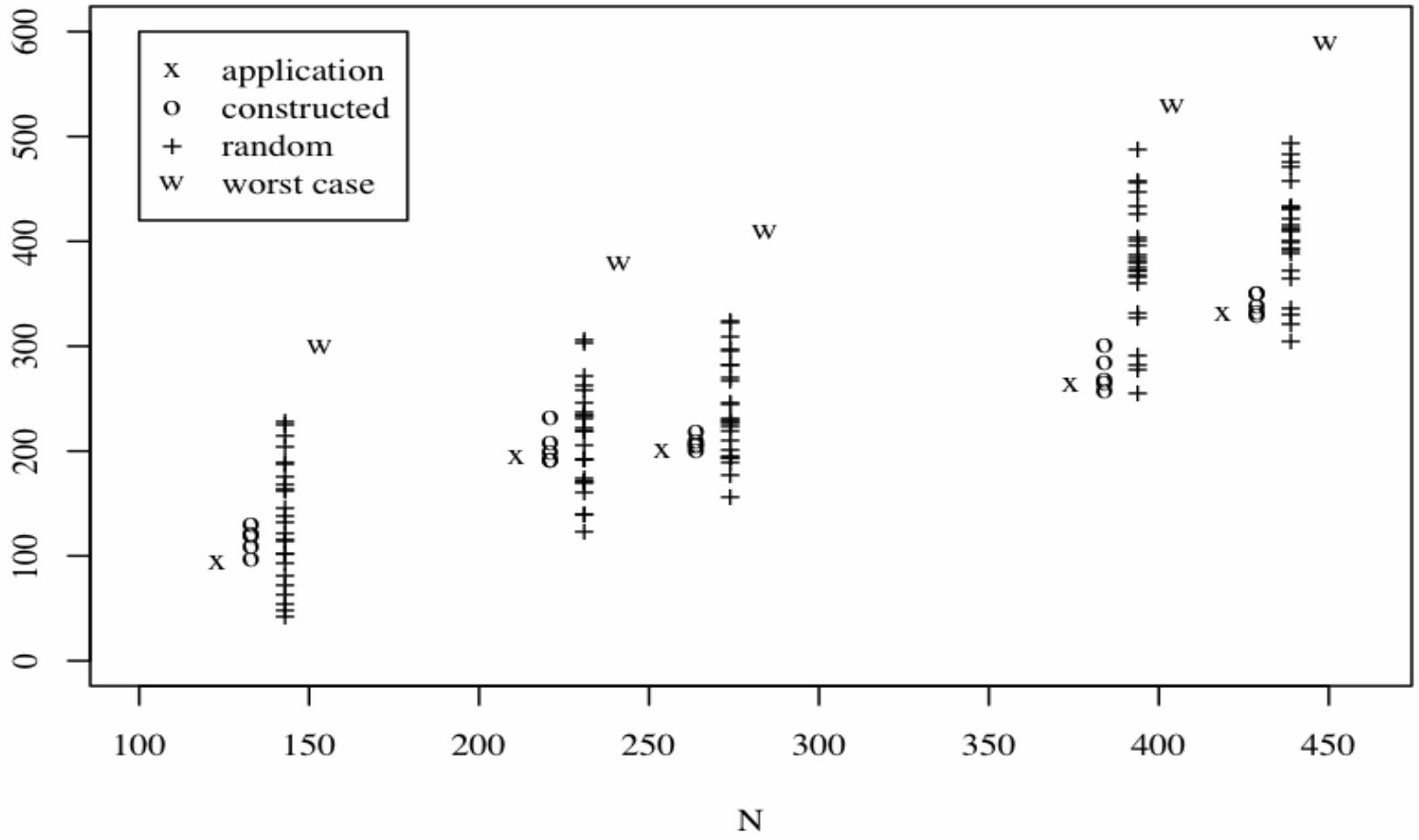
- Choose as much generality as your precision needs (to distinguish outcomes) will tolerate.
- Choose PI that *isolates* the interesting factors and *ignores* irrelevant factors.
- Choose indicators that are *directly comparable,* when making comparisons.
- Choose indicators that match the literature.
- OK to use more than one performance indicator when necessary.



What Factors Should I Study?

- ***Algorithm Factors***: data structure, initialization policy, stopping criteria, execution order, step size, any parameter ....
- ***Input Factors***: (1) input class, (2) instance parameters, ...
- ***Code Factors***: programming language, programmer skill, data types, generality of code ...
- ***Environment Factors***: compiler, optimizer, memory hierarchy, coprocessors, clock speed ...

Input Classes, Instance Parameters



Input Classes: Some Choices

- *Worst case, best case*: to define boundaries
- *True random*: to provide ranges & bounds
- *Constructed parameterized*: to test dependence of performance on parameters.
- *Benchmark*: to compare to results in literature.
- *Typical*: representative of practitioner's experience.
- *Perterbed typical (randomized or pieces)*: to increase the pool of representation

Planning the Next Experiment

- ***Manipulate*** the factors that are (next) most important to performance.
- ***Fix*** the factors that you know have no effect on cost.
- ***Randomize*** and/or ***record*** the factors you aren't sure about.
- Move factors from ***manipulate*** to ***randomize*** to preserve the broad range of possible outcomes.
- Move factors from ***manipulate*** to ***fix*** to narrow the scope and focus on an interesting subproblem.
- Move factors from ***manipulate*** to ***record*** if they are similar to, or depend, on other factors.

Tips on Choosing Factors

- With a platform--independent performance indicator, you can *fix* all environment and code factors.
- Numerical factors can be *interpolated*, qualitative factors can not. (Sometimes you can transform qualitative to quantitative factors.)
- Consider the previous literature: What is already known? What is not yet known?
- Consider your audience: They can match the algorithm (and sometimes code) factors; practitioners can't choose environments or inputs.

Choosing Input Classes and Instance Parameters

- *Classes*: Be compatible with old, and add something new.
- *Classes*: Match to the instantiation level of the experiment.
- *Sizes*: Max *N* possible, then reduce by *k* constant, or *m* multiplicative, for a total of *b*+1 levels. (For a a b-degree polynomial).
- *Instance sizes*: Convert numerical levels to quantitative groups (hi, lo) in application instances.
- Factorial designs: Choose the same instance sizes (and parameters) across all classes!

The Pilot Study

- *Critical for defining the boundaries of your experiment:*
- *What are the largest and smallest problem sizes possible?*
- *How much time does it take?*
- *How much variance in the data?*
- *What is most / least important to performance?*
- *etc.....*

Questions About ...

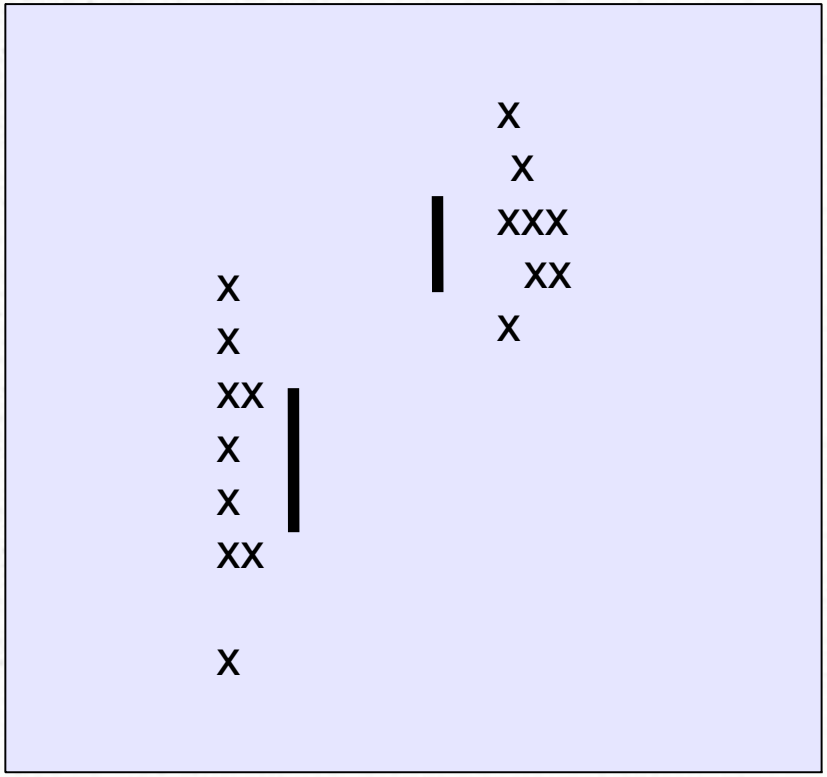
- Choosing performance indicators and instantiation?
- Factors and levels?
- Number of trials?
- The Pilot Study?

Variance Reduction Techniques

- How to create experiments that produce analysis-friendly data samples.

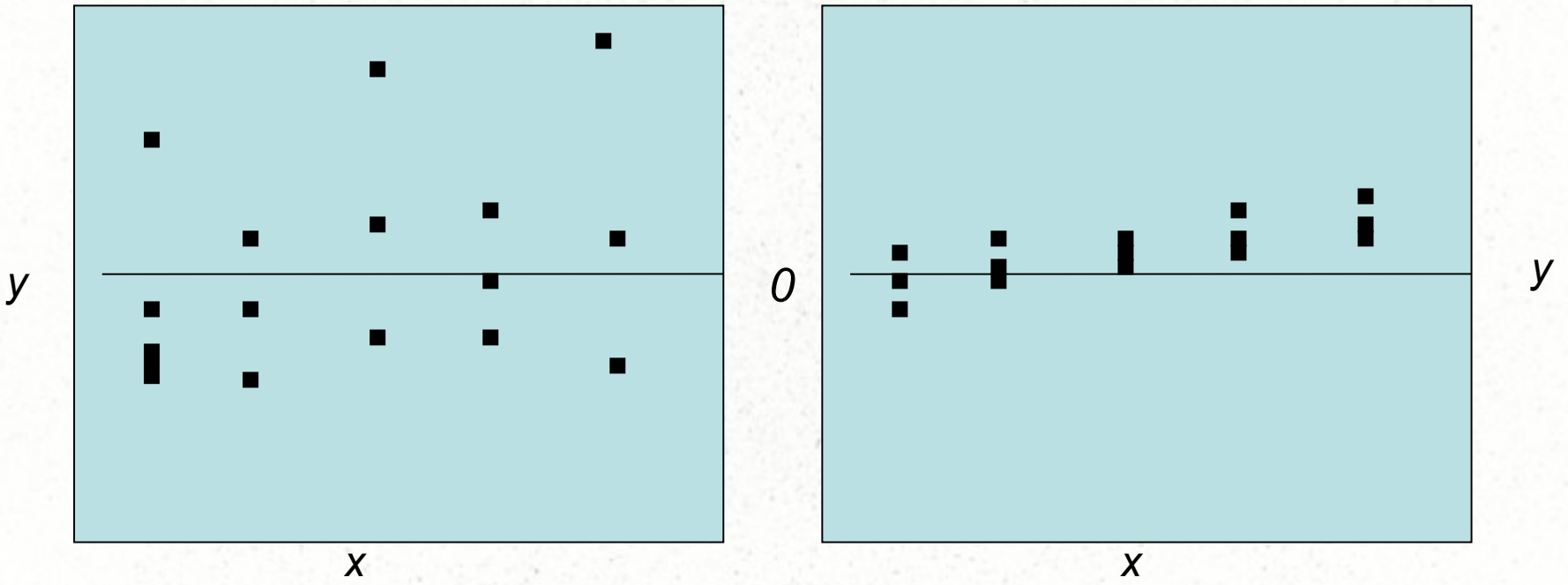
What is Analysis-Friendly?

- Comparing means: Small variance *within* a design point, compared to variation *between* design points.
- Greater separation in dp.: more variation between.
- More trials per dp: less variance within.



Variance reduction techniques produce less variance in same number of trials.

What is Analysis-Friendly?



Is the mean of y asymptotically positive or negative?

Example: Self-organizing Sequential Search

- Given a sequence of *m* requests for items *1..n*.
- Cost = location in list,
- *P* = request probabilities.
- What is the expected *Cost (Zipf, n=20,m=1000)* of 2 reorganizing rules M and T?
- *Implement M and T, generate inputs, measure costs …*

```
L: 3, 5, 7, 1, 2,
4, 9, 8,6
```

Request *r = 2*
Cost = *5*

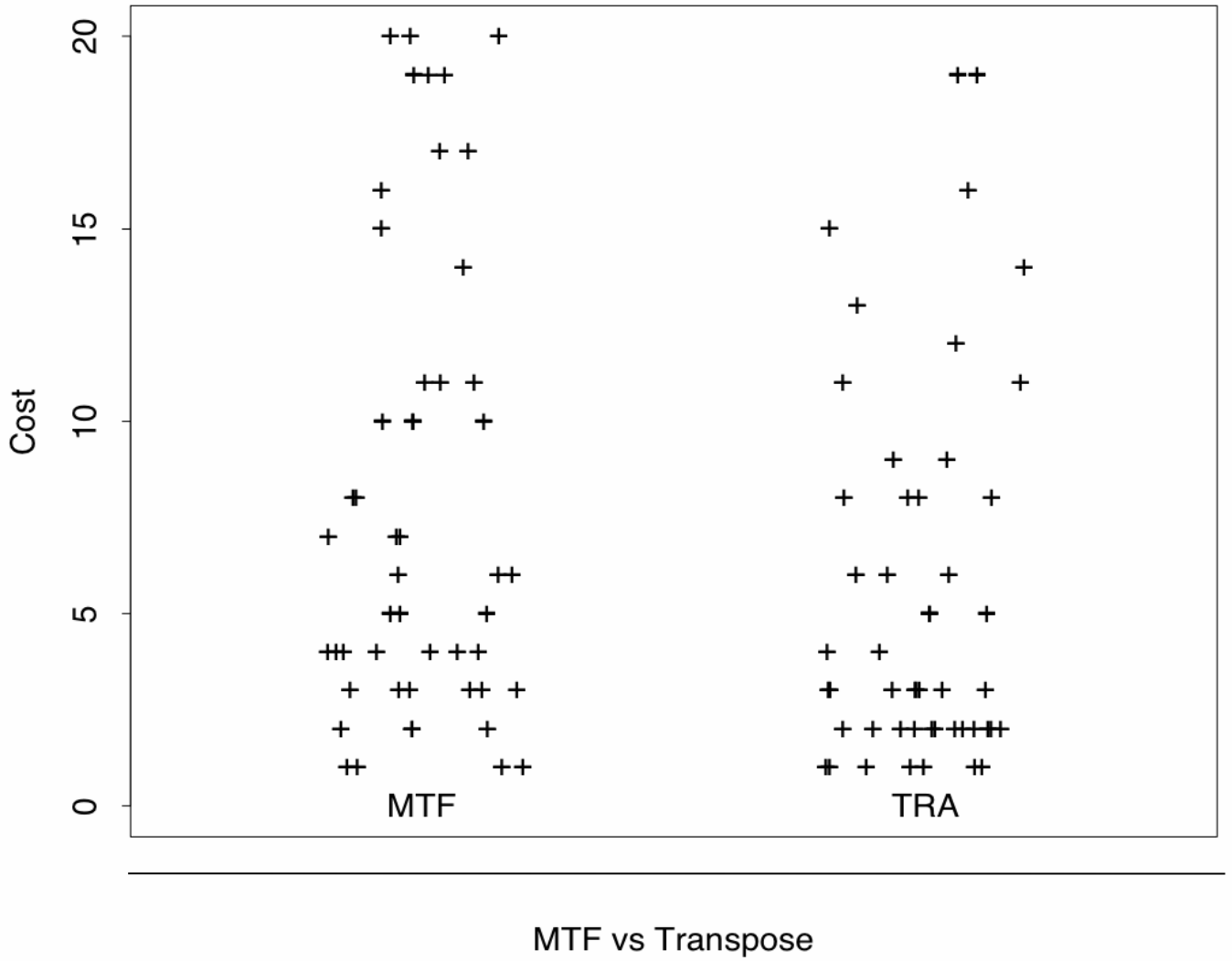
```
M: 2, 3, 5, 7, 1,
4, 9, 8, 6
```

Move to Front Rule

```
T: 3, 5, 7,  2, 1,
4, 9, 8,6
```

Transpose Rule

Comparison of MTF and Tr



VRT: Common Random Numbers

- If comparing two outcomes, and ...
- Outcomes are positively correlated w.r.t a random component in the experiment, then ....
- Run both experiments using common random number sequence. .

Sequential Search: When comparing cost of M & T, use the same random request list.

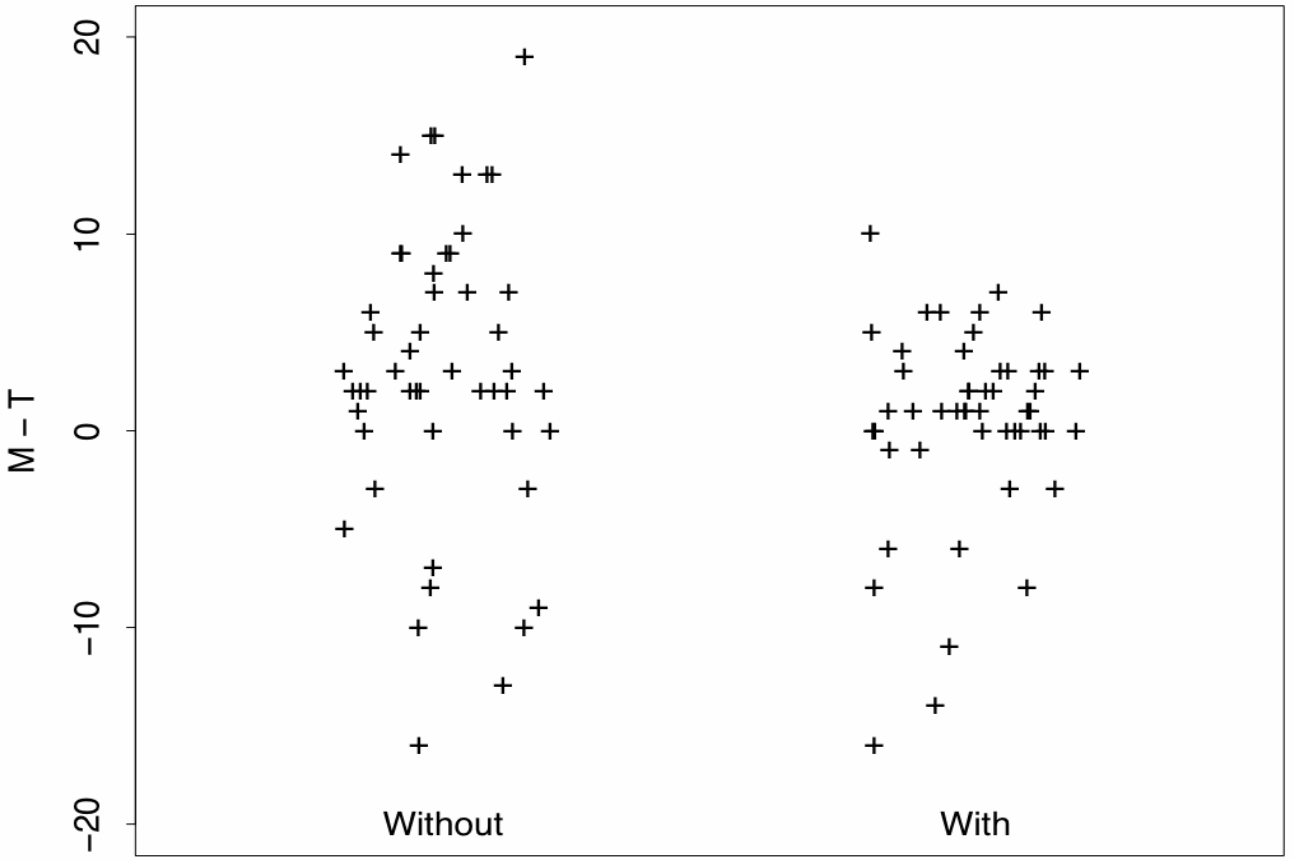
$$D[L] = M[L] - T[L]$$

$$E[D] = E[M - T] = E[M] - E[T]$$

$$Var(D) = Var(M) + Var(T) - 2Cov(M, T)$$

f covariance > 0, variance is reduced.

Common Random Numbers



St. Dev = 7.50

W/WO Paired Inputs

St. Dev = 5.03

n=20, m=1000, t=50 random trials, P = Zipf's Law

VRT: Control Variates

- If Cost is correlated with another variate Y, and ...
- Expectation $E[Y] = y$ is known, then ...
- Measure Y and look at the difference.
- Y is a ``control variate'' for C.

$Cov(C, Y) > 0$

$E[Y] = y$ (known)

$E[C] = c$ (unknown)
$E[C - Y] = c - y$

$Var[C - Y] = Var(C) + Var(Y) - 2Cov(C,Y)$.

Measure $D_i = C_i - Y_i$
Estimate $E[C]$ with $D_i - y$

Control Variates

Cost of request *r* in Transpose list is positively correlated with cost of *r* in Optimal list.

Cost of *r* in Optimal list = *r*

Measure *D* = *T* – *r* *for each request.*

Estimate *E[T] with D + r*

R is a control variate for T.

T: 3, 6, **2**, 5, 1, 4, 7

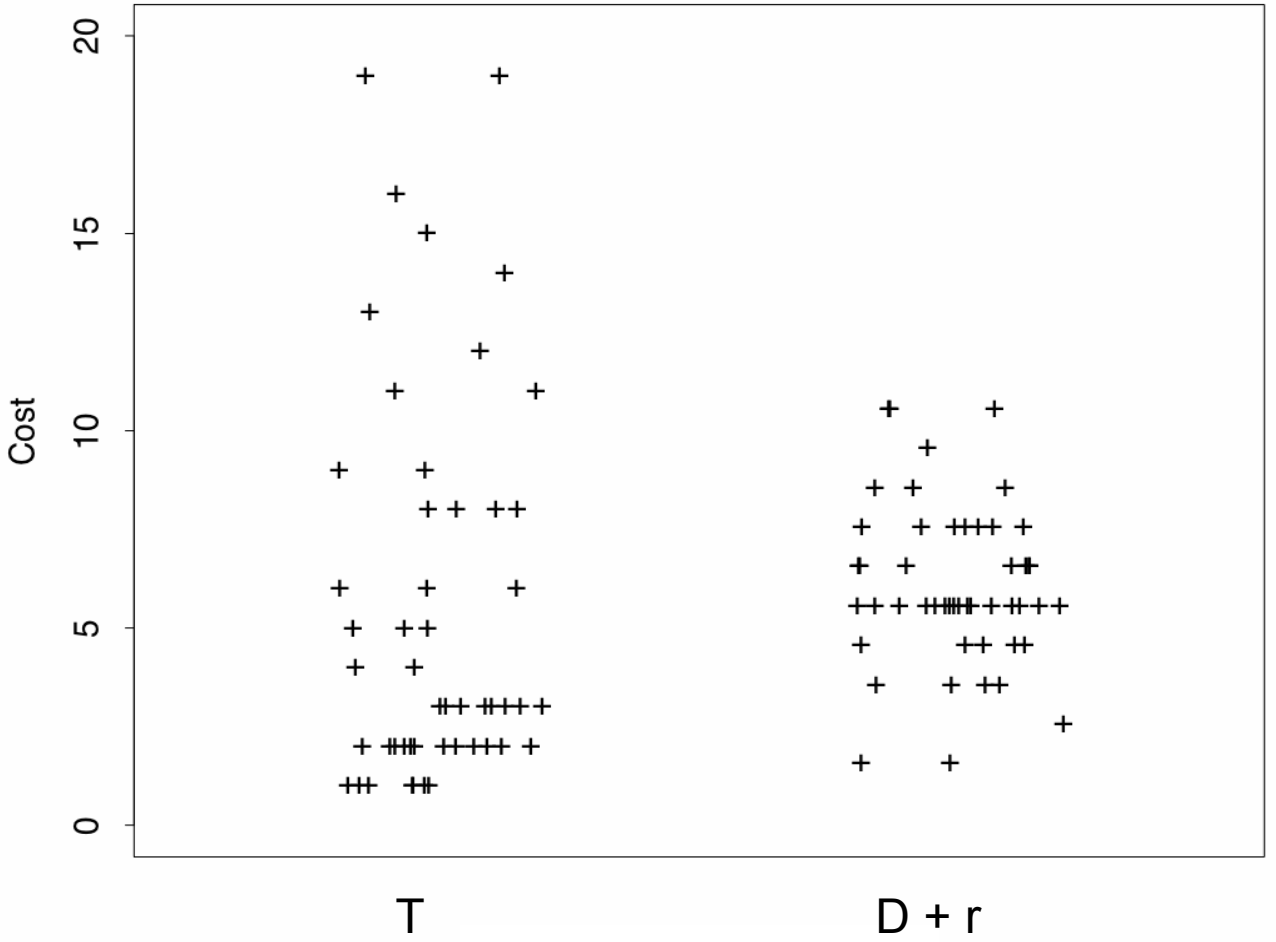
Cost = 3
D = 1 = 3 - 2

Opt: 1, **2**, 3, 4, 5, 6, 7

Assume

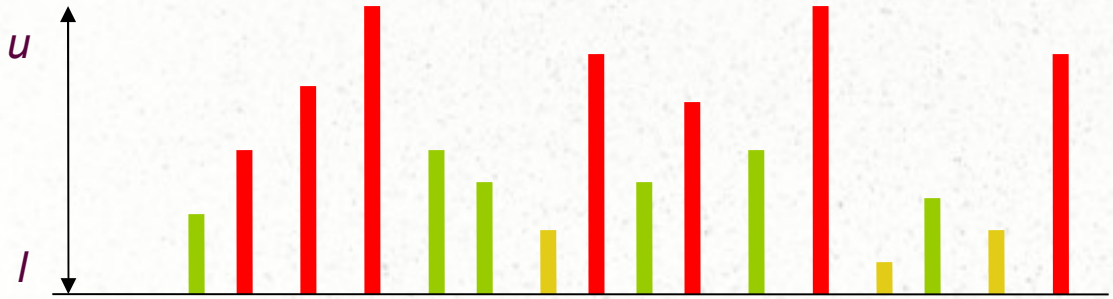
$$P = (p_1 \geqslant p_2 \ldots \geqslant p_n)$$

Control Variates

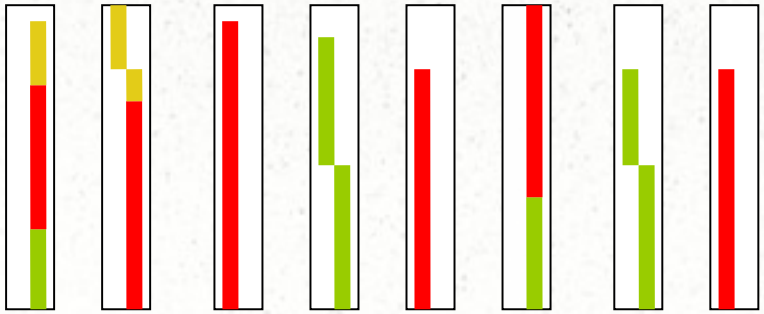


n=20, m=1000, t=50, P = Zipf's Law

FF Bin Packing: Control Variates



Consider weights in order of arrival; pack each into the first (leftmost) bin that can contain it.



Given a list of *n* weights uniform on *(0,u)*, pack into unit-sized bins using the First Fit heuristic.

What is the expected cost (bins used) of *FF(n,u)*?

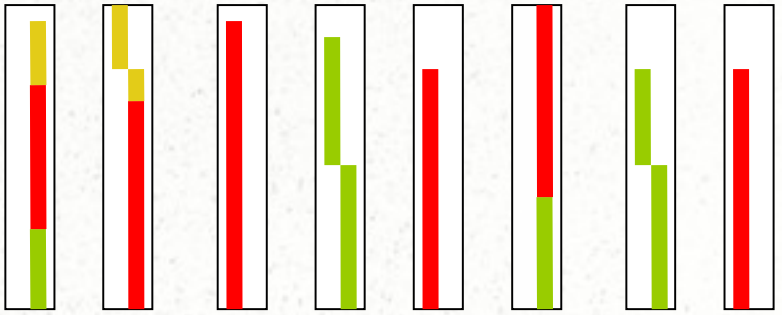
FF Control Variates

Bins = Weight sum + Empty Space

Bins is positively correlated with weight sum.

E[W] = nu/2

Measure empty space. Estimate mean bins with E[B] = E[S] +nu/2

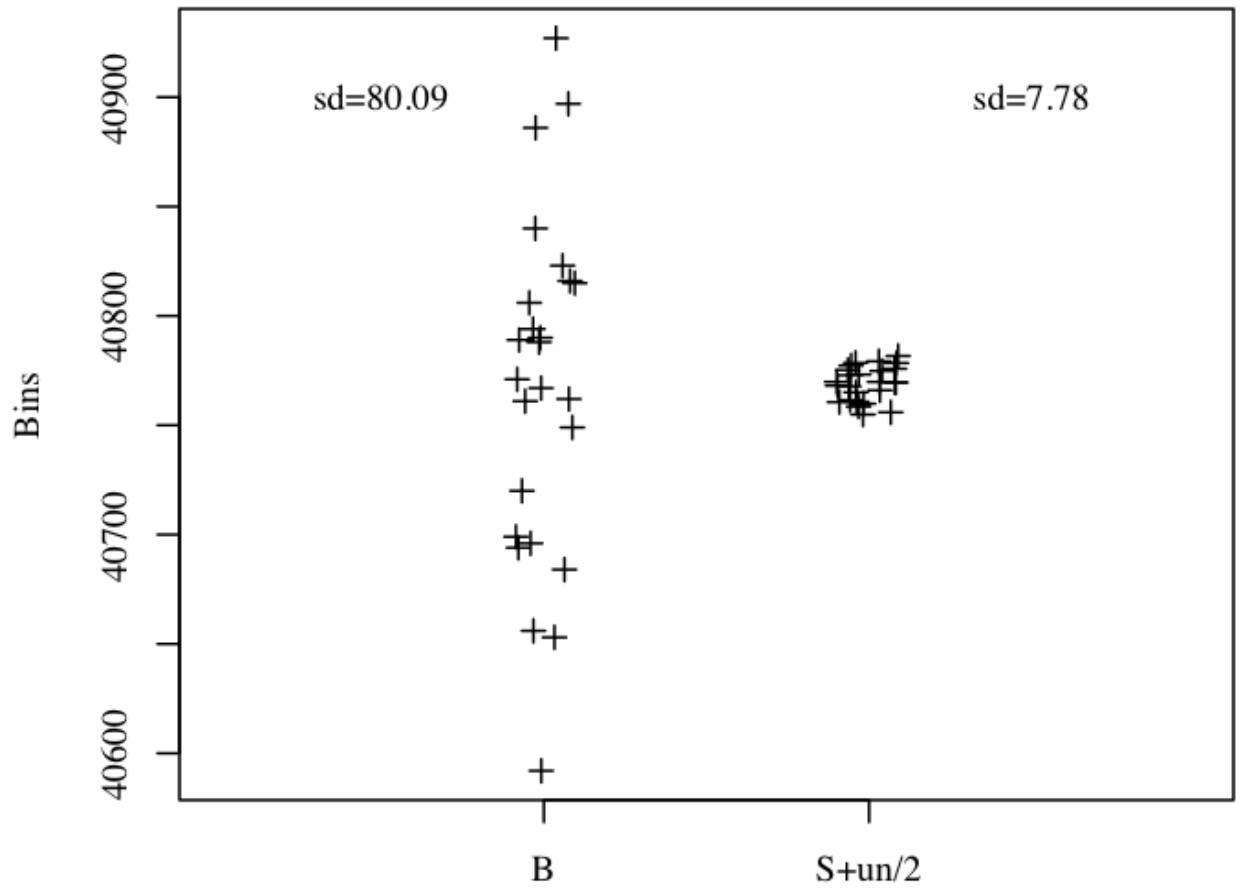


Weight sum is a control variate for number of bins.

Empty space S has less variance than B.

S + nu/2 is easier to analyze than B.

FF Control Variates



Sources of Control Variates

- Lower bound on solution (bin packing).
- Optimal solution (Transpose).
- Initial solution in iterative algorithm.
- Solution to a simpler problem.
- Algorithm sub-structure ...

1. Is the variate X correlated with solution cost? (Check it experimentally.)

2. Do I know the expectation x of this variate?

If YES: Measure the difference D = C – X. . Substitute x to estimate mean cost.

E[C] = E[D] + x

VRT: Conditional Monte Carlo

- If Cost depends on another variate *Z*, and ...
- the *conditional expectation* of C re Z can be calculated, ... then
- Generate Z_i, and calculate $f(Z_i)$.

$$E[C_i] = c$$

$$E[C_i | Z_i] = f(Z_i)$$

$$E[f(Z_i) | Z_i] = c$$

MTF: Conditional Monte Carlo

- Cost of request *r* at time *t* depends on prior list order.
- Expected cost of the list *f(L)* can be calculated.
- Generate list M at time *t*, then calculate *f(M)*

M: 3, 5, **2**, 1, 4

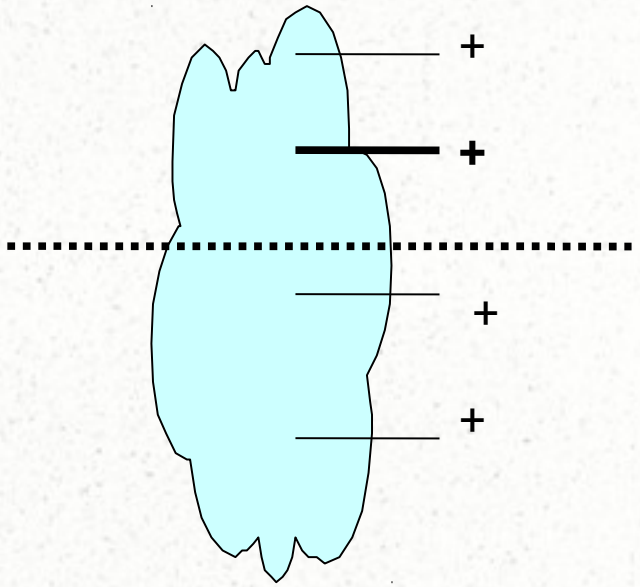
Instead of one request:
r = 2, *cost* = 3

Calculate the expected cost of all requests for this list:

$$c = \frac{1\,p_3 + 2\,p_5 + 3\,p_2 + 4\,p_1 + 5\,p_4}{5}$$

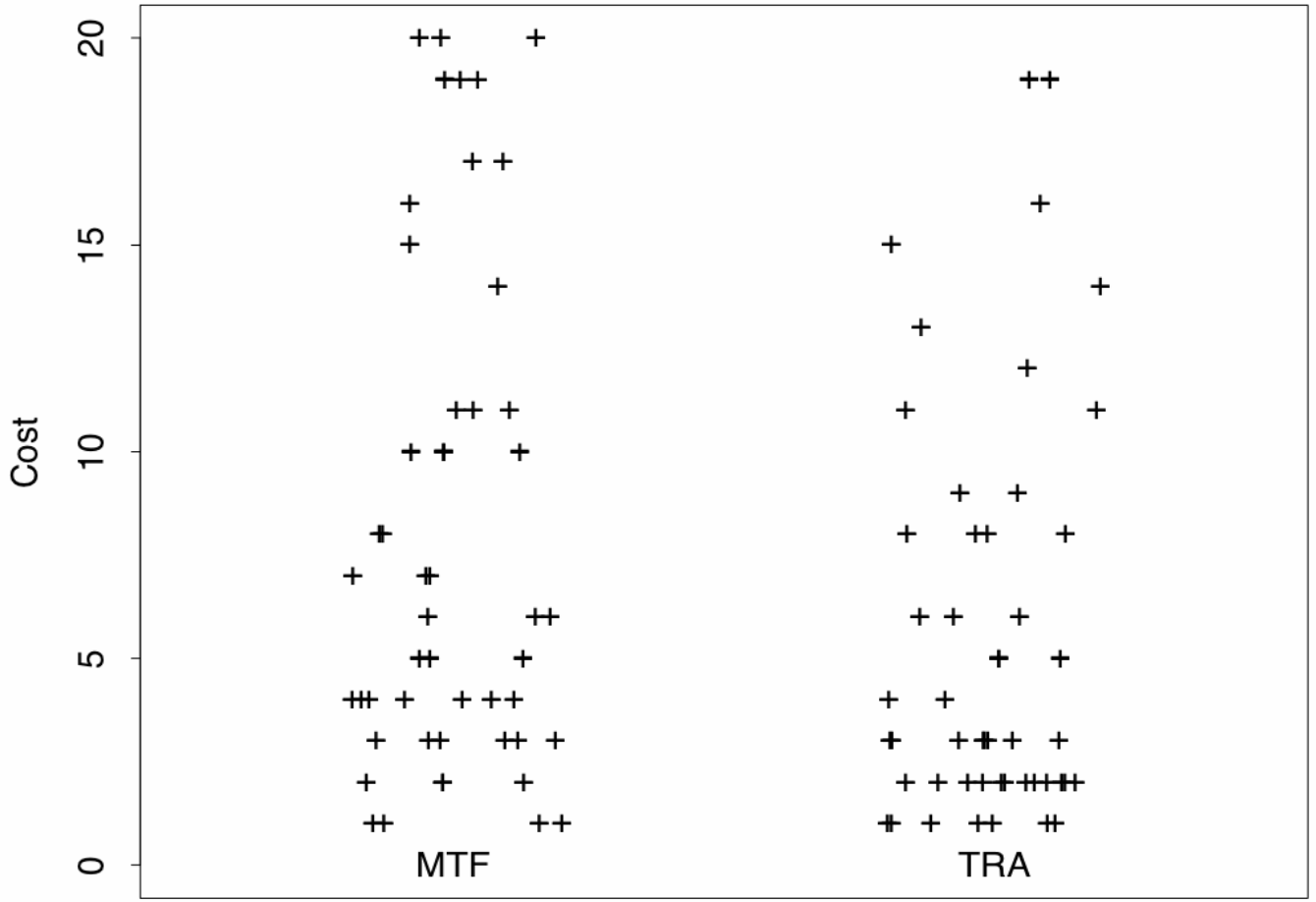
Ideas for Conditional Monte Carlo

- Iterative, data structure-based, and, random walk algorithms ...
- Generate state S at time t.
- Instead of sampling a random cost (+) based on S, compute expected cost (--) of S.



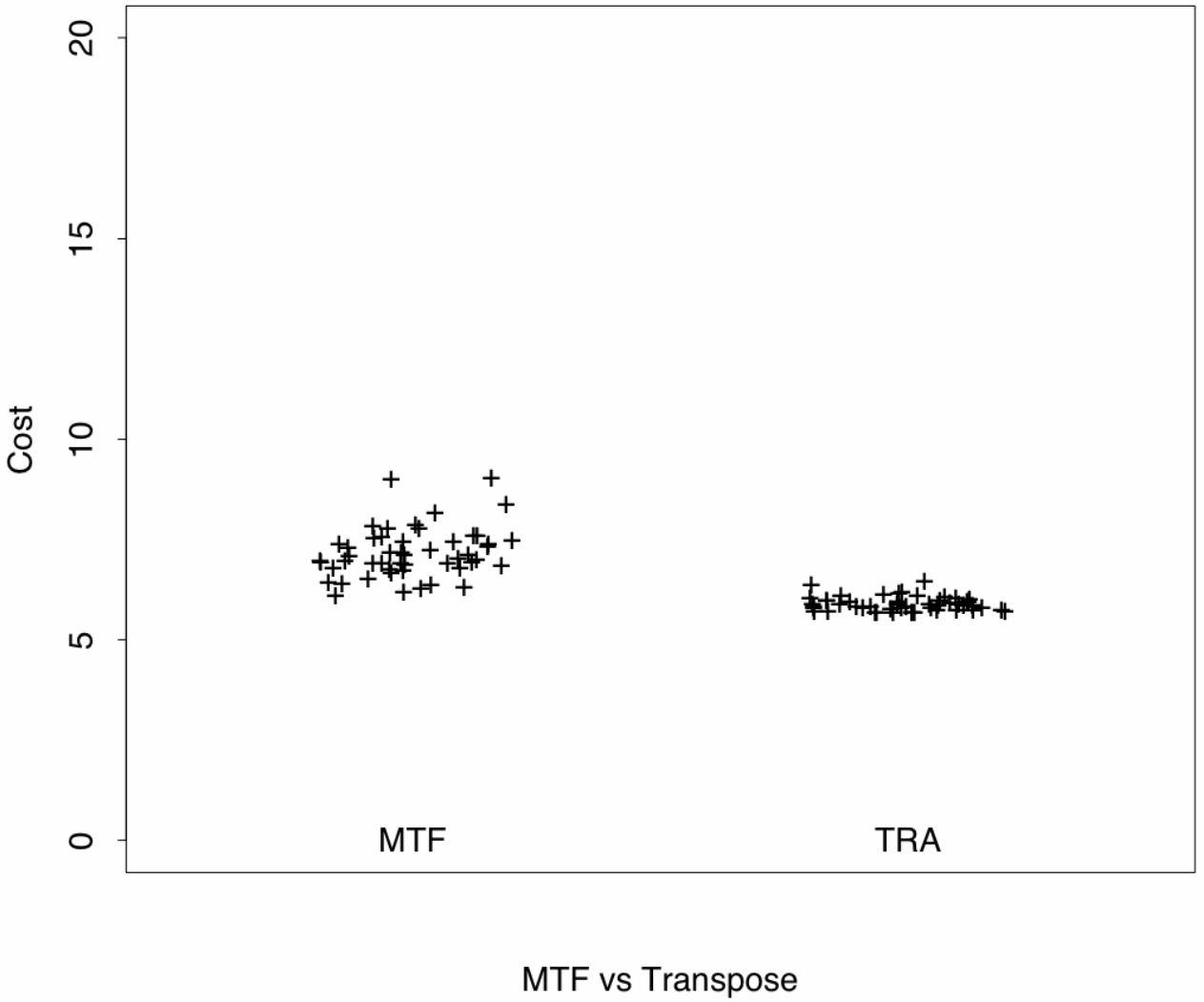
Splitting: if E[S] can't be calculated, samplethe cost many times from S.

Before: No Variance Reduction



MTF vs Transpose

After: With Conditional MC



Many Other VRTS

- Antithetic Variates
- Splitting
- Stratification
- Poststratification
- Importance sampling
- ....

No Guarantees that any will work.

But when they do, very powerful!

Questions about VRTS?

- Bratley, Fox, Schrage, ``A Guide to Simulation," Springer-Verlag.
- CCM, `` Analyzing Algorithms by Simulation: Variance Reduction Techniques and Simulation Speedups," ACM Computing Surveys JUNE 1992.

Statistical Methods for Algorithm Evaluation – Three Lessons

I. **Control = Responsibility: How not to do it.** Avoiding rookie mistakes.

II. **Control = Power: Before the (next) experiment.** DOE, planning experiments, variance reduction techniques.

III. **Control = Choice: Statistics and data analysis.** The right tools for unusual jobs.

Part III. Control = Choice: Statistics and Data Analysis

1. **Making comparisons.**
2. **Analyzing trends**.

Algorithms and Statistics Mismatch

Algorithm – type question:

What is the asymptotic cost of algorithm X, as a function of *n*, on instances randomly selected from class C?

Statistics – type answer:

With certainty at least .95, the difference between cost at *n=100* and cost at *n=1000* is not zero, assuming cost is normally distributed and variance is constant in *n*.

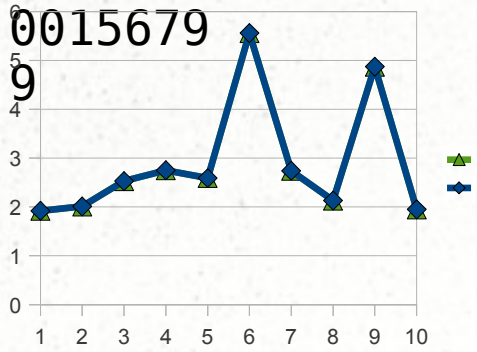
Choose carefully!

Methods of Data Analysis

- Descriptive statistics
- Exploratory data analysis (EDA)
- Graphical methods
- Parametric and nonparametric inference

Mean = 2.91, Var = 1.61

```
5: 6
      Stem and
4: 9
       Leaf
3:
      Chart
2: 0015679
1: 9
```



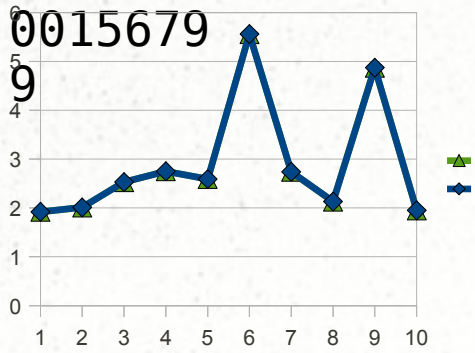
95 percent confidence interval:
μ = [2.0046, 3.81866]

Methods of Data Analysis

- Descriptive statistics
- Exploratory data analysis (EDA)
- Graphical methods
- Parametric and nonparametric inference

Mean = 2.91, Var = 1.61

```
5: 6
      Stem and
4: 9
       Leaf
3:
      Chart
2: 0015679
1: 9
```



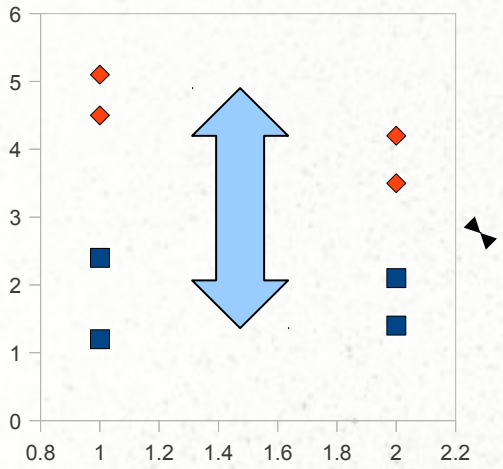
Today!

95 percent confidence interval:
μ = [2.0046, 3.81866]

Why Nontraditional Methods

- Less well known to computer scientists, operations researchers, and AI types.
- Good for big data sets.
- Good for high-dimensional data.
- Good for unusual (non-normal) distributions. .
- Can capture complex patterns and relationships.

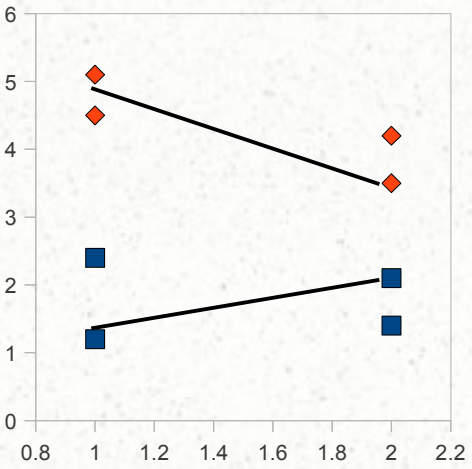
Three Goals of Data Analysis



Comparison:

Bigger/smaller

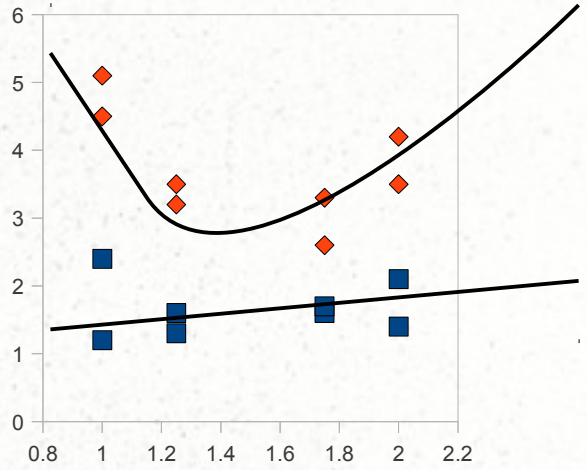
By how much



Line Fitting:

Apply a convenient model (function family), usually a line.

Interpolate



Model Fitting:

Find the ``true'' model.

Extrapolate

Today's Topics

- **Making comparisons**: Descriptive/summary statistics, graphical analysis of distributions, graphical summaries, correlations, multivariate data.
- **Analyzing trends:** Fitting lines; fitting nonlinear models, data transformation, hazards of interpolation and extrapolation.

Comparison

- Measure the cost of five levels (A,B,C,D,E) of some factor.
- Fifty random independent trials each level.
- How do they compare?
- (Which is best?)

```
A      1      132.26187
A      2      109.57494
A      3       80.17537
A      4      104.45386


...




E      48     169.64956
E      49     209.01645
E      50      48.37730
```

Descriptive (aka Summary) Statistics

- Statistics of *location*: mean, median, mode, etc....
- Statistics of *dispersion*: variance, standard deviation, range, etc. ...
- Statistics of other properties: skew, kurtosis, covariance, homoscedascity, ...

Choosing Summary Statistics

Ideal qualities:

- **Concise:** Easy to express and compare.
- **Stable**: Not likely to change much from one experiment to the next.
- **Revealing**: Doesn't hide important facts about the data sample.
- **Informative:** Supports insight.

Data Sample Notation

$$X = \{x_1, x_2, \ldots x_n\}$$

A sample of ***n*** measurements.

$$X^o = \{x_{(1)} \leqslant x_{(2)} \leqslant \ldots \leqslant x_{(n)}\}$$

The **order statistics** of the sample are the elements in nondecreasing order. Element $x_{(r)}$ with **rank** r is the r^{th}-smallest in the set.

$$x_{(2.5)} = \left(\frac{x_{(2)} + x_{(3)}}{2} \right)$$

If the rank is non-integral, average the two nearby order statistics.

Statistics of Location

The mean: Sum of the elements, divided by n

$$\overline{X} = \left(\frac{1}{n}\right)\sum_{i=1}^{n} x_i$$

The median: The *n/2* nd order statistic. The middle-most element.

$$\widehat{X} = x_{(n/2)}$$

The trimmed mean: Discard *t* elements from the top and bottom ranks, then compute the mean.

$$\overline{X^t} = \left(\frac{1}{n-t}\right)\sum_{i=t+1}^{n-2} x_{(i)}$$

Which one do you choose?

Looking at Distributions

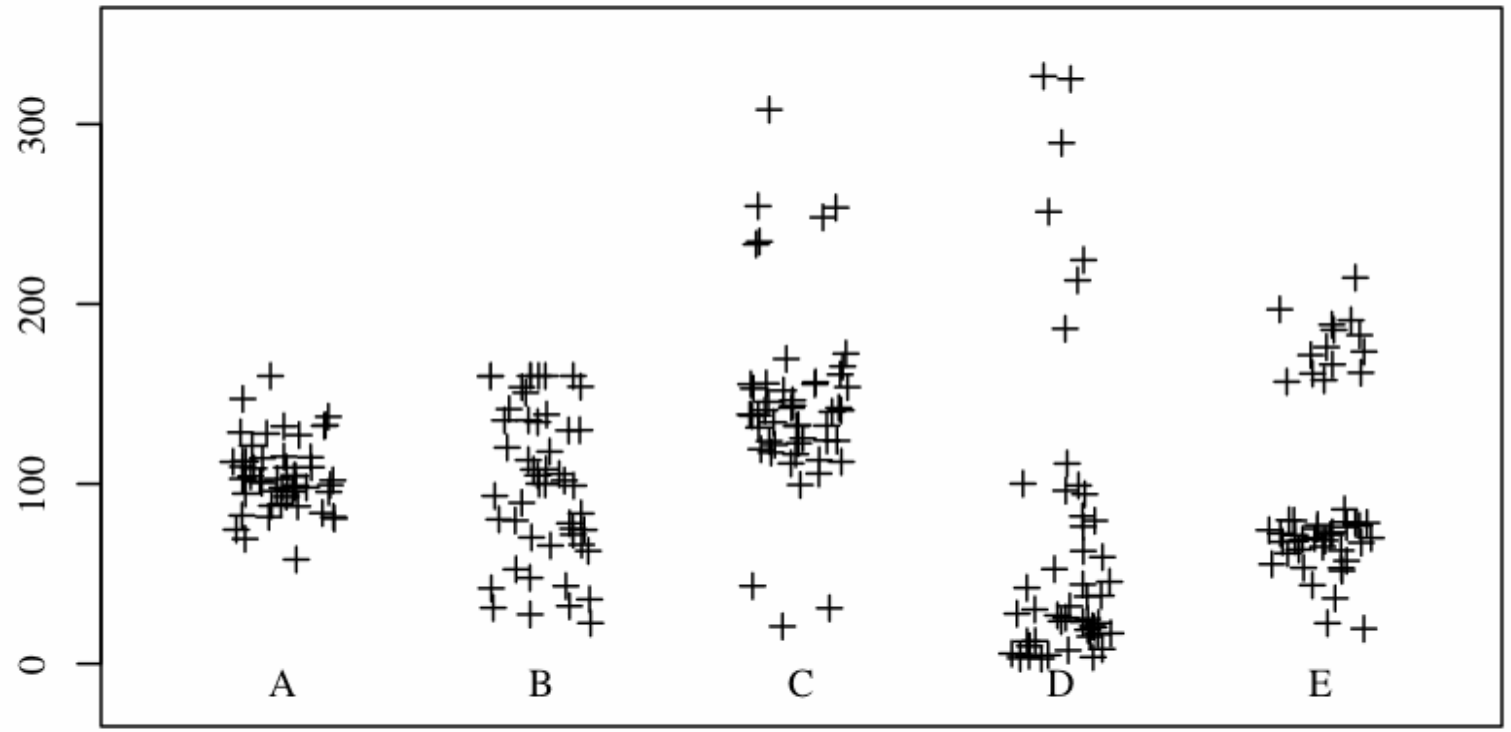
Good data:
Symmetric, no outliers, unimodal.

Bad data:

Outliers

Outliers and skew

Bimodal

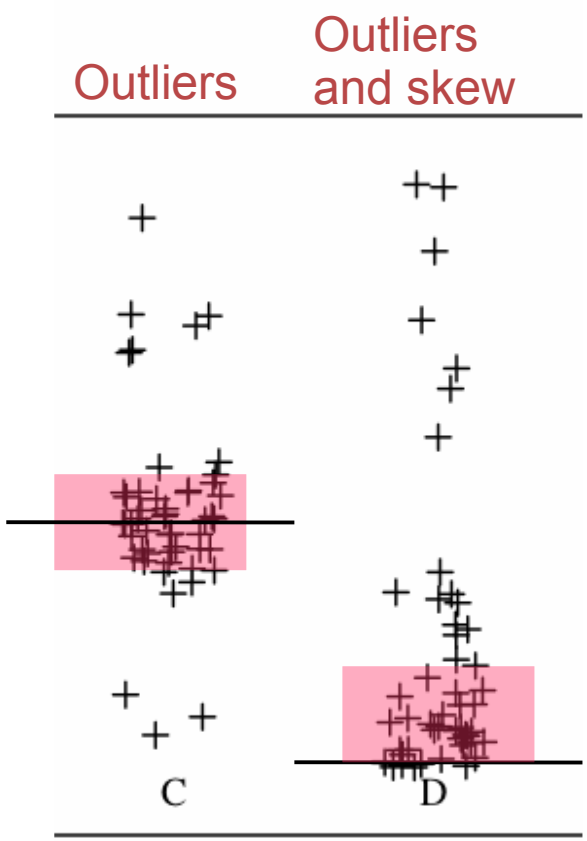


(This is a graphical technique called a jittered scatterplot.)

Choice of Statistic

The mean is sensitive to outliers. Trimmed mean and median are not.

The mean is drawn towards the long tail of a skewed distribution. The median is not. Trimmed mean depends on how much is trimmed.



Statistics of Location

- **Symmetric, unimodal, no outliers**: Any statistic works.
- **Outliers**: Don't use means -- not stable. Choose trimmed means or medians.
- **Skew**: Think carefully about the meaning of ``location'': median or mean? Maybe report both.
- **Bimodal, multimodal**: Treat each group separately if you can separate them. If you can't, don't summarize.

Statistics of Dispersion

- Variance
- Standard deviation
- Interquartile range
- Quartiles
- Hinges

$$Var(X)=\left(\frac{1}{n-1}\right)\sum_{i=1}^{n}\left(x_i-\overline{X}\right)^2$$

$$SD(X)=\sqrt{Var(x)}$$

$$IQR(X)=x_{(3n/4)}-x_{(n/4)}$$

$$Q(X)=[x_{(n/4)},x_{(3n/4)}]$$

$$H(X)=[x_{(1)},x_{(n/4)}.x_{(n/2)},x_{(3n/4)},x_{(n)}]$$

Which one to use when?

Looking at Distributions

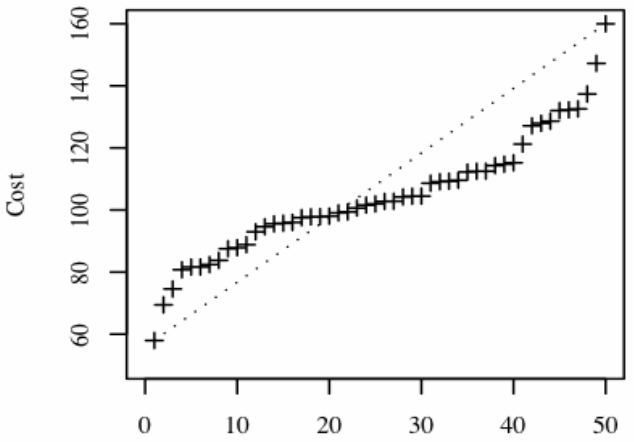
Empirical Distribution Plots (with sight lines) reveal:

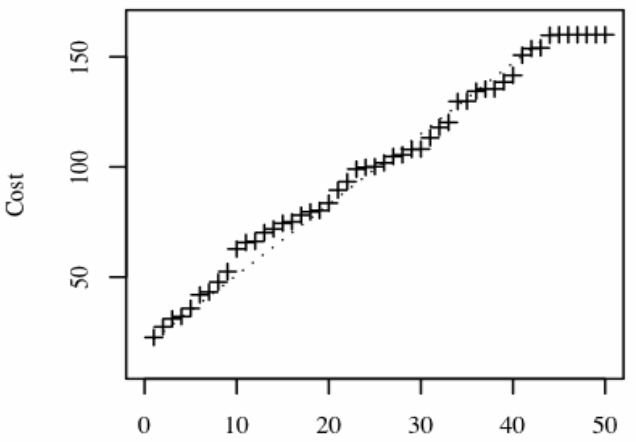
(a) is normal, mesokurtic

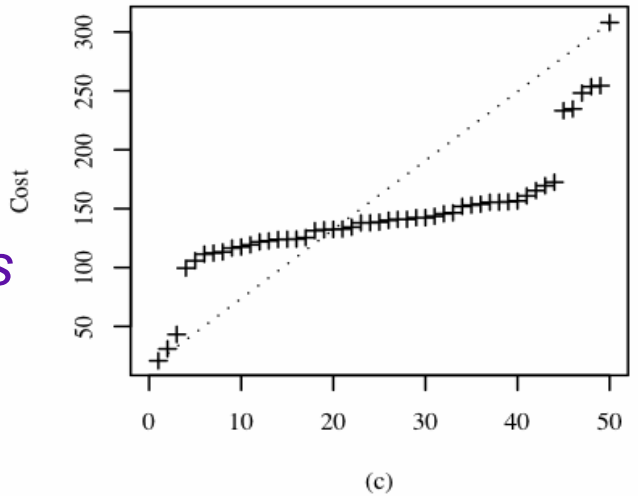
(b) is uniform + ceiling effect?

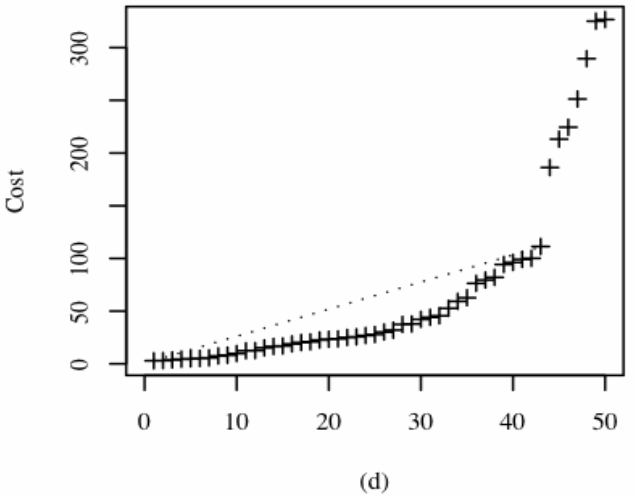
(c) has low kurtosis + outliers

(d) has skew + outliers.



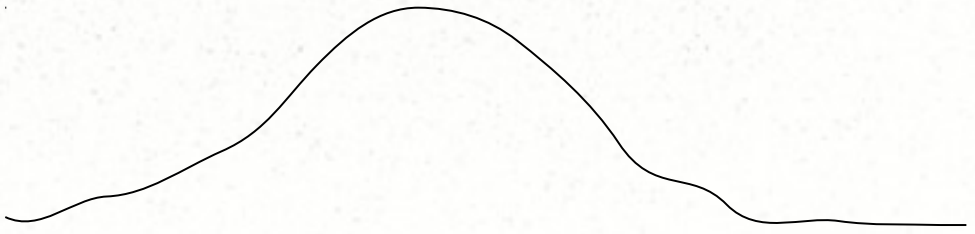






Kurtosis

- How flat or pointy is the distribution that produced this data?
- Kurtosis statistic $k = [-1 \ldots 0 \ldots +1]$
- Negative kurtosis $k < 0$. Fat tails associated with outliers, possibly bimodal.
- Mesokurtic $k = 0$. Normal-looking
- Positive kurtosis $k > 0$. Pointy distribution with thin tails.



Censored Data

- When a measurement is partially recorded, as an upper bound (right censoring) and/or lower bound (left censoring), *due to a limitation of the experimental apparatus*.
- **Ceiling effect:** For example, when all tests are stopped after a time limit is reached.
- **Random censoring**: When censoring is independent of the measurement, e.g. when some trials crash due to power failure.

Descriptive Statistics of Dispersion

- SD is on same scale as mean, but variance is a better estimator.
- The ***Empirical Rule*** uses SD: 68% of data is within μ +/- σ, 95% is within μ +/- 2σ, 99% of data is within μ +/- 3σ.
- **Normal data.** Any statistic ok, Empirical Rule holds.
- **Symmetric, no outliers**. Any statistic ok, but Empirical Rule fails.
- **Censored data**. Think carefully about whether to ***include, omit, or trim*** censored data. Usually ok to omit randomly censored points. (Any descriptive statistic is ok if its not really censored.)

- **Outliers**. Variance, st. dev, and range are not stable.
- **Skew**. Do not hide the asymmetry; use quartiles or hinges.

Boxplots

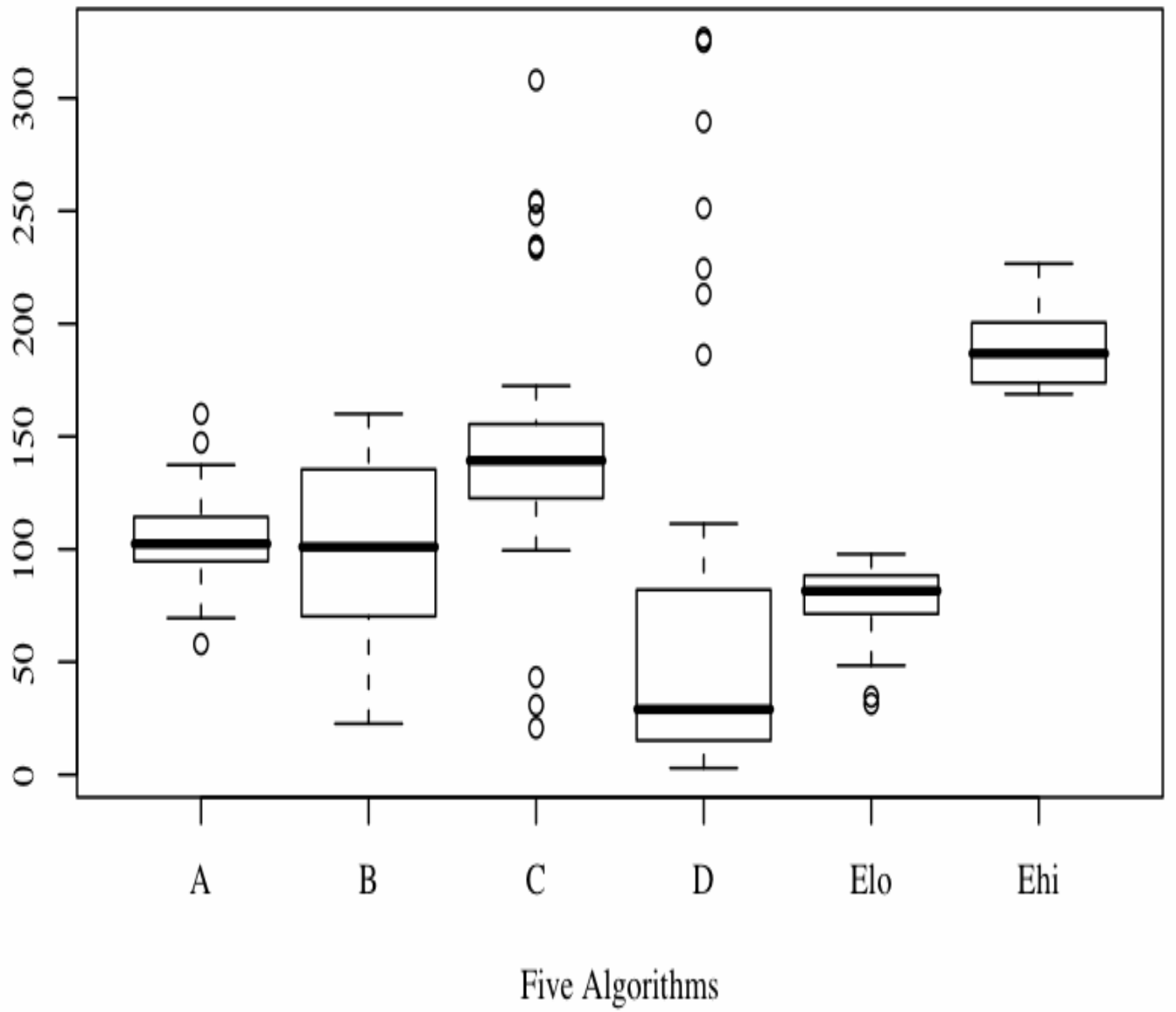
Visual Summaries of Hinge Data:

Bar: Median

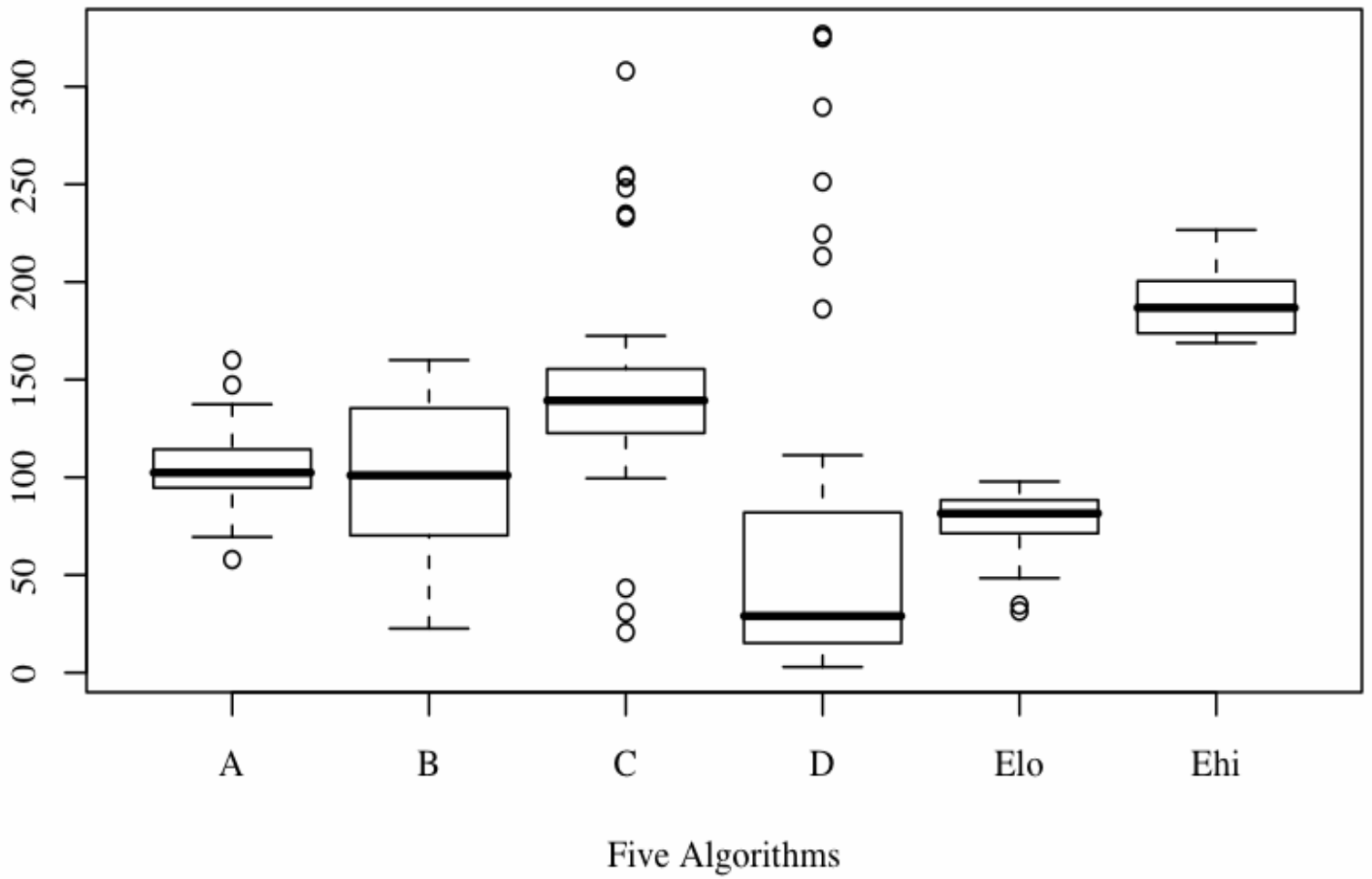
Box ends: Quartiles

Whisker ends: Q +/- 1.5*(IQR), or range.

Dots: Anything beyond whiskers: call them outliers if separated from whiskers.



What can we observe?



Choosing Summary Statistics

Ideal qualities

- **Concise:** Easy to express and compare.
- **Stable:** Not likely to change much from one experiment to the next.
- **Revealing**: Doesn't hide important observations.
- **Informative**: Supports insight.

Choosing Summary Statistics

- **Concise**: Prefer single numbers -- mean, median, variance, st. dev -- when data is well behaved.
- **Stable**: Avoid means, variance, st.dev, range (statistics that are sensitive to outliers), when outliers are present.
- **Revealing**: Use quartiles or hinges on skewed data.
- Separate multimodal groups when possible; otherwise don't summarize.
- Choose graphical summaries for more than, say, 5 numbers.
- Always report both ***location*** and ***dispersion***.

Multidimensional Data

Examples:

- Pairwise comparisons on identical inputs.
- Multiple input parameters & factorial designs.
- Measurements on multiple platforms.
- Multiple costs: e.g. time vs solution quality.
- ... when is it *not* a multidimensional analysis problem?

New Experiment

- Compare cost C of algorithms F and G.
- Three factors: Choice of *algorithm*, of *input class*, of *instance size*.
- Paired tests: same instances for both algorithms.
- Full factorial design, with some extra levels, some omitted design points, ...

New Experiment

- Measure costs of algorithms F and G,
- On 32 identical instances: 10 application, 20 random, 2 worst case.
- Application: 5 small, 5 big, n = 225 ... 931
- Random: 10 n=225, 10 n=931
- Worst case: 1 n=225, 1 n=931

$$F = \{f_1, f_2, \cdots f_s\}$$

Note n = instance size, not sample size.

$$f_i \Leftrightarrow g_i$$

Paired = both measured on the same instance.

New Questions

- Are costs of F and G ***correlated***?
- What input properties are most important to ***explaining*** performance?
- How do F and G compare, using paired instances (which algorithm is better, under what circumstances)?

Correlation between paired X and Y

- How well does x_i predict y_i?
- Does correlation depend on instance class, size, individual?
- We can compute a correlation coefficient r = *[-1 ... +1]*,
- But graphical analysis reveals more ....

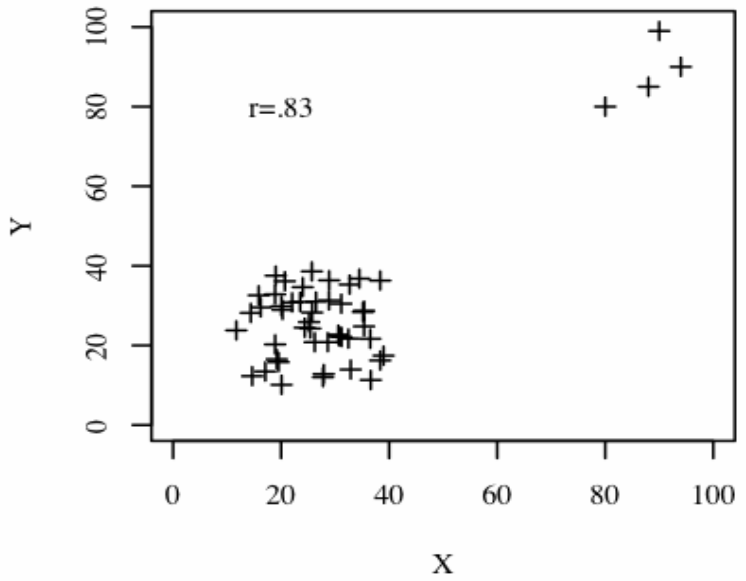
$$r = \frac{\sum_{i=1}^{s} (x_i - \overline{X})(y_i - \overline{Y})}{SD(X) \cdot SD(Y)}$$

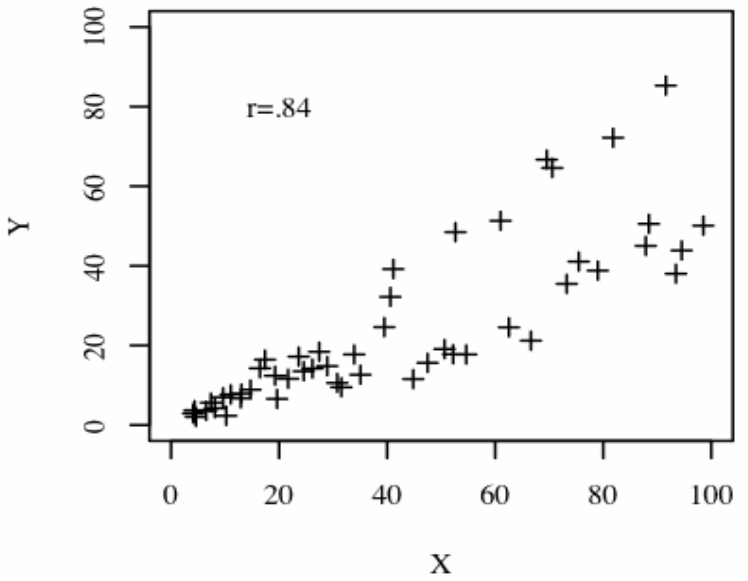
Correlation in X and Y

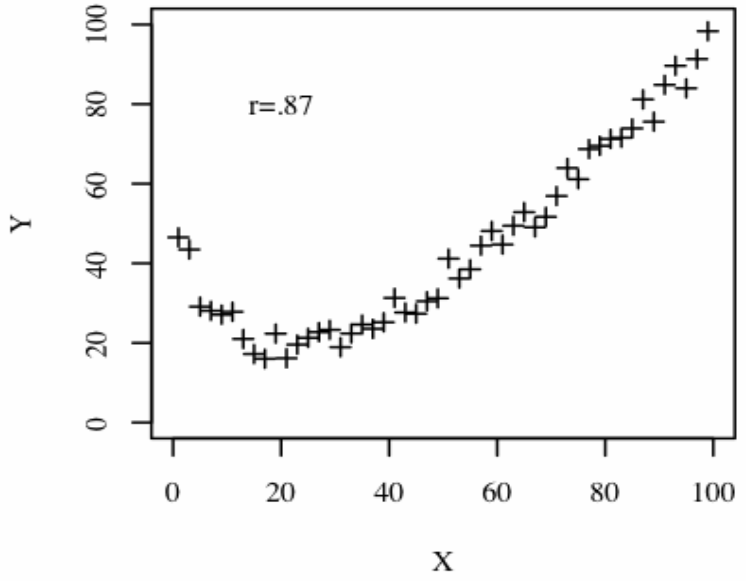
These are **scatterplots** of x_i vs y_i

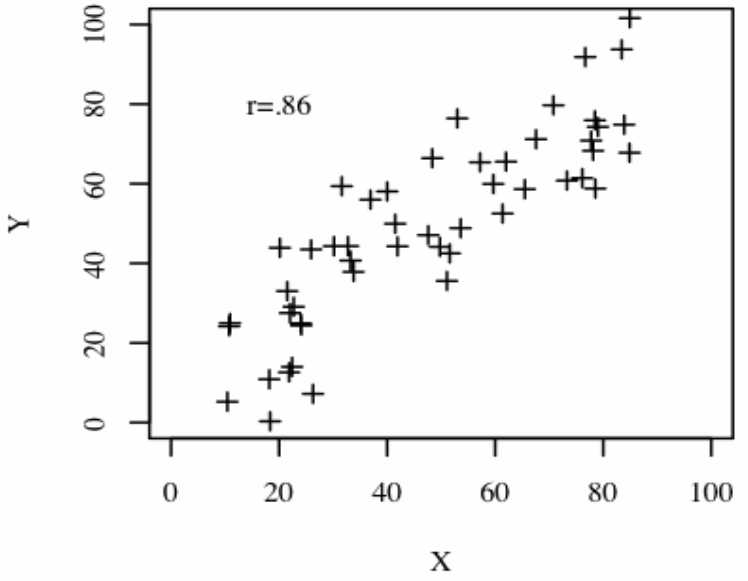
Correlation statistic *r* is between .83 and .87 here, indicating fairly strong positive correlation.

But the nature of the correlation is quite different in each case!









Correlation in F and G

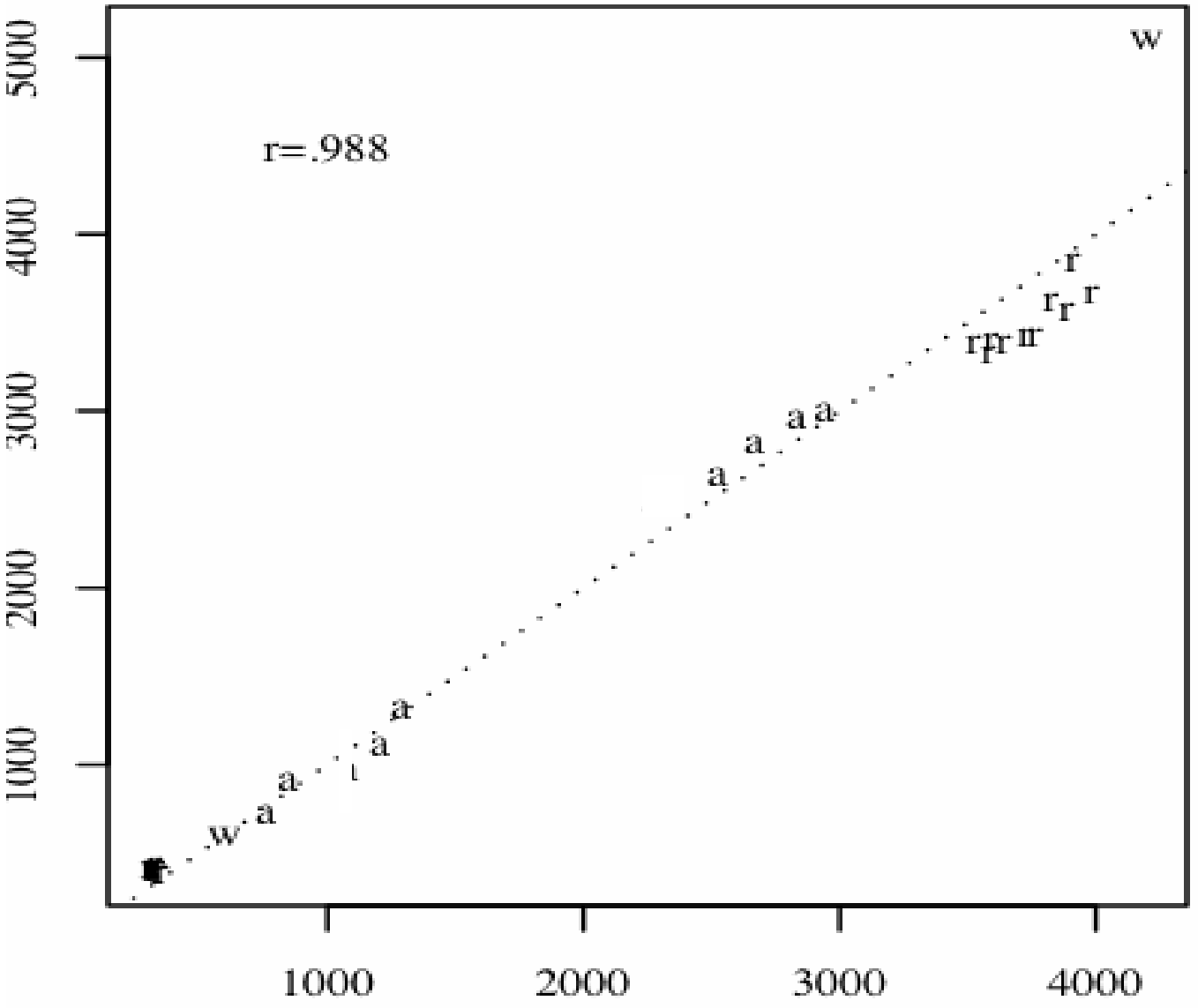
Scatterplot of (paired) F vs G

Coded by instance class, sight-line at *x=y*

Strong positive correlation, possibly by instance size.

Some positive correlation *by class.*

F < G on large random instances. F>G on large wc...



F vs G by Input Class

These are parallel coordinate plots.

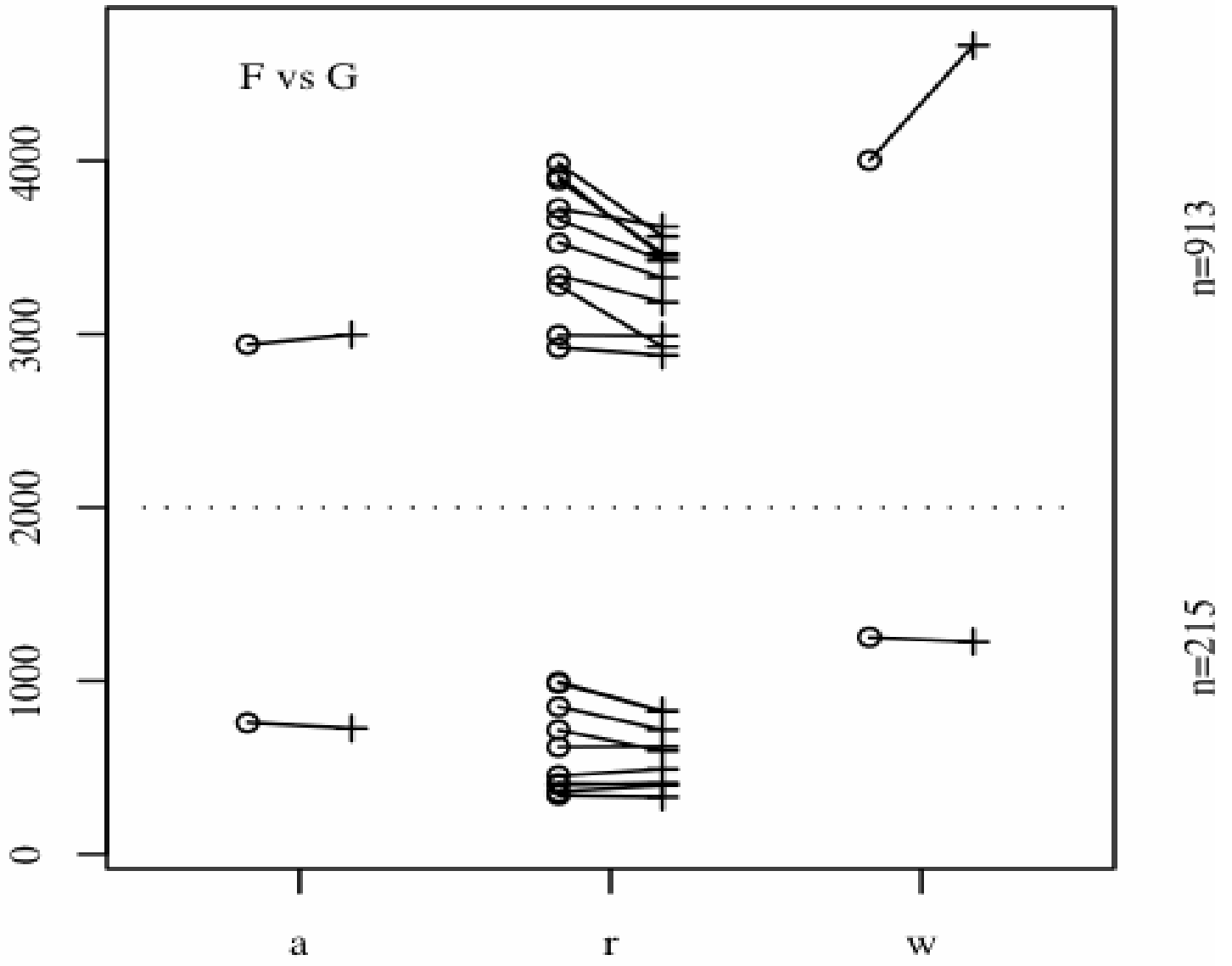
Each line connects same instance: *(f_i, g_i).*

Top: n=913
Bottom: n= 215

Comparison: look at slope.

Correlation: look for parallel lines.

What can you observe?



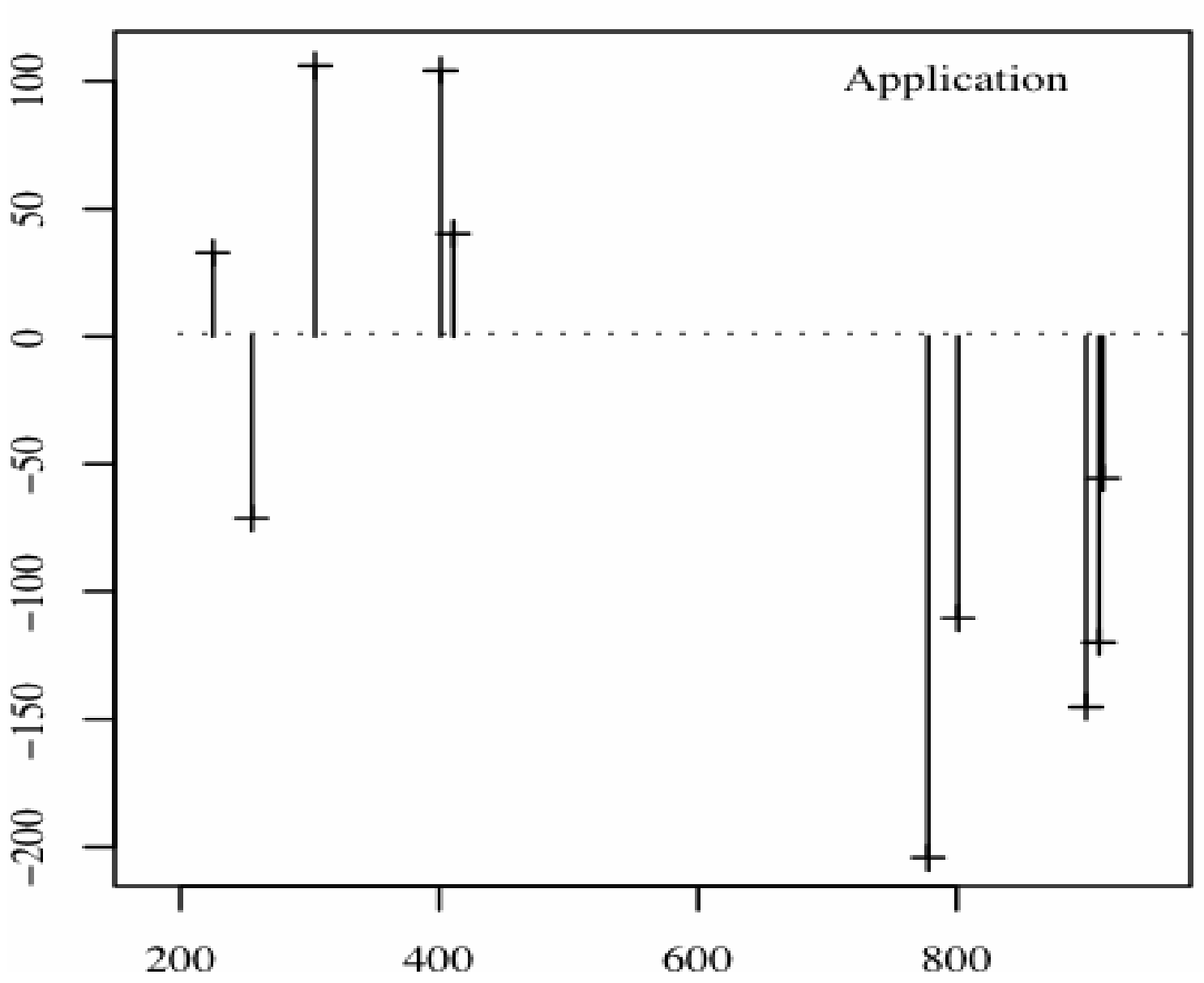
Application Data Only

This is a scatterplot of differences, with sight lines and a baseline.

x axis: *n*
y axis: *F – G*

Five small, five big instances.

What can you observe?

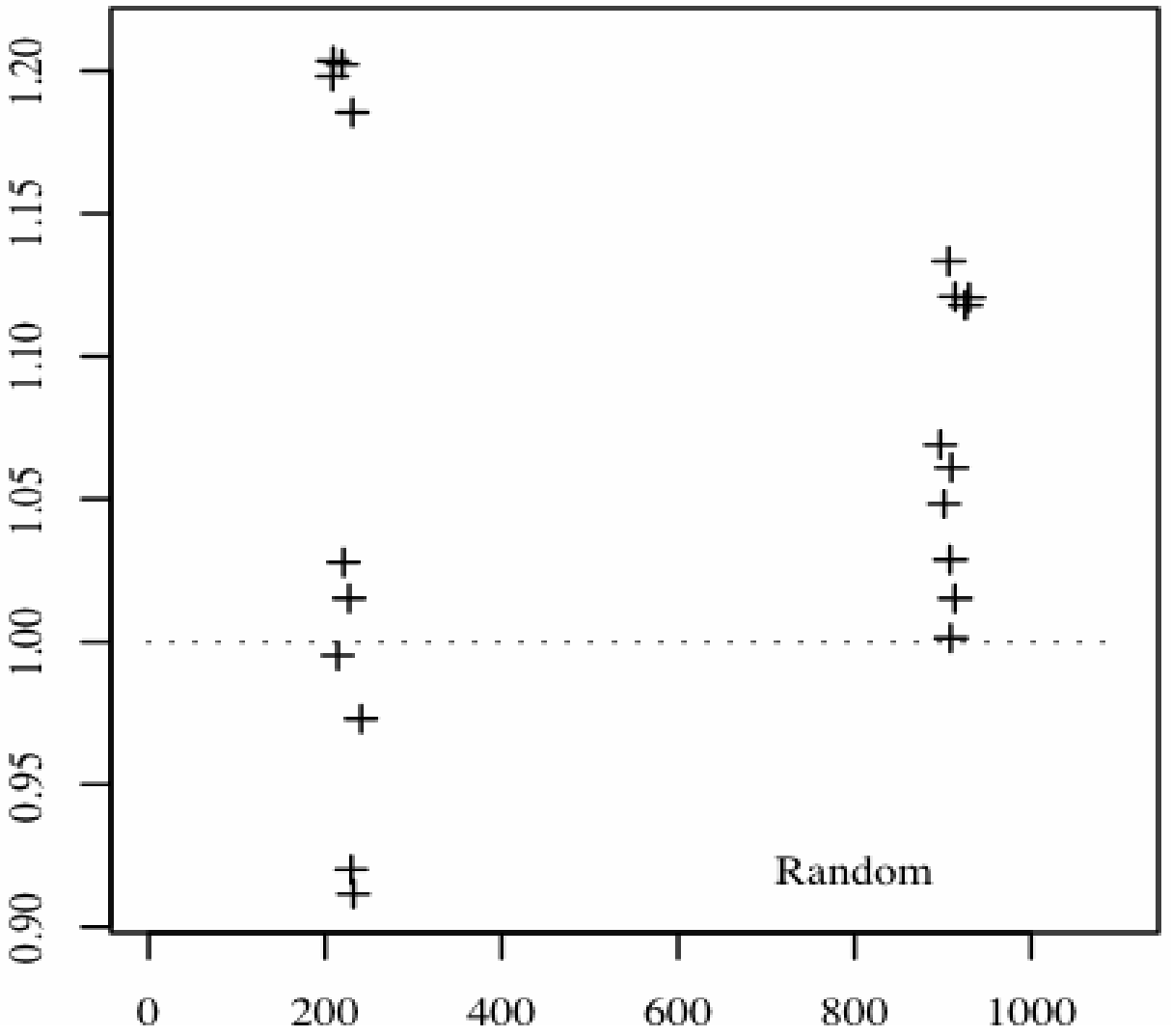


Random Data Only

A scatterplot of ratios, with a baseline.

x axis: *n, jittered*
y axis: *F / G*

What can you observe?



Observations: F vs G

- Strongest correlation is w.r.t instance size.
- For fixed *n*, Worst case instances cost highest, Application instances least, Random instances range between.
- F is generally better (diff <200) on large Application instances.
- F is generally worse (ratio <1.15) on large Random instances.
- F is better (4665 vs 1225) on large WC inst.
- Mixed results on small instances
- (Too bad I made it all up) ...

Review of Graphing Techniques

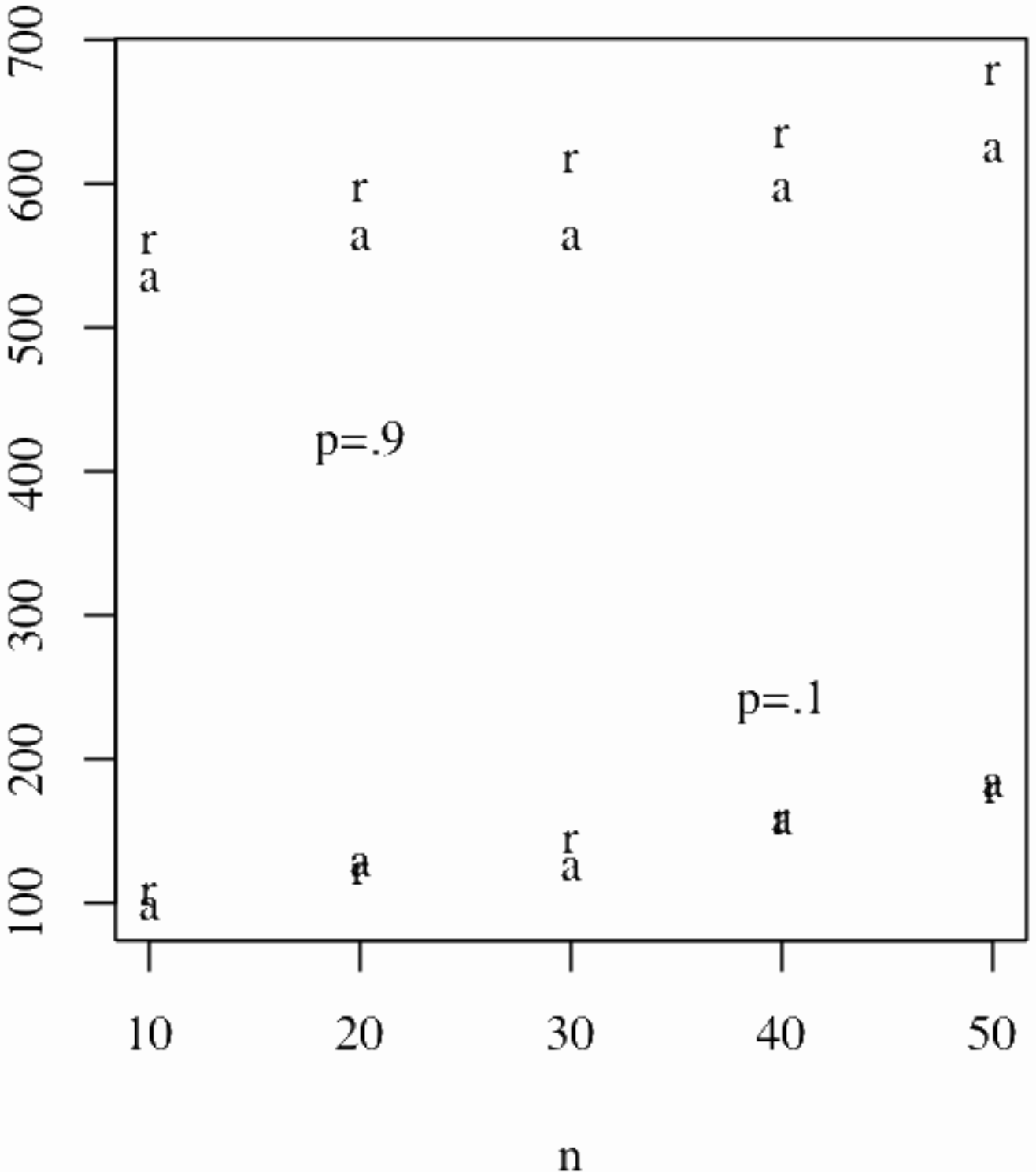
- Scatterplots for comparisons
- Jittering
- Empirical distribution plots
- Boxplots
- Scatterplots for correlation.
- Coded data
- Parallel coordinate plots.
- Plots of ratios
- Plots of differences.

When Graphs are Necessary

- When you have:
- Multiple factors, levels, and cost functions.
- Scores of trials per design point.
- Hundreds of data points.
- And you want to know:
- What factors are most important?
- How does cost depend on factors?
- What patterns can be discovered?

Multivariate Data

Prob	N	Class	Time	Solution
0.1	10	a	110.7537109161053	1102.48655381176
0.1	20	r	127.6420896160162	1169.825559515544
0.1	30	a	144.047945997189	1230.037344234463
0.1	40	r	168.7202684863741	1387.315513387292
0.1	50	a	166.3188842051977	1427.980180359228
0.1	10	r	99.74386802155193	1104.623310469199
0.1	20	a	110.815911303562	1102.192088427884
0.1	30	r	128.4191657401647	1256.033498184
0.1	40	a	151.6013840061871	1336.24623368607
0.1	50	r	186.4322009059989	1494.097819924586
0.9	10	a	516.870264067132	1940.465268650558
0.9	20	r	601.431178009019	2818.803698472815
0.9	30	a	577.6148758940964	3661.210315657438
0.9	40	r	636.372297681587	4598.610920137633
0.9	50	a	629.749199060474	5396.638455965651
0.9	10	r	584.0884648126813	1908.982443543109
0.9	20	a	529.6217152450284	2793.447280277583
0.9	30	r	601.1122116820361	3660.132302887245
0.9	40	a	586.8481878251291	4611.372430578932
0.9	50	r	657.7696450992082	5454.108832521787



Scatterplot
Y-axis: *Cost 1*
X-axis: *n*
Coded by input class, and by prob.

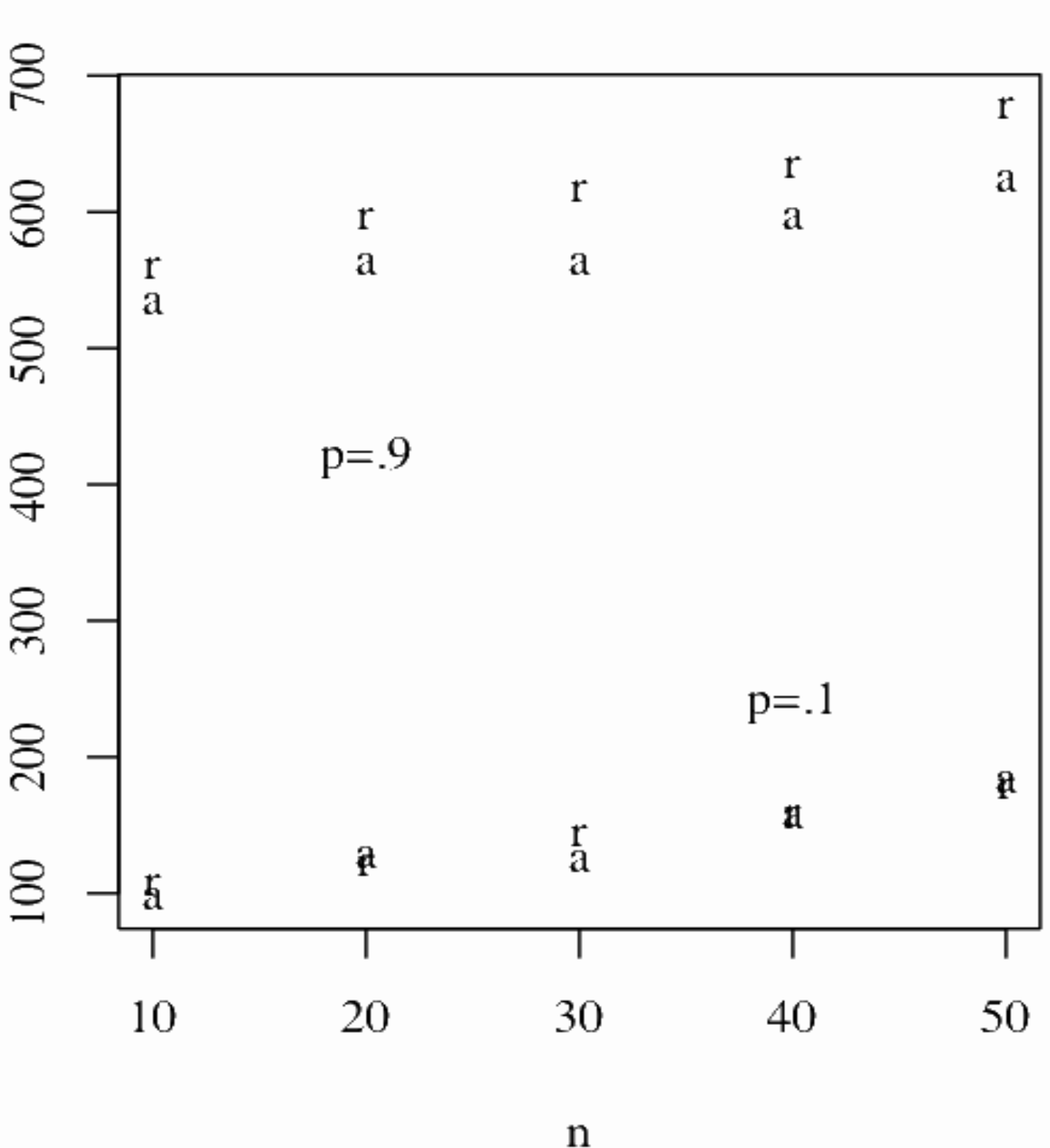
Observations:

Probability p is most important factor.

Class (a vs r) is least important.

Apps have less cost than Random when p=.9. Mixed when p=.1

Approx linear growth in n?

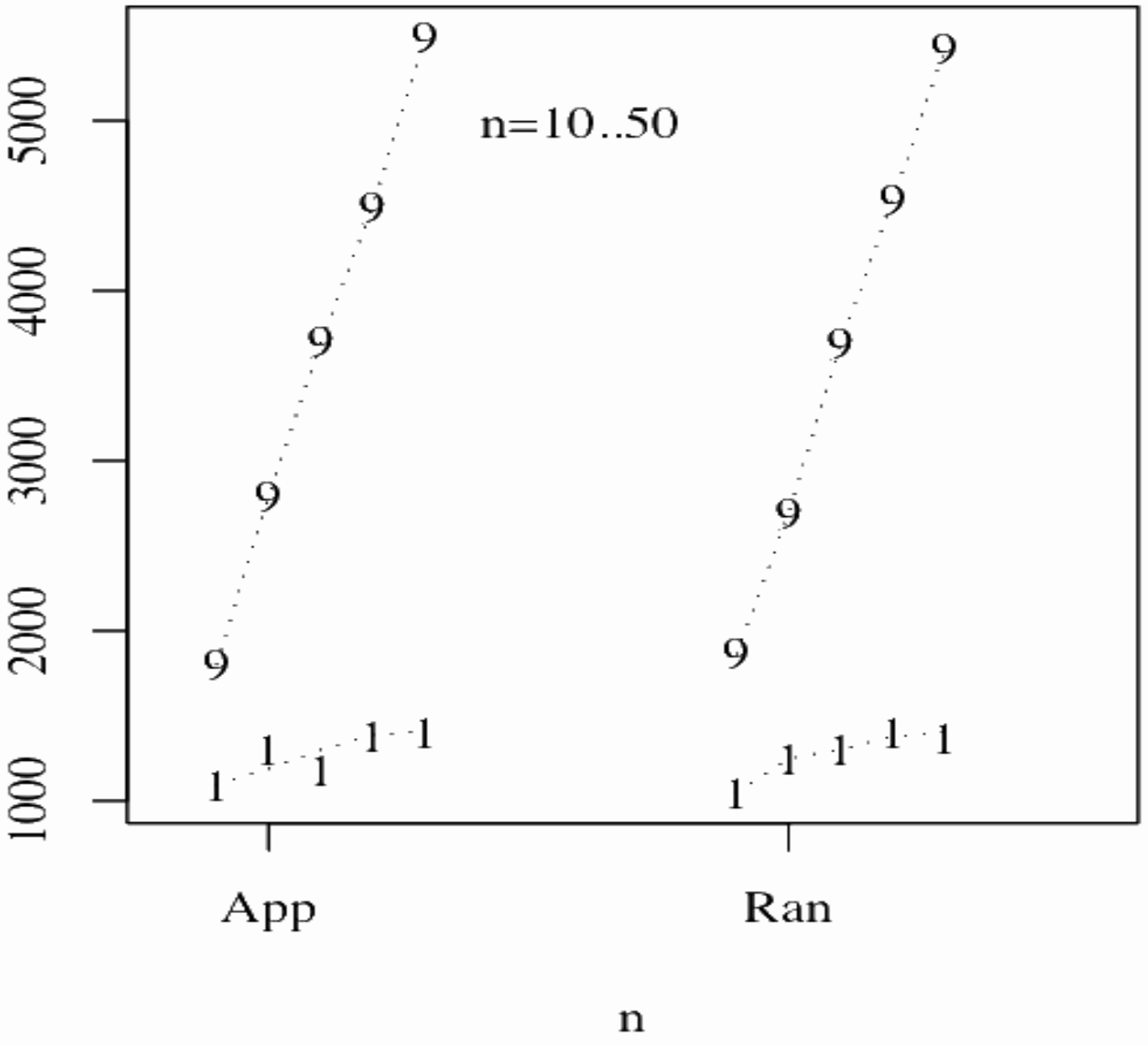


Visualization techniques:

Plot one factor on x-axis

Coding: label points according to other factors.

If points are visually separate, ok to use text labels.



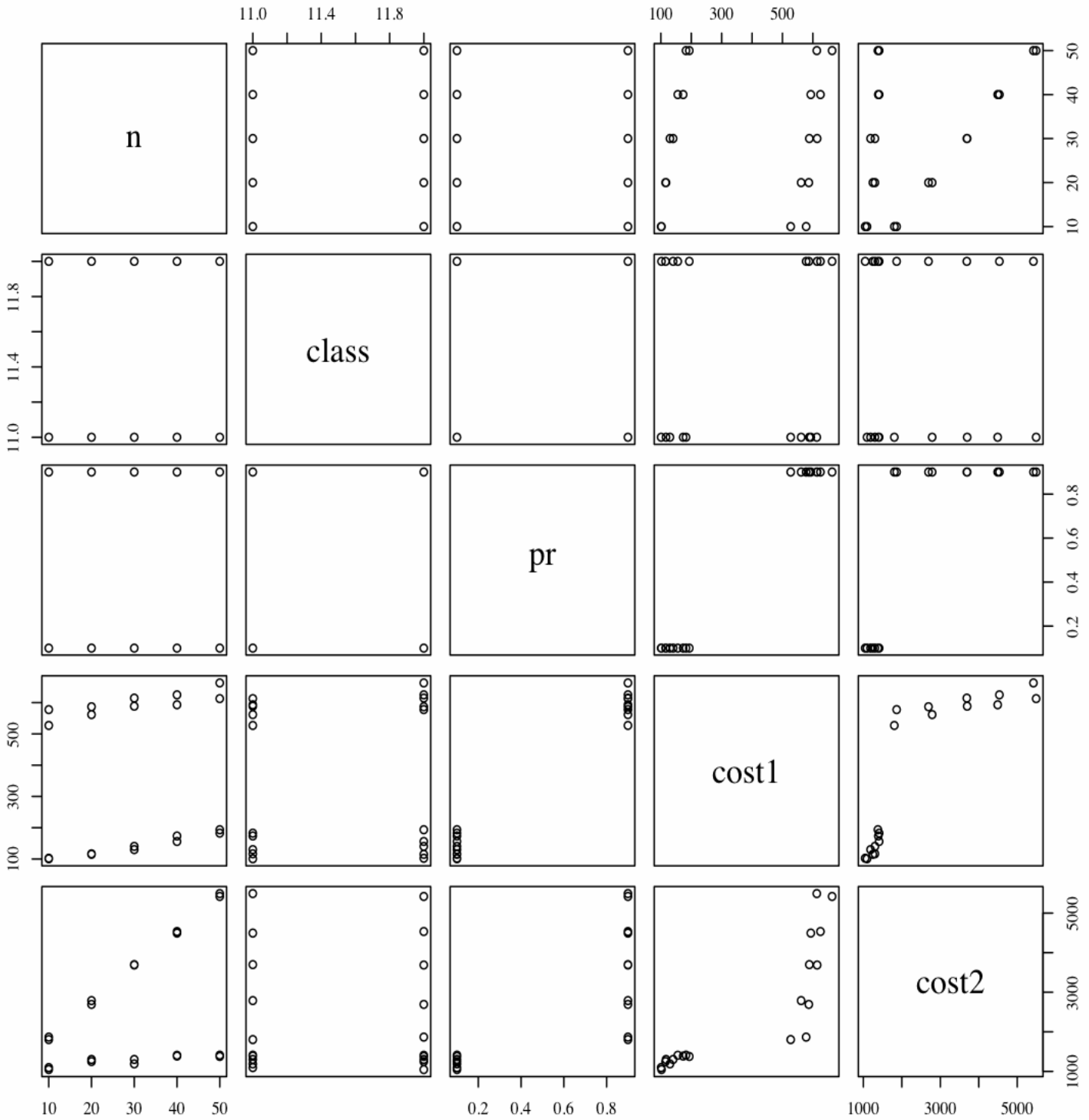
Y axis: *Cost 2*
X axis: *Class (0,60) + n*

Class is coded (qualitative -> numeric) and added to n (2 factors become 1).

Observations:

Cost 2 depends on both n and p.

Application remains similar to Random.

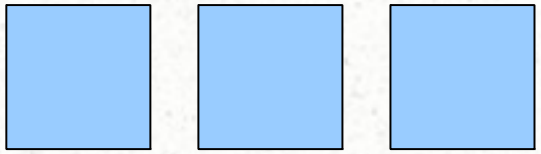


A Pairs Plot of all factors and costs.

Good for comparing multiple costs, non-orthogonal factors.

...

Multivariate Visualization Tips



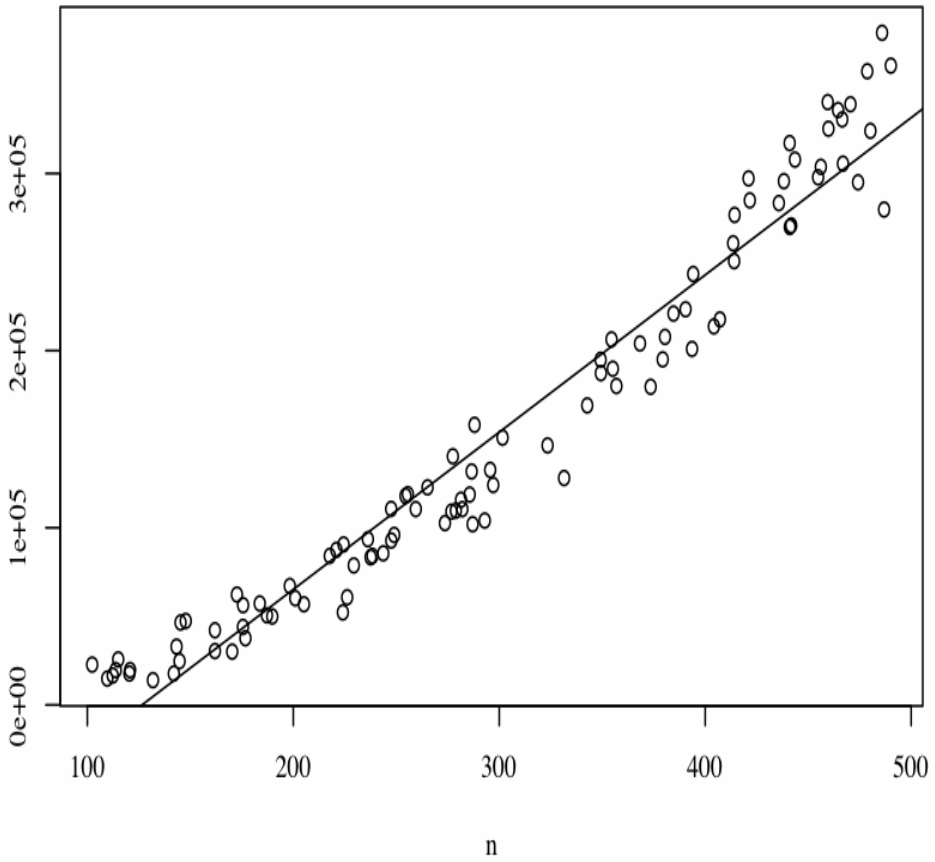
- Multiple panels
- Labels/colors on points
- Labels on regions of points
- One or more factors on *x* axis, via arithmetic. *n*100 + p*5*
- Code qualitative to quantitative (a,r) -> (10,60)
- Code quantitative --> qualitative cost1 --> (hi,md,lo)

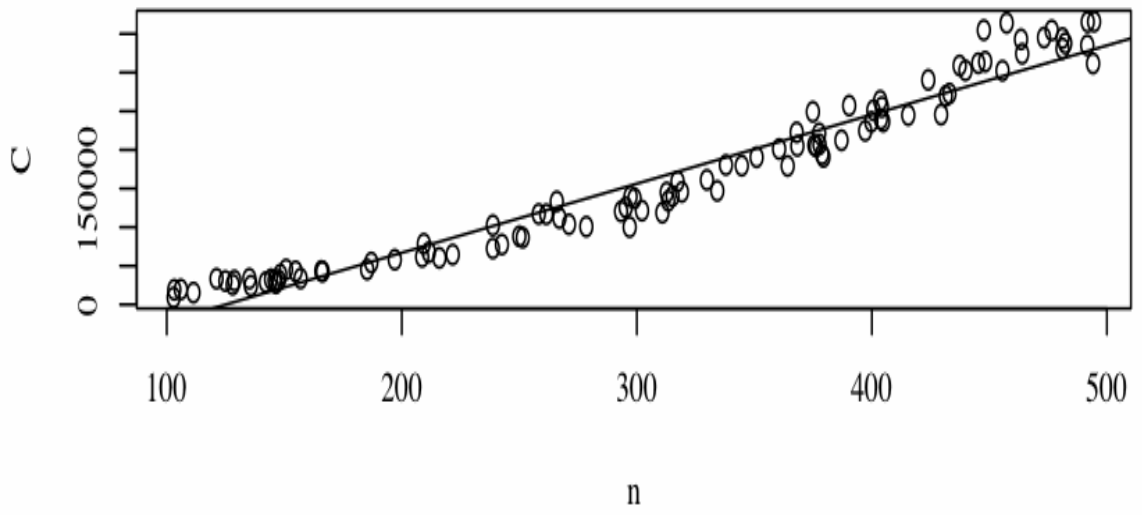
Questions About ...

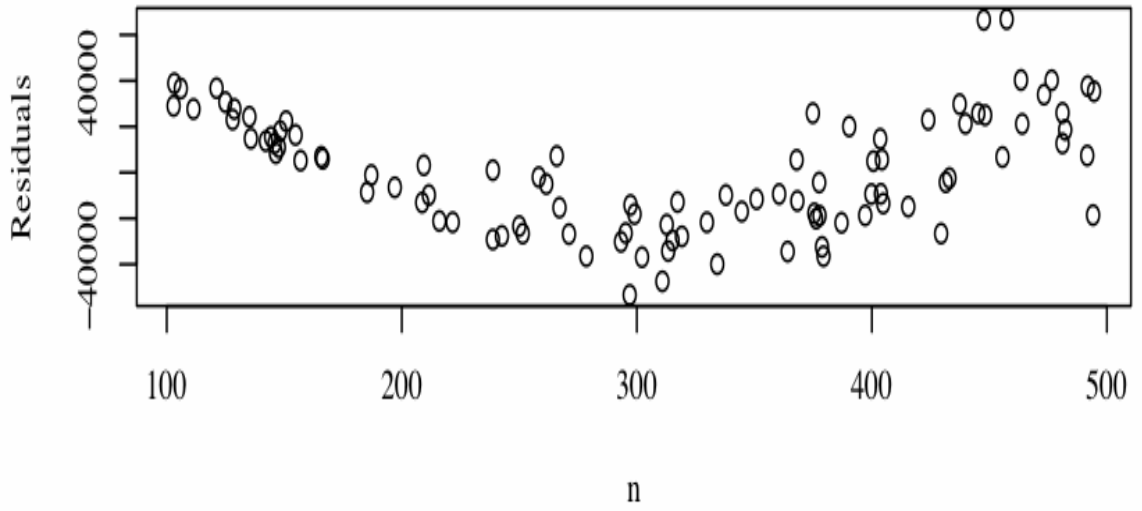
- Summary statistics for comparisons?
- Properties of distributions?
- Correlations?
- Visualizing multidimensional data sets?

Line Fitting = A Type of Summary

- Fit a convenien line to the data sample:
- $C = a\,n + b$
- Report the coefficients a,b
- *Concise!*







Residuals = Points – Line

Residuals curve upwards: the points are growing faster than the line.

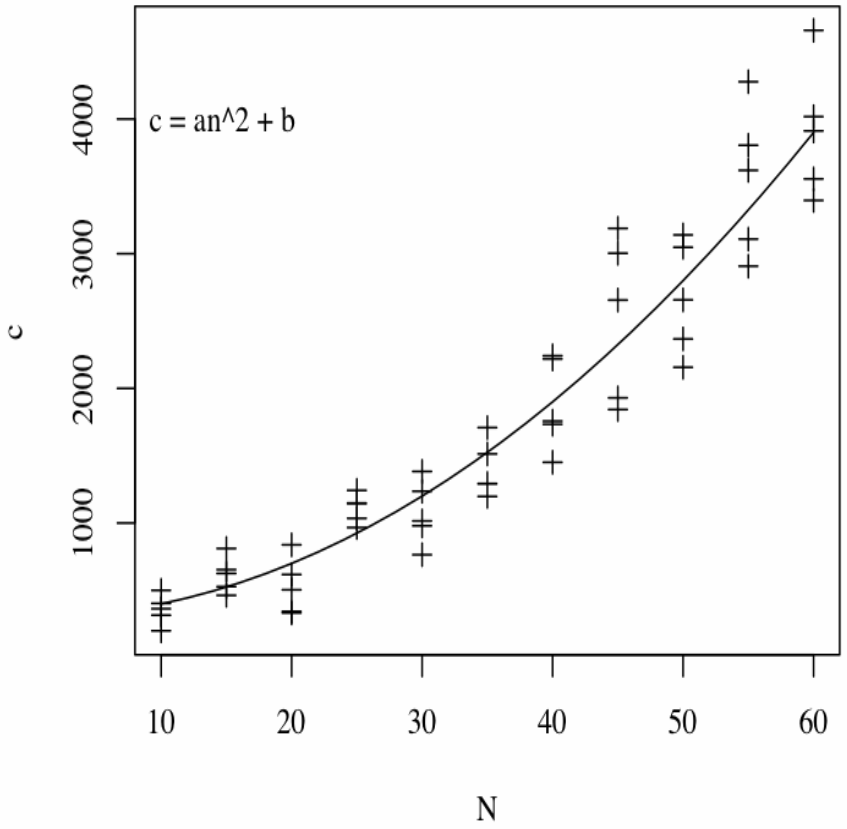
This line will underestimate at endpoints, overestimate at center.

Line Fitting Guidelines

- Find a convenient function to describe the relationship between x and y.
- Tukey's rule: Report both the *smooth* -- the general trend, and the ***rough***-- how data deviates from the general trend.
- (Analogous to location and dispersion.)

Two Approaches to Model Fitting

- How to find the right model if a straight line won't work.
- *Search-based methods.*
- *Transformation-based methods.*



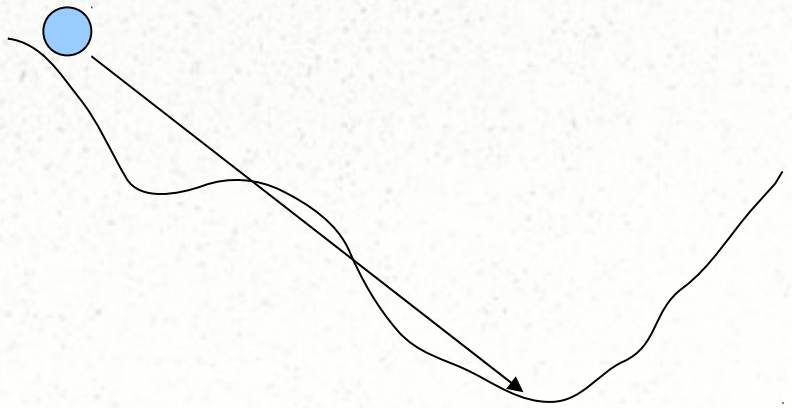
Search Methods

- Start with a model,
- Iteratively add / subtract terms, fit model to data,
- Stop when the Residual Sum of Errors (RSE) is minimized.
- *(What does this remind you of?)*

$$C = an^2 + bn\log(n) + cm + d$$

$$C = an^2 + bn\log(n) + cm^2 + d$$

$$C = an^2 + bn + cm^2 + d$$



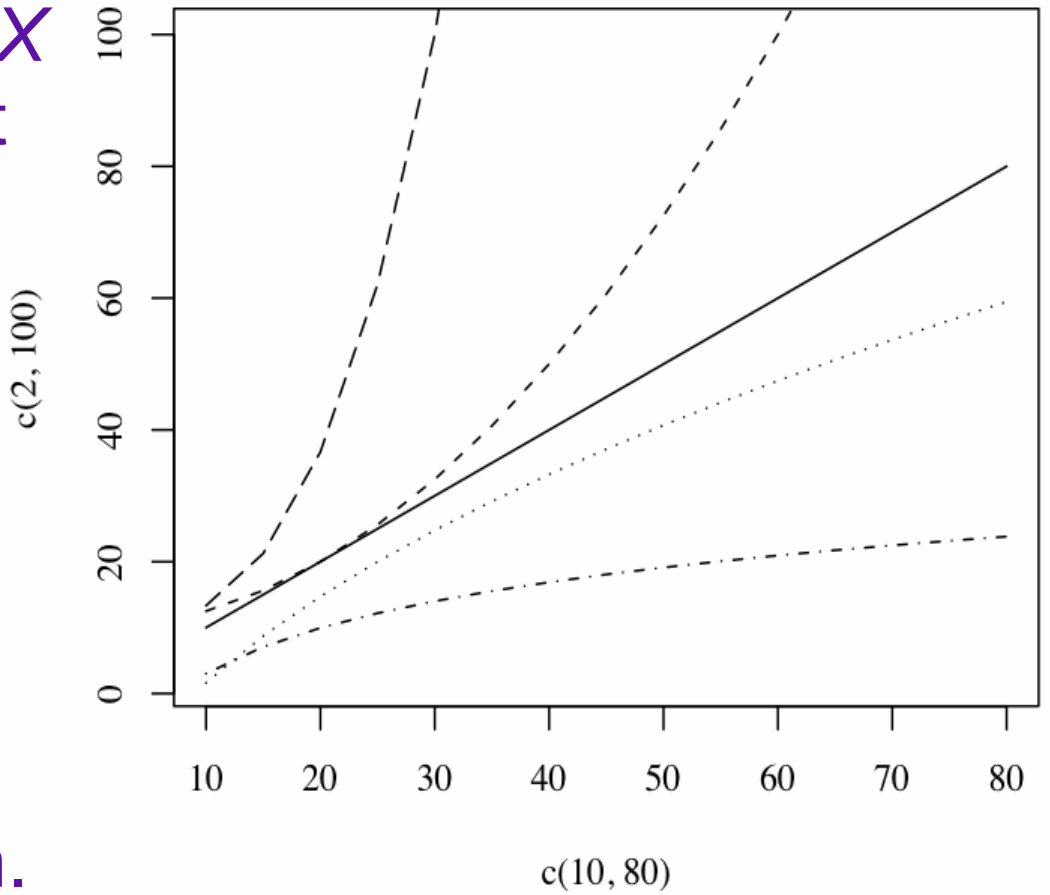
Search-Based Model Fitting

- Hill climbing with a simple step function and a sketchy objective function.
- (Could use some help from heuristic search experts!)

- RSE does not distinguish (leading terms of) models.
- No stopping criteria.
- Results are usually an artifact of stepping rule.

Transformation Methods

- Apply a function to *X* or *Y* (or both), to get a straight line.
- Apply linear regression to *(X', Y')* *in the transformed domain.*
- Extrapolate conclusions back to original *X, Y* domain.



Transformations to Straighten a Curve

- Power Law: plot on a log-log scale.
- Tukey's Ladder of re-expression: a sequence of functions to try.
- Box Cox transformation: formalizes Tukey's Ladder.

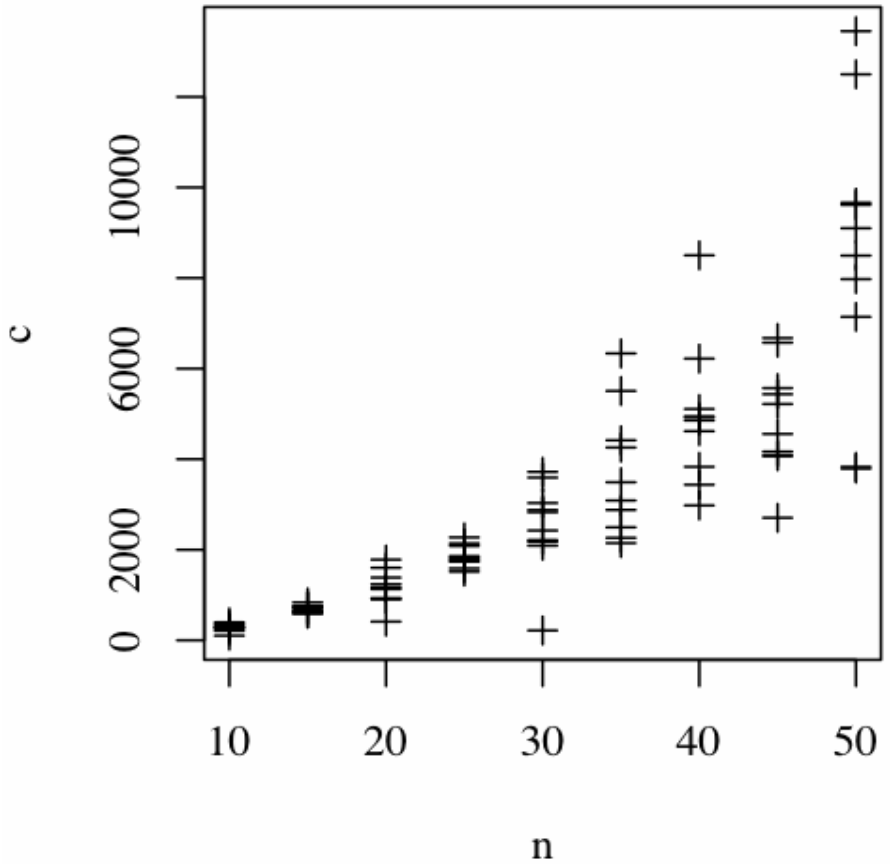
Power Law

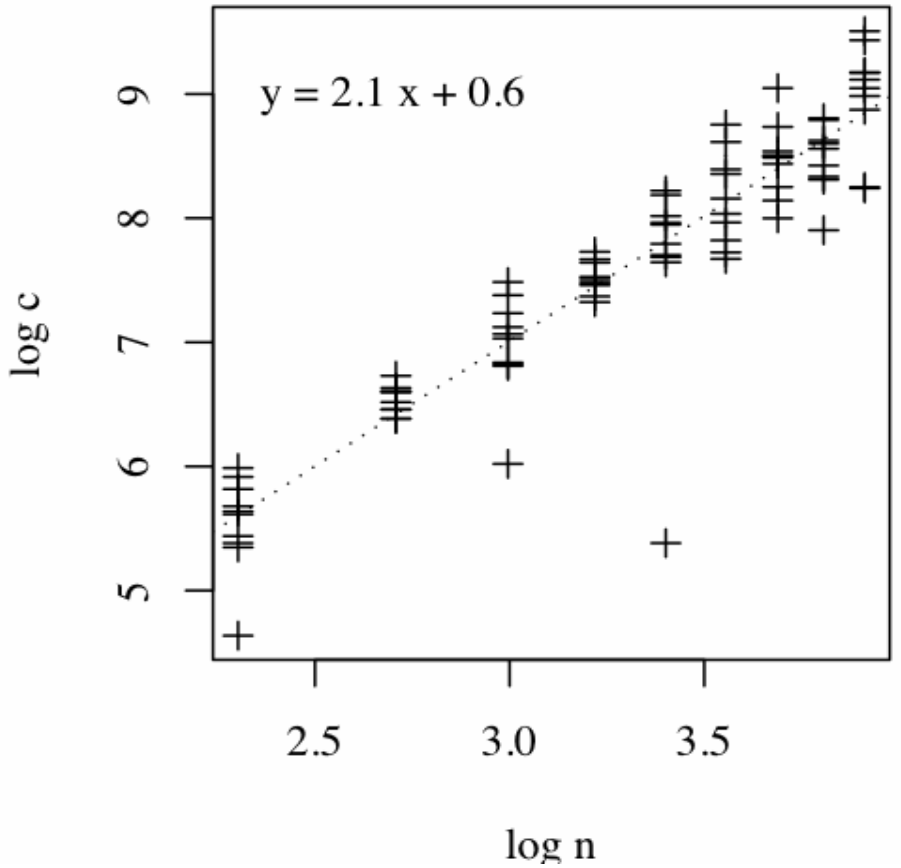
- Y' = log(Y)
- X' = log(X)
- One term power curves are linear on log-log scale.
- Degree in original scale = slope in log log scale.
- Low order terms mean *(X',Y')* converges asymptotically to a line.

$$y = a\,x^{b}$$

$$\log(y) = b\log(x) + \log(a)$$

Power Law





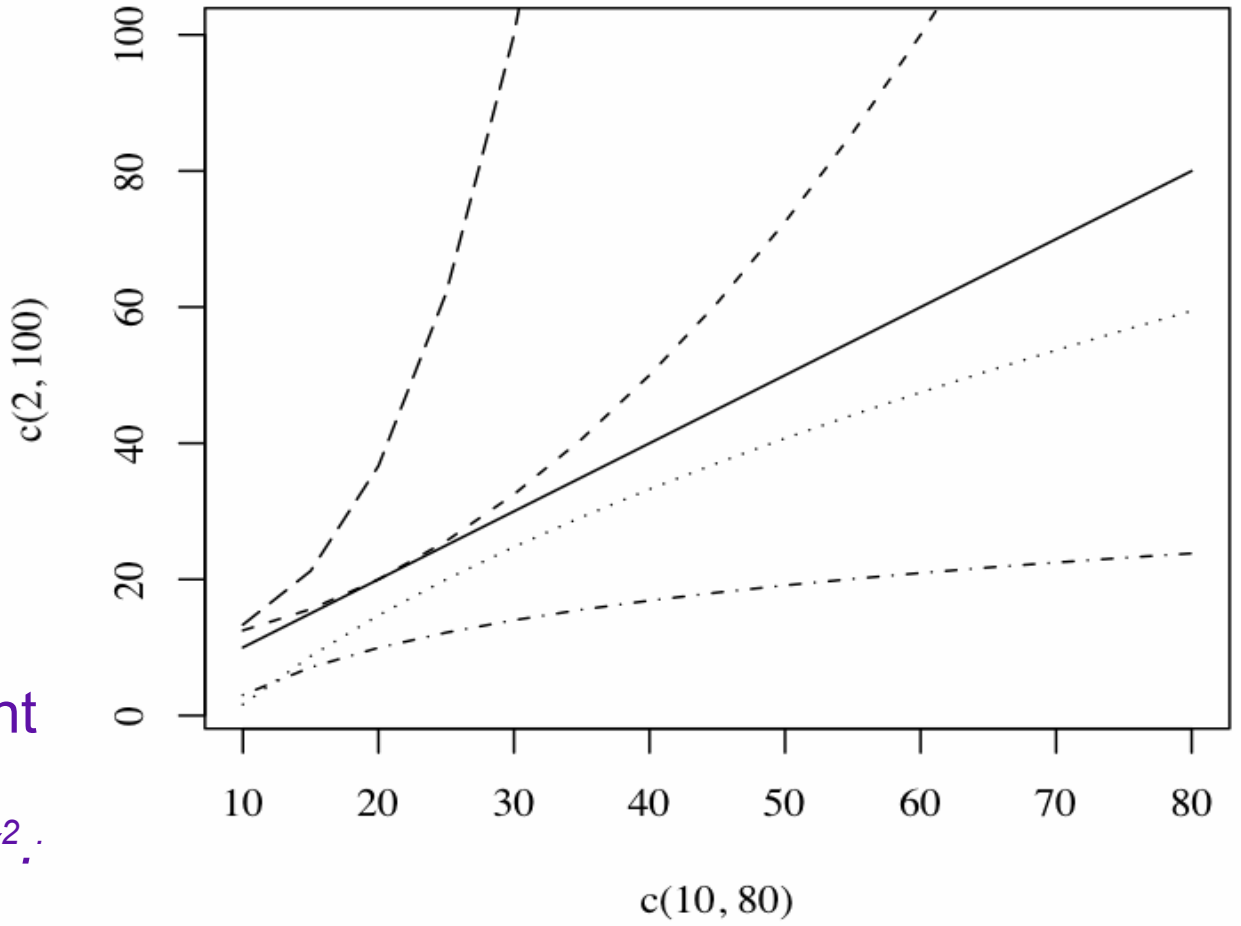
Conclude: c grows approximately as $(e^{0.6})\ n^{2.1}$

Note transformation also stabilizes variance n.

Tukey's Transformation Ladder

Apply transformations according to a *ladder of choices*:

$$y' = y^2$$
$$y' = y^{1/2}$$
$$y' = \log(y)$$
$$y' = 1/y$$
$$y' = 1/y^2$$



Until you see a straight line. If sqrt(y) is straightest, then $y = x^2$.

Transform x, or y, or both .

Box Cox Transformation

- Formalizes Tukey's ladder, using parameter λ.
- Allows direct comparison of residuals from regression fits at different scales.
- *(M(y)* is the geometric mean of sample y.)

$$y_i^{\lambda} = \frac{y_i^{\lambda} - 1}{\lambda M(y)^{\lambda - 1}}, \ \ if\ \lambda \neq 1$$

$$y_i^{\lambda} = M(y) \log(y_i), \ \ if\ \lambda \neq 1$$

$$M(y) = (y_{1,} \ldots y_n)^{1/n}$$

Data Transformations

- A generally useful tool in data analysis:
- Apply transformation,
- Analyze data in transformed domain,
- Transform conclusions back to original domain.

- *To straigthen a curve.*
- *To normalize scale when* $x_{(1)} << x_{(n)}$.
- *To make variance constant throughout range.*
- *To make skewed data symmetric.*

Any Questions About ...

- Fitting lines to data?
- The smooth and the rough?
- Fitting nonlinear models to data?

Statistical Methods for Algorithm Evaluation – In Three Parts

I. **Control = Responsibility: How not to do it.** Avoiding naive but common mistakes.

II. **Control = Power: Before the experiment.** DOE, planning experiments, variance reduction techniques.

III. **Control = Choice: Statistics and data analysis.** The right tools for unusual jobs.

Tools and References

- All graphs were plotted using R (aka Gnu S): free software environment for statistical computing and graphics.
  Visit: www.r-project.org
- Books on statistics and data analysis ...
- Papers on applications to algorithmic and computational problems ...

Some Useful Books

- Cohen, *Empirical Methods for Artificial Intelligence*
- Baron, *Probability and Statistics for Computer Scientists*
- Tukey, *Exploratory Data Analysis*
- Chambers, Cleveland, Kleiner, Tukey, *Graphical Methods for Data Analysis*
- Cleveland, *Visualizing Data*
- McGeoch, *Guide to Experimental Algorithmics*, Cambridge Press, to appear, 2011.

Some Papers on Methodology

- CCM, `` Analyzing Algorithms by Simulation: Variance Reduction Techniques and Simulation Speedups," *ACM Computing Surveys*, June 1992.
- D.S. Johnson, ``A theoretician's guide to experimental analysis of algorithms," in *Data Structures, Near Neighbor Searches, and Methodology, Fifth and Sixth DIMACS Implementation Challenges,* Goldwasser, Johnson, McGeoch, Eds.

Questions about ....

- anything?

Thanks for Your Attention!

Inferring Asymptotic Bounds

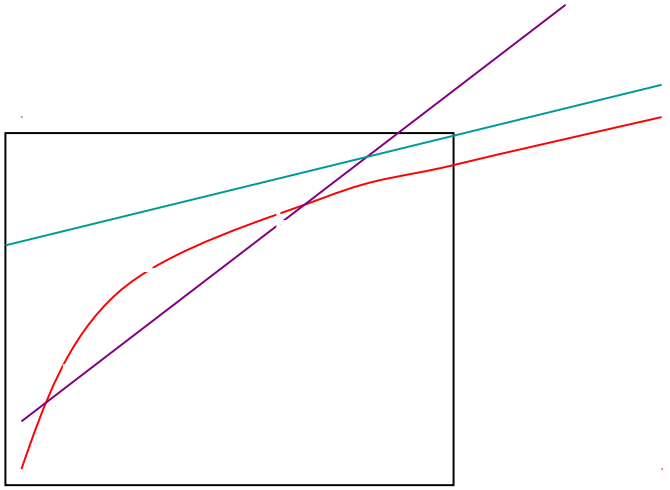
Theory: Any finite sample of data may come from an arbitrarily high-order function. Therefore any big-oh bound you find could be wrong.

Practice: Given dataset D from unknown function f(n), evaluate guess g(n) like this:

Does *f(n)/g(n)* tend to unbounded, positive constant, or zero? *r*

Does *f(n) - g(n)* decrease, then increase?

What is the asymptotic slope of log *n vs* log*(f(n))?*



What doesn't work:

- Classic regression analysis, function finding.
- Curve-fitting is not the same as curve-bounding!

What is Experimental Algorithmics?

Experimental analysis: Using computational experiments to predict algorithm performance in new scenarios. We prefer:

Predictions based on input parameters and properties. Functional descriptions of the relationship between input properties and algorithm performance.

Predictions that are platform independent, yet more realistic than the RAM model.

Experiments are needed to develop those realistic **models of computation**.

Algorithm engineering. A systematic process for bringing abstract algorithms to production-quality implementations, with an emphasis on efficiency. This process relies on computational experiments, to produce an efficient and well-understood product.

Experimental Algorithmics

Johnson, *Theoretician's Guide*

1. Perform newsworthy experiments.
2. Tie your paper to the literature.
3. Use instance testbeds that support general conclusions.
4. Use efficient and effective experimental designs.
5. Use reasonably efficient implementations.
6. Ensure reproducability
7. Ensure comparability.
8. Report the full story.
9. Draw well-justified conclusions and look for explanations.
10. Present your data in informative ways.

Contributions of Experimental Algorithmics

- Cache-efficient analysis of algorithms and data structures; new I/O models. (*LaMarca and Ladner, many others)*
- Identification of phase transitions for some NP-Hard problems: simple parameters that characterize hard/easy instances. (*Culberson, Cheeseman et al. , many others*)
- Highly tuned, well understood data structures and algorithms for particular problems, together with guidelines about which works best for which types of instances (TSP *Johnson + many others*)
- Well tested and reliable code libraries. (LEDA *Mehlhorn + others,* Stanford GrafBase *Knuth + others*
- Authoritative repository sites for codes, test suites, and research results (*Culberson's* Graph Coloring page, *Mike Trick's* page, *Skiena's* OR Stony Brook Algorithm Repository)

Contributions of Experimental Algorithmics

Demetrescu and Italiano, *What can we learn from experimental algorithmics?*

- Defining a clear methodology for algorithm engineering.
- Creating software support for implementation, debugging, evaluation of algorithms.
- Identifying useful input test sets and generators.
- Providing standard, documented libraries with efficient implementations of data structures and algorithms.
- Performing empirical studies to identify best ideas for given applicaitons: identifying algorithm separators; evaluating heuristics; characterizing asymptotic behavior; understanding parallel speedups; understanding effects of memory heirarcy and communication on performance; predicting performance; finding bottlenecks in real implementation
- Helping the practitioner.