The Endeavour

http://www.johnd-cook.com/blog

Unix doesn’t follow the Unix philosophy

The Unix philosophy is a noble idea, but even Unix doesn’t follow it too closely. The Unix philosophy, as summarized by Doug McIlroy, says

1. Write programs that do one thing and do it well.
2. Write programs to work together.
3. Write programs to handle text streams, because that is a universal interface.

Here is an example from ***James Hague*** where the first point has been strained.

> The UNIX `ls` utility seemed like a good idea at the time. It's the poster child for the UNIX way: a small tool that does exactly one thing well. Here that thing is to display a list of filenames. But deciding exactly what filenames to display and in what format led to the addition of over 35 command-line switches. Now the man page for the BSD version of `ls` bears the shame of this footnote: "To maintain backward compatibility, the relationships between the many options are quite complex."

James Hague gives this as only one small example of how programmers have allowed things to become unnecessarily complicated. He concludes

> We did this. We who claim to value simplicity are the guilty party. See, all those little design decisions actually

> matter, and there were places where we could have stopped and said “no, don’t do this.” And even if we were lazy and didn’t do the right thing when changes were easy, before there were thousands of users, we still could have gone back and fixed things later. But we didn’t.

He’s right, to some extent. But as I argued in ***Where the Unix philosophy breaks down***, some of the growth in complexity is understandable. It’s a lot easier to maintain an orthogonal design when your software isn’t being used. Software that gets used becomes less orthogonal and develops diagonal shortcuts.

Why does `ls` have dozens of tangled options? Because users, even Unix users, are not overly fond of the first two points of the Unix philosophy. They don’t want to chain little programs together. They’d rather do more with

the tool at hand than put it down to pick up new tools. They do appreciate the ideal of single-purpose tools that work well together, but only in moderation.

I agree that “all those little design decisions actually matter, and there were places where we could have stopped and said ‘no, don’t do this.’” Some complexity has come from a lack of foresight or a lack of courage. But not all of it. Some of it has come from satisfying what complex humans want from their software.

Related post:

100x better approach to software?

First changes to a new computer

Here are the first three changes I make to a new computer.

1. Remap Caps Lock to be a control key.
2. Delete all icons from the desktop.
3. Mute the sound.

Related posts:

Why and how to remap Caps Lock (Windows, Linux, and Mac)

Deleting the Windows recycle bin icon

The Book of Inkscape

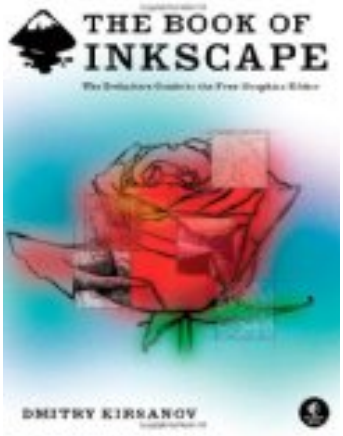
When I first started using Inkscape, I read ***Inkscape: Guide to a Vector Drawing Program*** by Tavmjong Bah, 3rd edition. It's now in its 4th edition, which I have not seen.

I received a copy of ***The Book of Inkscape*** by Dmitry Kirsanov recently, and it looks like the book I would have preferred to start with. Both books are fine introductions, but Kirsanov's book is more my style.

Bah's book is more inductive. It teaches you the elements of Inkscape by first taking you through a series of projects. Kirsanov's book is organized more like a textbook or a reference. Some people would prefer Bah's

book, especially if it were their intention to work through all the exercises. I prefer Kirsanov's book, organized more by topic than by project. It's easier to dip in and out of as needed.

I'd like to learn Inkscape well. I could imagine going through a book slowly, carefully working all the examples, exploring side roads, etc. But that's not realistic for me any time soon. For now, I expect I'll learn more about Inkscape ***just-in-time*** as I need to make illustrations. And Kirsanov's book is better suited for that.



Related posts:

Including LaTeX in an Inkscape drawing

Including an Inkscape drawing in LaTeX

Plotting functions in Inkscape

Should you walk or run in the rain?

One of the problems in ***X and the City***, a book I mentioned ***the other day***, is deciding whether you'll get wetter by walking or running in the rain.

The author takes several factors into account and models the total amount of water a person absorbs as

$$T = \frac{Iwd}{v}\left(ct\cos\theta + l(c\sin\theta + v)\right).$$

T = \frac{Iwd}{v}\left(ct \cos\theta + l(c \sin\theta + v)\right).

This assumes a person is essentially a rectangular box of height l, width w, and thickness t. The rain is falling at an angle θ to the vertical (e.g. $\theta = 0$ for rain coming straight down). The distance you need to walk or run is d and your speed is v. The rain is falling with speed c. The parameter I is the rain intensity, ranging from 0 for no rain to 1 for continuous flow. The book goes into greater detail, deriving the formula and estimating numerical values for the parameters.

Conclusion?

> If the rain is driving into you from the front, run as fast as you safely can. On the other hand, if the rain is coming from behind you, and you can keep pace with its horizontal speed by walking, do so!

Using SciPy with IronPython

Three years ago I wrote a ***post*** about my disappointment using SciPy with IronPython. A lot has changed since then, so I thought I'd write a short follow-up post.

To install NumPy and SciPy for use with IronPython, follow the instructions ***here***. After installation, NumPy works as expected.

There is one small gotcha with SciPy. To use SciPy with IronPython, start `ipy` with the command line argument `-X:Frames`. Then you can use SciPy as you would from CPython. For example.

```
c:\> ipy -X:Frames
>>> import scipy as sp
>>> sp.pi
3.141592653589793
```

Without the `-X:Frames` option you'll get an error when you try to import `scipy`.

```
AttributeError: 'module' object has no attribute
'_getframe'
```

According to ***this page***,

> The issue is that SciPy makes use of the CPython API for inspecting the current stack frame which IronPython doesn't enable by default because of a small runtime performance hit. You can turn on this functionality by passing the command line argument "-X:Frames" to on the command line.

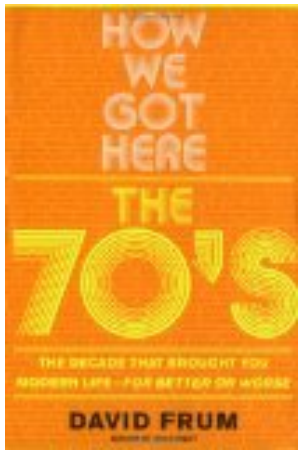
The 1970s

Here’s a perspective on the 1970s I found interesting: The decade was so embarrassing that climbing out of the ’70s was a proud achievement.

> The 1970s were America’s low tide. Not since the Depression had the country been so wracked with woe. *Never* — not even during the Depression — had American pride and self-confidence plunged deeper. But the decade was also, paradoxically, in some ways America’s finest hour. America was afflicted in the 1970s by a systemic crisis

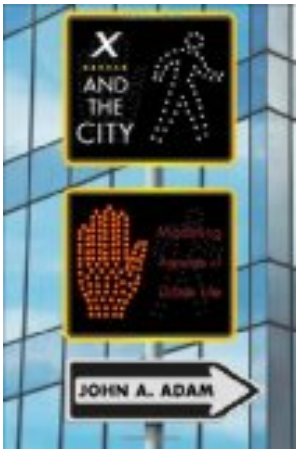
> analogous to the one that struck Imperial Rome in the middle of the third century A.D. … But unlike the Romans, Americans staggered only briefly before the crisis. They took the blow. For a short time they behaved foolishly, and on one or two occasions, even disgracefully. Then they recouped. They rethought. They reinvented.

Source: ***How We Got Here: The 70's: The Decade That Brought You Modern Life—For Better or Worse***



Differential Equations and the City

This afternoon I got a review copy of~***X and the City: Modeling Aspects of Urban Life*** by John A. Adam. It's a book about mathematical model, taking all its examples from urban life: public transportation, growth, pollution, etc. I've only skimmed through the book so far, but it looks like most of the applications involve differential equations. Some depend on algebra or probability.



The book looks interesting. I hope to say more about the book once I've had a chance to read it. The examples are all short, so it may be any easy book to read a little at a time.

I also got a review copy of ***The Book of Inkscape*** today, and I'm expecting several other books soon. It may take a while to get through these since this is a busy time for me. When it rains, it pours.

Castles and quantum mechanics

How are castles and quantum mechanics related? One connection is rook polynomials.

The rook is the chess piece that looks like a castle, and used to be called a castle. It can move vertically or horizontally, any number of spaces.

A rook polynomial is a polynomial whose coefficients give the number of ways rooks can be arranged on a chess board without attacking each other. The coefficient of x^k in the polynomial $R_{m,n}(x)$ is the number of ways you can arrange k rooks on an m by n chessboard such

that no two rooks are in the same row or column.

The rook polynomials are related to the Laguerre polynomials by

$R_{m,n}(x) = n!\ x^n\ L_n^{m-n}(-1/x)$

where $L_n^k(x)$ is an "associated Laguerre polynomial." These polynomials satisfy Laguerre's differential equation

$x\ y'' + (n+1-x)\ y' + k\ y = 0,$

an equation that comes up in numerous contexts in physics. In quantum mechanics, these polynomials arise in the solution of the Schrödinger equation for the hydrogen atom.

Related:

Relations between special functions

Mars, magic squares, and music

About a year ago I wrote about ***Jupiter's magic square***. Then yesterday I was listening to the ***New Sounds podcast*** that mentioned a magic square associated with Mars. I hadn't heard of this, so I looked into it and found there were magic squares associated with each of solar system bodies known to antiquity (i.e. Sun, Mercury, Venus, Moon, Mars, Jupiter, and Saturn).

Here is the magic square of Mars:

11	24	7	20	3
4	12	25	8	16
17	5	13	21	9
10	18	1	14	22
23	6	19	2	15

The podcast featured ***Secret Pulse*** by Zack Browning. From the ***liner notes***:

> Magic squares provide structure to the music. Structure provides direction to the composer. Direction provides restrictions for the focused inspiration and interpretation of musical materials.

> The effect of this process? Freedom to compose.
>
> The compositions on this CD use the 5×5 Magic Square of Mars (Secret Pulse), the 9×9 Magic Square of the Moon (Moon Thrust), and the ancient Chinese 3×3 Lo Shu Square found in the Flying Star System of Feng Shui (Hakka Fusion, String Quartet, Flying Tones, and Moon Thrust) as compositional models.~ The musical structure created from these magic squares is dramatically articulated by the collision of different musical worlds …

I don’t know how the composer used these magic squares, but you can listen to the title track (Secret Pulse) on the ***podcast***.

Related posts:

Jupiter’s magic square

A magic king’s tour

A magic knight’s tour

A knight’s random walk

Machine Learning in Action

A couple months ago I briefly reviewed ***Machine Learning for Hackers*** by Drew Conway and John Myles White. Today I'm looking at ***Machine Learning in Action*** by Peter Harrington and comparing the two books.

Both books are about the same size and cover many of the same topics. One difference between the two books is choice of programming language: *ML for Hackers* uses R for its examples, *ML in Action* uses Python.

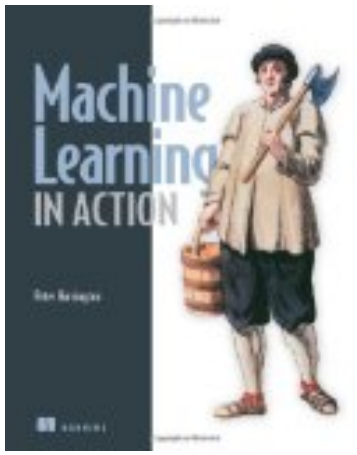
ML in Action doesn't lean heavily on Python libraries. It mostly implements its algorithms from scratch, with a little help from NumPy for linear algebra, but it does not use ML libraries

such as ***scikit-learn***. It sometimes uses Matplotlib for plotting and uses Tkinter for building a simple GUI in one chapter. The final chapter introduces Hadoop and Amazon Web Services.

ML for Hackers is a little more of a general introduction to machine learning. *ML in Action* contains a brief introduction to machine learning in general, but quickly moves on to specific algorithms. *ML for Hackers* spends a good number of pages discussing data cleaning. *ML in Action* starts with clean data in order to spend more time on algorithms.

ML in Action takes 8 of the top 10 algorithms in machine learning (as selected by ***this paper***) and organizes around these algorithms. (The two algorithms out of the top 1o that didn't make it into *ML in Action* were PageRank, because it has been covered well elsewhere, and EM, because its explanation requires too much mathematics.) The algorithms come first in *ML*

in Action, illustrations second. *ML for Hackers* puts more emphasis on its examples and reads a bit more like a story. *ML in Action* reads a little more like a reference book.



http://www.johndcook.com/blog/2008/06/27/w
tistics/#comment-170809

Criteria for a computing setup

“My setup” articles have become common. These articles list the hardware and software someone uses, usually with little explanation. The subtext is often the author’s commitment to the Apple brand or to open source, to spending money on the best stuff or to avoid spending money on principle. I don’t find such articles interesting or useful.

Vivek Haldar has written a different kind of~ “my setup” article, one that emphasizes the problems he set out to solve and the reasons for the solutions he chose. Here are a cou-

ple excerpts describing his goals for preserving his data and his health.

> Try to remember the oldest digital artifact that you can still retrieve, and more importantly, decode and view. Can you? How old is it? That should give you some idea of how hard and full of unknowns the problem of long-term preservation is. …

> If a significant fraction of your working life is spent working with computers, and you do not yet have even the mildest RSI, you should consider yourself extremely lucky, but not immune. Act like you do have RSI, and change your set up right now to avoid it.

I thought the best part of the article was the criteria, not the solutions. It's not that I disapprove of his choices, but I appreciate more his

explanation of the rationale behind his choices. I don't expect anybody is going to read the article and say "That's it! I'm going to copy that setup." I gather that in at least one detail, his choice of version control system, Vivek wouldn't even copy his own setup if he were to start over. But people will get ideas to consider in their own circumstances.

Related post: ***Ford-Chevy arguments in tech***

Solutions to knight's random walk

My ***previous post*** asked this question:

> Start a knight at a corner square of an otherwise-empty chessboard. Move the knight at random by choosing uniformly from the legal knight-moves at each step. What is the mean number of moves until the knight returns to the starting square?

There is a mathematical solution that is a little arcane, but short and exact. You could also approach the problem using simulation, which is more accessible but not exact.

The mathematical solution is to view the problem as a random walk on a graph. The vertices of the graph are the squares of a chess board and the edges connect legal knight moves. The general solution for the time to first return is simply $2N/k$ where N is the number of edges in the graph, and k is the number of edges meeting at the starting point. Amazingly, the solution hardly depends on the structure of the graph at all. It only requires that the graph is connected. In our case $N = 168$ and $k = 2$.

For a full explanation of the math, see this ***online book***, chapter 3, page 9. Start there and work your way backward until you understand the solution.

And now for simulation. The problem says to pick a legal knight's move at random. The most direct approach would be to find the legal moves at a given point first, then choose one of those at random. The code below achieves

the same end with a different approach. It first chooses a random move, and if that move is illegal (i.e. off the board) it throws that move away and tries again.~ This will select a legal move with the right probability, though perhaps that's not obvious. It's what's known as an accept-reject random generator.

```
from random import randint

# Move a knight from (x, y) to a random new position
def new_position(x, y):

    while True:
        dx, dy = 1, 2

        # it takes three bits to determine a random
knight move:
        # (1, 2) vs (2, 1), and the sign of each
        r = randint(0, 7)
        if r % 2:
            dx, dy = dy, dx
        if (r >> 1) % 2:
            dx = -dx
        if (r >> 2) % 2:
```

```
            dy = -dy

        newx, newy = x + dx, y + dy
        # If the new position is on the board, take it.
        # Otherwise try again.
        if (newx >= 0 and newx < 8 and newy >= 0 and
newy < 8):
            return (newx, newy)

# Count the number of steps in one random tour
def random_tour():
    x, y = x0, y0 = 0, 0
    count = 0
    while True:
        x, y = new_position(x, y)
        count += 1
        if x == x0 and y == y0:
            return count

# Average the length of many random tours
sum = 0
num_reps = 100000
for i in xrange(num_reps):
    sum += random_tour()
print sum / float(num_reps)
```

A theorem is better than a simulation, but a simulation is a lot better than nothing. This problem illustrates how sometimes we think we need to simulate when we don't. On the other hand, when you have a simulation *and* a theorem, you have more confidence in your solution because ***each validates the other***.

A knight’s random walk

Here’s a puzzle I ran across today:

> Start a knight at a corner square of an otherwise-empty chessboard. Move the knight at random by choosing uniformly from the legal knight-moves at each step. What is the mean number of moves until the knight returns to the starting square?

There’s a slick mathematical solution that I’ll give later.

You could also find the answer via simulation: write a program to carry out a knight random walk and count how many steps it

takes. Repeat this many times and average your counts.

knight

Related post: ***A knight's tour magic square***

Web architecture

Robert “Uncle Bob” Martin argues that an application’s purpose should determine its architecture; its delivery mechanism should not. If you look at an architectural drawing of a church, for example, you can tell it’s a church. But if you look at an architectural drawing of a web application, the fact that it’s a web application should not be the first thing you notice.

> The web is a delivery mechanism. The web is a detail. The web is not particularly important to your application. … The web is a pipe. It is nothing more than a pipe. It is not the central abstrac-

> tion of your application. It is not the grand, over-arching thing that makes your application the application it is. The web is just a dumb detail. And yet it dominates our code.

From Robert Martin's ***keynote*** at Ruby Midwest 2011. The quote starts about 9 minutes into the presentation.

He goes on to explain how to let the functionality of the application determine its architecture. In the dependency diagram arrows point *from* the delivery mechanism *to* the rest of the code, but none point the other way. By following this design you get something easier to test.

Related posts:

Recap of the Robert Martin/Joel Spolksy brouhaha

Why there will always be programmers

How do you know when someone is great?

Roger Peng asks a good question on ***his blog*** this morning: How would you know if someone is great at data analysis? He says that while he has worked with some great data analysts, the nature of their work is that it's hard to evaluate the work of someone you don't know personally. And as ***Josh Grant*** pointed out, this isn't unique to data analysts.

I immediately thought of a database administrator I know. Everyone who works with her knows she's great at her job, but I doubt any-

one who doesn't know her has ever said "They must have a great DBA!"

Matthew Crawford argues in ***Shop Class as Soulcraft*** that white collar work in general is hard to objectively evaluate and that this explains why offices are so political. Employees are judged on their sensitivity and other nebulous attributes because unlike a welder, for example, they can't be judged directly on their work. He argues that blue collar people workers greater freedom of speech at work because their work can be objectively evaluated.

Colleagues can identify great data analysts, DBAs, and others whose work isn't on public display. But this isn't easy to do through a bureaucratic process, and so technical competence is routinely under-compensated in large organizations. On the other hand, reputation spreads more efficiently outside of organizational channels. This may help explain why

highly competent people are often more appreciated by their professional communities than by their employers.

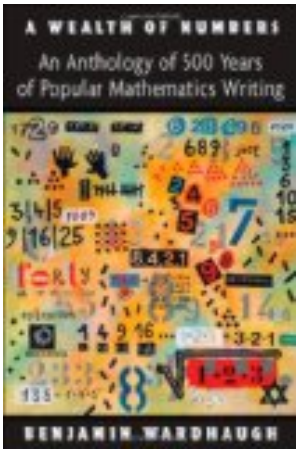
Related post: ***It doesn’t pay to be the computer guy***

An algebra problem from 1798

The Lady's Diary was a popular magazine published in England from 1704 to 1841. It contained mathematical puzzles such as the following, published in 1798.

> What two numbers are those whose product, difference of their squares, and the ratio or quotient of their cubes, are all equal to each other?

From Benjamin Wardhaugh's ***new book*** A Wealth of Numbers: An Anthology of 500 Years of Popular Mathematics Writing.



See also my ***brief review*** of ***How to Read Historical Mathematics*** by the same author.

Beer with a confidence interval

Medalla is a Puerto Rican beer. On the side of a can it says

> Alcohol by volume over 4%, not more than 6%.

I'd never seen a beer give a confidence interval for its alcohol content. I'd only seen point estimates before. For example, Budweiser announces that its beer contains 5%alcohol by volume. Just 5%, no statistical details.

I wonder why Medalla reports an interval.

- Is their manufacturing process so variable that they can't just report an average?
- Is their manufacturing process no more variable than others, but they disclose their uncertainty?
- Are they required to report alcohol content the way they do by local law or by the law of countries they export to?

Related posts:

Beer, wine, and statistics

Wine and politics

Visiting Puerto Rico

I've been in San Juan this week, visiting the University of Puerto Rico. I've been here several times before, but here are a few things I noticed about Puerto Rico on this trip.

Coffee: Coffee means espresso here; I haven't seen it brewed any other way. And it's cheap. At UPR, a 4 oz *pocillo* is only $0.70, and at a bakery near my hotel a 6 oz *espresso con leche* is $1.25.

Gasoline: You can't pay at the pump with a credit card. You have to go inside to pay for gas. Some gas stations used to let you pay at the pump, but these have gone back to having you pay inside.

Electronics: I've been told that you can't buy electronics from Apple's web site for delivery in Puerto Rico. You can buy Apple products in stores like Best Buy on the island, but you can't have them shipped directly here from their web site.

***

Related post: ***Beer with a confidence interval*** (Medalla)

Python as a Lisp dialect

From Peter Norvig:

> Basically, Python can be seen as a dialect of Lisp with “traditional” syntax … Python supports all of Lisp’s essential features except macros, and you don’t miss macros all that much because it does have `eval`, and operator overloading, and regular expression parsing, so some — but not all — of the use cases for macros are covered.

Source: ***Python for Lisp Programmers***

Reading historical math

I recently received review copies of two books by Benjamin Wardhaugh. Here I will discuss ***How to Read Historical Mathematics***. The other book is his ***anthology*** of historical popular mathematics which I intend to review later.

Here is the key passage, located near the end of How to Read Historical Mathematics, for identifying the author's perspective.

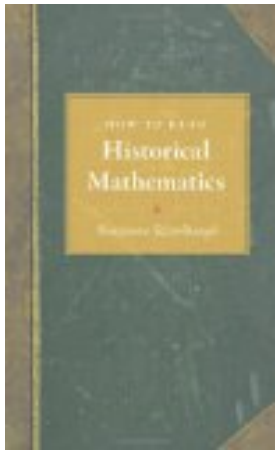
> But not all historical mathematics *is* significant. And perhaps there is a second kind of significance, where something can be historically significant without being mathematically signif-

> icant. Some historians (I'm one of them) delight in investigating mathematical writing that contains little or no important or novel mathematics: popular textbooks, self-instruction manuals, … or old almanacs and popular magazines with mathematical news or puzzles in them. These kinds of writing … are certainly significant for a historian who wants to know about popular experiences of mathematics. But they're not significant in the sense of containing significant mathematics.

Wardhaugh's perspective is valuable, though it is not one that I share. My interest in historical math is more on the development of the mathematical ideas rather than their social context. I'm interested, for example, in discovering the concrete problems that motivated mathematics that has become more abstract

and formal.

I was hoping for something more along the lines of a mapping from historical definitions and notations to their modern counterparts. This book contains a little of that, but it focuses more on how to read historical mathematics as a *historian* rather than as a *mathematician*. However, if you are interested in more of the social angle, the book has many good suggestions (and even exercises) for exploring the larger context of historical mathematical writing.



Big data is easy

Big data is easy; big models are hard.

If you just wanted to use simple models with tons of data, that would be easy. You could resample the data, throwing some of it away until you had a quantity of data you could comfortably manage.

But when you have tons of data, you want to take advantage of it and ask questions that simple models cannot answer. (”Big” data is often*indirect* data.) So the problem isn’t that you *have* a lot of data, it’s that you’re using models that *require* a lot of data. And that can be very hard.

I am not saying people should just use simple models. No, people are right to want to take advantage of their data, and often that does require complex models. (See Brad Efron's ***explanation*** why.) But the primary challenge is intellectual, not physical.

Related post:***http://www.johndcook.com/blog/2010/12/15/big-data-is-not-enough/ Big data and humility***

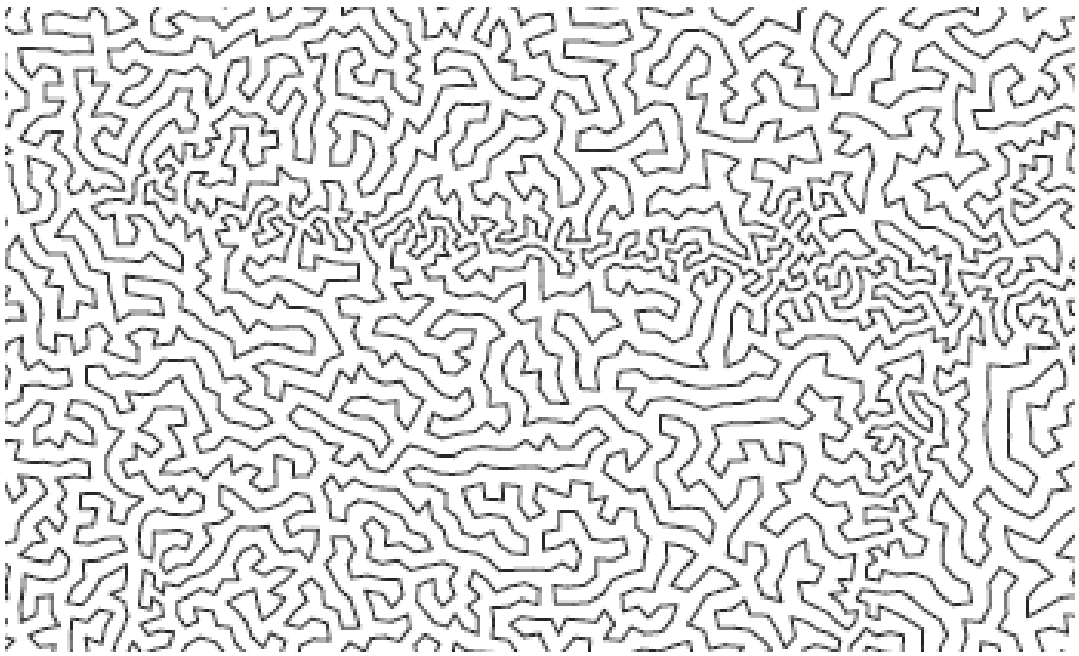
Traveling salesman art

Bill Cook sent me a file yesterday that renders the Endeavour photo on my blog as the solution to a 66,290-city Traveling Salesman problem. His iPhone app ***Concord TSP*** chose 66,290 points and then solved for the shortest path connecting these points, a feat that would have strained a supercomputer a few years ago. (Bill Cook and I are not related as far as I know.)

Here is a thumbnail image of the full TSP tour:

You can find the full PDF ***here*** (1.24 MB). To show some of the detail, here is a close-up from near the top-left corner of the image:

I asked how the tour was constructed:



> How do you construct a set of points whose TSP solution resembles a photograph? Is it sufficient to add more “cities” in regions where you want darker shading? And are the cities added at random with a density specified by color depth?

Bill Cook replied:

> By default, the app will select the points along the line you describe: it splits the image into a grid, computes the average gray scale in each grid region, and drops a number of random cities into each grid region in proportion to the square of the average gray scale. This technique was first proposed by Bob Bosch and Adrianne Herman at Oberlin College. It is the default since it takes almost no time to compute, but I include two other options,

> that each take about a minute to render a large image on an iPhone 4.
>
> The image of The Endeavour was created with a method Jim Bumgarnder proposed in his ***Stipple Cam project***.

Related post:

Moore’s law squared

Conforming for tenure

From AnnMaria De Mars' most recent ***blog post***:

> Recently, a young person told me that *I* could hold to my principles about the importance of my family, honesty and equality — and any of a hundred other things because I had "made it".
>
> This troubled me. It troubles me when I hear the same thing from new Ph.Ds who are trying to get tenure.**I don't see how you can pretend to be someone else for 5 or 10 years until you have "made it" and then**

> **be your true self**.

Emphasis added.

Open source and pride

Liz Quilty explains how becoming an expert in open source software changed her life.

> [I was a] high school dropout, I had no education, I was nobody. I'd made some poor choices, and I think at this point suddenly I was known for my knowledge rather than for my poor choices. And so it was quite good for me that I had some pride. I was like, this is really awesome, I'm really good at this, I really know this.

Source: ***FLOSS Weekly*** podcast, starting around 12:00.

A tip on using a French press

When I first bought a ***French press***, the instructions said to pour hot but not boiling water over the coffee. They were emphatic about what the temperature should *not* be, but vague about what it *should* be. (Boiling water extracts oils that you'd rather leave in the grounds; water a little cooler brings out just the oils you want.)

I asked someone online — sorry, I don't remember who — and he said that after your water boils, set the kettle off the burner and check your mail. When you come back, the

water should be at the right temperature. That turned out to be good advice.

I don’t know whether he meant postal mail or email, but it doesn’t make much difference. Unless you get caught in replying to email and come back to room-temperature water.

Now if you really wanted to geek out on this, you could use ***Newton’s law of cooling*** along with the surface area, thickness, and material composition of your kettle to compute the time to let your water cool to 200 °F (93 °C). You could assume a kettle is a half sphere …

Related post: ***A childhood question about heat***

How variable are percentiles?

Suppose you're trying to study the distribution of something by simulation. The average of your simulation values gives you an estimate of the mean value of the thing you're simulating.

Next you want to have an idea how much the thing you're simulating varies. You estimate the percentiles of your distribution by the percentiles of your samples. How variable are your estimates? For example, you estimate the 90th percentile of your distribution by finding the 90th percentile of your samples. How much is your estimate likely to change if you rerun

you simulation with a new random number generator seed?

How does the variability of the percentiles depend on the percentile you're looking at? For example, you might expect the median, i.e. the 50th percentile, to be the most stable. And as you start looking at more extreme percentiles, say the 5th or 95th, you might expect more variability. But *how much* more variability? A lot more or a little more?

To explore the questions above, let's suppose we're sampling from a standard normal distribution. (Of course if we really were sampling from something so well known as a normal distribution, we wouldn't be doing simulation. But we need to pick something for illustration, so let's use something simple.)

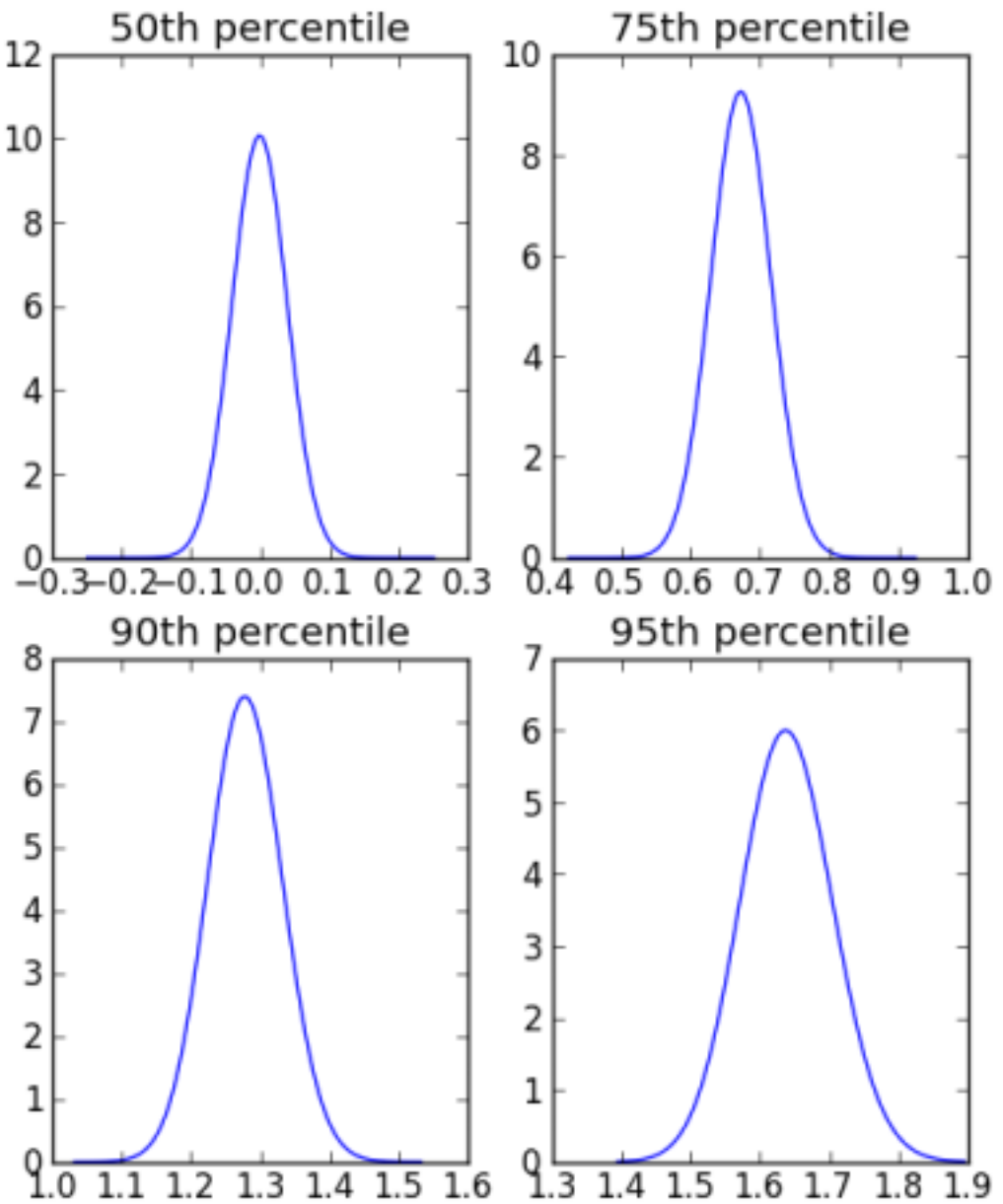
Suppose we simulate n values, sort them, and take the kth largest value.~ This is called the kth order statistic. For example, we might

estimate the 90th percentile of our distribution by using $n = 1000$ and $k = 900$. The distribution of the kth order statistic from n samples is known analytically (it's a short derivation; see ***here***) and so we can use it to make some plots.

These plots use $n = 1000$. Each uses the same horizontal scale, so you can see how the distributions get wider as we move to higher percentiles. The distribution for the 95th percentile is about twice as wide as the distribution for the median.

Each plot is centered at the corresponding quantile for the standard normal distribution. For example, the plot for the distribution of the 75th percentile of the samples is centered at the 75th percentile of the standard normal.

The distribution for the sample median is unbiased, symmetric about 0. The distributions for the 75th and 90th percentiles are slightly biased to the left, i.e. they slightly underes-



timate the true values on average. The 95th percentile, however, is slightly biased in the opposite direction. [Update: As pointed out in the comments below, it seems the appearance

of bias was a plotting artifact.]

If you'd like to see the details of how the plot was made, here is the Python code.

```
import scipy as sp
import scipy.stats
import scipy.special
import matplotlib.pyplot as plt

def pdf(y, n, k):
    Phi = scipy.stats.norm.cdf(y)
    phi = scipy.stats.norm.pdf(y)
    return scipy.stats.beta.pdf(Phi, k, n+1-k)*phi

def plot_percentile(p):
    center = scipy.stats.norm.ppf(0.01*p)
    t = sp.linspace(center-0.25, center+0.25, 100)
    plt.plot(t, pdf(t, 1000, 10*p))
    plt.title( "{0}th percentile".format(p) )

plt.subplot(221)
plot_percentile(50)
plt.subplot(222)
plot_percentile(75)
plt.subplot(223)
plot_percentile(90)
```

```
plt.subplot(224)
plot_percentile(95)
plt.show()
```

Simmer reading list

One of my friends mentioned his “simmer reading” yesterday. It was a typo — he meant to say “summer” — but a simmer reading list is interesting.

Simmer reading makes me think of a book that stays on your nightstand as other books come and go, like a pot left to simmer on the back burner of a stove. It’s a book you read a little at a time, maybe out of order, not something you’re trying to finish like a project.

What are some of your simmer reading books?

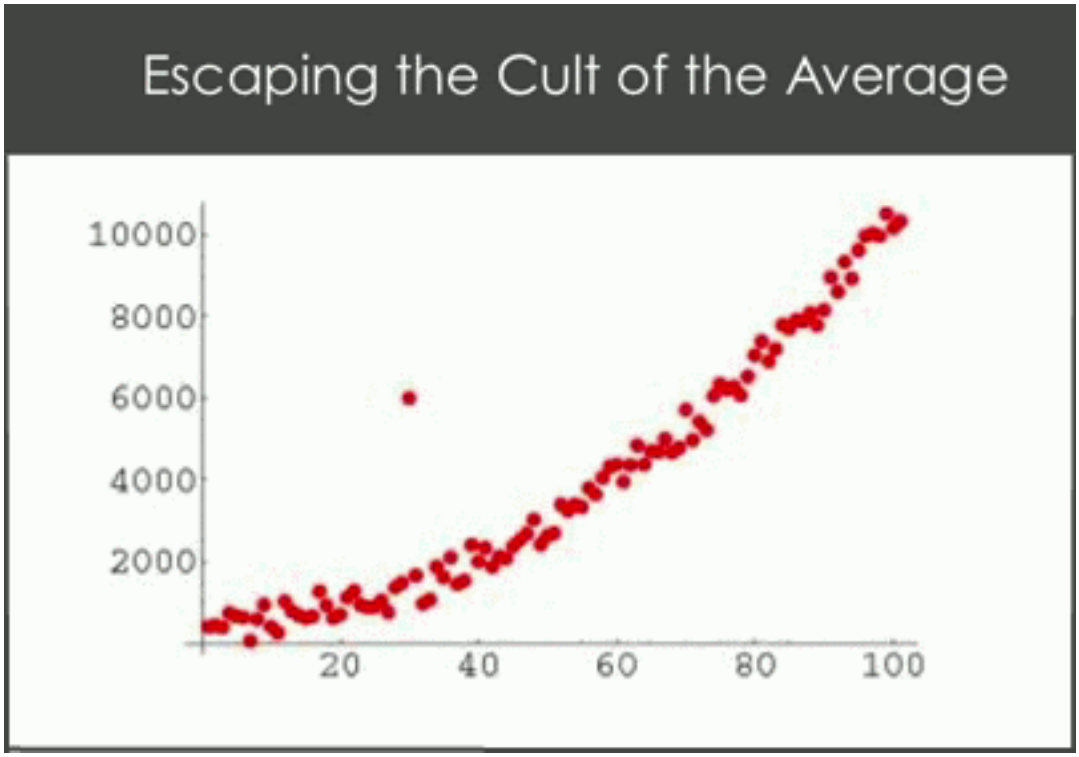
Related post: ***A book so good I had to put it down***

The cult of average

Shawn Achor comments on “the cult of the average” in science.

> So one of the very first things we teach people in economics and statistics and business and psychology is how, in a statistically valid way, do we eliminate the weirdos. How do we eliminate the outliers so we can find the line of best fit? Which is fantastic if I’m trying to find out how many Advil the average person should be taking — two. But if I’m interested in potential, if I’m interested in your potential, or for happi-

> ness or productivity or energy or creativity, what we're doing is we're creating the cult of the average with science. … **If we study what is merely average, we will remain merely average**.



Related posts:

Stupidity scales

Above average legs

Achievement is not normal

Chaotic versus random

From John D. Barrow's chapter in ***Design and Disorder***:

> The standard folklore about chaotic systems is that they are unpredictable. They lead to out-of-control dinosaur parks and out-of-work meteorologists. …
>
> Classical … chaotic systems are not in any sense intrinsically random or unpredictable. They merely possess **extreme sensitivity to ignorance**. Any initial uncertainty in our knowledge of a chaotic system's state is rapidly am-

> plified in time.
>
> … although they become unpredictable when you try to determine the future from a particular uncertain starting value, there may be a particular stable statistical spread of outcomes after a long time, regardless of how you started out.

Emphasis added.

Related post:

Random is as random does

100x better approach to software?

Alan Kay speculates in ***this talk*** that 99%or even 99.9%of the effort that goes into creating a large software system is not productive. Even if the ratio of overhead and redundancy to productive code is not as high as 99 to 1, it must be pretty high.

Note that we're not talking here about individual programmer productivity. Discussions of 10x or 100x programmers usually assume that these are folks who can use basically the same approach and same tools to get much

more work done. Here we're thinking about better approaches more than better workers.

Say you have a typical stem of 10,000,000 lines of code. How many lines of code would a system with the same desired features (not necessarily all the *actual* features) require?

1. If the same team of developers had worked from the beginning with a better design?

2. If the same team of developers could rewrite the system from scratch using what they learned from the original system?

3. If a thoughtful group of developers could rewrite the system without time pressure?

4. If a superhuman intelligence could rewrite the system, something approaching ***Kol-***

mogorov complexity?

Where does the wasted effort in large systems go, and how much could be eliminated? Part of the effort goes into discovering requirements. Large systems never start with a complete and immutable specification. That's addressed in the first two questions.

I believe Alan Kay is interested in the third question: How much effort is wasted by brute force mediocrity? My impression from watching his talk is that he believes this is the biggest issue, not evolving requirements. He said ***elsewhere***

> Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves.

There's a rule that says bureaucracies fol-

low a 3/2 law: to double productivity, you need three times as many employees. If a single developer could produce 10,000 lines of code in some period of time, you'd need to double that 10 times to get to 10,000,000 lines of code. That would require tripling your workforce 10 times, resulting in over 57,000 programmers. That sounds reasonable, maybe even a little optimistic.

Is that just the way things must be, that geniuses are in short supply and ordinary intelligence doesn't scale efficiently? How much better could we do with available talent if took a better approach to developing software?

Related post:

Where does programming effort go?

No Silver Bullet