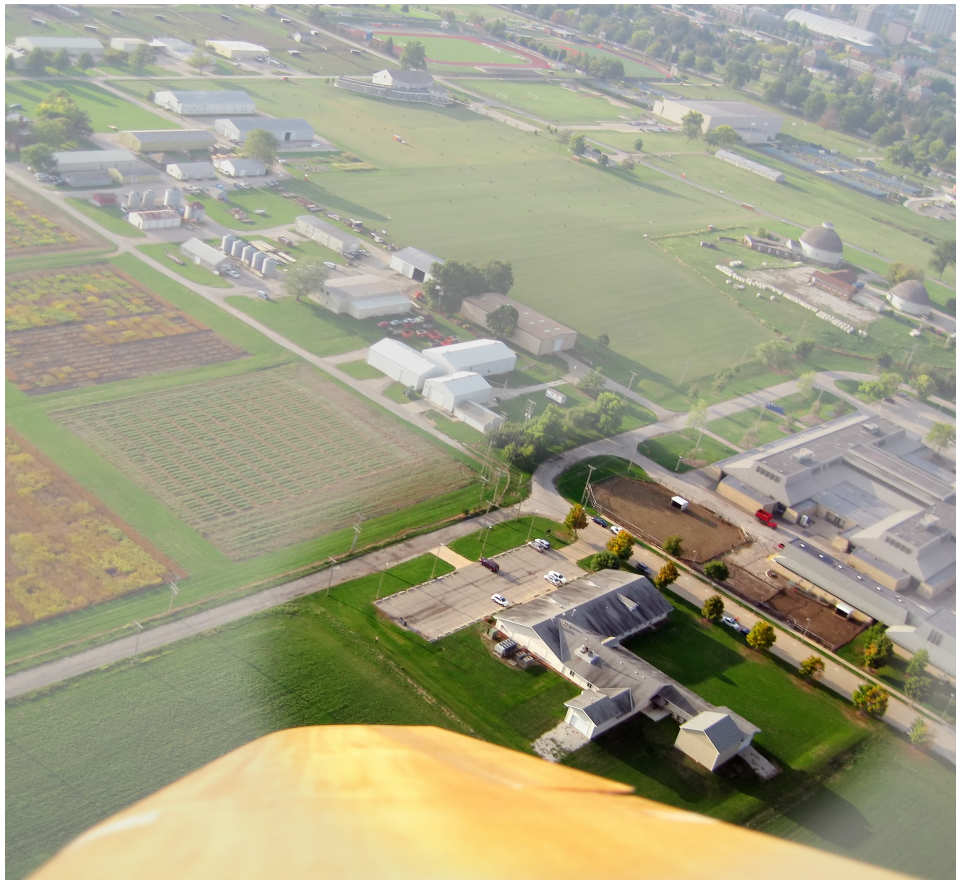
# Mechanized GTD

Camille Goudeseune
cog@illinois.edu

Illinois Simulator Lab,
Beckman Institute
www.isl.uiuc.edu



# Better living through stochastics

List all your projects.
Give each project a cost.

Pick a budget for these costs:

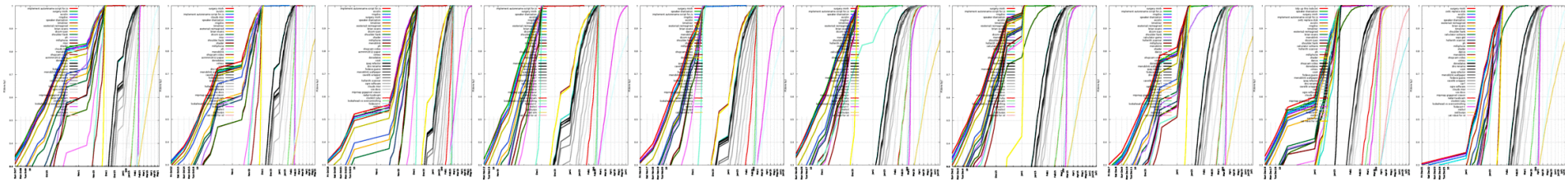
- 500,000 $/year from NSF
- 9 lab hours/day
- 3 dog-walking hours/day
- 5 group-meeting hours/week

Monte Carlo simulate the rest of your life:
One day at a time,
spend the daily budget stochastically to projects.
Record on which day each project completes.

# Better living through stochastics, cont'd

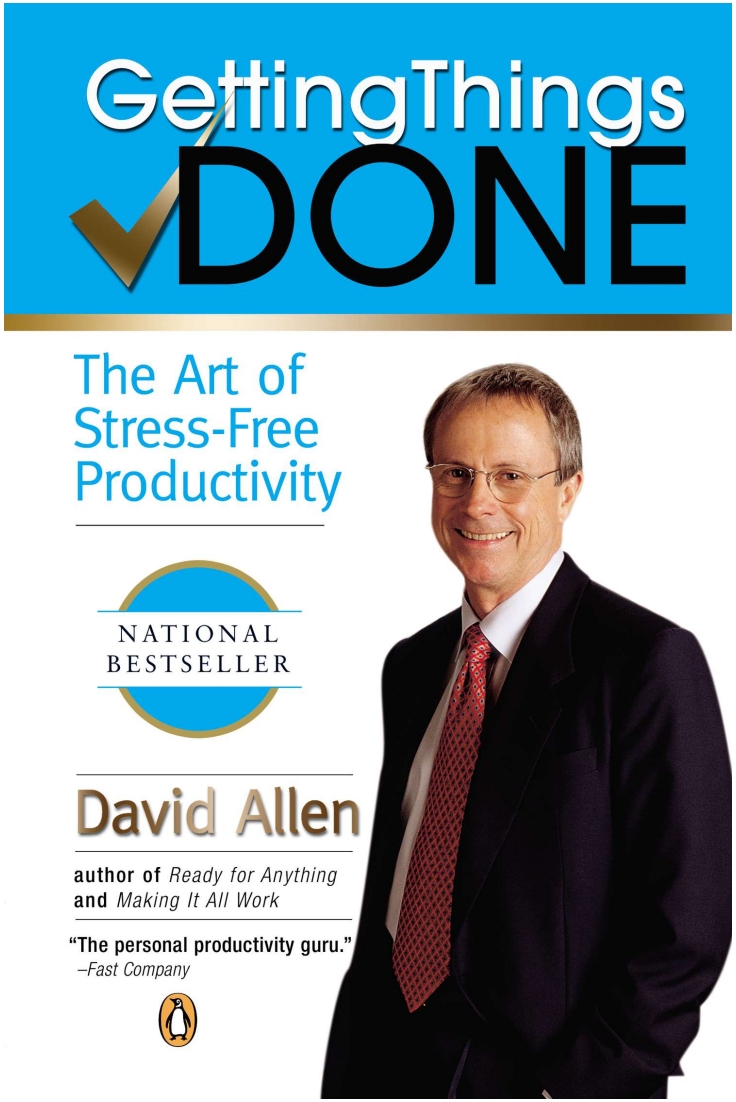
Do the Monte-Carlo hokey-pokey 1000 times,
to get 1000 completion dates. Plot that distribution.

(Here's my life for the past 10 weeks, at 1-week intervals.)



Observe which resources are bottlenecks (lab time!).
Choose an order of execution for projects.
Abandon projects that threaten the deadlines of more important projects.

# Getting Things Done




GTD is a

- 12-year-old book
- set of trench-warfare techniques
- religious cult
- #gtd hashtag

# GTD in postdoc-ese

Your mind is good at making creative associations.
Your laptop and filing cabinet are good at storing facts.
Use each only for what it's good at.

Offloading remembering to *trustworthy* external storage
frees up your mind to be creative.

Street cred: `http://dx.doi.org/10.1016/j.lrp.2008.09.004`
Cool buzzword: stigmergy.

`</evangelizing>`

# Motivation 1 of 3: Predict completion dates

Predict completion dates of projects,
to optimize their order of execution.

GTD advocates regularly seeing all your projects at a glance.
So, extend that high-level view, from just today,
to every day in the future for several months.

# Motivation 2 of 3: Ward off neophilia

High-level view = antidote to neophilia
(curiosity, 3-hour Wikipedia binges).

Neophilia in the lab = "this would be cool."
Dangerous, when you're bright enough to have many neat ideas,
*and* old enough to know how to implement all of them.

Neophilia = welcome distraction from long hard tasks
like writing a dissertation or a grant proposal.

# Motivation 3 of 3: Continuous review

GTD advocates a manual weekly review of all active projects.

This mechanism does an automatic nightly review,
warning you (by email or whatever) of impending snafus.

# Projects in GTD

A *project* is anything that has:

- a specific, finite result
- specific, finite costs
- more than one step (so it's worth managing)
- optionally, a deadline.

# An example project

On your computer, one folder
/projects.

That has subfolders for each project, such as
/projects/osculating-foliation/.

That subfolder then has (among other things) a plaintext file
/projects/osculating-foliation/a.txt:

```
result:  raytraced image of nested osculating spheres
cost:  hComputer 5
start:  12/10 # after ICASSP conference
due:  12/25 # Christmas present
prerequisite:  SOPS-presentation reinstall-Win7
nextproject:  NIH-grant-proposal
```

# Doing the Monte Carlo

Costs for projects are measured in arbitrary units.
The budget for each unit is in any convenient time unit:

- 2 hours per day practising piano, but 4 on weekends
- $100,000 per year spent from an awarded grant

Running the completion-date predictor:

- Inputs: budget and a set of projects.
- Algorithm: each day, spend that day's budget on random projects, until all projects are completed.
  As each project completes, log the date.
  *Do that 1000 times.*
- Outputs: for each project, a plot of how likely it'll be completed by a given date (and how likely it'll meet its deadline).

# Implementation details

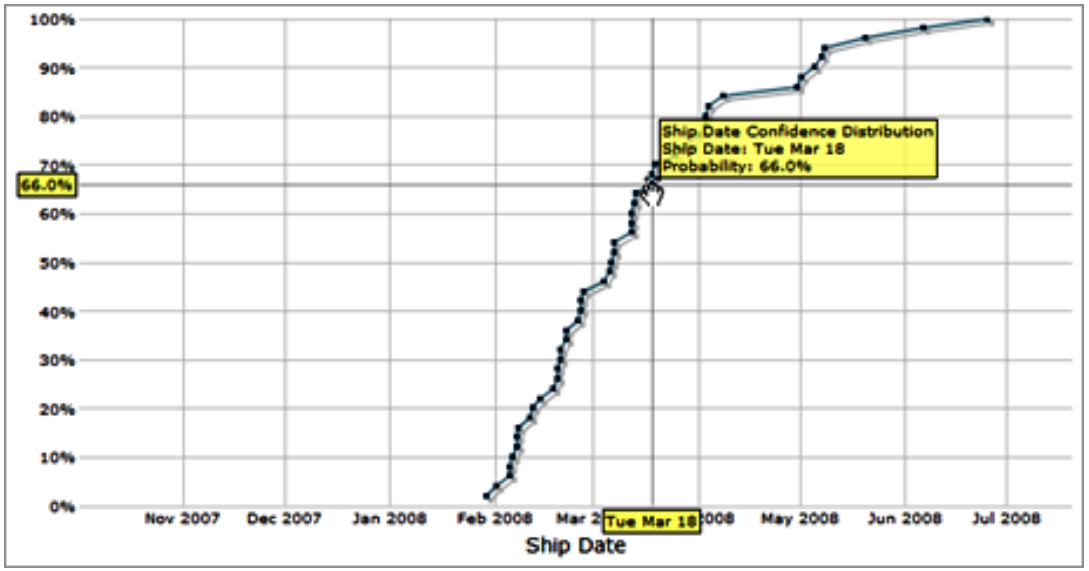
A 900-line Ruby script. Flexible but slow:
1000 trials $\times$ 200 projects $\times$ 365 days takes a few hours.
But slow is ok when you make decisions only once per day.

Plots are made with Ruby wrappers around gnuplot and imagemagick.

To model uncertainty, costs in the budget and in each project are dithered (randomly spread).

# Estimating costs

To correct endemic underestimation,
recalibrate future predictions with $\frac{\text{past prediction}}{\text{past measured cost}}$.



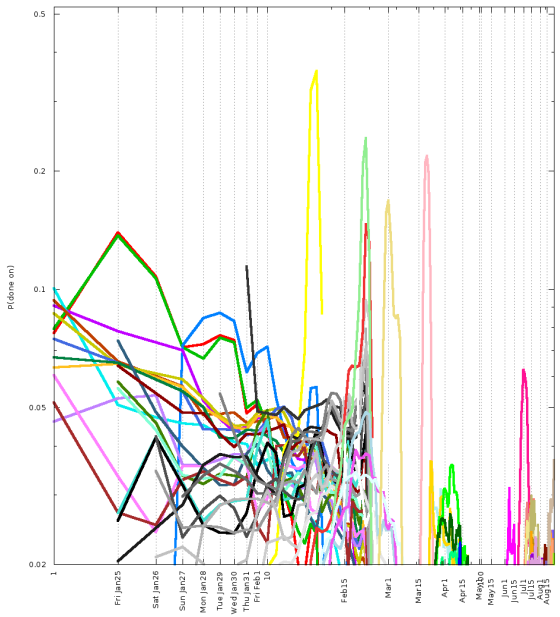

Joel Spolsky, "Evidence Based Scheduling,"
`www.joelonsoftware.com/items/2007/10/26.html`

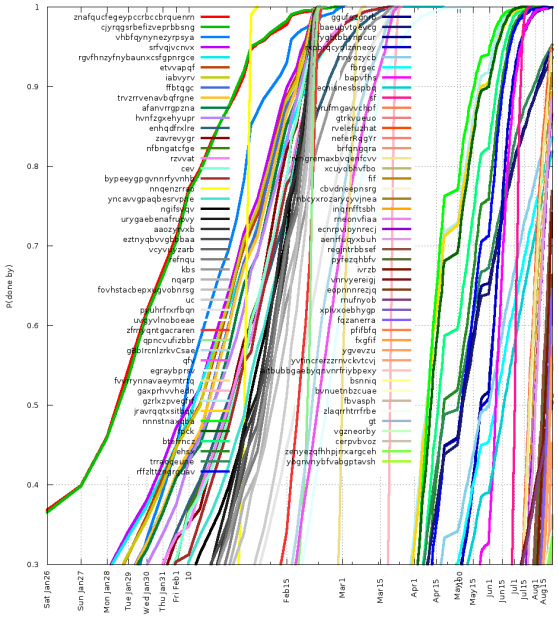
I don't bother. This matters less than you'd expect....

# How many plots?

One plot per project is useless for several hundred projects (usually I have 60 active and 140 pending).

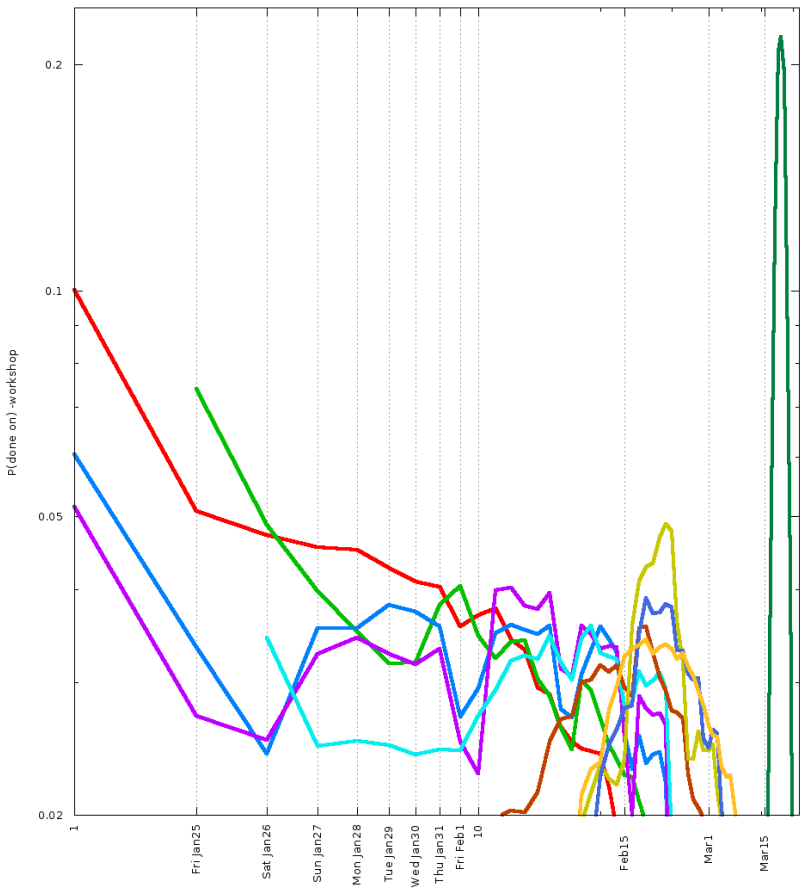
Equally useless is one plot overlaying all active projects.

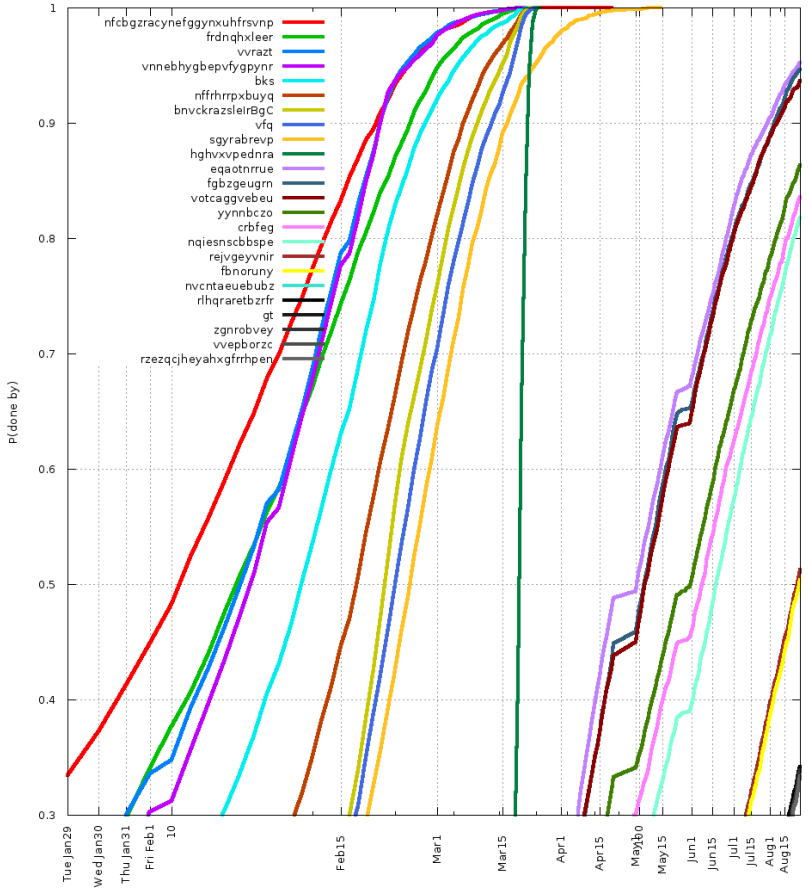


# Just a few plots

Happy medium: a few plots, each of a subset of projects related by a shared cost (competing for the same scarce resource).

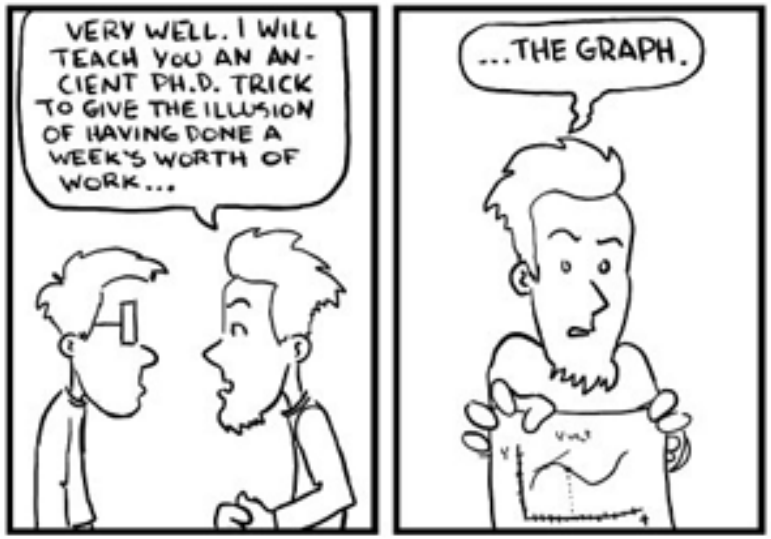



(This is my honey-do list.)

# Managing

So what use *are* these plots?




For managing these projects. In the technical sense of figuring out how to optimize your use of limited resources (the costs in the budget).

"How can I be maximally productive, within the budget I've measured?"

# Deadlines

When a deadline is in danger of being missed, or a completion date is amusingly distant and hence highly uncertain:

- Apply prerequisites, to change the order of execution. (Unlike the process scheduler of an OS, it's ok for projects to hang or kill or be infinitely delayed.)
- Change goals: reduce a project's cost by reducing its scope, or dividing it into milestones.
- Change the budget. Give up WoW, to get more hours for writing grant proposals.

# Poorly estimated costs

Inaccurate cost estimates are tolerable because:

- You probably skew all your estimates the same way.
- The skew slowly corrects as you update the remaining cost every week or so.
- Many of these daily or weekly decisions (postpone, reorder, subdivide, or abandon a project) are affected only by projects' relative costs.

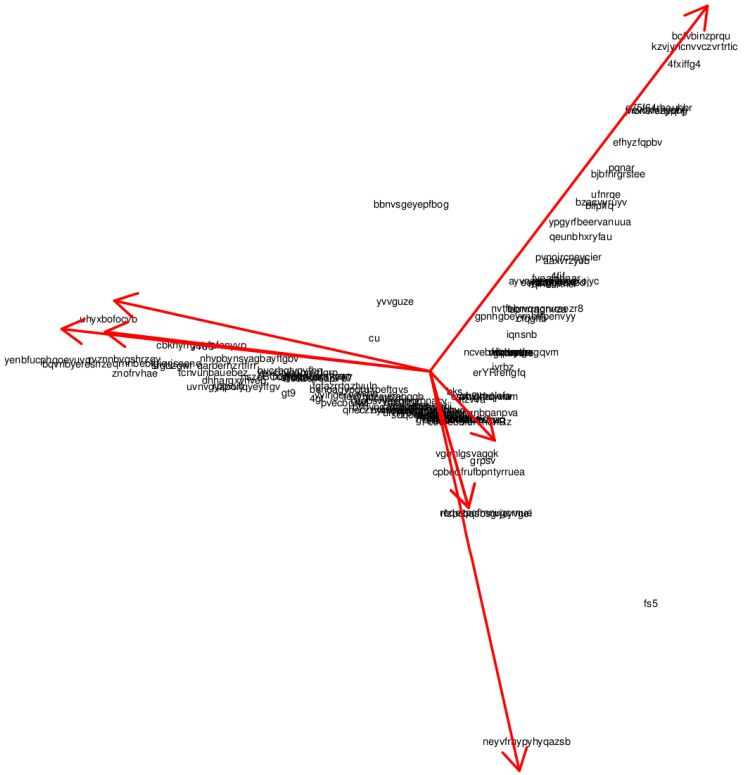
# Bonus: integrity checks

Besides predicting completion dates,

- A written-down list of projects is itself a useful high-level view.
- Automated integrity checks: well-defined cost, result, start date, etc.
- GTD integrity checks: each project has a "next action" defined on some "action list" (calendar, at laptop, running errands, ...).

# Failed bonus: clustering projects

- Plot all projects by cost, projecting vectors of cost-units down to 2D with prcomp and biplot in stats package 'R.'




(75% of my projects' cost vectors have *only one* non-zero entry!)

# Bonus: multi-level view

To make these decisions about projects *confidently*, you bounce between high-level multi-project overviews and icky details.

*Quickly*, within a few seconds, to preserve your train of thought.

These plots and integrity checks let you do exactly that.

# Thanks:


- The Prophet David Allen
- My supervisor, Hank Kaczmarski
- `apt-get install latex-beamer`
- `http://tex.stackexchange.com`