software carpentry

CUAHSI

UNIVERSITIES ALLIED FOR WATER RESEARCH

Schedule: Tues 7/16

- 9:00-10:00: Intro: what is software carpentry? (Ben Morris)
- 10:00: **COFFEE BREAK**
- 10:30-12:00: classes and objects in Python (Ben Morris)
- 12:00: **LUNCH**
- 1:00-2:30: program design (David Tarboton)
- 2:30: **COFFEE BREAK**
- 3:00-4:30: testing in Python (Ben Morris)

Schedule: Wed 7/17

- 9:00-10:30: using the shell (Ethan White)
- 10:30: **COFFEE BREAK**
- 11:00-12:30: version control with Git and GitHub (Ben Morris)
- 12:30: **LUNCH**
- 1:30-3:00: SQL databases (Ethan White)
- 3:00: **COFFEE BREAK**
- 3:30-4:30: conclusion (Ben Morris)

What is Software Carpentry?

Ben Morris

(thanks to Steve Crouch, Greg Wilson, Ethan White)

What is Software Carpentry?

“Software Carpentry helps researchers be more productive”

In the Seven Years' War, 1754-1763...

Britain lost 1,512 sailors to enemy attacks.

In the Seven Years' War, 1754-1763...

Britain lost 1,512 sailors to enemy attacks.

*...and nearly **100,000** to scurvy!*

The first (?) controlled medical experiment

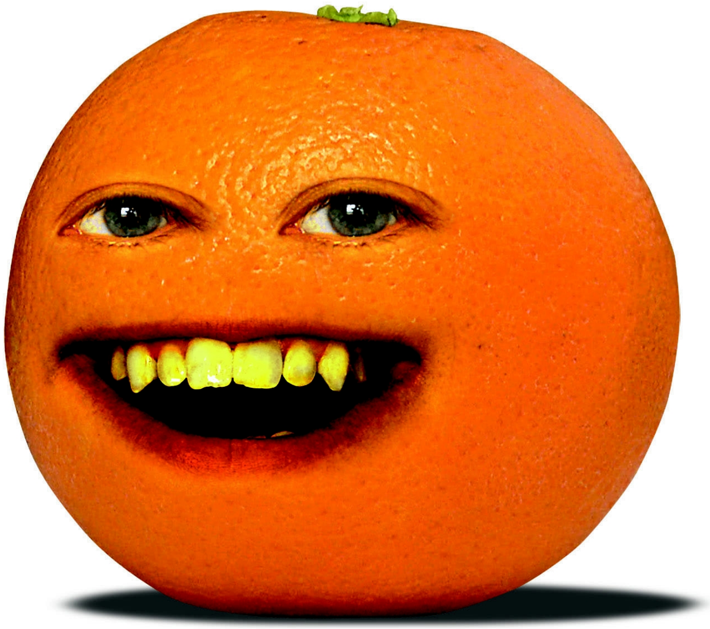
- James Lind, British scientist, in 1747
- Tested the efficacy of many substances thought to prevent scurvy:
  - Cider
  - Sea water
  - Sulphuric acid
  - Oranges
  - Vinegar
  - Barley water

The first (?) controlled medical experiment

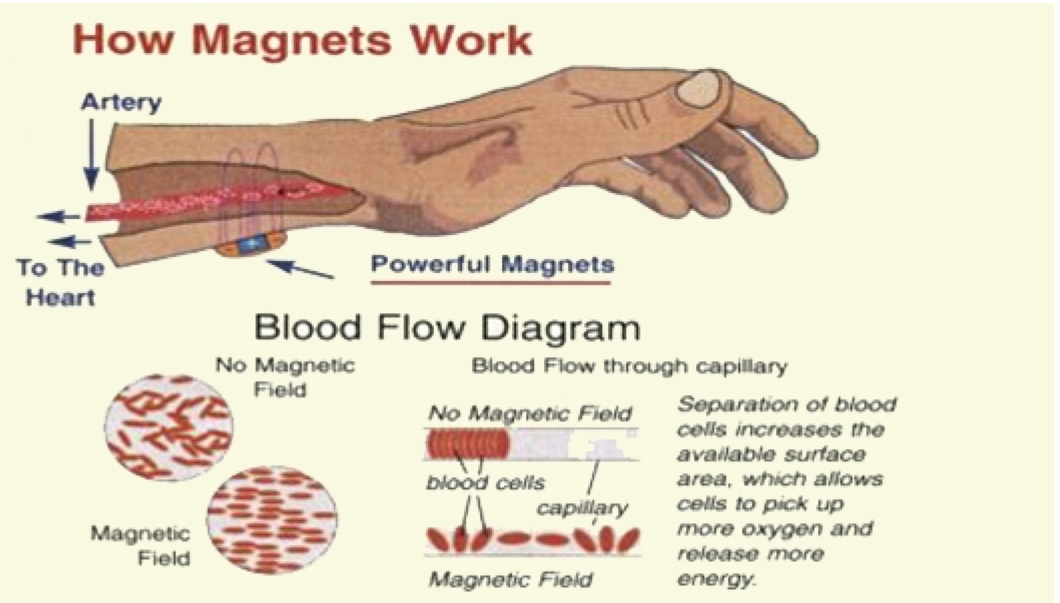
- James Lind, British scientist, in 1747
- Tested the efficacy of many substances thought to prevent scurvy:
  - Cider
  - Sea water
  - Sulphuric acid
  - **Oranges < == we have a winner!**
  - Vinegar
  - Barley water

The first (?) controlled medical experiment

- Yet the British Admiralty didn't listen (Lind wasn't an English gentleman) until 1794
- After 1794, dramatic worldwide decrease in deaths due to scurvy
- The scientific method worked!



Modern medicine



How Magnets Work

Blood Flow Diagram

Separation of blood cells increases the available surface area, which allows cells to pick up more oxygen and release more energy.

Modern medicine

- How do we distinguish what works and what doesn't?
- **Evidence-**based medicine
  - Randomization
  - Double-blind studies
  - Transparency, data accessibility

Software is no different!

- Should be based on **evidence** of what works, not superstition or anecdotes
- What do we know about how to effectively develop software, and how do we know it?
- A certain amount of skepticism towards common software engineering anecdotes is healthy!
- “This works because many people believe it does”

A bold claim

- “The best programmers are up to **28** times more productive than the worst”
  - Sackman, Erikson, and Grant, “Exploratory experimental studies comparing online and offline programming performance” (1968)

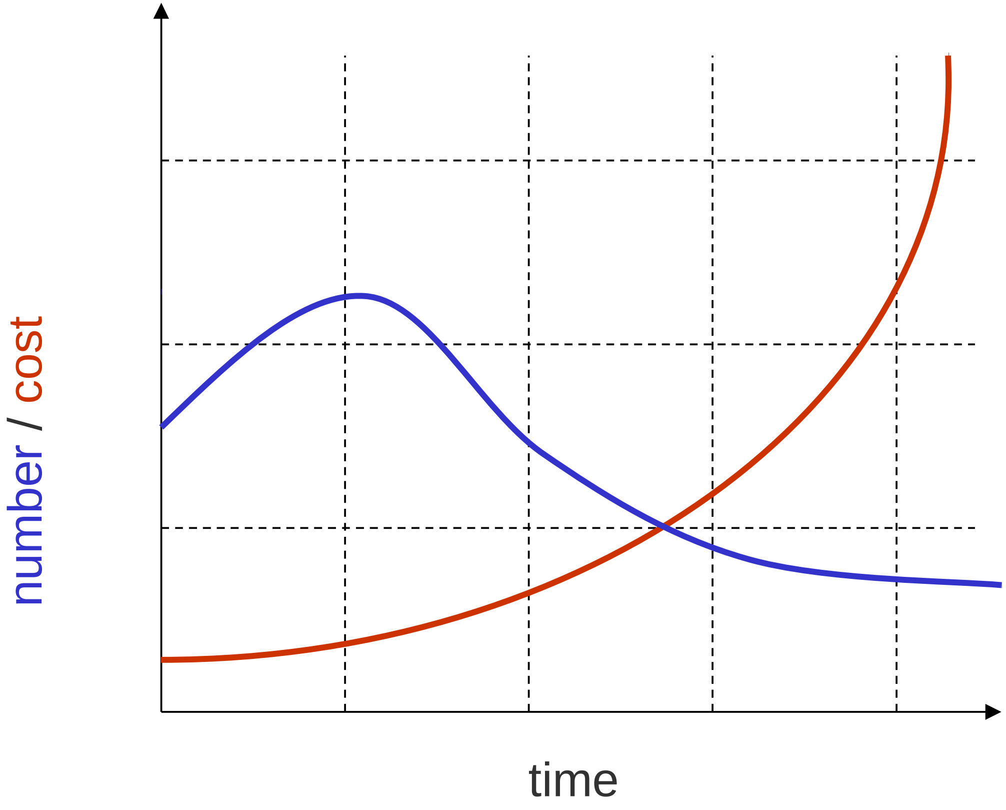
A bold claim

- “The best programmers are up to **28** times more productive than the worst”
  - Sackman, Erikson, and Grant, “Exploratory experimental studies comparing online and offline programming performance” (1968)
- Hold up...
  - **1968**
  - Study involved 12 programmers for an afternoon
  - Designed to compare batch vs. interactive

So what do we know?

- Most errors are introduced during the early stages of development (design and requirements analysis)
- The later an error is detected, the more costly it is to address

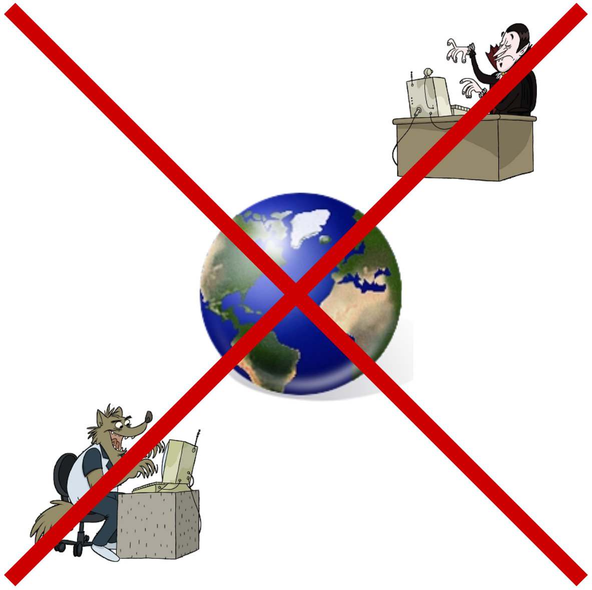
Boehm et al (1975)

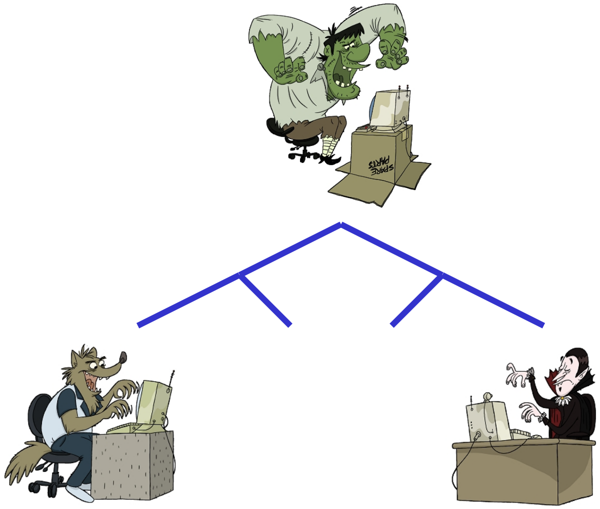


So what do we know?

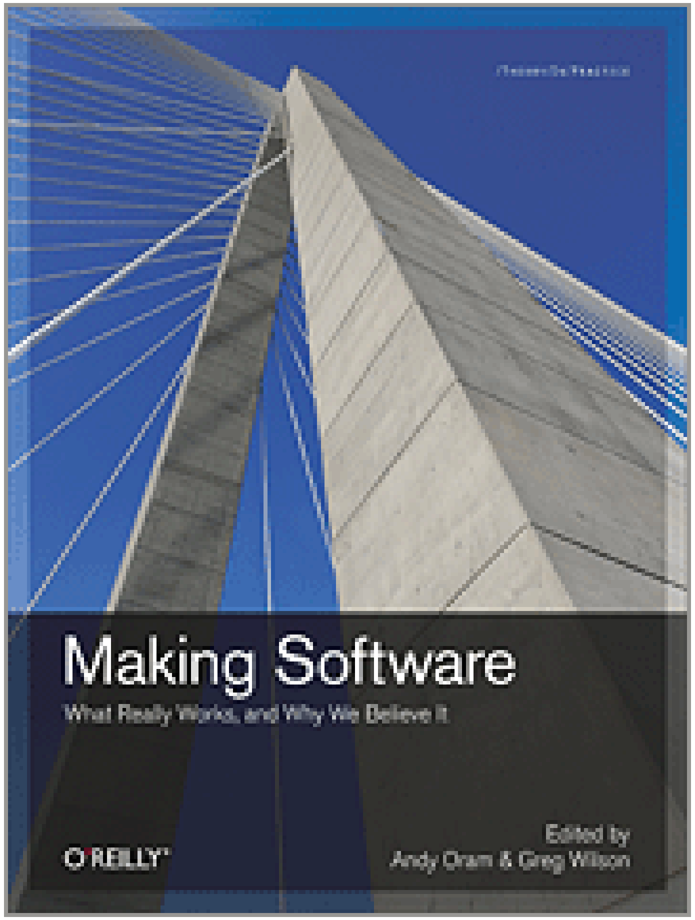
- Physical distance doesn't matter
- Organizational distance does

Nagappan et al. (2007), Bird et al. (2009)





So what do we know?



http://software-carpentry.org/about/biblio.html

Optimization

- What are some things we can optimize in software development?
  - Computing time (often fairly cheap)
  - Programmer time (expensive)
  - Cognitive load
    - Your brain can juggle about 7 ± 2 chunks of information at once in its short term memory

- Whatever we choose to optimize, there should be a reason

Why automate?

- Optimize programmer time: let the machine handle things without your supervision
  - e.g. on a computing cluster
- Optimize cognitive load: record those pesky command line options that you can never seem to remember, and forget them!
- For yourself – *repeatability*
- For others – *reproducibility*

Reproducibility

“Commonly research involving scientific computations are reproducible in principle, but not in practice.”

“In our laboratory, we noticed that after a few months or years, researchers were usually unable to reproduce their own work without **considerable agony**.”

Schwab, Matthias, et al. "Making scientific computations reproducible." Computing in Science & Engineering 2.6 (2000): 61-67.

What we'll learn today

- Program design
  - Computational thinking – how do you approach a problem?
  - How to break your program into logical pieces that are easy to understand, remember, and come back to later

What we'll learn today

- Object-oriented programming (classes, objects)
  - Optimizes cognitive load by allowing us to model the data in the same way we picture it in our minds
  - Structures your code in a logical way, so that when you come back to it in [weeks/months] you'll be able to find it, use it, or modify it
  - Minimize code duplication

What we'll learn today

- Testing
  - As program complexity grows, it becomes less feasible to test manually
  - Proper automated unit tests give you the confidence that changes to your code didn't break important functionality

What we'll learn today

- Shell scripting: “why not just do this from the file browser?”
  - Record a series of actions and you or someone else can repeat those actions later with precision
  - Command-line utilities provide a powerful way to manipulate and analyze files quickly
  - Sometimes you don't have a graphical environment, but the terminal is everywhere

What we'll learn today

- Version control (git): “why not just use (Dropbox/a USB drive/e-mail attachments)?”
  - Share your data and code with others, easily!
    - You can limit this to just collaborators or release it to the public
  - Provenance: keep a record of every change you make to a file, and *why* you made the change

Be a skeptic

- We're claiming that we can improve your efficiency as researchers; do you believe us? Why or why not?
- Make us convince you – don't just take our word for it!
- Do these tools provide an improvement over what you use now?

Logistics

- Etherpad
- Sticky notes
  - No sticky: you're working
  - Green: success
  - Red: you need help or have a question
- Continuous feedback – notecards
- Participate!
  - Ask questions
  - Follow along
  - Point out our mistakes (and we will make mistakes)