Getting computing into the classroom: computational reasoning

Erik Spence

SciNet HPC Consortium

9 October 2014

Who is this guy?

My name is Erik Spence.

- I'm an Applications Analyst at SciNet.
- SciNet is a High-Performance-Computing consortium, one of six in Canada.
- These consortia run massively parallel computers, to perform computations that couldn't be done otherwise.
- At SciNet our two biggest machines have about 32000 and 40000 cores.
- My job is to help users make their code run on these machines.
- My job is also to educate users on how to write fast, efficient code.

Notes for today's class

A few notes before we begin:

- The website for the class can be found here: `http://wiki.scinethpc.ca/wiki/index.php/Teacher_PD`
- The slides for today's class can be found there.
- Questions, comments? Email me: `ejspence@scinet.utoronto.ca`

About this class

Why are we here?

- The purpose of this class is to assist you, the teachers, in getting computing into your classrooms.
- This will presumably take the form of developing materials that you can work into your lesson plans.
- Here is a list of topics we might cover:
  - Computational (model-based) thinking and exploration.
  - Simulation and data analysis tools that can be used in the classroom.
  - Basic Python programming.
  - Technologies for large-scale computation, including in-class clusters.
- However, the class will only be successful if you tell us what your needs are, so that we can taylor the class appropriately.

The truth of the matter

Full disclosure: we don't know what we're doing!

- We, at SciNet, are experts in simulation, computation and data analysis.
- We are NOT experts in teaching these materials to high-school students.
- NOR are we experts in developing curricula, class schedules, lesson plans...
- We need your help so that we can help you:
  - Give us feedback.
  - Tell what works and what doesn't.
  - Give us suggestions on how we might approach material and its presentation.
  - Tell us what material to cover.

Simply put: this is the first time that we've attempted to teach teachers. This is an experiment.

Today's class: an introduction to computational reasoning

Goal: understanding that computer models require the merging of mathematics, programming and science.

- Develop a working definition of computational reasoning.
- Understand some ways in which computational reasoning can be infused into teaching.
- Recognize that probability and random numbers are important mathematical ideas that can be modelled using the tools of computational reasoning.
- Understand that probability can be used to simulate real-world phenomena, and make predictions.
- Recognize the importance of graph interpretation skills in understanding model behaviour.

What is computational reasoning?

What is computational reasoning, and why should we teach it?

Computational reasoning is:

- the ability to analyse, visualize and represent data using mathematical and computational tools.
- using computer models to support theory and experimentation in scientific inquiry.
- using models and simulations as interactive tools for understanding complex concepts in science and mathematics.

In a nut-shell, computational reasoning is the use of computational models to advance understanding in science.

Who cares?

Why should we care about computational reasoning?

- It's probably part of your curriculum standards, in one form or another. (If it's not, we humbly suggest that it should be.)
- It's an excellent way of getting students excited about science.
  - developing and programming models.
  - using models to explore physical effects.
  - applying mathematical ideas to real situations.
  - ability to test hypotheses and analyse results.
- All branches of physical science, and many other branches of science, require research to be backed up by simulations.

Computation is the 'third leg' of science, and computational reasoning is needed to use it properly.

Today's example: hands-on probability

Let's do an example lesson of computational reasoning: theoretical probability versus the real world.

- What is the probability of getting a head when you toss a coin?
- If you toss your coin 10 times, will you get an equal number of heads and tails?
- In 1000 tosses, will you get an even split?
- If you toss 10 times and get 10 heads, is it more likely that you'll get tails next time?
- How would you design an (physical) experiment to answer these questions?
- How would you design a simulation (model) to answer these questions?

Probability via the computer

Let's examine the second case: developing a model for flipping coins.

- A computer uses a formula to generate numbers uniformly between 0 and 1 in no discernable pattern ("random numbers").
- To simulate flipping a coin, we generate a random number and use the rule that random numbers less than 0.5 represent heads, and random numbers above 0.5 represent tails.
- If thousands of numbers are generated, approximately half of those numbers should be less than 0.5, and the rest should be greater than 0.5.
- If only 10 numbers are generated, will half be less than 0.5?

Getting started with Python

Before we start coding, we need to set up our coding environment.

1. On the Desktop (or where-ever you wish), create a folder to hold your code.
2. Under the Start menu, go to Anaconda, then Spyder.
3. In the lower-right corner of the Spyder window, type the following command, using the path to the folder you just created:

```
In[1]:
In[1]: cd C:\Users\ejspence\Desktop\my_code
In[2]:
```

This will tell Python to work in the directory which will hold the code for today's class, so that Python can find it.

Coin flipping using Python

How do you run this code?

The random() function generates a random number between 0 and 1.

Recall that the flip is heads if the random number is less than 0.5.

```
# coin_flip.py
import sys, random   # Import packages.

# Get n from the command line.
if (len(sys.argv) == 2):
  n = int(sys.argv[1])
else: n = 10

actual = 0        # Actual # heads.
theory = n / 2    # Theoretical # heads.

# For each flip, get a random number
for i in range(n):
  if (random.random() < 0.5):
    # ...and increment if it's heads.
    actual = actual + 1

print(actual, "heads out of", n)
print("Percent error:", 100.0 *
  abs(actual - theory) / theory)
```

Coin flipping using Python

How do you run this code?

```
In[2]:
In[2]: run coin_flip.py
4 heads out of 10
Percent error: 20.0
In[3]: run coin_flip.py 20
10 heads out of 20
Percent error: 0.0
In[4]: run coin_flip.py 50
24 heads out of 50
Percent error: 4.0
In[5]:
```

The random() function generates a random number between 0 and 1.

Recall that the flip is heads if the random number is less than 0.5.

```python
# coin_flip.py
import sys, random   # Import packages.

# Get n from the command line.
if (len(sys.argv) == 2):
 n = int(sys.argv[1])
else: n = 10

actual = 0        # Actual # heads.
theory = n / 2    # Theoretical # heads.

# For each flip, get a random number
for i in range(n):
  if (random.random() < 0.5):
    # ...and increment if it's heads.
    actual = actual + 1

print(actual, "heads out of", n)
print("Percent error:", 100.0 *
  abs(actual - theory) / theory)
```

Coin flipping questions

Run the code on the previous slide, and consider the following questions:

1. Run the program, flipping your 10 coins 10 times. Record the number of heads that you get on each trial.
2. How do the trials compare to each other?
3. How many heads did you get in total? How does that compare to others in the class?
4. Try running the code with 100 flips. What do you notice about the error, as compared to 10 flips?
5. Try 1000 flips, 10000 flips. What happens to the error?

Advanced coin flipping

Consider a more-advanced question:

- if I flip a coin 5 times, what is the probability of getting 4 heads?
- How can we design an numerical experiment to answer this question?

Advanced coin flipping

Consider a more-advanced question:

- if I flip a coin 5 times, what is the probability of getting 4 heads?
- How can we design an numerical experiment to answer this question?

```
# coin_flip2.py
import random
from numpy import zeros

# Create a function we can use.
def coin_dist():

  num_flips = 5
  num_tries = 200

  # An array of integer zeros.
  ans = zeros(num_tries, dtype = int)

  for j in range(num_tries):
    for i in range(num_flips):
      if (random.random() < 0.5):
        # Add up number of heads.
        ans[j] += 1

  return ans
```

Advanced coin flipping

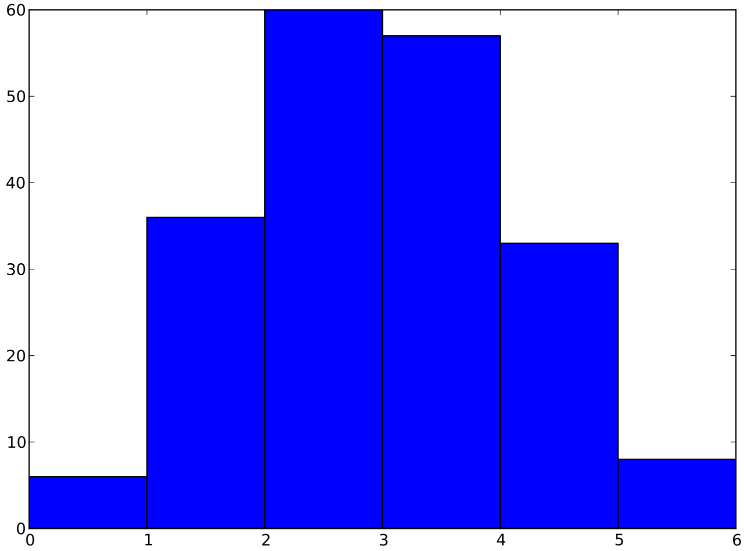
Consider a more-advanced question:

- if I flip a coin 5 times, what is the probability of getting 4 heads?
- How can we design an numerical experiment to answer this question?

```
In[5]:
In[5]: import coin_flip2
In[6]: answer = coin_flip2.coin_dist()
In[7]: from pylab import *
In[8]: hist(answer, bins = range(7))
In[9]:
```

```
# coin_flip2.py
import random
from numpy import zeros

# Create a function we can use.
def coin_dist():

  num_flips = 5
  num_tries = 200

  # An array of integer zeros.
  ans = zeros(num_tries, dtype = int)

  for j in range(num_tries):
    for i in range(num_flips):
      if (random.random() < 0.5):
        # Add up number of heads.
        ans[j] += 1

  return ans
```

Our flipping data

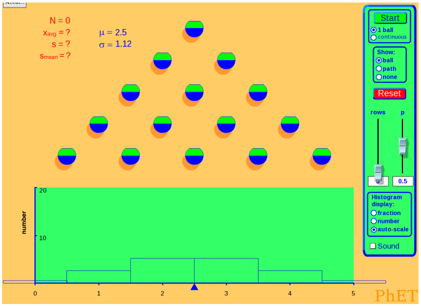


An interactive approach

`http://phet.colorado.edu/sims/plinko-probability/plinko-probability_en.html`

How does it work?

- Set the number of rows to 5.
- Drop balls. If a ball moves to the right, consider it to be a head.
- The histogram at the bottom counts the number of 'heads' you get.
- To get a large number of balls, switch to 'continuous' mode.


N = 0
x_{avg} = ?
s = ?
s_{mean} = ?
μ = 2.5
σ = 1.12
[✓ 1 ball] continuous
Show: [✓ ball] path none
rows 5 p 0.5
Histogram display: fraction number [✓ auto-scale]

More coin flipping questions

Play with the simulation, then consider the following questions:

1. Do just a few dropped balls tell you much about the probability of an individual event?
2. As the number of balls increases, does how does the histogram compare to the histogram on the previous slide?
3. Should the histogram be symmetric? Explain.

Conclusions from the coin flipping simulations

After conducting the simulations, consider these questions:

- Will a simulation that uses random numbers give the same result every time it is run? Explain.
- Is such a simulation a valid representation of reality? Explain.
- What can you learn from a simulation if it doesn't always give the same result? Explain.
- How does the sample size, meaning the number of times the simulation is run, affect the average outcome?

Appendix: our original question

Appendix: for those interested in the answer to the earlier question (what's the probability of getting 4 heads in 5 flips?), the probability of choosing k successes in n independent trials, with probability of success being p, is

$$\binom{n}{k} p^k (1-p)^{n-k} = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

In this case we have $n = 5$, $k = 4$ and $p = 1/2$, which gives

$$\frac{5!}{4!1!} \left(\frac{1}{2}\right)^4 \left(\frac{1}{2}\right)^1 = \frac{5}{32} = 0.15625$$

Appendix: our original question

Appendix: for those interested in the answer to the earlier question (what's the probability of getting 4 heads in 5 flips?), the probability of choosing k successes in n independent trials, with probability of success being p, is

$$\binom{n}{k} p^k (1-p)^{n-k} = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

In this case we have $n = 5$, $k = 4$ and $p = 1/2$, which gives

$$\frac{5!}{4!1!} \left(\frac{1}{2}\right)^4 \left(\frac{1}{2}\right)^1 = \frac{5}{32} = 0.15625$$

The number from the graph was about $\dfrac{33}{200} = 0.165$.

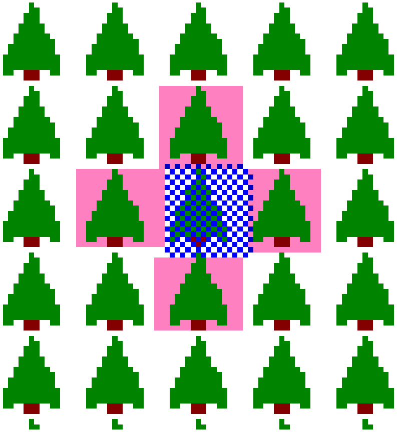
Another probability exercise: forest fire simulation

We will run a forest-fire simulation to explore:

- Probability
- Averages
- Predictions and hypothesis-testing
- Assumptions

Possible guiding questions:

- What can we learn by observing this simulation?
- What should we look for?

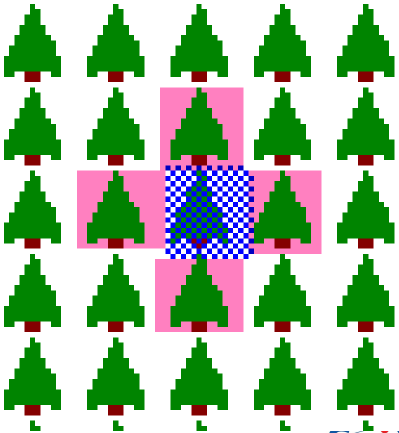


Simulating a forest fire

```
http://www.shodor.org/interactivate/activities/Fire
```

How does it work?

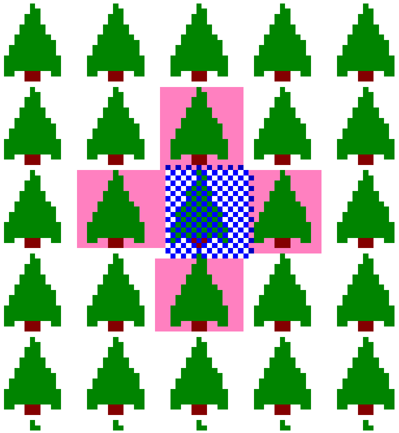
- Set the probability. This indicates the likelihood that an adjacent tree will catch on fire.
- Start a fire by clicking on one of the trees. We imagine this to be like a lightning strike.
- Note the percentage of trees burned.



Simulating a forest fire, continued

`http://www.shodor.org/interactivate/activities/Fire`

- Does the percent of trees burned stay the same for a given burn probability?
- How does the percentage of burned trees change as the burn probability is changed?
- Does the location of the lightning strike affect the percentage of trees burned?



Forest fire data collection

Question: How do you think the percentage of trees burned is related to the burn probability?

Procedure:

- You will be assigned a burn probability. Run the simulation 10 times and average your results.
- Share your average with the class to create a comprehensive dataset.
- Sketch a graph of percentage burned versus burn probability. Or better yet, plot it using python.

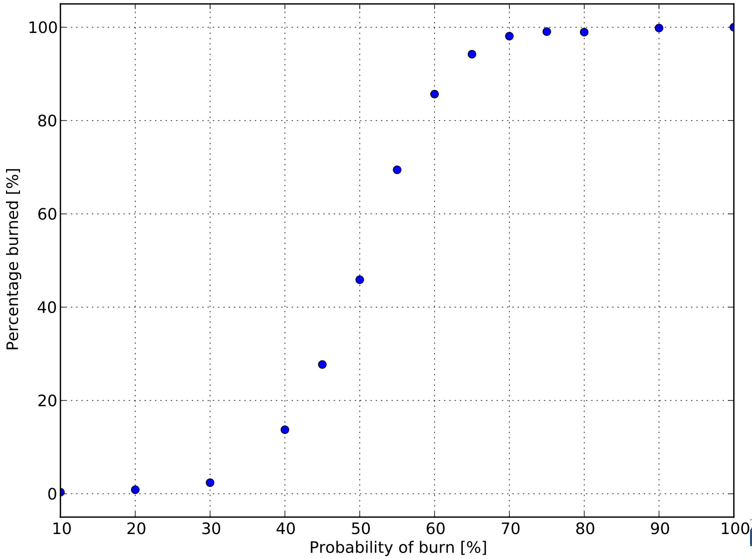
```
In[9]:
In[9]: from numpy import mean
In[10]: data = [21.1, 10.38, 9.68, 3.11, 0.34, 7.26, 25.95, 21.1,
 32.17, 6.22]
In[11]: print(mean(data))
Out[11]: 13.73
In[12]:
```

Plotting our data

How do we plot the results? We use the pylab module, which is part of matplotlib.

```
In[12]:
In[12]: from pylab import *
In[13]: percent = [10, 20, 30, 40, 45, 50, 55, 60, 65, 70, 75, 80, 90, 100]
In[14]: data = [.34, .86, 2.38, 13.73, 27.71, 45.88, 69.44, 85.67, 94.22,
  98.09, 99.06, 98.95, 99.83, 100]
In[15]: plot(percent, data, 'o')
In[16]: ylim([-5, 105])
In[17]: xlabel("Probability of burn [%]")
In[18]: ylabel("Percentage burned [%]")
In[19]: grid()
In[20]:
```

Plotting our data, continued



Evaluating the simulation

Questions about what we have seen:

- How realistic is this simulation? What is wrong with it?
- What are its limitations?
- What are some other factors that influence the spread of a forest fire?
- How could the simulation be modified to incorporate these other factors?

Flipping coins and forest fires

The three simulations which we have used

- What is similar about the three simulations we have run today?
- What is different about the three simulations?
- How might this impact your teaching about the concept of probability?

What have we learned?

What have we learned in today's class?

- Uncertainty in the real world can be modelled using random number generators.
- Model outcomes will vary when random numbers are used to model probabilities, but trends can be observed through graphs of data collected with multiple runs.
- The assumptions behind a computer model must be made explicit to understand the model, especially its shortcomings.
- Computer models can be used to test predictions about the behaviour of a system under varying conditions.
- Problems that seem different on the surface may have characteristics in common when looked at from a modelling perspective.

But more importantly, how are you going to use this?

Assignments for next class

Your homework:

1. In the next week or so, email me with ideas, thoughts, hopes, dreams about what you might like to get out of the course:
   - just a bunch of lesson plans?
   - a focus on fun widgets that can supplement existing lessons?
   - a greater focus on programming?
   - a focus on cluster computing?
   - something else?
2. Next class (October 30), bring in a rough outline of your class plans for the next two weeks. We will discuss how the material from next class might be used in your plans.