IAML Online: The First Practical

October 29, 2015

TODO: Add in intro to pair programming. This should take roughly an hour, allowing for time for an additional chat.

1 Learning Objectives

Python: This is an opportunity for you to review basic Python concepts such as libraries and functions.

Machine Learning: You will learn how to simulate a data set, reinforce your understanding of the binomial distribution, how to create a histogram, discover the substantial variation in outcomes that can be generated by the same probability distribution, and see how statistical models can be linked to real-world outcomes.

Programming: This will be your first time programming in a pair. You will learn how to set up your connection and work together productively.

Reprise: You have already installed and become familiar with iPython in Practical 0. Here, you will need two types of cell, the `markdown` cell, where you document what you are doing, and the `code` cell, where you type in Python code. **Remember to run each cell after you have added it.**

2 Introduction to Pair Programming

TBD

Chat: background, why taking IAML, machine and configuration, type of connection, from where are you connecting.

Initial screen sharing, initial IMs

Configuration: web browser with iPython and one more tab for looking up stuff, either Google Hangout or Skype set up to allow both screen sharing and instant messaging. If you are the helper and want to pass on coding suggestions to the coder, you can use the Instant Messaging system.

3 The Practical

The productivity of the Unseen University's School of Nobody's Business is at an all-time low, even after all social media and gaming sites have been blocked. The School's Strategy Committee decides that excessive amounts of spam email are the problem - staff must be spending too much time dealing with unsolicited emails. So they want to install a spam blocker. As the Committee's resident Big Data Wizard, they have asked you to produce an estimate of the amount of time the University's preferred spam blocker, From Spamalot to Camelot (FSC), will save staff.

In your own experience, around 70% of your inbox are ham (legitimate email). This includes emails from various university committees. 30% are spam. You time yourself, and find that you need 5 seconds to deal with a spam email. You know that you get around 200 emails per day.

Start: Coder: Person 1, Observer: Person 2

Prepare by adding a code cell that imports the libraries `numpy as np` and `matplotlib.pyplot as plt`

Task 1: How much time (in minutes) do you lose to spam per day?
Implementation: Add a markdown cell where you describe the main parameters. Add a code cell that contains variables for each of the main parameters. Add a markdown cell that describes the calculation. Add a code cell where you perform the appropriate calculation.

Task 2: Simulate a typical day and compute how much time you lose to spam.
Implemenation: Add a markdown cell with the task description. Add a code cell that uses `random.binomial` from `numpy` to generate the number of spam emails, and a second code cell that then converts that information into the number of minutes lost.

P2: Check out the function's entry in the scipy documentation in your browser, so that you can help with any questions. You can also use the `help()` function.

Task 3: Now repeat the simulation for a whole year, assuming 250 work days, using new markdown and code cells. Compute the number of spam emails for each work day.
Hint: You can use the same function as before.

Handover: Coder: Person 2, Observer: Person 1

P1, save your workbook in .json format and commit it to the svn. P2, update your svn and load the workbook into your iPython instance. Run all cells.

Task 4: Using pyplot (which you imported as plt), plot the number of spam emails received.
Add a markdown cell before the plot where you explain the content of the plot.

P1: It is now your turn to help. Here is the tutorial for pyplot. The library works in a way that is similar to several other plotting algorithms, including R's `ggplot` package. You first specify the properties of the graph to be plotted, then use `plt.show()` to actually plot it.

Task 5: Compute the input for a histogram of the spam frequencies, using a bin width of 1. Derive the number of bins required from the maximum and minimum of the spam frequencies.
Again, don't forget to add a markdown cell before the actual computation.

P1: Here is the documentation of the histogram function.

Task 6: Using the data structure generated by the histogram, write a function that tells you the number of days when you get a given number of spam emails, and execute it for days=50, 60, 70. What is special about the number 60? Hint: Go back to the very first code cell. On what percentage of days does the simulation actually show 60 spam emails?
Use six cells: A markdown to explain the function, a code cell for the function itself, a markdown for the computations, a code cell where you generate the number of days, a markdown to explain that you are now looking at the percentage of days, and a code cell where you generate the percentage.

P1: Help on working with lists and arrays is here.

Handover: Coder: Person 1, Observer: Person 2

Follow the protocol described above: Save your worksheet, commit, check out, load, run to make sure your instance of python is up to date.

Task 7: Write a function that takes as input the output of the histogram function, and that plots a histogram with the x axis limits between min(bin edges)-5 and max(bin edges)+5, and the y axis limits between 0 and max(frequencies) + 5.
Hint: Histograms are bar charts, which can be plotted as

```
plt.bar(bin_edges[:-1], frequencies, bin_width = 1)
```

. Remember to add a markdown cell explaining the function before the function itself.

P2: Make sure you have the tutorial for pyplot on hand to help with setting axis limits.

Task 8: How much time would a spam detector save over a year? Compute the time saved in minutes for each day and determine mean, median, standard deviation, and total sum, and convert the total sum into hours.

P2: Here is some documentation on basic statistics in numpy.

3.0.1 Handover: Parallel Work

Follow the protocol described above.

Task 9: **Both** create a code cell that generates the number of spam emails you get in a year, then plots the histogram, gives you the time saving statistics from Task 8, and tells you on how many days the person received exactly 60 spam emails. Copy the cell 4 more times, make sure to assign different variable names to the array of spam statistics in each cell (e.g., Spam1, Spam2, Spam3, Spam4, Spam5), and run each instance.

How much variation do you see?

Task 10: Discuss what could be done to make the simulation more realistic. Is the number of emails likely to stay the same during the week? Might the frequency of spam emails show patterns of variation? If you have time, experiment with different spam frequencies and different times for dealing with spam emails.