Unit testing: what's it for?

An automated way of ensuring:

- That your code's API works (i.e. that others can use it)
- That the individual parts of your code work correctly
- That you don't break your code when changing it

Also useful when developing a tricky algorithm (using test-first, or test-driven development)

Version control: what's it for?

Software to help keep track of changes made to files

Tracks the **history** of your work:

- How do you get back the working version you had last week?
- Can you reproduce the results from last year's journal paper?

Helps you **collaborate** with others:

- Check you all have the same version of the same code
- Share and merge your changes

Popular examples include git, Mercurial, Subversion

An awkward question

"How do I know your results come from the method you're describing, and not from bugs in your software?"

Validation and verification

Validation: Establishing how well your model represents reality

- Research papers are expected to do this

Validation and verification

Validation: Establishing how well your model represents reality

- Research papers are expected to do this

Verification: Establishing that you have implemented your model

- This is what formal software testing is about
- It's also what most informal good practice aims at
- Research papers seldom attempt any of this

“Feedback cycles”

Write code → learn how code is wrong → change or fix code

- Software development “best practice” is often about trying to shorten or simplify this cycle
- Learn about your mistakes as early as possible

What we're about to do

Simple audio analysis using a test-driven development process

Software used:

- Python 2 (version 2.6 or 2.7, not Python 3)
- NumPy – Python numerical modules
- Nose – Python unit-test framework
- EasyMercurial – Version control user interface

Optionally:

- IPython – improved interactive console for Python
- Matplotlib – plotting modules