



SNP Pipeline Code Review

Hugh Rand and Yan Luo 4/4/2014

Thanks to

- Yan - graciously letting me rearrange her code.

- Jamie - for help testing and troubleshooting.

- John I. - for an idea about how do one thing.

- Errol - for prodding me along.

- Fish - for prodding on code reviews.

Code History

===== Errol - 1st version Yan - 2nd version Jamie and Yan - use for publication

Errol, ... - use for outbreak investigation CDC pipeline (Lee Katz) <https://github.com/lskatz/lyve->

SET

Goals

===== Make it publishable - CDC person - CVM person

Make it a package Reuseable code Easily installed Locked down Testable

Some background

===== Running with anaconda Using python 2.7.6 PyVCF (If you want to

get good at writing, then read a fair bit.) (But, be judicious about what you read.) reasonably

well written. does something useful for me. `$python -m easy_install PyVCF` 131 Searching for

PyVCF 132 Best match: PyVCF 0.6.7 133 Processing PyVCF-0.6.7-py2.7-linux-x86_64.egg 134

PyVCF 0.6.7 is already the active version in easy-install.pth 135 Installing vcf_filter.py script to

/home/hugh.rand/anaconda/bin 136 Installing vcf_melt script to /home/hugh.rand/anaconda/

bin

The 'ecosystem'

Python instalation - Anaconda IDE - Spyder Code standards PEP8 Google Code Testing pylint

[unittest](#) [doctest](#) [Distribution](#) [distutils](#) [setuptools](#) [Documentation](#) [PEP8](#) [sphinx](#) [Source Code](#)

Control - git Presentations

What I did / What is to do

- Add comments

- Broke much of code out as functions

- Pythonify names of functions and variables

- Create tests

- All global variables gone

- Break up code into 'main' and utilities

- Move main 'script' into function

- Update argument passing package (2.6->2.7)

- Organize code as a python package

- Move from threads to processes

- General cleanup

- Add use of PyVCF

- Improve pylint score

- Add release versioning and all under version control

- Add new flags (includeReference, various parameters).

- Add new flags (verbose, useOldPileups).

- Trap keyboard interrupts so can halt code cleanly.

- Turn it into an "Egg"

- Get it out on gitHub

- Move to SVN

Package layout

snppipeline build dist doc LICENSE.txt MANIFEST.in notes presentations README.txt setup.py

snppipeline README_developmentNotes snppipeline.py utils.py test codeComparisonFiles

testAgonaMOM testLambdaVirus test_snppipeline.py test_utils.py

snppipeline.egg-info

Functions (no classes)

```
egrep 'def|\\\"\\\"' snppipeline/snppipeline.py def run_snp_pipeline(options_dict): """Create SNP
```

matrix


```
egrep 'def|\\\"\\\"' snppipeline/utils.py def pileup_wrapper(args): """Wraps pileup to use multiple
```

arguments with multiprocessing package. def pileup(filePath, options_dict): """Run samtools to

```
generate_pileup.  
def get_consensus_base_from_pileup(base, length, data): """Call the base for
```

each SNP position

```
def create_consensus_dict(pileup_file_path): """Create a dict based on the
```

information in a pileup file. def write_list_of_snps(file_path, snp_list_dict):

```
"""Write out list of snps for all samples to a single file. def
```

```
write_reference_snp_file(reference_file_path, snp_list_file_path, ""Write out the snp fasta file
```

for the reference.fasta using the snp

Function call graph

pycallgraph

Testing environment

virtual machine vmware - vmplayer ubuntu install process samtools pyvcf ????

Where to put the code?

svn: <https://xserve19.fda.gov/svn/bioin/mapping?>

Code layout

pylint

2014-03-31:

pylint snppipeline/snppipeline.py Your code has been rated at 2.42/10

pylint snppipeline/utils.py Your code has been rated at 6.15/10

2014-04-01:

```
pylint snppipeline/utils.py | grep "rated" No config file found, using default configuration Your
```

code has been rated at 7.72/10 (previous run: 7.72/10)

```
pylint snppipeline/snppipeline.py | grep "rated" No config file found, using default
```

configuration Your code has been rated at 5.68/10 (previous run: 5.68/10)

The git log

Making the code public

github?

The tests

===== * Packages unittest doctest * Some simple unit tests * Two integration tests

lambda virus agona

More on testing

```
./test/test_utils.py -v ... ok 5 items had no tests: utils utils.pileup utils.pileup_wrapper
```


utils.write_list_of_snps utils.write_reference_snp_file 2 items passed all tests: 3 tests in

utils.create_consensus_dict 9 tests in utils.get_consensus_base_from_pileup 12 tests in 7

items. 12 passed and 0 failed. Test passed. ok

snppileline.py

```
./test/test_snppipeline.py -v test_snppipeline_agona (main.Test) Run snppipeline with agona 5
```

samples example. ... [mpileup] 1 samples in 1 input files ... Match, Mismatch, Errors: 8, 0, 0 ...

ok test_snppipeline_lambda_virus (**main**.Test) Run snppipeline with synthetic virus example. ...

[mpileup] 1 samples in 1 input files ... Match, Mismatch, Errors: 7, 0, 0 ... ok

Test output

References

=====

Python Eggs

A "Python egg" is a logical structure embodying the release of a specific version of a Python

project, comprising its code, resources, and metadata. There are multiple formats that can be

used to physically encode a Python egg, and others can be developed. However, a key

principle of Python eggs is that they should be discoverable and importable. That is, it should

be possible for a Python application to easily and efficiently find out what eggs are present on

a system, and to ensure that the desired eggs' contents are importable.

The .egg format is well-suited to distribution and the easy uninstallation or upgrades of code,

since the project is essentially self-contained within a single directory or file, unmingled with

any other projects' code or resources. It also makes it possible to have multiple versions of a

project simultaneously installed, such that individual programs can select the versions they

wish to use.

(<http://stackoverflow.com/questions/2051192/what-is-a-python-egg>)

Building Python Eggs

Packages distutils setuptools Code setup.py