labibi Documentation

Release 0.1

C. Titus Brown

November 09, 2016

Contents

See http://ged.msu.edu/courses/2013-fall-cse-801/ for class information.

CHAPTER 1

Helpful resources

1.1 Learning Python

- Python for Biologists: http://pythonforbiologists.com/
- Practical Computing for Biologists

  This is a highly recommended book for people looking for a systematic presentation on shell scripting, programming, UNIX, etc.

1.2 Shell tutorials and references

https://github.com/JorySchossau/shell

http://explainshell.com/

- Biostar

  A high quality question & answer Web site.

- SEQanswers

  A discussion and information site for next-generation sequencing.

- Software Carpentry lessons

  A large number of open and reusable tutorials on the shell, programming, version control, etc.

1.3 Additional Web sites

The NGS course tutorials: http://ged.msu.edu/angus/tutorials-2013/

UW Seattle "zero entry" workshop: http://2013-uw-zero-entry.readthedocs.org/

1.4 Recommended books and other resources

- Practical Computing for Biologists

  This is a highly recommended book for people looking for a systematic presentation on shell scripting, programming, UNIX, etc.

1.5 Computing and Data

- A Quick Guide to Organizing Computational Biology Projects, Noble, PLoS Comp Biology, 2009.
- Willingness to Share Research Data Is Related to the Strength of the Evidence and the Quality of Reporting of Statistical Results, Wicherts et al., PLoS One, 2011.
- Got replicability?, McCullough, Economics in Practice, 2007.

Also see this great pair of blog posts on organizing projects and research workflow.

1.5.1 Other Links

1.6 Humor

- Data Sharing and Management Snafu in 3 Short Acts

1.7 Blogs

- http://www.genomesunzipped.org/

  Genomes Unzipped.
- http://ivory.idyll.org/blog/

  Titus's blog.
- http://bcbio.wordpress.com/

  Blue Collar Bioinformatics
- http://massgenomics.org/

  Mass Genomics
- http://blog.nextgenetics.net/

  Next Genetics
- http://gettinggeneticsdone.blogspot.com/

  Getting Genetics Done
- http://omicsomics.blogspot.com/

  Omics! Omics!

Homework 9 (due Tuesday, December 10, at 11:59pm PST)

2.1 1. Tutorials

I've prepared several tutorial videos for this homework.

Working with the evolutionary digital model: http://www.youtube.com/watch?v=kXKJQHaFWnU

Saving data to a file in Python: http://www.youtube.com/watch?v=irnj19jz8uI

Computing and plotting estimated 95% confidence intervals in Python: http://www.youtube.com/watch?v=4N5Uo3XOTNQ

There are also several useful tutorials below.

pandas & SciPy tutorial: http://www.randalolson.com/2012/08/06/statistical-analysis-made-easy-in-python/

pandas video tutorial: http://vimeo.com/59324550

matplotlib tutorial: http://matplotlib.org/users/pyplot_tutorial.html

2.2 2. Final project report

This week, you will prepare a 2-page final report on your findings from homework 8. Prepare it as a scientific report:

- introduction to the problem and what you're studying
- motivate why it's interesting
- discuss related work (optional, but may strengthen your case for why it's interesting)
- explain your methods for exploring the problem
- show the data that you measured from your experiments
- use statistics and your data to show what you discovered
- discuss the possible implications of the findings
- discuss the limitations of the findings
- suggest possible followup projects

Deliverables should include:

- Final project report in PDF format
- Code for the extended model
- Raw data and code used for all data analysis and plotting (as a nbviewer link)

Make sure to include the names of everyone in your group in the final project report. Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

CHAPTER 3

Wed, Dec 4, 2013 – The future of digital evolutionary modeling

3.1 Lectures

We will have lectures on the following topics:

Realistic simulations, e.g. EvoSphere: http://www.evosphere.org/

Hybrid digital/biological models, e.g. https://www.sciencemag.org/content/337/6099/1181.summary (viewable w/ MSU subscription)

Robots! e.g., http://users.ox.ac.uk/~bioc1048/Research_files/floreanoetal07.pdf

3.2 Homework 9 - Final project report

Prepare a 2-page report for your final project. Only the representative for your group needs to turn the report in. See details in the homework 9 section.

Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

CHAPTER 4

Wed, Nov 27, 2013 – pandas, SciPy, and matplotlib for data analysis, statistics, and plotting

4.1 Installing pandas on EC2

Since pandas does not come installed by default on EC2, we need to install it ourselves. Run the following commands to update your EC2 node:

```
# Update your apt-get:
apt-get update

# Pre-requisities
apt-get install build-essential gfortran gcc g++ curl wget python-dev

# Make sure you have the latest setup tools
wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py -O - | python2.7

# Get pip
curl --show-error --retry 5 https://raw.github.com/pypa/pip/master/contrib/get-pip.py | python2.7
```

The pandas module is built off of the NumPy module. Now we need to update NumPy to the latest version of it. Unfortunately, the automatic upgrading tools on EC2 don't seem to work right, so we need to update it manually:

```
# download the latest version of NumPy
wget --no-check-certificate https://pypi.python.org/packages/source/n/numpy/numpy-1.8.0.zip#md5=6c918

# unzip the NumPy file and move into the install directory
unzip numpy-1.8.0.zip
cd numpy-1.8.0/

# build NumPy
python setup.py build

# install NumPy
python setup.py install

# move back to the home directory and clean up
cd ~/
rm -r numpy-1.8.0/
rm numpy-1.8.0.zip
```

Finally, we can install pandas:

```
pip install pandas
```

You should be able to use pandas on your EC2 nodes now. Try it out:

```
# download some test data
wget https://raw.github.com/rhiever/ipython-notebook-workshop/master/parasite_data.csv

# import pandas and read some data
from pandas import *

test_data = read_csv("parasite_data.csv")
print test_data
print test_data["Virulence"]
```

4.2 pandas, scipy, and matplotlib

We will go over some tutorials on pandas, SciPy, and matplotlib.

pandas & SciPy tutorial: http://www.randalolson.com/2012/08/06/statistical-analysis-made-easy-in-python/

pandas video tutorial: http://vimeo.com/59324550

matplotlib tutorial: http://matplotlib.org/users/pyplot_tutorial.html

CHAPTER 5

Homework 8 (due Tuesday, November 26, at 11:59pm PST)

5.1 1. IPythonBlocks

See the IPythonBlocks introduction from earlier in the semester if you need a refresher for IPythonBlocks.

5.2 2. Tutorial videos

I've prepared several tutorial videos for this homework.

Working with the evolutionary model: http://www.youtube.com/watch?v=kXKJQHaFWnU

Saving data to a file in Python: http://www.youtube.com/watch?v=irnj19jz8uI

Computing and plotting estimated 95% confidence intervals in Python: http://www.youtube.com/watch?v=4N5Uo3XOTNQ

5.3 3. Final project

Download the evolutionary digital modeling homework notebook. You will be using the model from this notebook for the final project.

For the first week of the final project, you will break into groups of up to 4 and decide as a group on an interesting (and feasible^) extension of the model. You may work alone on the final project, but I do not recommend it. Once you've decided on a group, assign a group representative. They will be the one to send in all homework submissions and reports for the group.

Possible project ideas:

- food regrowth/decay
- starvation
- predation
- topographical barriers
- mating and recombination
- direction competition
- migration
- explore a variety of world sizes, patch sizes, and patch numbers to investigate how evolution of ARS changes in response to these variables.

- add a predator
- add different types of food resources with different values
- add poison food type that harms the forager

I expect your group to formulate a testable hypothesis, with alternative hypotheses, that the extension of this model will explore. You should also begin running the model in replicate and gathering data to explore your hypotheses. I do not expect you to have all of the data gathered by next Wednesday, but I do expect to see reasonable progress.

Deliverables for this week should include:

- List of everyone in the group (including email addresses)
- Brief explanation of the hypotheses your group is exploring and why
- Code for the extended model
- Any progress on data gathering/analysis so far

Turn in your progress report via the usual method (commit a notebook to Github + email me the nbviewer link). Be sure to include the data you used to generate any graphs.

Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

^ Please contact me if you're unsure about the extension you want to pursue.

CHAPTER 6

Wed, Nov 20, 2013 – Running evolutionary models in replicate, calculating and plotting 95% confidence intervals

6.1 We will have class next week, Nov. 27th

For those who missed class last week, we *will* be having class next week, Nov. 27th.

Next week will be a hands-on workshop covering some useful Python scientific computing libraries. Be sure to bring your laptop.

6.2 Homework 7 solutions

We will go over the solution to the evolutionary digital modeling homework.

This homework was tough, so we will dedicate plenty of class time to walk through it. Come to class with specific questions about the issues you ran in to.

Here is the notebook that we covered in class showing you how to plot the average and SEM for multiple files: http://nbviewer.ipython.org/7572328

6.3 Homework 8 - Final project

We will be expanding the model from the evolutionary digital modeling homework for a final project. This project will cover the next two weeks. You may work on this project in groups of up to 4 people, but you can do a solo project if you prefer.

For the first week, you will decide as a group on an interesting (and feasible^) extension of the model and begin implementing it and gathering data from the model. For the second week, you will analyze the data, plot the data, and write up a 2-page report.

Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

^ Please contact me if you're unsure about the extension you want to pursue.

CHAPTER 7

Homework 7 (due Tuesday, November 19, at 11:59pm PST)

7.1 1. IPythonBlocks

See the IPythonBlocks introduction from earlier in the semester if you need a refresher for IPythonBlocks.

See the class homework that explains the relevant details for this homework

7.2 2. Tutorial videos

I've prepared several tutorial videos for this homework.

Working with the evolutionary model: http://www.youtube.com/watch?v=kXKJQHaFWnU

Saving data to a file in Python: http://www.youtube.com/watch?v=irnj19jz8uI

Computing and plotting estimated 95% confidence intervals in Python: http://www.youtube.com/watch?v=4N5Uo3XOTNQ

7.3 3. Homework

Download the evolutionary digital modeling homework notebook.

All of the homework problems are listed in the notebook. Turn in the completed notebook via the usual method (commit to Github + email me the nbviewer link). Be sure to include the data you use to generate the graphs.

Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

CHAPTER 8

Wed, Nov 13, 2013 – More plotting, data wrangling, and coding an evolutionary digital model

8.1 Class on Nov. 27th?

Nov. 27th is the day before Thanksgiving. I don't have a lecture planned for that day, but I can run a hands-on workshop to teach you how to use some advanced data management and plotting tools that will be valuable for your final project and future research if you work with data.

Do we want to have class on Nov. 27th?

Having class means I will be there to help you debug as we walk through the tools. You will also have video tutorials to optionally watch afterward.

Not having class means you will only have online video tutorials and emails to help you with these tools.

8.2 Homework 6 solutions

We will briefly go over the solution to the Avida homework assignment.

We will cover how to wrangle and plot the data after Avida has generated it. Video here: http://www.youtube.com/watch?v=gyBI6arfMzs

We will also discuss the biological implications of the experiment as a class.

8.3 Model in Homework 7

We will walk through the model in the evolutionary digital modeling homework. You will be working with this model for your homework for the rest of the semester.

Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

CHAPTER 9

Homework 6 (due Tuesday, November 12, at 11:59pm PST)

9.1 1. Avida

Avida documentation is here: http://avida.devosoft.org/documentation/page/Main_Page

The latest stable version of Avida can be downloaded here: http://avida.devosoft.org/download/

See the class homework that explains the relevant details for this homework

Videos are available showing you how to download and install Avida and how to work with Avida once it's installed.

9.2 2. Homework

Download the Avida homework notebook.

All of the homework problems are listed in the notebook. Turn in the completed notebook via the usual method (commit to Github + email me the nbviewer link). Be sure to include the data you use to generate the graphs.

Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

CHAPTER 10

Wed, Nov 6, 2013 – Introduction to evolutionary digital modeling

10.1 Evolutionary digital modeling

For the first portion of the evolutionary digital modeling (EDM) classes, we will go over what EDMs are and why they are used. Several experts who use EDMs in their research will give brief introductions to the three major types of EDMs.

Arend Hintze will be giving a 30-minute introduction to evolutionary game theory (EGT).

Chris Adami will be giving a 30-minute introduction to Avida / digital evolution.

Randy Olson will be giving a 20-minute introduction to embodied models of artificial life.

10.2 Homework

We will briefly go over the Avida homework assignment.

Email the homework solutions to Randy Olson: olsonran AT msu DOT edu

CHAPTER 11

Homework 5 (due Tuesday, November 12, at 11:59pm PST)

For solutions, see `hw5-solutions`.

11.1 1. Fitting

Do a linear fit to each of the Anscombe's Quartet data sets, and plot the lines and data points. Here is some simple example code for doing a linear fit to x and y in Python:

```
x = [1,2,3,4]
y = [3,5,7,10] # 10, not 9, so the fit isn't perfect

fit = polyfit(x,y,1)
fit_fn = poly1d(fit) # fit_fn is now a function which takes in x and returns an estimate for y

plot(x,y, 'yo', x, fit_fn(x), '--k')
```

Submit as usual (done in an IPython Notebook, submitted via a gist; send me the nbviewer URL).

11.2 2. Variant calling

(Please do either this, or expression analysis, or both. You only need to do one.)

The below data is from one of Rich Lenski's LTEE papers, the one on the evolution of citrate consumption in LTEE.

11.2.1 Install software

You'll want an m1.large or m1.xlarge for this.

First, we need to install the BWA aligner:

```
cd /root
wget -O bwa-0.7.5.tar.bz2 http://sourceforge.net/projects/bio-bwa/files/bwa-0.7.5a.tar.bz2/download

tar xvfj bwa-0.7.5.tar.bz2
cd bwa-0.7.5a
make

cp bwa /usr/local/bin
```

We also need a new version of samtools:

```
cd /root
curl -O -L http://sourceforge.net/projects/samtools/files/samtools/0.1.19/samtools-0.1.19.tar.bz2
tar xvfj samtools-0.1.19.tar.bz2
cd samtools-0.1.19
make
cp samtools /usr/local/bin
cp bcftools/bcftools /usr/local/bin
cd misc/
cp *.pl maq2sam-long maq2sam-short md5fa md5sum-lite wgsim /usr/local/bin/
```

11.2.2 Download data

Download the reference genome and the resequencing reads:

```
cd /mnt

curl -O http://athyra.idyll.org/~t/REL606.fa.gz
gunzip REL606.fa.gz

curl -O ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR098/SRR098042/SRR098042.fastq.gz
```

Note, this last URL is the "Fastq files (FTP)" link from the European Nucleotide Archive (ENA) for this sample: http://www.ebi.ac.uk/ena/data/view/SRR098042.

11.2.3 Do the mapping

Now let's map all of the reads to the reference. Start by indexing the reference genome:

```
cd /mnt

bwa index REL606.fa
```

Now, do the mapping of the raw reads to the reference genome:

```
bwa aln REL606.fa SRR098042.fastq.gz  > SRR098042.sai
```

Make a SAM file (this would be done with 'sampe' if these were paired-end reads):

```
bwa samse REL606.fa SRR098042.sai SRR098042.fastq.gz > SRR098042.sam
```

This file contains all of the information about where each read hits on the reference.

Next, index the reference genome with samtools:

```
samtools faidx REL606.fa
```

Convert the SAM into a BAM file:

```
samtools import REL606.fa.fai SRR098042.sam SRR098042.bam
```

Sort the BAM file:

```
samtools sort SRR098042.bam SRR098042.sorted
```

And index the sorted BAM file:

```
samtools index SRR098042.sorted.bam
```

At this point you can visualize with tview or Tablet.

'samtools tview' is a text interface that you use from the command line; run it like so:

```
samtools tview SRR098042.sorted.bam REL606.fa
```

The '.'s are places where the reads align perfectly in the forward direction, and the ','s are places where the reads align perfectly in the reverse direction. Mismatches are indicated as A, T, C, G, etc.

You can scroll around using left and right arrows; to go to a specific coordinate, use 'g' and then type in the contig name and the position. For example, type 'g' and then 'rel606:553093<ENTER>' to go to position 553093 in the BAM file.

For the Tablet viewer, click on the link and get it installed on your local computer. Then, start it up as an application. To open your alignments in Tablet, you'll need three files on your local computer: `REL606.fa`, `SRR098042.sorted.bam`, and `SRR098042.sorted.bam.bai`. You can copy them over using Dropbox, for example.

11.2.4 Calling SNPs

You can use samtools to call SNPs like so:

```
samtools mpileup -uD -f REL606.fa SRR098042.sorted.bam | bcftools view -bvcg - > SRR098042.raw.bcf
```

(See the 'mpileup' docs here.)

Now convert the BCF into VCF:

```
bcftools view SRR098042.raw.bcf > SRR098042.vcf
```

11.2.5 Examining SNPs

You can load the VCF file with this code in IPython notebook:

```
import csv

def load_vcf(filename):
    fp = open(filename, 'rb')
    r = csv.reader(fp, delimiter='\t')

    snp_calls = []
    for n, row in enumerate(r):
        if row[0].startswith('#'):
            continue
        chr, pos, _, ref, alt = row[:5]
        pos = int(pos)

        snp_calls.append((chr, pos, ref, alt))
    return snp_calls
```

The full filename should be `/mnt/SRR098042.vcf`.

Look at a few of the locations in tview or Tablet. Do you believe the variant calls?

11.2.6 Problems

1. Download the FASTQ.GZ file for SRR098038, and call SNPs.
2. Compare a few the SNPs in SRR098038 with the SNPs from SRR098042. Are the SNPs in one a subset or superset of the SNPs in the other?
3. SRR098038 is one of the Cit+ strains from 38,000 generations. Go find the location of the mutS mutation by finding the mutS gene in the genome of the reference strain here. Do you find the mutation? What is it, precisely? (Coordinates & ref -> variant.)

Hand in your two VCF files, together with your answer for #3.

11.3 3. RNAseq expression analysis

(Please do either this, or variant calling, or both. You only need to do one.)

Here we're going to look at RNAseq differential expression. We're going to be using data generated from running the Eel Pond mRNAseq protocol. You can feel free to run through all of this – it goes through a complete de novo assembly and quantitation of a bunch of mRNAseq data – but the whole thing takes about a week, so ... you might not want to. Below, I've given you access to some of the output files instead :).

11.3.1 Install software

You'll want an m1.large or m1.xlarge for this.

Start by installing R and some R packages:

```
apt-get -y install r-base-core r-cran-gplots
```

Next, install the EBSeq package for calculating differential expression, which comes as part of RSEM, the package we used in khmer-protocols for calculating expression levels:

```
cd /root
curl -O http://deweylab.biostat.wisc.edu/rsem/src/rsem-1.2.7.tar.gz
tar xzf rsem-1.2.7.tar.gz
cd rsem-1.2.7
make
cd EBSeq
make

echo 'export PATH=$PATH:/root/rsem-1.2.7' >> ~/.bashrc
source ~/.bashrc
```

11.3.2 Download data

Create a working directory:

```
cd /mnt
mkdir diffexpr
cd diffexpr

curl -O http://athyra.idyll.org/~t/rsem-data.tar.gz
tar xvf rsem-data.tar.gz
```

This will result in four files, `{1,2,6,7}.fq.genes.results`. These are the first two 0 Hour and the first two 6 Hour time points from the Tulin et al. study on Nematostella development, run through the entire Eel Pond protocol (see khmer-protocols, above).

11.3.3 Calculate differential expression

To do the differential expression calculation with EBSeq:

```
cd /mnt/diffexpr

rsem-generate-data-matrix {1,2,6,7}.fq.genes.results > genes.matrix
rsem-run-ebseq genes.matrix 2,2 genes.changed
```

Here, the '2,2' means there are 2 conditions, each with two replicates.

The EBSeq output will be in 'genes.changed'. Read the docs to understand what's in the output file – you're most interested in the PPDE (posterior probability that a transcript is differentially expressed) and the PostFC (posterior fold change) columns, columns 4 and 5.

11.3.4 Problem: Load in data & plot

1. Please build an IPython Notebook that loads in the sample 1 and sample 6 data sets and plots the expression levels of genes in sample 1 on the X axis and sample 6 on the Y axis; below is some code taken from the in class notebook that you can adapt.
2. Load in the 'genes.changed' file (hint: you only need column 1) and plot the locations of genes that have changed.

   A tip: instead of iterating over all the keys in the sample1 dictionary, you can iterate over just the keys in the dictionary created to hold the results of the 'genes.changed' file.
3. Grab the results in http://athyra.idyll.org/~t/rsem-data2.tar.gz – they are additional replicates (3.fq.genes.results and 8.fq.genes.results). Redo the differential expression analysis & plotting (note: use '3,3' for rsem-run-ebseq).

A function to load in a `genes.results` file:

```
import csv

def load_results(filename):
    r = csv.DictReader(open(filename, 'rb'), delimiter='\t')
    results = {}
    for row in r:
        gene = row['gene_id']
        results[gene] = row

    return results
```

Code to extract the expression values (FPKM) for two samples and put them in numpy array:

```
sample1 = load_results('1.fq.genes.results') # these are replicates
sample2 = load_results('2.fq.genes.results')

sample1v2 = []
for k in sample1:
    s1 = float(sample1[k]['FPKM'])
    s2 = float(sample2[k]['FPKM'])
```

```
    if s1 == 0 or s2 == 0:
        continue
    else:
        sample1v2.append((s1, s2))

sample1v2 = numpy.array(sample1v2)
```

Code to plot sample 1 vs sample 2:

```
x = plot(sample1v2[:,0], sample1v2[:,1], 'bo', alpha=0.1)
ax = axes()
ax.set_yscale('log')
ax.set_xscale('log')
```

Code to load in the '.changed' file:

```
def load_changed(filename, ppee_threshold=.05):
 d = {}
 r = csv.reader(open(filename, 'rb'), delimiter='\t')
 r.next()
 for row in r:
     gene_id, PPEE, PPDE, postfc, realfc = row
     PPEE=float(PPEE)
     PPDE=float(PPDE)
     postfc = float(postfc)
     realfc = float(realfc)
     if PPEE > ppee_threshold: # skip those with probability of equal expression > 0.05
         continue
     d[gene_id] = (PPEE, PPDE, postfc, realfc)
 return d
```

CHAPTER 12

Wed, Oct 30, 2013 – plotting, fitting, RNAseq, and mapping

12.1 HW overview

Let's go over the homework briefly.

12.2 mRNAseq

I have a full de novo mRNAseq pipeline here:

https://khmer-protocols.readthedocs.org/en/latest/

it involves many of the same steps that you have already run in other guises: filter and trim reads, normalize, and assemble. It also includes calling gene families, annotating (naming the sequences something), and doing differential expression calculation.

It takes about a week so I'm not going to ask you to run it :).

However, for the homework, one option will be to do some differential expression analysis, so I thought I'd show you around the data.

12.2.1 Data loading, plotting, fitting

See notebook at: http://nbviewer.ipython.org/7236897

Plotting:

Note, you can explore functioning code from the matplotlib gallery by using %loadpy inside of IPython Notebook – for example, put

```
%loadpy http://matplotlib.org/mpl_examples/showcase/integral_demo.py
```

into a notebook and execute the cell.

So, the point of all this Python stuff is that at some point you may want to investigate your data ...by hand! *gasp* In that case you will need to know, minimally, how to load your data in and poke at it.

12.3 Mapping and variant calling

While we don't have a general protocol for variant calling, it's in some ways easier. Basically, you "map" the reads to a reference, and then look to see where they differ.

We're going to be looking at data from one of Lenski's experiments: Genomic analysis of a key innovation in an experimental E. coli population.

Genome of the reference strain here.

Tablet viewer

CHAPTER 13

Homework 4 (due Tuesday, Oct 29th, at 11:59pm PST)

Watch the videos below, and do the homework.

13.1 Videos

Watch these videos (they're from last year, so there's a bit of a context gap at the beginning; to catch up, just connect to the IPython Notebook and run the contents of the 'course-00-update-notebooks' notebook, and you'll be good).

1. Loading data from local files.
2. Using modules in Python: introducing 'math' and 'csv'.

See the Python stdlib math docs and csv docs.

Also see Doug Hellmann's excellent intermediate introductions to these at his Python Module of the Week (PyMOTW) site: math – mathematical functions and csv – comma-separated value files. You don't *need* more than what I've shown in the screencast to do the homework, but you might consider skimming these to see what other goodies are available in these modules in the stdlib.

4. Introducing dictionaries.

Also see the Software Carpentry lecture on dictionaries: http://software-carpentry.org/4_0/setdict/dict/

5. Plotting in ipython notebook with matplotlib.

—

13.2 Homework 4

1. After you update the notebooks repository on your EC2 instance by running the 'course-00-update-notebooks' notebook, you will have a new directory '/usr/local/notebooks/quartet', containing four files: q1.csv, q2.csv, q3.csv, and q4.csv. Each of these files will be available to your program as

```
'/usr/local/notebooks/quartet/q1.csv'
```

   etc. You can see their contents here:

   https://github.com/beacon-center/intro-computational-science/tree/master/quartet

   Write a notebook to:

(a) Load in the data in these files into two lists. Please do this using a single function that takes a filename as an argument and returns two lists, x and y – you might want to name them x1 and y1 for q1.csv, x2 and y2 for q2.csv, etc.

(b) Calculate and print the average and standard deviation of y1, y2, y3, and y4.

(c) Plot the x and y points on four different plots.

Hand in the homework by uploading it as a gist: log in via ssh, and run

```
apt-get install -y ruby1.9.1 rubygems
```

Then install 'gist':

```
gem install gist
```

and when you're done with the notebook, save it, and execute '!gist notebook-name.ipynb'. Then send me the gist info.

2. Write a function to 'invert' dictionaries, i.e. the function should take a single dictionary, and create a new dictionary that contains all of the values from the first dictionary as keys in the second, and all of the keys from the first dictionary as values associated with the appropriate key in the second dictionary.

   One possible approach:

   > Do (b) by hand (i.e. no functions or loops), first; just write out the statements to create a new dictionary and assign the correct values to it. Then do (c), too.

   Another approach:

   > First, write a function that takes in a dictionary and returns a new, empty dictionary.
   >
   > Second, modify the function so that it prints out each key/value pair in the input dictionary.
   >
   > Third, modify the function so that it prints out each value, key pair. (i.e. flip the order of things in the print statement.)
   >
   > Fourth, modify the function so that it assigns the value/key pair to the new dictionary rather than just printing the value/key.

   For the homework,

   (a) Write the function; name it something like 'invert_d'.

   (b) Take the following test dictionary (you can copy/paste into the notebook):

   ```
   first_d = { 'first': 1, 'second': 2 }
   ```

   and make sure your function turns it into this:

   ```
   { 1: 'first', 2: 'second' }
   ```

   Write this as an assert statement:

   ```
   assert invert_d(first_d) == test_d
   ```

   Note! If you swipe some online code it will probably not pass this test, so be careful :).

   (c) Observe what happens when you pass in this dictionary:

   ```
   bad_d = { 'one': 1, 'uno': 1 }
   ```

   Explain what happens in your own words (English, please...). Is this 'expected' behavior? Why or why not?

   Bonus pixie dust: Name one or more possible alternatives to this behavior.

Also hand this notebook in as a gist.

CHAPTER 14

Wed, Oct 23, 2013 - More Python; Annotation; Scripting

14.1 Python homework

HW3 IPython Notebook solutions set: http://nbviewer.ipython.org/7120119

14.2 Prokka and annotation evaluation

What does Prokka do? Basically annotation: finds non-coding and protein-coding genes; provides provisional names.

Re HW3, how did we use Prokka here? Was there a clear "winner" in terms of k value? What further questions might we ask?

14.2.1 BLAST output and parsing with Python

Working with BLAST output: basic export from BLAST to CSV,

> https://github.com/ngs-docs/ngs-scripts/blob/master/blast/blast-to-csv.py

and also:

> https://github.com/ngs-docs/ngs-scripts/blob/master/blast/blast-to-ortho-csv.py

14.3 Automating stuff; scripting

One (a major) reason why the command line has been so successful is that it's pretty easy to write software that can mixed, matched, and placed in automated pipelines at the command line.

See:

```
for i in {19..51..2}
do
   /mnt/prokka-1.7/bin/prokka ecoli-k${i}.fa --outdir k${i} --prefix k${i}
   # other commands involving ${i}
done
```

Note, this is a *completely* separate language from Python! This is shell programming, not Python programming. It is possible to mix and match them but as a rule I try to write *complicated scripts* in Python and reserve shell programming for when I'm using the command line interactively. This works for me but Your Mileage May Vary (YMMV).

14.3.1 Shell scripts

"Scripts" can be written in many different languages: shell and Python are the two you have experience with, but R also.

The idea is that you just put the commands you want to run into a file and then run that file with the right command: 'bash' for shell scripts, 'python' for

To try it, do:

```
echo "echo hello, world" > hello.sh
bash hello.sh
```

and:

```
echo "print 'hello, world'" > hello.py
python hello.py
```

Creating and editing these files is tricky, however. Need to use a text editor: can use 'pico' locally (or other, more complicated editors); or TextEdit/TextWrangler remotely, then copy via Dropbox.

14.4 GitHub

Gists: place to store random "stuff".

Github more generally: version controlled.

Go to https://github.com/ngs-docs/ngs-scripts/; make sure you're logged in; click "fork". Click edit. Make some changes. Save, with commit.

CHAPTER 15

Homework 3 (due Tuesday, Oct 22nd, at 11:59pm PST)

Note: Need help? Go to http://angus.askbot.com/ to ask questions and see other people's answers!

Note: See `amazon/using-screen` for information on using screen.

15.1 1. Annotate E. coli; compare annotations

Hint: you definitely want to coordinate with some other people on this one. Each Prokka run will take about 45 minutes.

Start up an m1.xlarge. Install BLAST (just the part under "Next, install BLAST", in `week2/blastkit`) and download ngs-scripts:

```
git clone https://github.com/ngs-docs/ngs-scripts /usr/local/share/ngs-scripts
```

You'll also need screed:

```
pip install screed
```

Next: install prokka, as per `week3/prokka-annotation`.

Go to /mnt and download a bunch of velvet assemblies that I ran for you (what would be produced by running through `week2/assembly-lab` for many different k):

```
cd /mnt
curl -O http://public.ged.msu.edu.s3.amazonaws.com/velvet-dn-ecoli.tar.gz
tar xzf velvet-dn-ecoli.tar.gz
```

Run the command `ls`; you should see about 11 assemblies, named `ecoli-kXX.fa`.

Now, download the "official" E. coli MG 1655 file:

```
curl -O http://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/Escherichia_coli_K_12_substr__MG1655_uid57779/NC
```

and format it for BLAST:

```
formatdb -i NC_000913.faa -o T -p T
```

Then, build and compare annotations by doing the following: for each E. coli file, run Prokka and construct a set of pairwise BLAST files:

```
/mnt/prokka-1.7/bin/prokka ecoli-k19.fa --outdir k19 --prefix k19

formatdb -i k19/k19.faa -o T -p T
blastall -i k19/k19.faa -d NC_000913.faa -p blastp -e 1e-12 -o k19.x.mg1655
blastall -d k19/k19.faa -i NC_000913.faa -p blastp -e 1e-12 -o mg1655.x.k19
```

Finally, for each set of annotated E. coli files, calculate the orthologs and then count them:

```
python /usr/local/share/ngs-scripts/blast/blast-to-ortho-csv.py k19/k19.faa NC_000913.faa k19.x.mg165
wc -l k19-ortho.csv
```

You should see output like this: `3804 k19-ortho.csv`. Record the number (3804) and the k value (19); do this for each of the E. coli assemblies & annotations, then put the list in a spreadsheet and send to me.

Note: You can automate the above by putting it in a for loop in the shell:

```
for i in {19..51..2}
do
   /mnt/prokka-1.7/bin/prokka ecoli-k${i}.fa --outdir k${i} --prefix k${i}
   # other commands involving ${i}
done
```

Here, the for loop will run the commands between 'do' and 'done' once for every value of "i" between 19 and 51, skipping numbers forward by 2 each time (e.g. 19, 21, 23, 25, ...).

A good trick is to use 'echo' to see if your commands are correct before actually running them – e.g. run

```
for i in {19..51..2}
do
   echo /mnt/prokka-1.7.bin/prokka ... ${i}
done
```

and then look at the commands; after it looks like you're going to run the right commands, then take the 'echo' commands out.

15.2 2. Programming in Python: lists and dictionaries and functions, oh my! part 2

Log in to your EC2 instance via SSH and install ipythonblocks:

```
pip install ipythonblocks
```

and Ruby/Ruby gems:

```
apt-get install -y ruby1.9.1 rubygems
```

Then install 'gist':

```
gem install gist
```

and download the class 3 notebook:

```
cd /usr/local/notebooks
curl -O https://raw.github.com/beacon-center/2013-intro-computational-science/master/notebooks/class3
```

Connect to IPython Notebook ('https://' + your EC2 machine name) and solve the problems. Post the notebook as a github 'gist' (see last line for instructions) and send me the nbviewer link.

CHAPTER 16

Wed, Oct 16, 2013 - More sequencing; mapping mismatches; lists, dicts, fns

16.1 Presentation

See today's presentation.

16.2 Mapping

Let's explore some of the mismatch mapping via IPython Notebook.

Go pick a random machine: visit the Google doc for machine names and connect to IPython Notebook on that machine.

Open class3-mismatches.ipynb; copy it; rename it.

16.3 Lists, functions, and dictionaries

Open class3-lists-dicts-functions.ipynb on a remote machine; copy it; rename it.

Or see a static copy: in-class notebook.

CHAPTER 17

Homework 2 (due Tuesday, Oct 15th, at 11:59pm PST)

Note: Need help? Go to http://angus.askbot.com/ to ask questions and see other people's answers!

Note: See `amazon/using-screen` for information on using screen.

17.1 1. Assemble E. coli.

Follow `week2/assembly-lab`.

17.2 2. Retrieve CRP_ECOLI from an assembly.

Go to http://www.ncbi.nlm.nih.gov/ and search the protein database for CRP_ECOLI. Retrieve the FASTA version of this gene (just the amino acids).

Next, run an assembly with a k other than 31: pick any odd k between 19 and 51, and rerun the velvet etc. commands to produce an 'ecoli-kXX.fa' assembly.

Install this assembly in a BLAST server (see `week2/blastkit`) and search it with the CRP_ECOLI protein sequence.

Download the best match sequence, and e-mail it to me with your other HW.

17.3 3. Compare the mismatch profiles for 4 different stages of reads.

We're going to take four different collections of reads and map them to an assembled genome, then look at the read mismatch profile (where mismatches between the read and the assembly exist, plotted by position in read). This is a way of looking at the overall error rate.

First, install bowtie, which does read mapping:

```
cd /root
curl -O -L http://sourceforge.net/projects/bowtie-bio/files/bowtie/0.12.7/bowtie-0.12.7-linux-x86_64
unzip bowtie-0.12.7-linux-x86_64.zip
```

```
cd bowtie-0.12.7
cp bowtie bowtie-build bowtie-inspect /usr/local/bin
```

Now, make the E. coli assembly into something that bowtie can map reads to by indexing it:

```
cd /mnt/assembly
bowtie-build ecoli-k31.fa ecoli-k31
```

Next, collect the four read data sets. First, get 200k of the raw reads:

```
curl ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR390/SRR390202/SRR390202_1.fastq.gz | gunzip -c | head -80
```

Also collect the first 200k of the QC reads:

```
gunzip -c SRR390202.pe.qc.fq.gz | head -800000 > qc-reads.fq
```

Take 200k reads from the diginorm processing step, and the abundfilt processing step, too:

```
head -800000 SRR390202.pe.qc.fq.gz.keep > dn1-reads.fq
head -800000 SRR390202.pe.qc.fq.gz.keep.abundfilt > fa-reads.fq
```

Now, map all of them:

```
bowtie -p 2 ecoli-k31 -q raw-reads.fq raw-reads.map
bowtie -p 2 ecoli-k31 -q qc-reads.fq qc-reads.map
bowtie -p 2 ecoli-k31 -q dn1-reads.fq dn1-reads.map
bowtie -p 2 ecoli-k31 -q fa-reads.fq fa-reads.map
```

You should see output like this:

```
# reads processed: 200000
# reads with at least one reported alignment: 181936 (90.97%)
# reads that failed to align: 18064 (9.03%)
Reported 181936 alignments to 1 output stream(s)
```

Take a look at the mapping files – it shows which reads map where in the genome, and where the mismatches are for each read:

```
head raw-reads.map
```

Now, let's convert each of these mapping files into a set of counts:

```
git clone https://github.com/ngs-docs/ngs-scripts.git /root/ngs-scripts

for i in *-reads.map
do
python /root/ngs-scripts/bowtie/map-profile.py $i > $i.count
done
```

This will create a bunch of .count files, one for each .map file.

To graph these files, go to your IPython Notebook (the https URL) and create a new notebook called hw2-mismatches-USERID. In this notebook, put two cells. First,

```
cd /mnt/assembly
```

and second:

```
dn1_counts = numpy.loadtxt('dn1-reads.map.count')
fa_counts = numpy.loadtxt('fa-reads.map.count')
qc_counts = numpy.loadtxt('qc-reads.map.count')
raw_counts = numpy.loadtxt('raw-reads.map.count')
```

When you execute these, it will load the files into Python; you can plot them in a third cell like so:

```
plot(qc_counts[:,0], qc_counts[:,1], label='qc')
plot(raw_counts[:, 0], raw_counts[:, 1], label='raw')
axis(ymin=0, ymax=80000, xmin=0, xmax=160)
xlabel('position in read')
ylabel('number of reads with mismatches at that position')
legend()
```

This will plot the first column of the two given counts files (the position in the read) against the second column (the number of mismatches at that position).

Play with the data a bit by adding cells to graph different combinations against each other, changing axes, etc. Use the graphs to determine which data set has the most mismatches, save the notebook, and send it to me.

17.4 4. Programming in Python: lists and dictionaries and functions, oh my!

Log in to your EC2 instance via SSH and install ipythonblocks:

```
pip install ipythonblocks
```

and then download a few notebooks:

```
cd /usr/local/notebooks
curl -O https://raw.github.com/beacon-center/2013-intro-computational-science/master/notebooks/class2
curl -O https://raw.github.com/beacon-center/2013-intro-computational-science/master/notebooks/hw1-ip
curl -O https://raw.github.com/beacon-center/2013-intro-computational-science/master/notebooks/hw2-ip
```

Connect to IPython Notebook ('https://' + your EC2 machine name) and solve the problems in hw2-ipythonblocks.ipynb. Rename the notebook to something that has your identifier in it, download it, and e-mail it to me as an attachment.

Note that 'class2-ipythonblocks' contains an introduction to lists and dictionaries, while 'hw1-ipythonblocks-SOLUTIONS' shows the solutions to the HW.

CHAPTER 18

Wed, Oct 9, 2013 - Illumina; shotgun sequencing; read QC

Outline of day:

18.1 1. Technical challenges of homework; long-running jobs

Review: starting up machines; connecting with SSH or IPython Notebook.

Copy/paste and putting in complete commands.

Disconnections.

Long running jobs; using screen.

Dropbox.

Shell vs IPython Notebook.

(I will deliver videos of all of these tonight.)

18.2 2. Shotgun sequencing with Illumina

(To follow along with lecture go to http://jotwithme.com/ and enter 'titus'.)

How Illumina works.

Assumptions underlying shotgun sequencing.

How the data is delivered.

- FASTQ format (see http://en.wikipedia.org/wiki/FASTQ_format)
- Phred quality scores (see http://en.wikipedia.org/wiki/FASTQ_format#Quality)
- read length

18.3 3. Quality scores, and quality evaluation

Discuss some FastQC results here: http://athyra.idyll.org/~t/SRR390202_1_fastqc/fastqc_report.html

and here: http://athyra.idyll.org/~t/SRR390202.pe.qc.fq_fastqc/fastqc_report.html

What other ways might there be to evaluate the quality of your sequencing data? (Think about things you already know, or can assume...?)

18.4 4. What do you do with sequencing data?

Mapping vs assembly. (This will be a continuing theme :)

18.5 5. Homework solutions

See: solutions.

18.6 6. More Python

Last week was "control flow" – for and if. This week, "data structures" – ways of storing aggregate information.

See: in-class notebook.

CHAPTER 19

Homework 1 (due Tuesday, Oct 8th, by midnight PST)

Note: Need help? Go to http://angus.askbot.com/ to ask questions and see other people's answers!

Note: Keep getting disconnected? This is probably because of inactivity or time limits on network connections (on your computer, at your university, or whatever). See `amazon/using-screen` for the best way to deal with this.

19.1 0. Start up an EC2 machine.

Work through the instructions here: `amazon/index` for starting up an EC2 machine on AWS.

A brief checklist:

1. Create an AWS account and log in;
2. Launch a machine; use m1.xlarge.
3. Adjust your security rules for the machine and select 'default' security profile.
4. Make sure you can both log into your machine with SSH *and* with IPython Notebook.
5. Install Dropbox on your EC2 machine (see `amazon/installing-dropbox`)

There's a video from last year here that is only slightly out of date.

Two notes:

- be sure to save your work to Dropbox and shut down your machines when you are done working;
- for now, you will need to pay for your own EC2 machines. This should be less than $2-3/wk. If you are a BEACON-affiliated graduate student we can help you pay.

19.2 1. Work through some programming exercises

Log in to your EC2 instance via SSH and install ipythonblocks:

```
pip install ipythonblocks
```

download the homework notebook:

```
cd /usr/local/notebooks
curl -O https://raw.github.com/beacon-center/2013-intro-computational-science/master/notebooks/hw1-ip
```

Then, go to 'https://' + your EC2 machine name to connect to EC2. Use the password 'beacon' to log into the console and then open hw1-ipythonblocks. Solve the problems and then download the notebook (file... save as...) and e-mail it to me as an attachment, with 'cse801 hw1' in the subject line of the e-mail. Please rename the attachment so it includes your last name or NetID somewhere.

Note that you can see a static version of the notebook from class here, with a download link; if you want to work with it, you can upload this notebook to your EC2 machine following the instructions below, under "A few notes."

A few notes:

- if you aren't going to work on this problem for a while and want to shut down your machine, you can download your ipython notebook by going to 'File' and 'Save As ipynb'; this downloaded ipython notebook can be reuploaded by dragging it from your computer into the IPython Notebook console. This replaces the 'download' command above.
- it is not a failure to use Google to ask questions like "how do I test if a number is even in Python?" or "how do I multiply numbers in Python?"

19.3 2. Run FastQC on full E. coli data set and trimmed data set

Log in to your m1.xlarge EC2 machine via SSH and follow through all of `week1/reads_and_qc`.

If you are interested in what the results mean and don't want to wait 'til Wednesday, watch this 12 minute video.

CHAPTER 20

Wed, Oct 2, 2013 - ipythonblocks; read QC

Google doc for machine names

Outline of day:

20.1 0. Survey.

(until ~20 minutes after)

You should all have received a survey e-mail from Claudia Vergara. Please fill out survey; I will leave room and return at 20 after.

20.2 1. Introduction; office hours; grading and hand-in policy.

I've already done the introduction at the workshops, but basically we're going to show you three things in this class: (1) some basic programming, (2) how to run and think about sequence data analysis, and (3) how to do some simple evolutionary modeling.

20.2.1 Lectures

There will be some online lectures, especially about technical stuff like starting up machines, handing in homework, and various aspects of programming and running NGS programs.

20.2.2 Homework and grading

If you're auditing you're expected to hand in the homeworks until you stop auditing, and vice versa.

Grading will be P/F each week, with each week representing 10% of the final grade.

Homework handed in after the official due date/time may not be graded.

If I do not send you comments on your homework by the end of the weekend following hand-in, (Sunday evening), you get a PASS garde.

I would suggest that you get together in study groups to do the homework.

20.2.3 Office hours

Primary support will be via e-mail. I can schedule a Google Hangout with you if you need one-on-one or one-on-group time. I propose to specify one or two times a week where I will be available, decided on by Doodle polls. Pushback?

20.3 2. ipythonblocks introduction; in-class exercises.

Pick one of the machines randomly (see Google doc for machine names) and open up a new IPython Notebook link. (Password: 'beacon'.)

Select the notebook 'class1-ipythonblocks' & make a copy!

See a static copy of the notebook here.

20.4 3. Read trimming and QC; FastQC reports.

A short introduction to NGS sequencing, and what can go wrong; a FastQC report. (Lecture whiteboard: go to JotWithMe/titus801-1.)

To see a FastQC report, pick one of the machines randomly (see Google doc for machine names) and open up the associated FastQC link. To generate this, I did

```
cd /usr/local/share
curl -O http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.10.1.zip
unzip fastqc_v0.10.1.zip
chmod +x FastQC/fastqc

cd /root
curl -O http://athyra.idyll.org/~t/100k_1.fq
curl -O http://athyra.idyll.org/~t/100k_2.fq

mkdir /var/www/fastqc
/usr/local/share/FastQC/fastqc 100k_1.fq 100k_2.fq -o /var/www/fastqc
```

CHAPTER 21

Indices and tables

- genindex
- modindex
- search