Genome 540 Discussion

Conor Camplisson

February 16th, 2023
Outline

• Homework 7 overview

• Related topics:
  • Example Snakemake pipeline

• Homework 6 & 7 questions
Outline

• Homework 7 overview

• Related topics:
  • Example Snakemake pipeline

• Homework 6 & 7 questions
**Homework 7 Overview**

(Homework 6 Background Info)

**Data:** next-gen read alignments to genome, CHM13 chr16

**Observed symbols:** counts of read starts at each position
- Frequencies from Poisson dist. with appropriate mean

**Target regions:** heterozygous duplications
- One chrom = ref allele, other = dup, Poisson mean 1.5X background

<table>
<thead>
<tr>
<th>Position (chr16)</th>
<th>Avg. # Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>[... ]</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>14793</td>
</tr>
<tr>
<td>16</td>
<td>14794</td>
</tr>
<tr>
<td>16</td>
<td>14795</td>
</tr>
<tr>
<td>16</td>
<td>14796</td>
</tr>
<tr>
<td>[... ]</td>
<td></td>
</tr>
</tbody>
</table>
Homework 7 Overview

(Homework 6 Background Info)

Found mean observed read count

Denominator adjusted for N's in reference (see HW7)

```
# compute mean read count, adjusting for N's in denominator
N_CORRECTION = 8422401
ADJ_CHROM_SIZE = len(df) - N_CORRECTION
ADJ_MEAN_COUNT = df['num_reads'].sum() / ADJ_CHROM_SIZE

print(ADJ_MEAN_COUNT)
0.14936377712374954
```

Created Model Distributions

```
import numpy as np
from scipy.stats import poisson

# compute means of model Poisson distributions
mu_0 = ADJ_MEAN_COUNT
mu_1 = ADJ_MEAN_COUNT * 1.5

# create model Poisson distributions given N observed
x = np.arange(4)
y_0 = (poisson.pmf(mu=mu_0, k=x) * ADJ_CHROM_SIZE).astype(int)
y_1 = (poisson.pmf(mu=mu_1, k=x) * ADJ_CHROM_SIZE).astype(int)

print('X (counts):', x)
print('Background:', y_0)
print('Elevated CN:', y_1)

X (counts): [0 1 2 3]
Background: [70455754 10523537 785917 39129]
Elevated CN: [65385664 14649374 1641064 122557]
```
**Homework 7 Overview**

*(Homework 6 Background Info)*

**Created LLR Scoring Scheme**

```plaintext
# compute means of model Poisson distributions
mu_0 = ADJ_MEAN_COUNT
mu_1 = ADJ_MEAN_COUNT * 1.5

# create model Poisson distributions given N observed
x = np.arange(4)
y_0 = poisson.pmf(mu=m_0, k=x)
y_1 = poisson.pmf(mu=m_1, k=x)

# truncate distributions
y_0[-1] += 1.0 - np.sum(y_0)
y_1[-1] += 1.0 - np.sum(y_1)

# compute LLR scoring scheme
weights = np.log2(y_1 / y_0)

print(f'X (counts):
  Background: \{y_0.round(4)}
  Elevated CN: \{y_1.round(4)}

Weights: \{-0.1077  0.4772  1.0622  1.6748\}
```

**HW6 Scoring Scheme**

2. Run your program on this file using the following scoring scheme:
   - score for 0 reads: -0.1077
   - score for 1 read: 0.4772
   - score for 2 reads: 1.0622
   - score for >=3 reads: 1.6748
   - D = -20
   - S = -D = 20

**Read Start Count Distributions**

```
X (counts):    [0 1 2 3]
Background:    [0.8613 0.1286 0.0096 0.0005]
Elevated CN:   [0.7993 0.1791 0.0201 0.0016]
Weights:       [-0.1077  0.4772  1.0622  1.6748]
```

http://bozeman.mbt.washington.edu/compbio/mbt599/assignments/hw7.html
Homework 7 Overview
(Homework 6 Background Info)

Created LLR Scoring Scheme

```python
# compute means of model Poisson distributions
mu_0 = ADJ_MEAN_COUNT
mu_1 = ADJ_MEAN_COUNT * 1.5

# create model Poisson distributions given N observed
x = np.arange(4)
y_0 = poisson.pmf(mu=mu_0, k=x)
y_1 = poisson.pmf(mu=mu_1, k=x)

# truncate distributions
y_0[-1] += 1.0 - np.sum(y_0)
y_1[-1] += 1.0 - np.sum(y_1)

# compute LLR scoring scheme
weights = np.log2(y_1 / y_0)

print(f'X (counts): {x}')
print(f'Background: {y_0.round(4)}')
print(f'Elevated CN: {y_1.round(4)}')
print(f'Weights: {weights.round(4)}')
```

HW6 Scoring Scheme

2. Run your program on this file using the following scoring scheme:
   - score for 0 reads: -0.1077
   - score for 1 read: 0.4772
   - score for 2 reads: 1.0622
   - score for >=3 reads: 1.6748
   - D = -20
   - S = -D = 20

Read Start Count Distributions

http://bozeman.mbt.washington.edu/compbio/mbt599/assignments/hw7.html
1. Create LLR Scoring Scheme

Use segment results from HW6:
- Count observed read start counts:
  - \textit{Background}: in ALL segments
    - Sum counts for both types of segments
    - Correct for N’s in reference (see HW7)
  - \textit{Elevated}: in elevated segments only
    - No N correction
- Convert counts to frequencies
- Compute LLR with log2

- Empirical data doesn’t fit Poisson well
  - Amplification in sequencing library prep.
- Use HW6 results to refine our model

http://bozeman.mbt.washington.edu/compbio/mbt599/assignments/hw7.html
Homework 7 Overview

2. Generate simulated read counts
   • Create simulated read counts
   • Run maximal D-segment program
     • On real data file
     • On simulated data file
     • Use your new scoring scheme!
   • Generate a list of ratios
     • See HW7 for details
   • Answer questions based on Karlin-Altschul theory and your results

Simulation pseudocode

N = length of sequence to be simulated
bkgd[r] = frequency of background sites with r read starts (r = 0, 1, 2, 3).
for each i = 1...N
  x = random number between 0 and 1 (uniform distribution)
  if x < bkgd[0]
    sim_seq[i] = 0
  else if x < bkgd[0] + bkgd[1]
    sim_seq[i] = 1
  else if x < bkgd[0] + bkgd[1] + bkgd[2]
    sim_seq[i] = 2
  else
    sim_seq[i] = 3

Read Start Count Distributions

http://bozeman.mbt.washington.edu/compbio/mbt599/assignments/hw7.html
Homework 7 Overview

Simulated data:
- 5 (# of segments with score >= 5)
- 6 (# of segments with score >= 6)
- 7 (# of segments with score >= 7)

for scores between 5 and 30

Real data:
- 5 (# of segments with score >= 5)
- 6 (# of segments with score >= 6)
- 7 (# of segments with score >= 7)

Target frequencies:
- 0={#.#####}
- 1={#.#####}
- 2={#.#####}
- 3={#.#####}

Scoring scheme:
- 0={#.#####}
- 1={#.#####}
- 2={#.#####}
- 3={#.#####}

Ratios of simulated data:
- N_seg(5)/N_seg(6) (# of segments with score >= 5 / # of segments with score >= 6)
- N_seg(6)/N_seg(7) (# of segments with score >= 6 / # of segments with score >= 7)
- N_seg(7)/N_seg(8) (# of segments with score >= 7 / # of segments with score >= 8)

list all ratios
(list only first/last 3 shown here)

N_seg(27)/N_seg(28) (# of segments with score >= 27 / # of segments with score >= 28)
N_seg(28)/N_seg(29) (# of segments with score >= 28 / # of segments with score >= 29)
N_seg(29)/N_seg(30) (# of segments with score >= 29 / # of segments with score >= 30)

As discussed in lecture, Karlin-Altschul theory predicts that, for LLR scores using logarithmic base b, the number of D-segments with scores
>= s should be proportional to b^s (b to the power -s; this is the reciprocal of the corresponding LR). Since your scores used logarithmic
base 2, if N_seg(s1) is the number of D-segments found with score value >= s1, and N_seg(s2) is the number of D-segments found with
score value >= s2, then the ratio N_seg(s1)/N_seg(s2) should be approximately equal to 2^(-s2 - s1). Consider the following questions:
- Does this relationship appear to be true for the simulated data?
- Is it true for the real data?
- Would you expect it to be true for the real data?
- What score threshold is a reasonable one to use for the real data, to ensure a very low false positive rate?

http://bozeman.mbt.washington.edu/compbio/mbt599/assignments/hw7.html
Outline

• Homework 7 overview

• Related topics:
  • Example Snakemake pipeline

• Homework 6 & 7 questions
Intro to Snakemake

rule bwa_map:
  input:
    "data/genome.fa",
    "data/samples/{sample}.fastq"
  output:
    "mapped_reads/{sample}.bam"
  shell:
    "bwa mem {input} | samtools view -Sb - > {output}"

rule samtools_sort:
  input:
    "mapped_reads/{sample}.bam"
  output:
    "sorted_reads/{sample}.bam"
  shell:
    "samtools sort -T sorted_reads/{wildcards.sample} "
    "-O bam {input} > {output}"

rule samtools_index:
  input:
    "sorted_reads/{sample}.bam"
  output:
    "sorted_reads/{sample}.bam.bai"
  shell:
    "samtools index {input}"
Simple pipeline in Snakemake

PaintSHOP Pipeline
Snakemake pipeline for genome-scale mining of optimal homology sequences for PaintSHOP

yEvo Pipeline
Variant calling Snakemake pipeline for yEvo sequencing data
Snakemake Demo Plan: Image Processing

Image processing with python and Snakemake

• Multidimensional array computing with numpy
  • An image == a numpy array
  • Pre-processing, matrix operations, masking, etc.

• Ideal for parallelization
  • Many images per experiment
    • Multiple channels per image, parallelize

• Ideal use case for cluster deployment (large data)
  • Snakemake greatly facilitates

Pipeline Specification

Input: .nd2 files (3D hyperstacks)

Steps: split channels, z-project, detect fluorescent objects (puncta), compute & plot stats

Output:
• plots of pixel intensity, spot size
• .csv file with stats per sample
Images & Python

2D Binary Arrays

Consider the following two text files:

arr1.txt & arr2.txt
Images & Python

Load .txt files and convert to numpy

```
def load_arr(file_path):
    "Load a 2D array from a text file and convert it to a numpy array.""
    arr = pd.read_csv(file_path, sep="\t", header=None).values
    return arr

arr1 = load_arr('data/arr1.txt')
arr2 = load_arr('data/arr2.txt')
print(f'Array 1:
\n{arr1}
Array 2:
\n{arr2}')
```

Visualize with matplotlib plt.imshow()
Images & Python

Images as numpy arrays

- Matplotlib can load, save, and view images
  - `plt.imread(file_path)`
  - `plt.imsave(img, file_path)`
  - `plt.imshow(img)`

- Types of pixel values:
  - Single int (greyscale, chess)
    - e.g. 0, 1, 255
  - RGB Tuple (Boltzmann PNG)
    - e.g. [255, 0, 255]
    - e.g. [1.00, 0.00, 1.00]
  - RGB with alpha (transparency)
    - e.g. [0.33, 0.25, 0.33, 1.0]
    - (int/float, bit-depth, etc.)
Images & Python

Masking with numpy

Left mask:
```
[[1 1 1 1 0 0 0 0 0 0 0 0 0 0 
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 
 1 1 1 1 0 0 0 0 0 0 0 0 0 0]]
```

Middle mask:
```
[[0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0 
 0 0 0 0 1 1 1 1 0 0 0 0 0]]
```

Right mask:
```
[[0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 
 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1]]
```
Images & Python

Masking with numpy

Left mask:

```
[[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]]
```

Digit mask:

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 0 1 0 1 0 1 1 1 0]
 [0 1 0 0 0 1 0 1 0 1 0 1 0]
 [0 1 1 1 0 1 1 1 0 1 0 1 0]
 [0 0 0 1 0 0 0 1 0 1 0 1 0]
 [0 1 1 1 0 0 0 1 0 1 1 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

Left mask AND digit mask:

```
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```
Images & Python

Masking with numpy

```
use masks in combination to set each digit

# invert the image, for a background of 1's
img = 1 - img

# set individual digits
img[mask & left] = 5
img[mask & middle] = 4
img[mask & right] = 0

print(img)
```

```
# display image array
plt.imshow(img, cmap='Set1')
plt.show()
```
Snakemake Demo: Image Processing

Today’s examples as a jupyter notebook

Conda-based install of snakemake/img libs

Images as numpy arrays

This notebook shows the basics of working with images as numpy arrays in python.

NOTE: the code depends on the `img` conda environment found in `metadocs/snakemake.png`

configure notebook

```
# (1)
import matplotlib.pyplot as plt
import numpy as np
import os

def clear():
    if os.name == 'nt':
        _ = os.system('cls')
    else:
        _ = os.system('clear')

ONES = np.ones((2,2))
```

viewing images with matplotlib

```
# (2)
plt.imshow(ones, cmap='gray')
plt.show()
```

.example Snakemake pipeline for the Genome 540 course at UW Genome Sciences

Installation

1. Make sure you have conda installed.
2. Install Mamba to facilitate snakemake installation, as recommended in the Snakemake docs.

   $ conda install mamba
   $ conda install --channel conda-forge mamba
3. Clone this repo:

   $ git clone https://github.com/conorcamplisson/GS540_snakemake_demo.git

Access the demo pipeline repo:

https://github.com/conorcamplisson/GS540_snakemake_demo
Snakemake Demo: Image Processing

Commit history

- add python image example notebook and test images
  - conorcamplisson committed 16 minutes ago
- add jupyter to conda env
  - conorcamplisson committed 17 minutes ago
- add image processing conda env for pipeline
  - conorcamplisson committed 17 minutes ago
- start pipeline directory
  - conorcamplisson committed 18 minutes ago

Commits on Feb 15, 2023

- update docs
  - conorcamplisson committed 3 hours ago
- add conda env with snakemake
  - conorcamplisson committed 3 hours ago
- Initial commit
  - conorcamplisson committed 5 hours ago

Repo structure

- master
  - 1 branch
  - 0 tags
  - add python image example notebook and test images
    - committed 17 minutes ago
  - notebooks
    - add python image example notebook and test images
      - committed 17 minutes ago
  - workflow
    - add image processing conda env for pipeline
      - committed 18 minutes ago
  - .gitignore
    - Initial commit
      - committed 5 hours ago
  - LICENSE
    - Initial commit
      - committed 5 hours ago
  - README.md
    - update docs
      - committed 3 hours ago
  - environment.yml
    - add jupyter to conda env
      - committed 17 minutes ago

Access the demo pipeline repo:

https://github.com/conorcamplisson/GS540_snakemake_demo
Images & Python

Automated Acknowledgements

- This image is a matplotlib figure
- Names and images are scraped from https://www.beliveau.io/team
- Using numpy, a circular mask is created and placed over the downloaded (square) image to create the circular cropped result
- Each lab member gets a subplot
  - (can filter myself out with regex)
- Can export to .svg, .png, .pdf, .eps
- Re-run notebook as people rotate
Outline

• Homework 7 overview

• Related topics:
  • Example Snakemake pipeline

• Homework 6 & 7 questions
Homework 6 & 7 Questions?

Avg. # Reads

Position (chr16)

chm13.chr16.txt

<table>
<thead>
<tr>
<th>Pos</th>
<th>Avg Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>[ ... ]</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>14793</td>
</tr>
<tr>
<td>16</td>
<td>14794</td>
</tr>
<tr>
<td>16</td>
<td>14795</td>
</tr>
<tr>
<td>16</td>
<td>14796</td>
</tr>
<tr>
<td>[ ... ]</td>
<td>0</td>
</tr>
</tbody>
</table>

Read Start Count Distributions

- Observed Counts
- Background Model
- Elevated CN Model
Reminders

• Homework 6 due this Sunday Feb. 19, 11:59 pm

• Homework 7 due next Sunday Feb. 26, 11:59 pm