Genome 540 Discussion

Conor Camplisson

February 16th, 2023

Genome Sciences
UNIVERSITY OF WASHINGTON

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)

chm13.chr16.txt

```
16      1       0
16      2       0
16      3       0
16      4       0
16      5       0
      [ ... ]
16      14793   0
16      14794   1
16      14795   3
16      14796   0
      [ ... ]
```

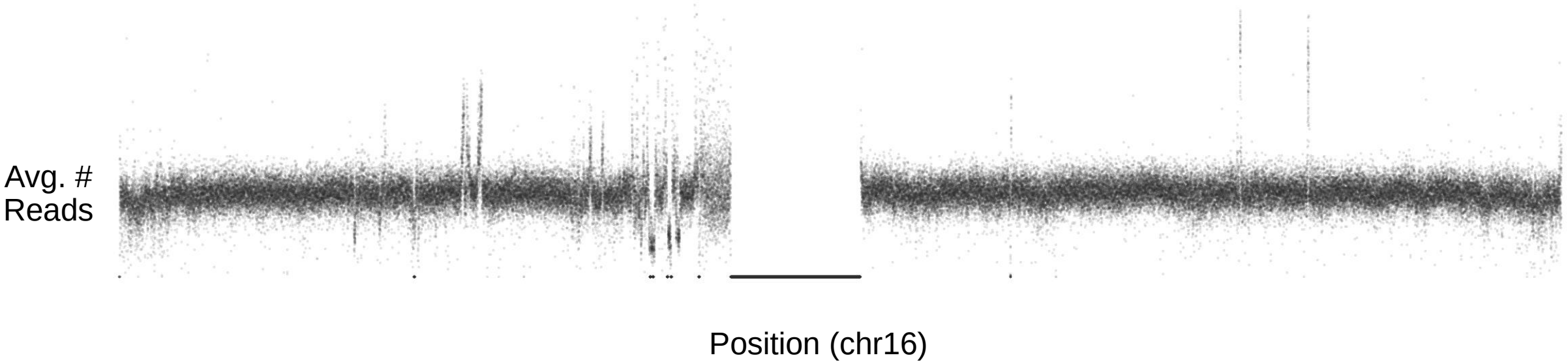
Data: next-gen read alignments to genome, CHM13 chr16

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



Homework 7 Overview

(Homework 6 Background Info)

Found mean observed read count

chm13.chr16.txt

```
16      1       0
16      2       0
16      3       0
16      4       0
16      5       0
      [ ... ]
16      14793   0
16      14794   1
16      14795   3
16      14796   0
      [ ... ]
```

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

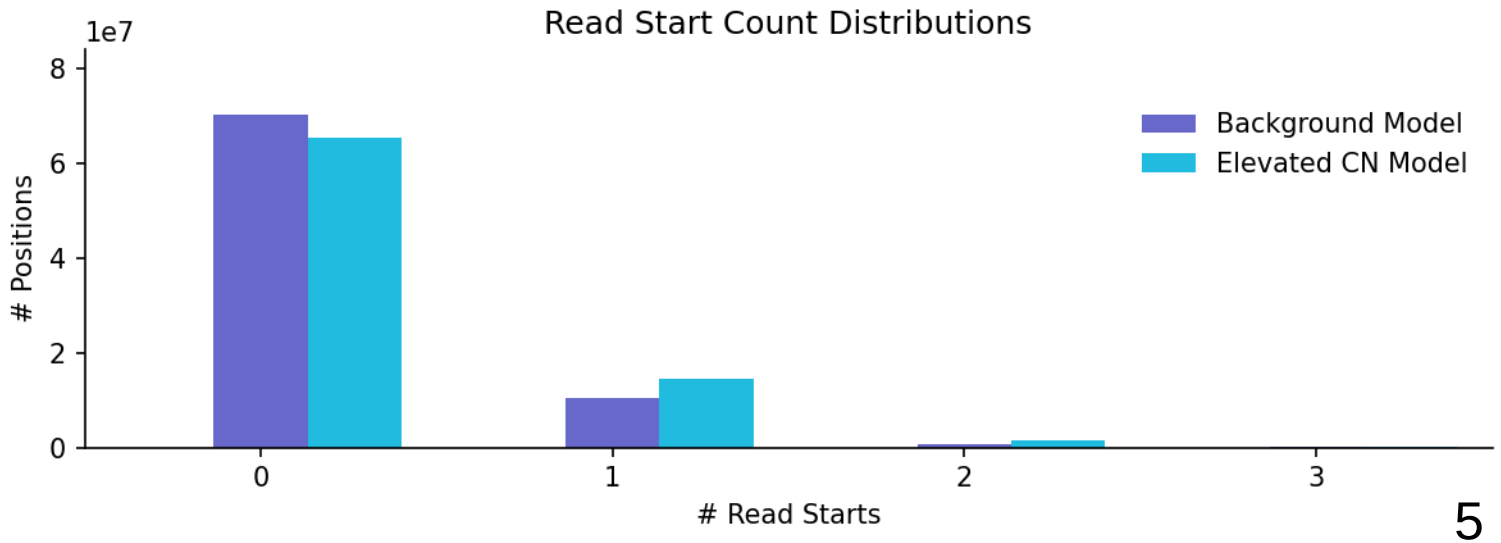
```python
# compute mean read count, adjusting for N's in denominator
N_CORRECTION = 8422401
ADJ_CHROM_SIZE = len(df) - N_CORRECTION
ADJ_MEAN_COUNT = df['num_reads'].sum() / ADJ_CHROM_SIZE

print(ADJ_MEAN_COUNT)
```

```
0.14936377712374954
```

Created Model Distributions

```python
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(f'X (counts):\t{x}')
print(f'Background:\t{y_0}')
print(f'Elevated CN:\t{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)

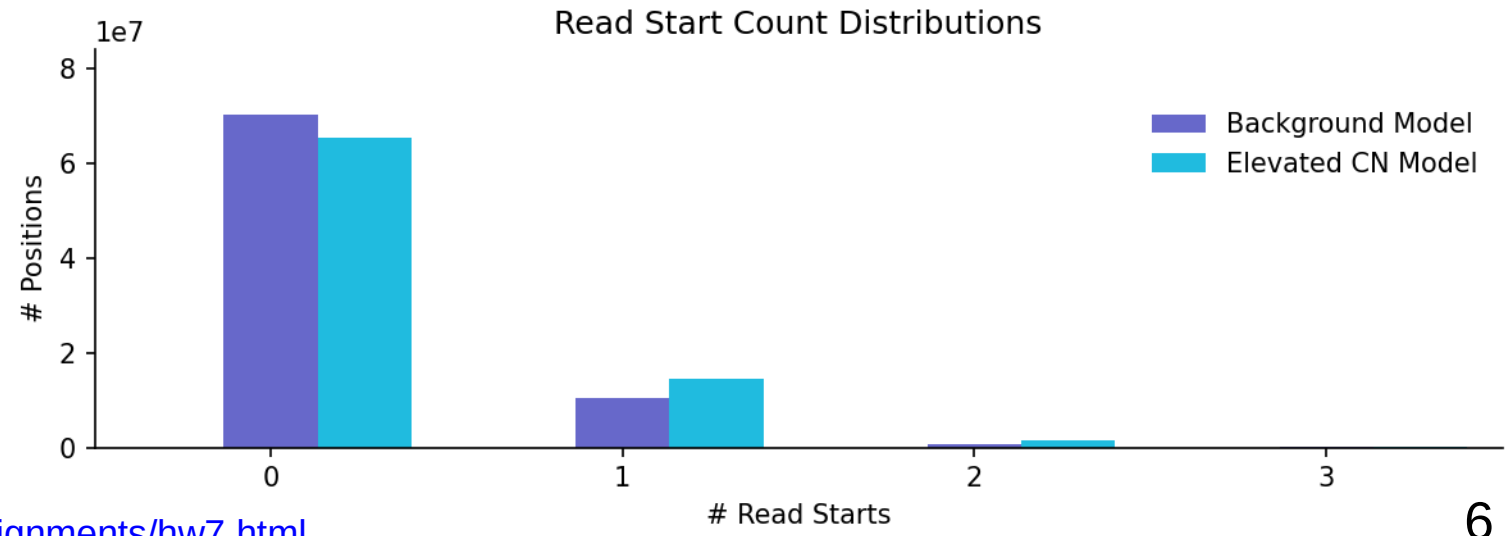
Created LLR Scoring Scheme

```
# 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):\t{x}')
print(f'Background:\t{y_0.round(4)}')
print(f'Elevated CN:\t{y_1.round(4)}')
print(f'\nWeights:\t{weights.round(4)}\n')
```

```
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]
```

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



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

Homework 7 Overview

(Homework 6 Background Info)

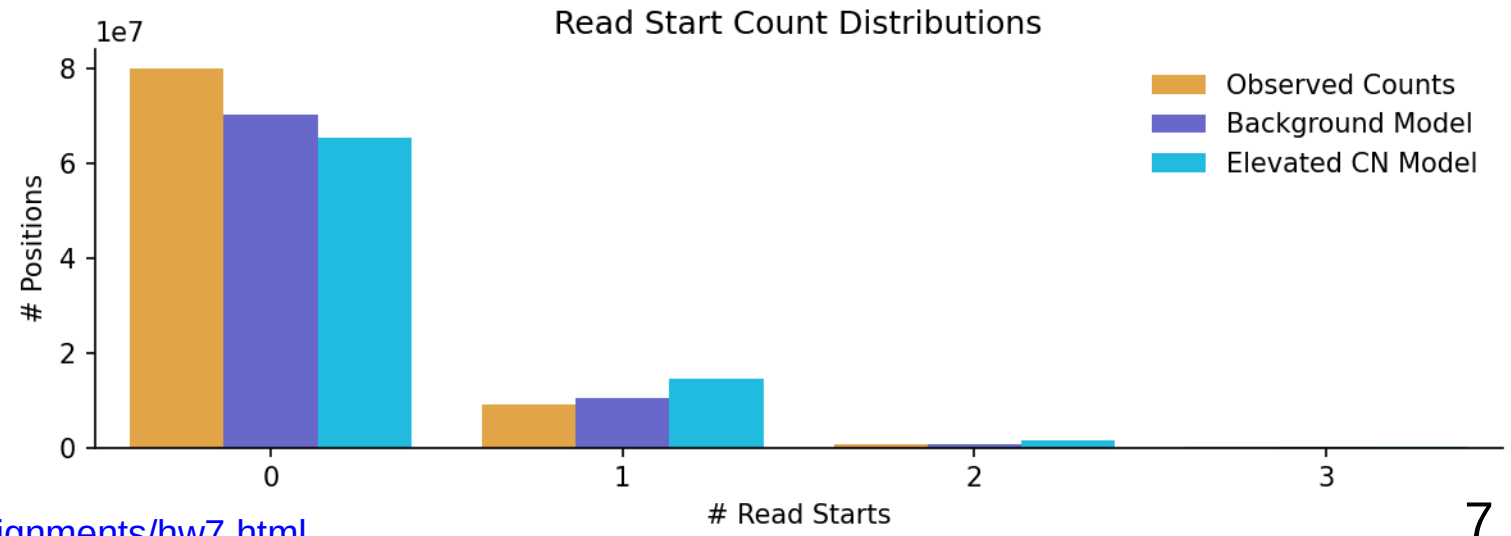
Created LLR Scoring Scheme

```
# 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):\t{x}')
print(f'Background:\t{y_0.round(4)}')
print(f'Elevated CN:\t{y_1.round(4)}')
print(f'\nWeights:\t{weights.round(4)}\n')
```

```
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]
```

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



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

Homework 7 Overview

1. Create LLR Scoring Scheme

- Empirical data doesn't fit Poisson well
  - Amplification in sequencing library prep.

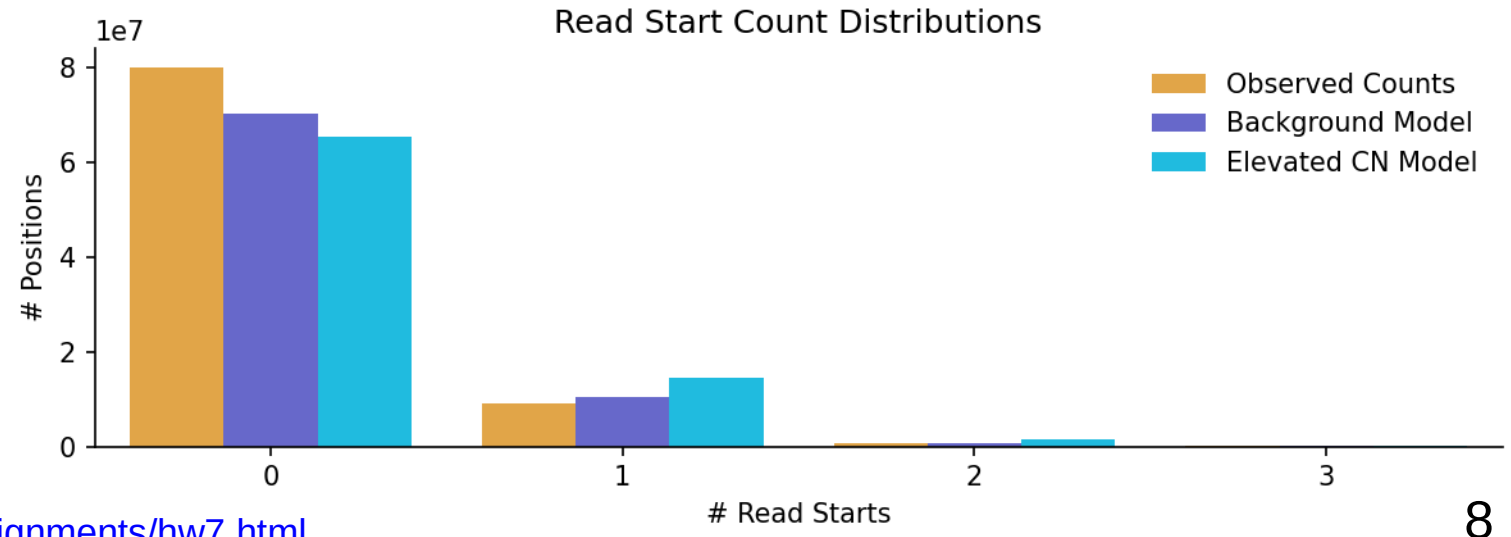
- Use HW6 results to refine our model

Use segment results from HW6:

- Count observed read start counts:
  - *Background*: in ALL segments
    - Sum counts for both types of segments
    - Correct for N's in reference (see HW7)
  - *Elevated*: in elevated segments only
    - No N correction

- Convert counts to frequencies

- Compute LLR with log2



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

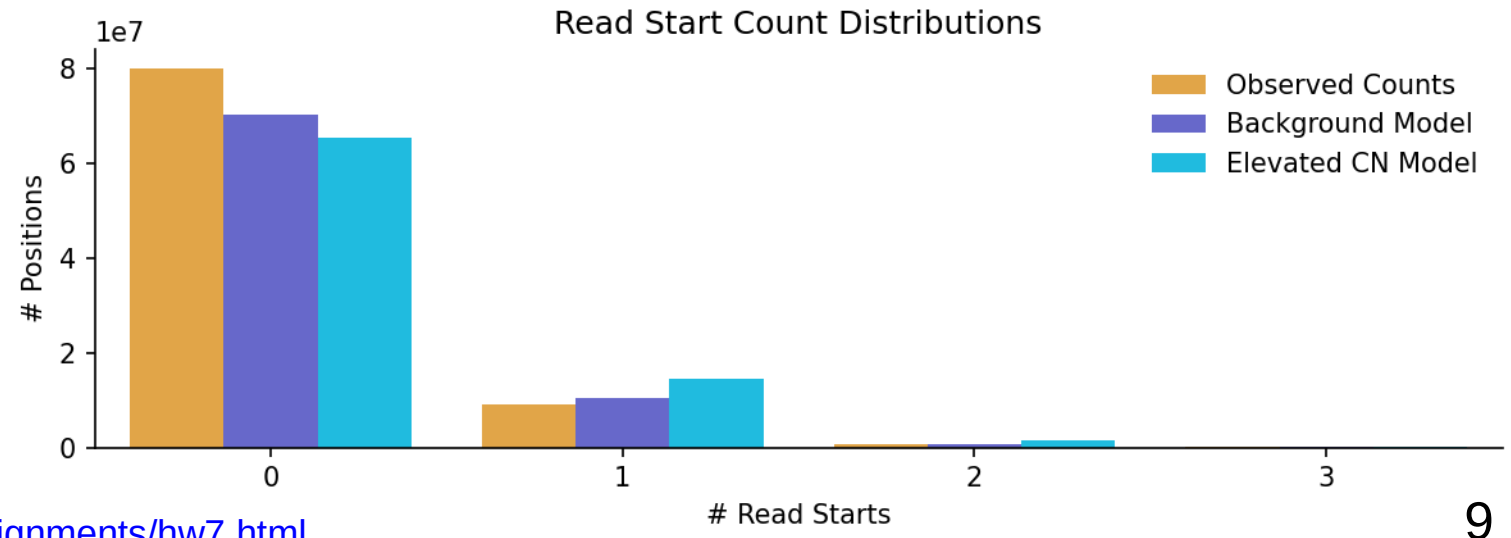
Homework 7 Overview

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
```



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

Homework 7 Overview

```
Assignment: GS540 HW7
Name: {YOURNAME}
Email: {YOUREMAIL}
Language: {YOURLANGUAGE}
Running time: {YOURRUNTIME}

Background frequencies:
0={#.####}
1={#.####}
2={#.####}
>=3={#.####}

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

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

```
Simulated data:
5 {# of segments with score >= 5}
6 {# of segments with score >= 6}
7 {# of segments with score >= 7}
```

```
Real data:
5 {# of segments with score >= 5}
6 {# of segments with score >= 6}
7 {# of segments with score >= 7}
.
.
.
list all the segment score counts for
(only first/last 3 shown here)
.
.
.
28 {# of segments with score >= 28}
29 {# of segments with score >= 29}
30 {# of segments with score >= 30}
```

for scores between 5 and 30

```
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
(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

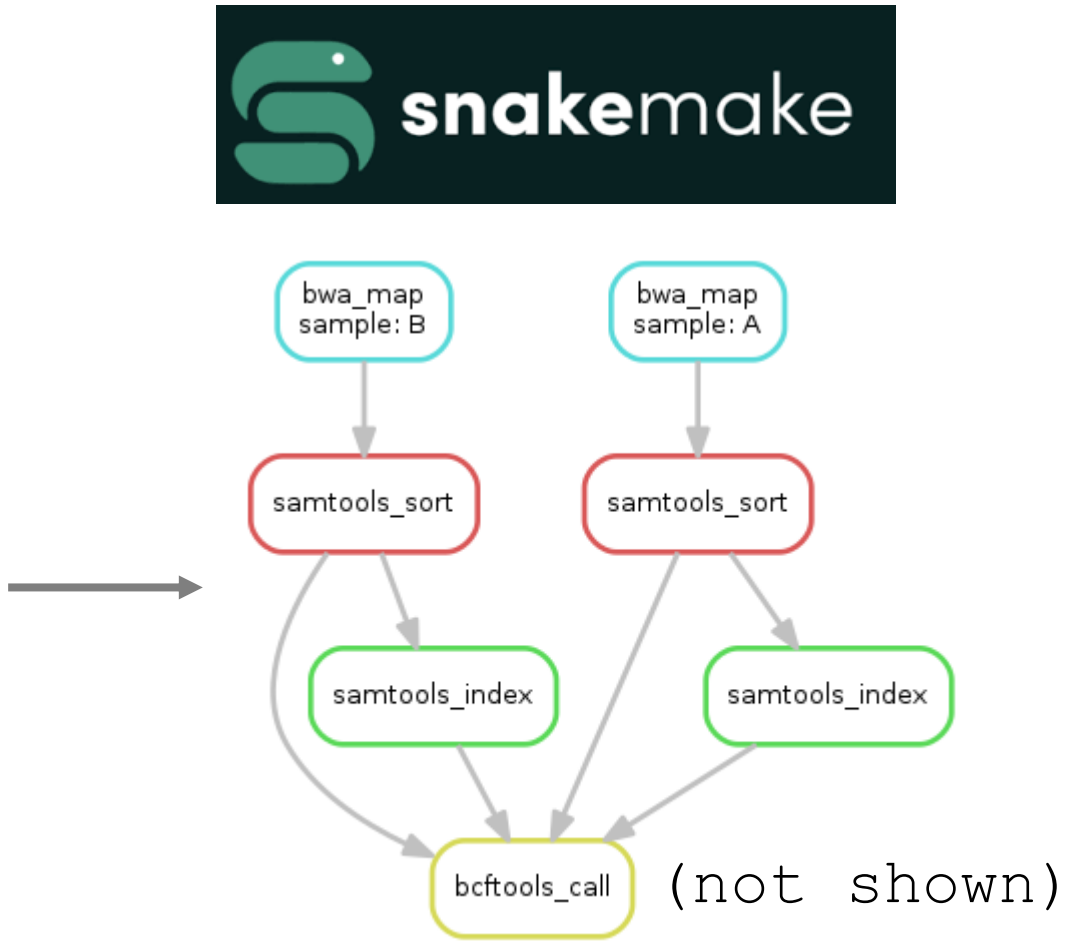
- Homework 7 overview

- Related topics:
  - Example Snakemake pipeline

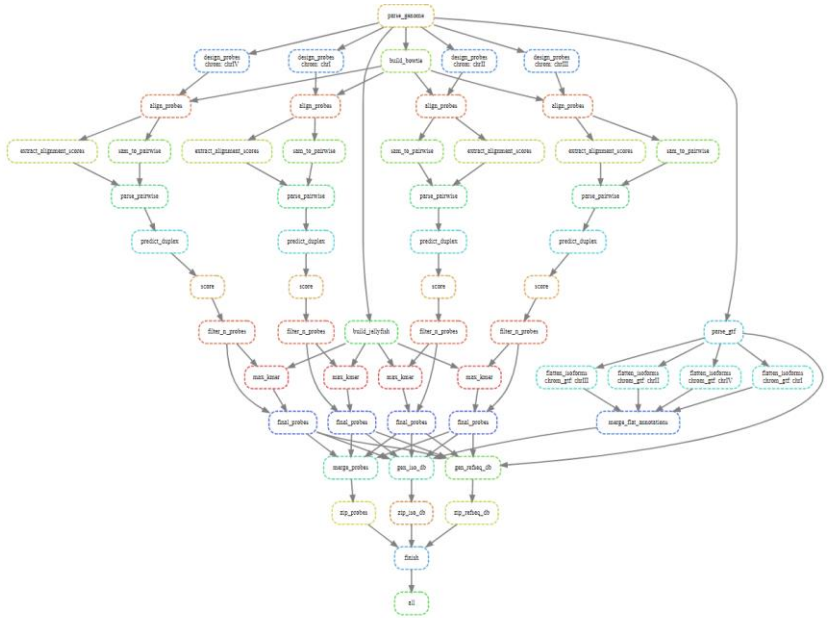
- Homework 6 & 7 questions

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}"
```

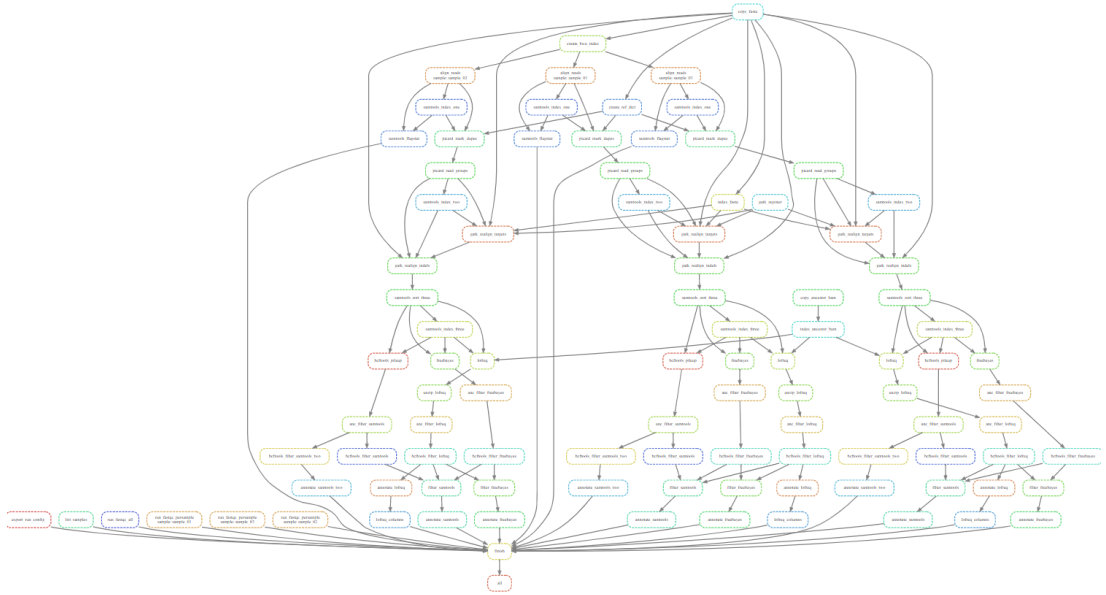


Simple pipeline in Snakemake



PaintSHOP Pipeline

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



yEvo Pipeline

Variant calling Snakemake pipeline for yEvo sequencing data

GitHub

CONDA

snakemake

Snakemake Demo Plan: Image Processing

Pattern 1: 3-color side-by-side

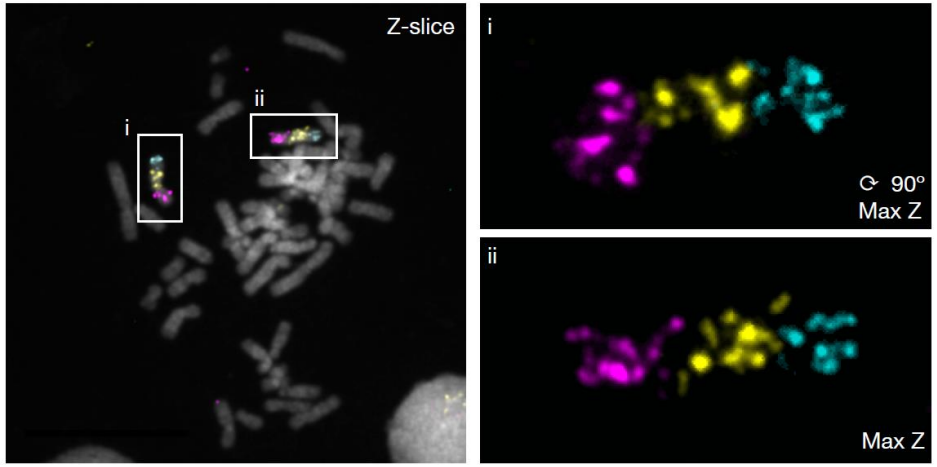


Image processing with python and Snakemake

- 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

```
concamp@b001:~/workspace/GS540_snakemake_demo/notebooks/data $ cat arr1.txt
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
concamp@b001:~/workspace/GS540_snakemake_demo/notebooks/data $ cat arr2.txt
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1	1	0
0	1	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0
0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1	1	0
0	1	0	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0	0	1	0
0	0	1	1	0	0	0	1	0	1	0	0	0	1	1	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
```

Images & Python

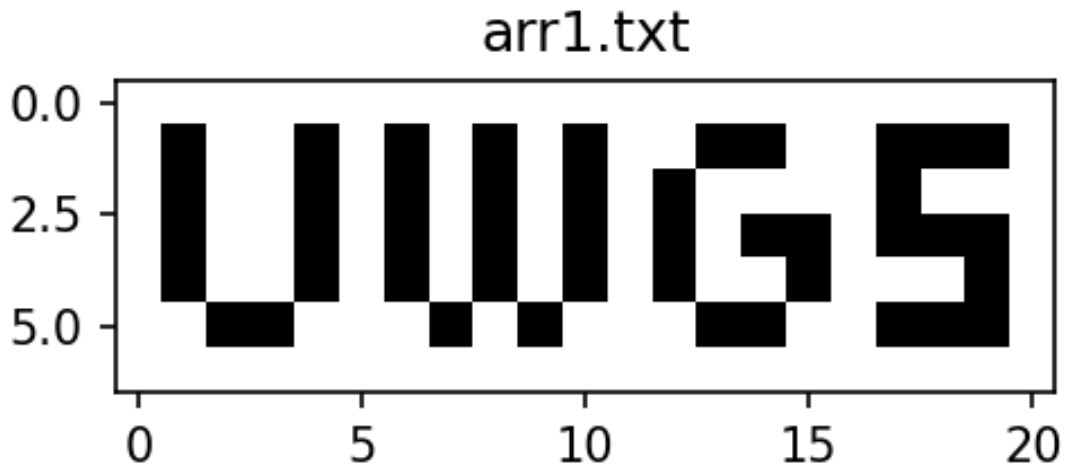
Load .txt files and convert to numpy

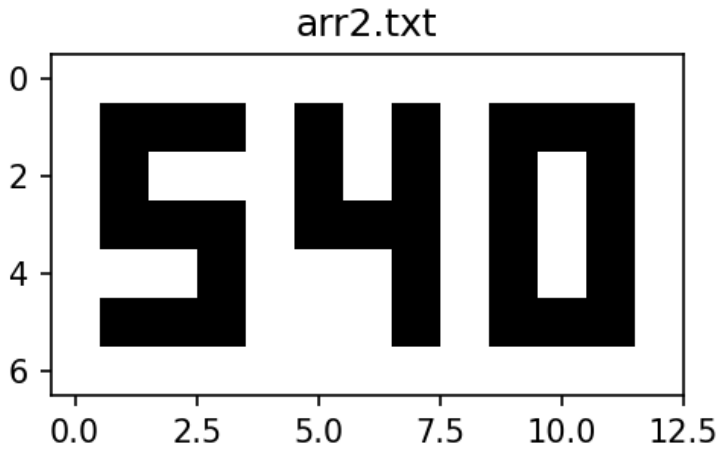
```
def load_arr(file_path):
    '''Load a 2D array from a text file and convert it to a numpy array.'''
    arr = pd.read_csv(file_path, sep='\t', header=None).values
    return arr

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

```
Array 1:
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1 1 0]
 [0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0]
 [0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0]
 [0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 1 0]
 [0 0 1 1 0 0 0 1 0 1 0 0 0 1 1 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]]
Array 2:
[[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]]
```

```
plt.imshow(arr1)
plt.imshow(arr2)
```

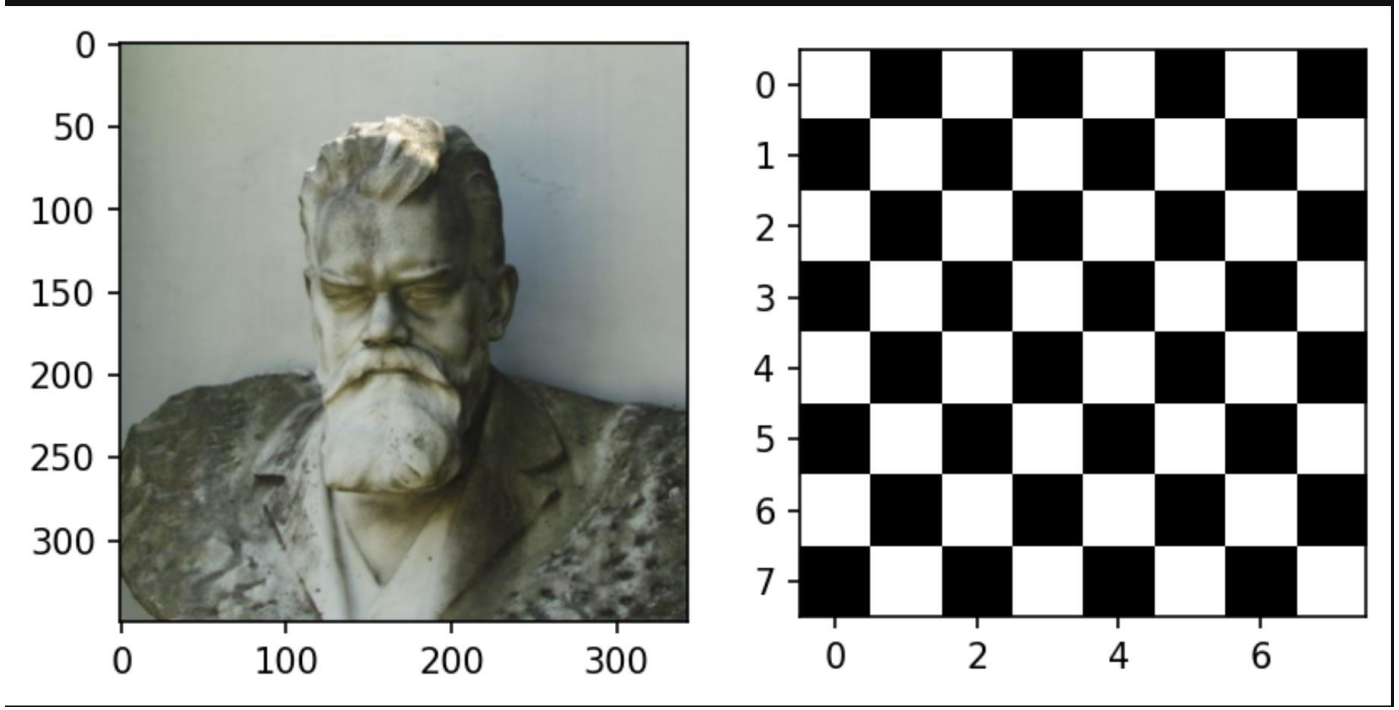
Visualize with matplotlib plt.imshow()





Images & Python

```
boltzmann = plt.imread('data/boltzmann.png')
chess = np.array([
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
])

fig, ax = plt.subplots(1, 2, dpi=150)
ax[0].imshow(boltzmann, cmap='Greys')
ax[1].imshow(chess, cmap='Greys')
plt.show()
```



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

create a mask as a handle to only digit pixels

```
# work with array2 for now
img = arr2.copy()

# create a mask for digit pixels in the 540 image
mask = (img == 1)
print(f'540 Digit Mask:\n{mask.astype(int)}')
```

```
540 Digit Mask:
[[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]]
```

use the mask to selectively change digit-pixel values

```
img[mask] = 99
print(img)
```

```
[[ 0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 99 99 99  0 99  0 99  0 99 99 99  0]
 [ 0 99  0  0  0 99  0 99  0 99  0 99  0]
 [ 0 99 99 99  0 99 99 99  0 99  0 99  0]
 [ 0  0  0 99  0  0  0 99  0 99  0 99  0]
 [ 0 99 99 99  0  0  0 99  0 99 99 99  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0]]
```

```
Left mask:
[[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]]
```

```
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]]
```

```
Right mask:
[[0 0 0 0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 0 0 0 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]
 [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]]
```

```
Digit mask:
[[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]]
```

AND gate

```
Left mask AND digit mask:
[[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 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 1 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]]
```

Images & Python

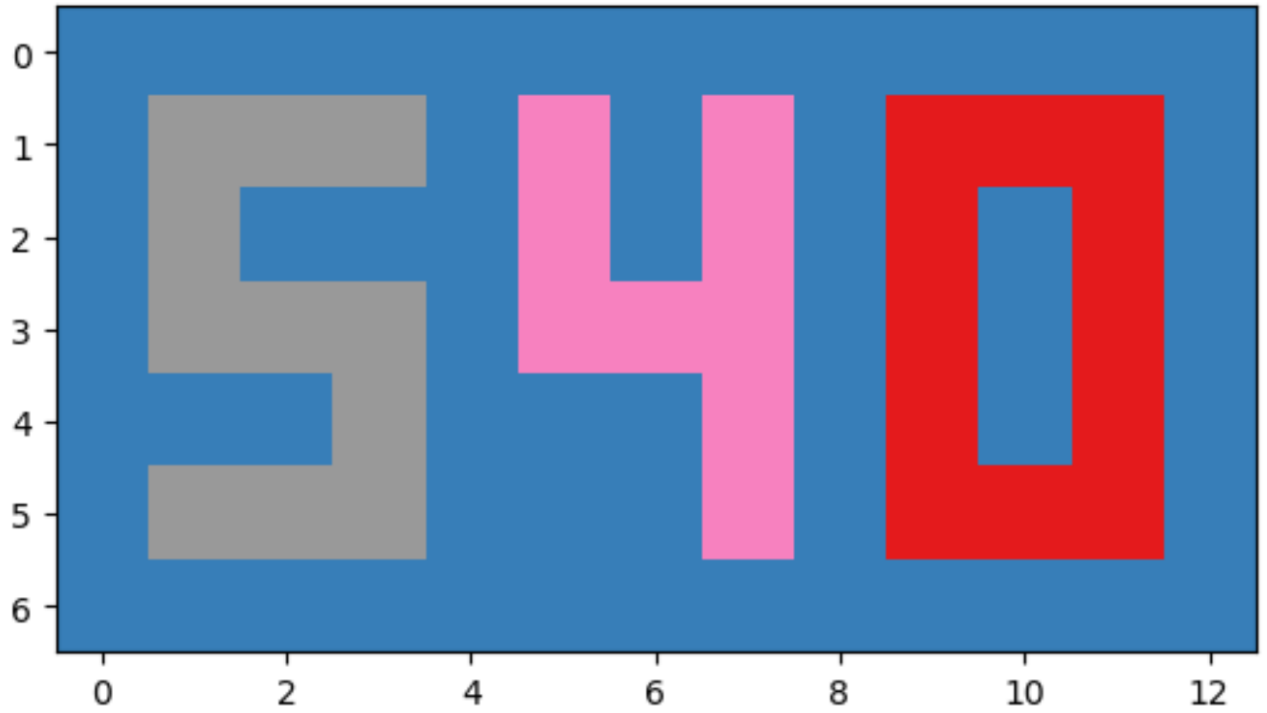
Masking with numpy

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)
```

```
[[1 1 1 1 1 1 1 1 1 1 1 1 1]
 [1 5 5 5 1 4 1 4 1 0 0 0 1]
 [1 5 1 1 1 4 1 4 1 0 1 0 1]
 [1 5 5 5 1 4 4 4 1 0 1 0 1]
 [1 1 1 5 1 1 1 4 1 0 1 0 1]
 [1 5 5 5 1 1 1 4 1 0 0 0 1]
 [1 1 1 1 1 1 1 1 1 1 1 1 1]]
```

```
# display image array
plt.imshow(img, cmap='Set1')
plt.show()
```



Snakemake Demo: Image Processing

Today's examples as a jupyter notebook

master GS540_snakemake_demo / notebooks / 01_binary_array_images.ipynb Go to file ...

conorcamplisson add python image example notebook and test images Latest commit a6d5e75 9 minutes ago History

1 contributor

417 lines (417 sloc) 356 KB Raw Blame

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_env` conda environment found in workflow/envs/main.yml

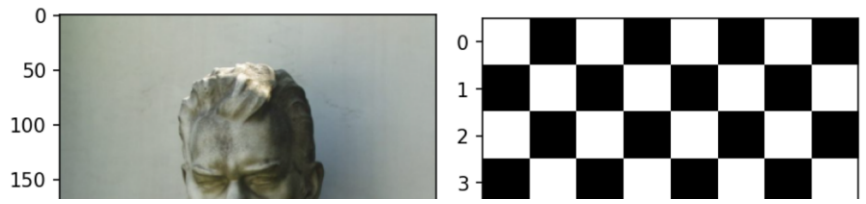
configure notebook

In [1]:
```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

print('DONE!')
```

```
DONE!
```

viewing images with matplotlib

In [2]:
```
boltzmann = plt.imread('data/boltzmann.png')
chess = np.array([
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
])

fig, ax = plt.subplots(1, 2, dpi=150)
ax[0].imshow(boltzmann, cmap='Greys')
ax[1].imshow(chess, cmap='Greys')
plt.tight_layout()
plt.savefig('figures/boltzmann_chess.png', dpi=150)
plt.show()
```



Conda-based install of snakemake/img libs

README.md

UWGS Genome 540 Snakemake Demo

Install with conda

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 -n base -c conda-forge mamba
```

3. Clone this repo:

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

GitHub

Access the demo pipeline repo:

https://github.com/conorcamplisson/GS540_snakemake_demo

Snakemake Demo: Image Processing

Commit history

Commits on Feb 16, 2023

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 | Go to file | Add file | Code

conorcamplisson add python image example notebook and test images — a6d5e75 — 17 minutes ago — 7 commits

docs/img	update docs	3 hours ago
notebooks	add python image example notebook and test images	17 minutes ago
workflow	add image processing conda env for pipeline	18 minutes ago
.gitignore	Initial commit	5 hours ago
LICENSE	Initial commit	5 hours ago
README.md	update docs	3 hours ago
environment.yml	add jupyter to conda env	17 minutes ago

GitHub

Access the demo pipeline repo:

https://github.com/conorcamplisson/GS540_snakemake_demo

Images & Python

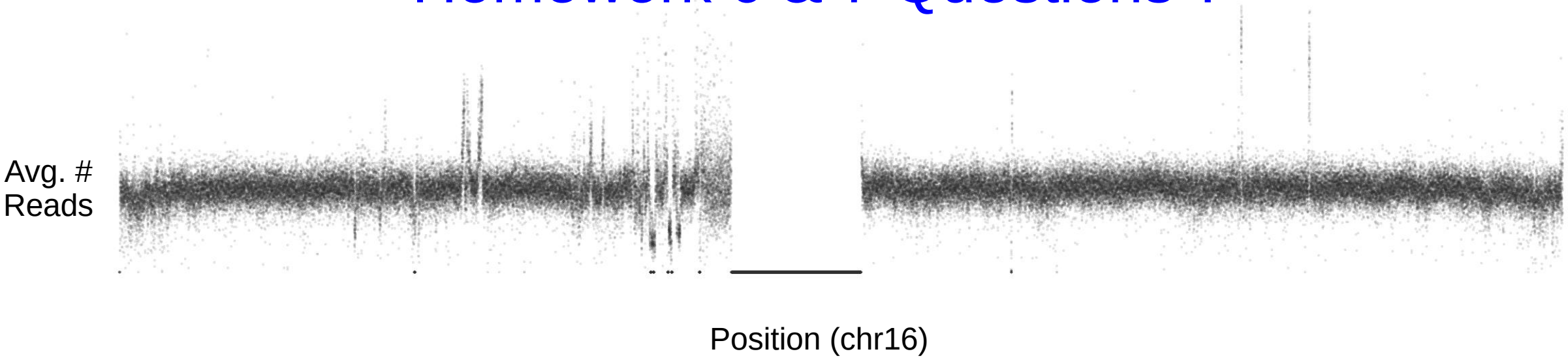
Automated Acknowledgements

- 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

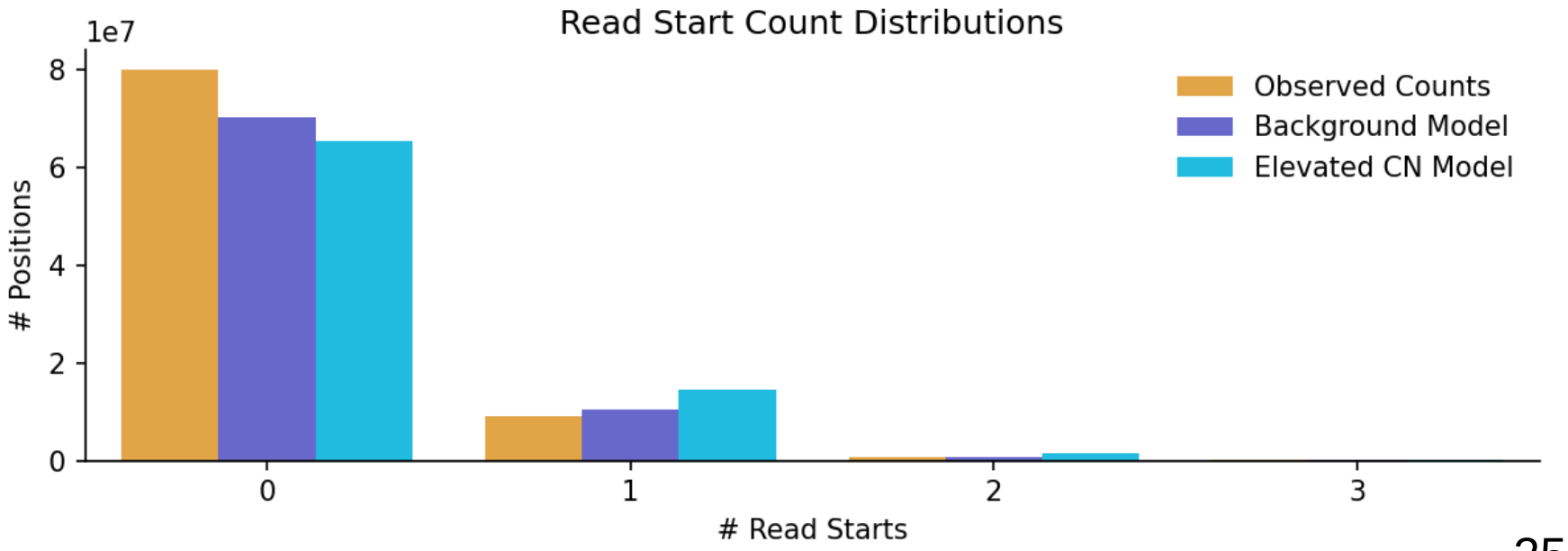
- Homework 7 overview
- Related topics:
  - Example Snakemake pipeline
- Homework 6 & 7 questions

Homework 6 & 7 Questions ?



chm13.chr16.txt

```
16      1       0
16      2       0
16      3       0
16      4       0
16      5       0
      [ ... ]
16      14793   0
16      14794   1
16      14795   3
16      14796   0
      [ ... ]
```



Reminders

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

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

. . .