Genome 540 Discussion

Conor Camplisson

February 21st, 2023

Genome Sciences
UNIVERSITY OF WASHINGTON

Outline

- Homework 6 wrap-up
- Related topics:
  - Image processing concepts, algorithms
    - Image segmentation, de-convolution, object detection
  - Information theory → imaging experiments
    - Hyperstacks, sampling, bit-depth
- Homework 7 questions

Outline

- Homework 6 wrap-up

- Related topics:
  - Image processing concepts, algorithms
    - Image segmentation, de-convolution, object detection
  - Information theory → imaging experiments
    - Hyperstacks, sampling, bit-depth

- Homework 7 questions

Homework 6 Wrap-up

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
      [ ... ]
```

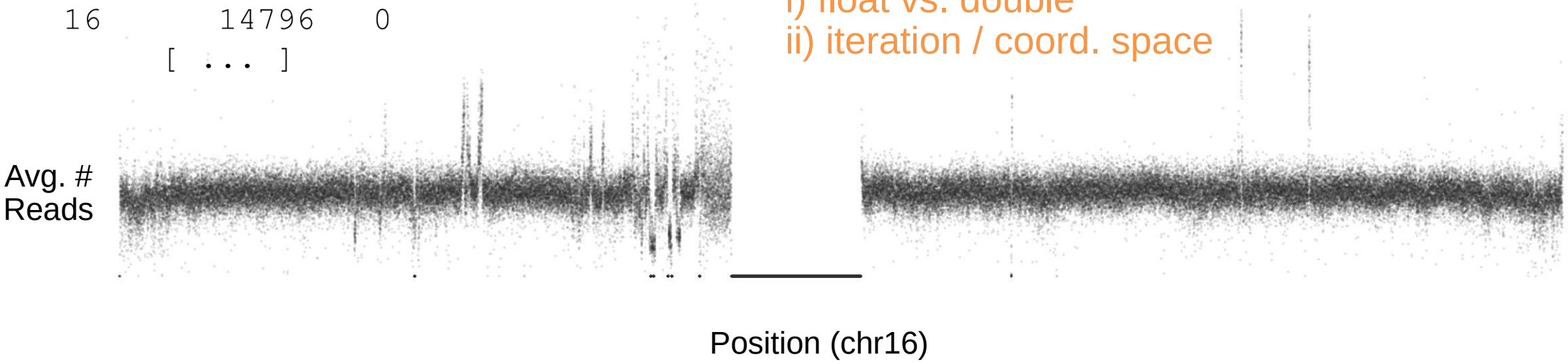
Goal: to find CNVs using D-segments

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

i) float vs. double
ii) iteration / coord. space



Outline

- Homework 6 wrap-up

- Related topics:
  - Image processing concepts, algorithms
    - Image segmentation, de-convolution, object detection
  - Information theory → imaging experiments
    - Hyperstacks, sampling, bit-depth

- Homework 7 questions

Outline

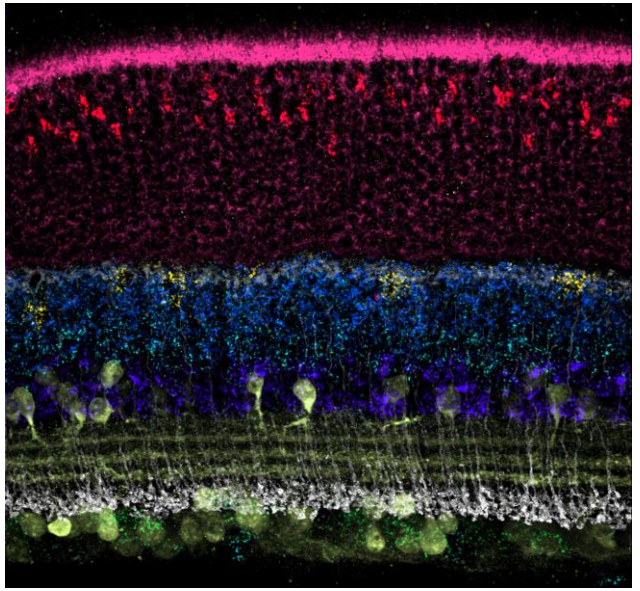
- Homework 6 wrap-up

- Related topics:
  - Image processing concepts, algorithms
    - Image segmentation, de-convolution, object detection
  - Information theory → imaging experiments
    - Hyperstacks, sampling, bit-depth

- Homework 7 questions

Image Segmentation

Segmentation Problems



GS 540:

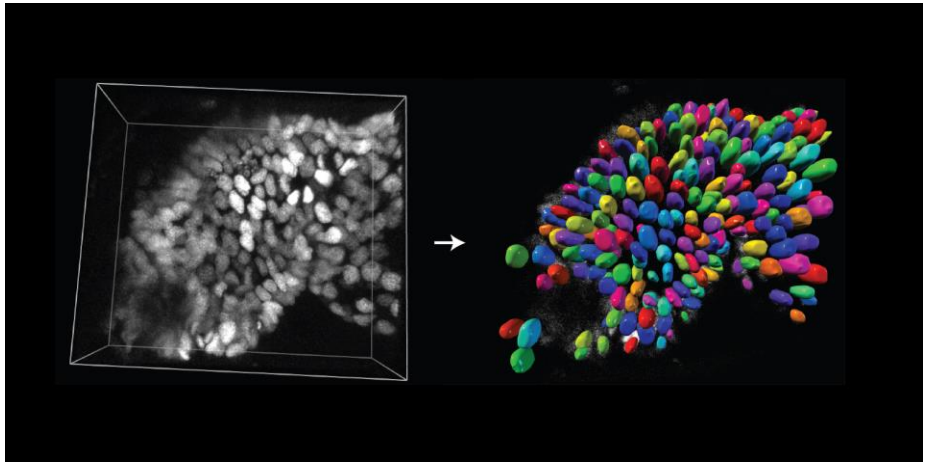
Segment:

- A Chromosome into elevated/non-elevated CN (HW6, HW7)
- A genome into GC-rich/AT-rich states (HW8)
- An alignment into conserved/neutral states (HW9)

Microscopy:

Answer for all pixels:

- [cell segmentation] Is this pixel in a cell?
  - Which pixels does this cell occupy?
- [nuclear segmentation] Is this pixel in the nucleus?
  - Which pixels does the nucleus occupy?



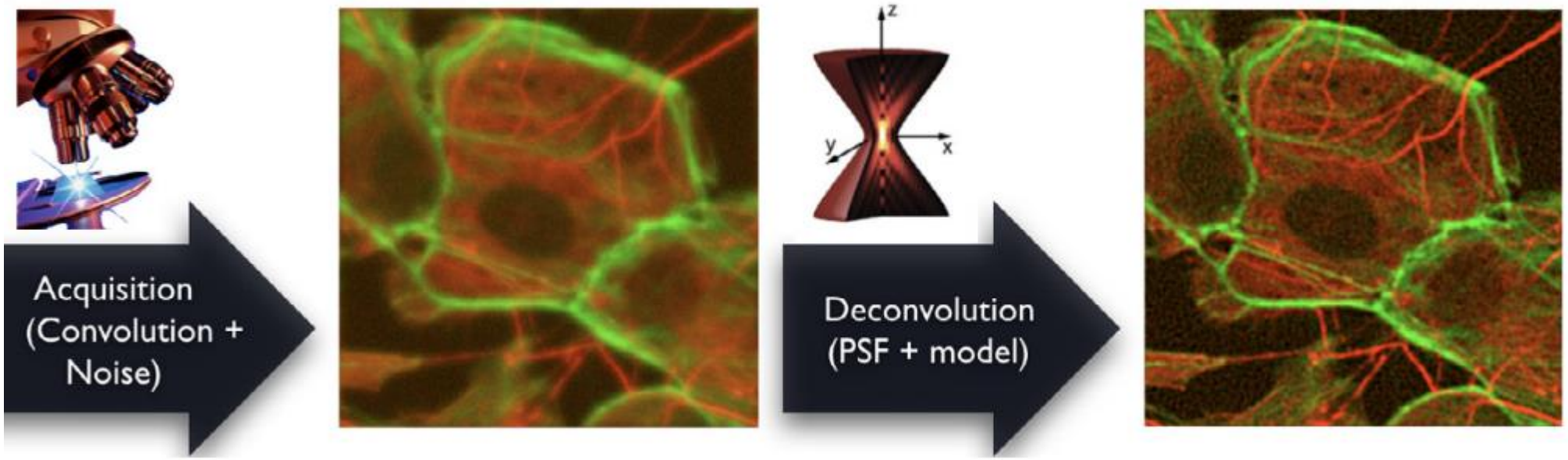
Active area of research:

- necessary to cash in on spatial bio wet lab technologies
- hard problems, diverse cell shapes, crowding, 3D
- Many recent machine learning approaches

i) Kishi, J.Y., Lapan, S.W., Beliveau, B.J. et al. *Nat Methods* **16**, 533–544 (2019)
ii) https://github.com/stardist/stardist

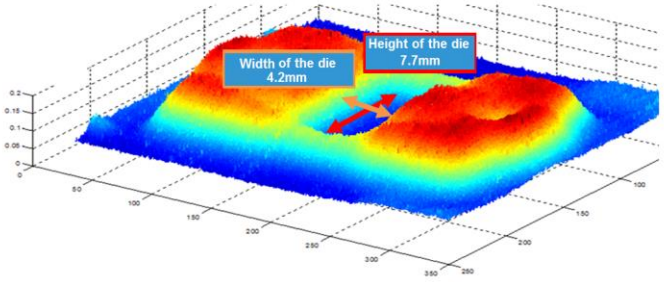
Image Deconvolution

- Instruments are not perfect; signal is convoluted during acquisition
- Model the convolutions using a Point Spread Function (PSF)
  - measure standards, determine PSF on each microscope / optical configuration
- Using the PSF model, correct for the convolutions



https://blog.biodock.ai/image-deconvolution/

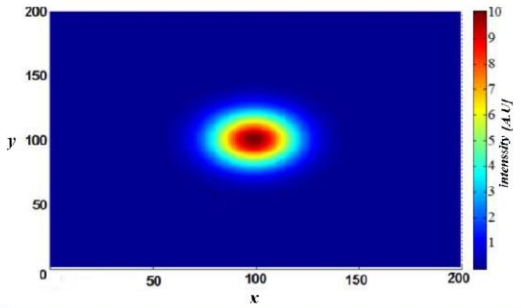
Point Spread Functions (PSFs)



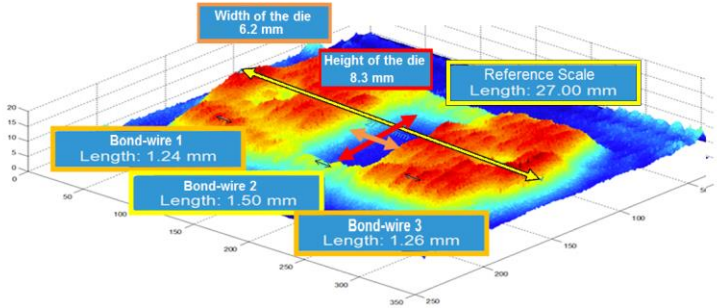
Low resolution THz Image

$^{-1}$

Devonvolution



Mathematically modelled PSF



High resolution THz Image

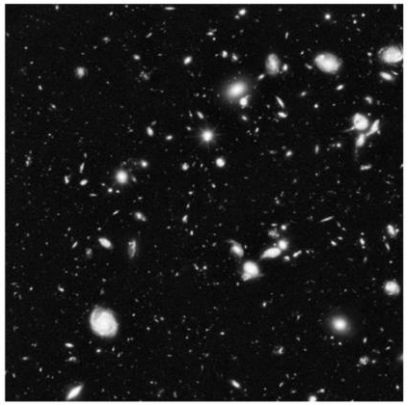
- Point spread function (PSF) used in deconvolution
- Gaussian common model for point sources / laser systems
- Measure empirically to diagnose optics issues in microscopy

https://en.wikipedia.org/wiki/Point_spread_function

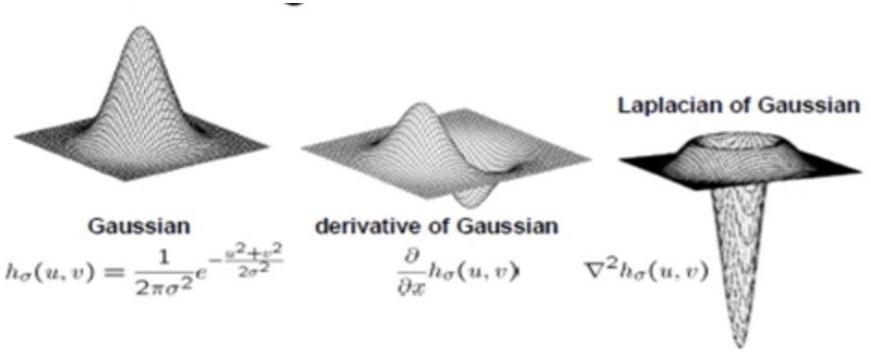
Blob detection with Gaussians

Hubble eXtreme Deep Field

To greyscale



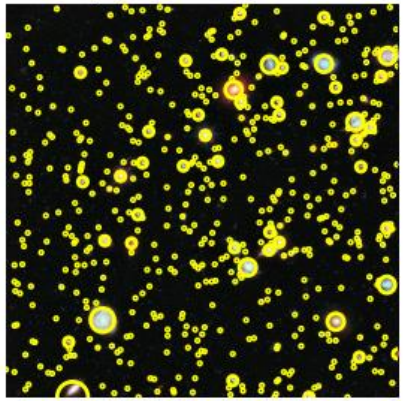
Choosing a Gaussian Model



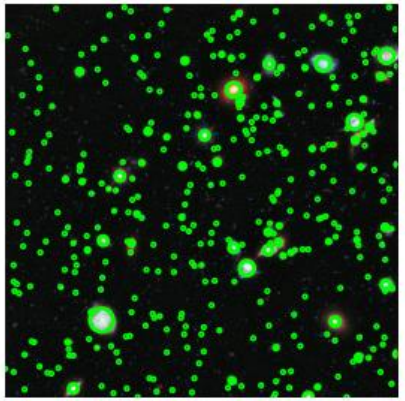
Each bright dot in the image is a star or a galaxy.

Three different blob finding algorithms (all using Gaussian models) are used:

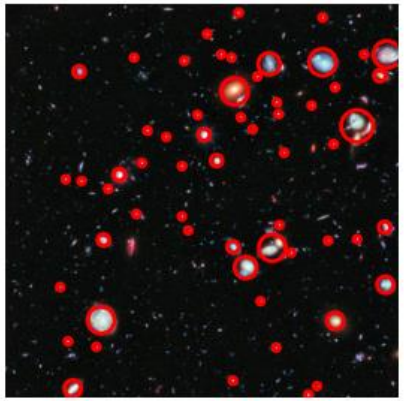
Laplacian of Gaussian



Difference of Gaussian



Determinant of Hessian



https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_blob.html

Some conceptual overlap

	Segmentation Problems	"Object Finding" Problems
GS 540:	• elevated/non-elevated CN (HW6, HW7) • GC-rich/AT-rich states (HW8) • conserved/neutral states (HW9)	Where are the "sites"? • Build a data structure (HW1) or train a site model (HW3) • Scan through every position in the 1D sequence and assess that position using model
Microscopy:	• Cell segmentation • Nuclear segmentation • Other applications (astronomy, computer vision, etc.)	Where are the fluorescent spots? • Use a Gaussian model • Scan through every position in the 2D image and assess that position using model

Outline

- Homework 6 wrap-up

- Related topics:
  - Image processing concepts, algorithms
    - Image segmentation, de-convolution, object detection
  - Information theory → imaging experiments
    - Hyperstacks, sampling, bit-depth

- Homework 7 questions

Snakemake Demo Plan: Image Processing

Pattern 1: 3-color side-by-side

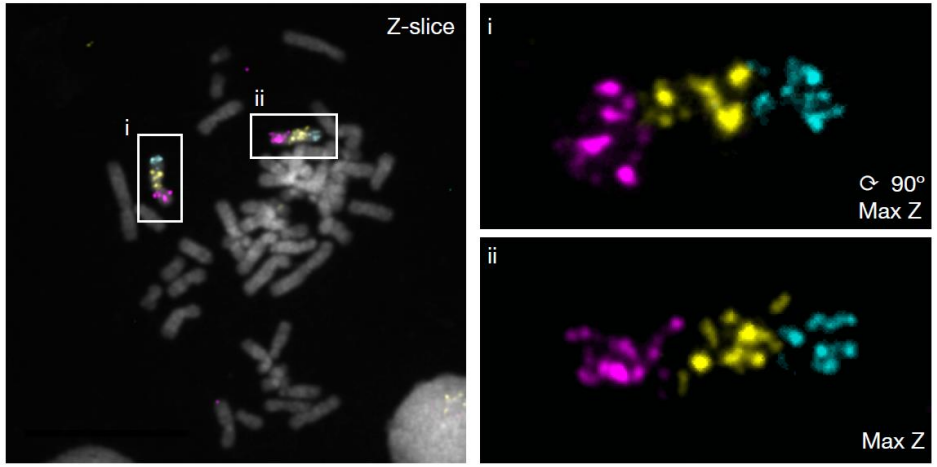


Image processing with python and Snakemake

- 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

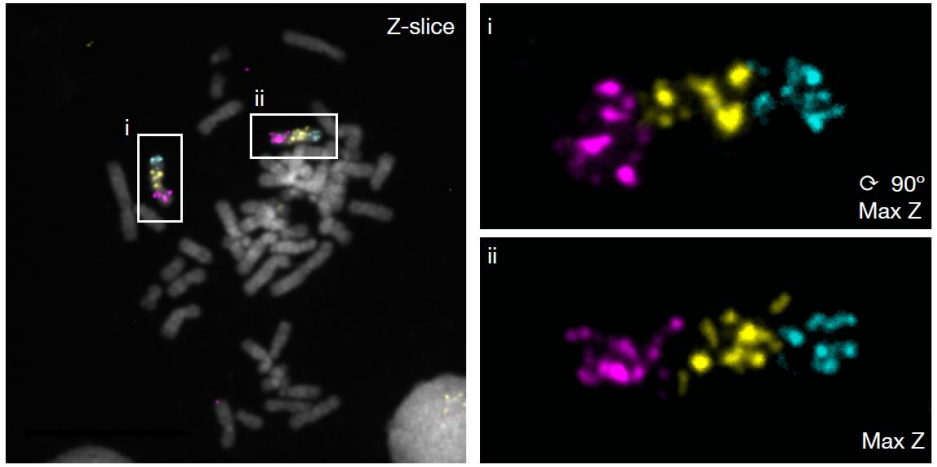
Intro to Hyperstack Images

Hyperstack Dimensions: (x, y, z, c, t)

- (x, y) move microscope stage to one/many region(s) of interest (R.O.I)
- (z) while ^ there, acquire images at one/many focal planes (moving stage in z)
- (c) in one/many fluorescent channels
- (t) at one/many timepoints

Our Demo Pipeline Inputs

- .nd2 hyperstack images, each with:
  - A single (x, y) field of view
  - Several (z) slices
  - Several (c) fluorescent channels
  - A single (t) observation only



Pipeline Specification

Input: .nd2 files (3D hyperstacks)

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

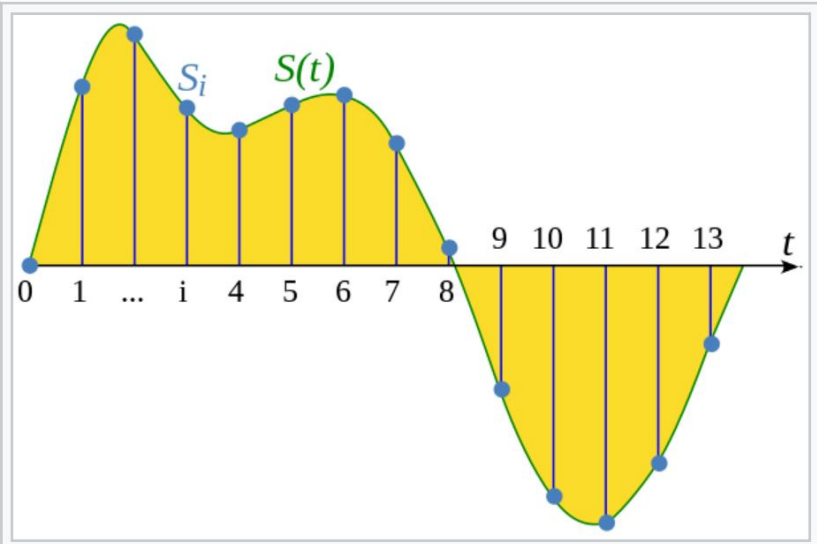
Output:

- plots of pixel intensity, spot size
- .csv file with stats per sample

Nyquist–Shannon sampling theorem

Sampling (signal processing)

Discrete observations
of a continuous signal

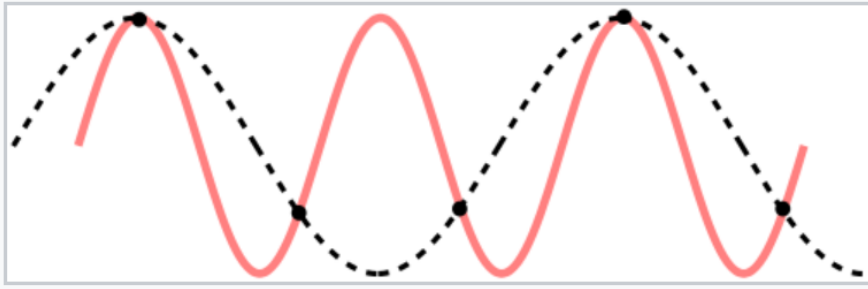


Signal sampling representation. The continuous signal $S(t)$ is represented with a green colored line while the discrete samples are indicated by the blue vertical lines.

Sampling Rate

Need enough samples to
capture info in signal

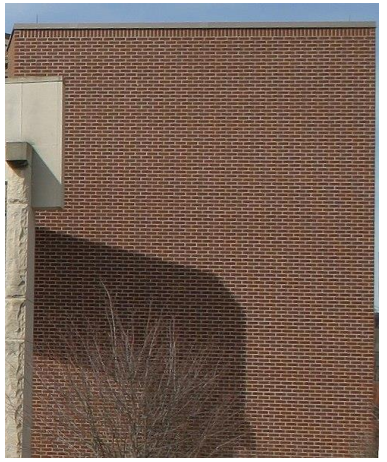
(recall: “info” ~
resolution of uncertainty ~
loss of entropy/ambiguity)



The samples of two sine waves can be identical when at least one of them is at a frequency above half the sample rate.

Subsampled

Proper
sampling

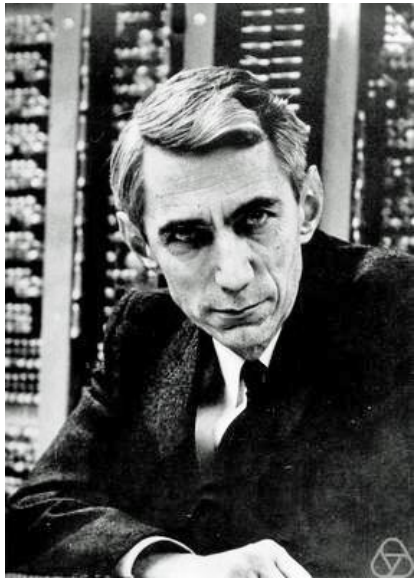


https://en.wikipedia.org/wiki/Sampling_%28signal_processing%29

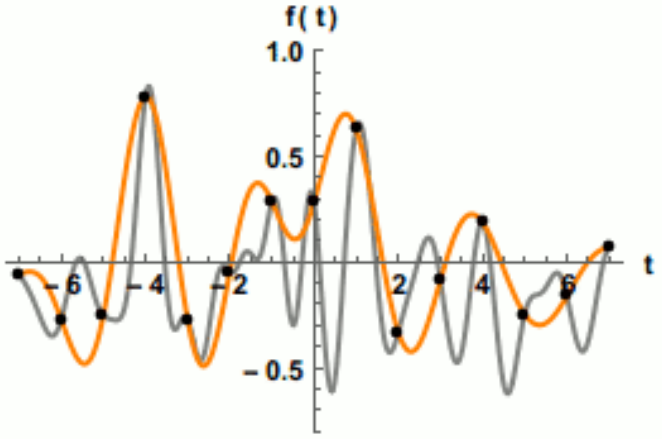
Nyquist–Shannon sampling theorem

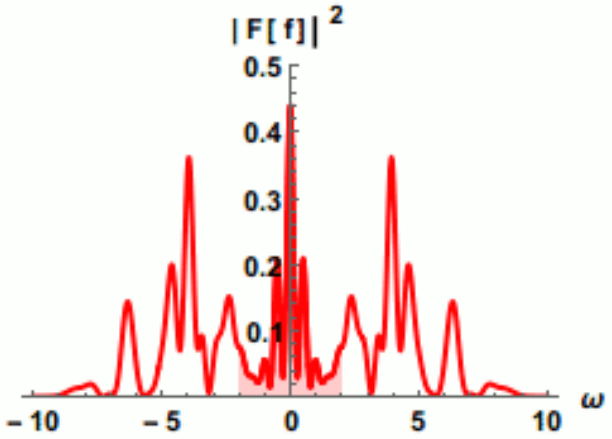
- Sample at (at least) double the highest frequency in the signal
- Fundamental bridge between continuous-time signals and discrete-time signals

Harry Nyquist



Claude Shannon





As sampling (black markers) rate increases, the reconstruction (gold) of they continuous signal (grey) improves.

(see link for detailed figure explanation)

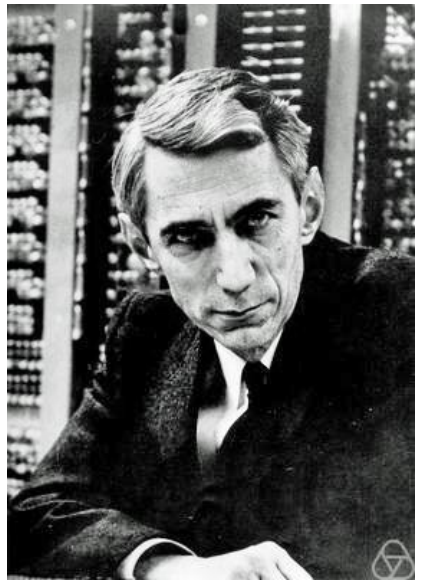
https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

Recall: 8bit-int encoding DNA

8-bit encoding DNA, 1 byte per nucleotide

```python
# generate a random int DNA sequence using different data types
k = 1000
print(f'DNA sequence length = {k}. Size in RAM:\n')
print(f'default ({np.random.randint(0, 4, k).dtype}):\t{np.random.randint(0, 4, k).nbytes} bytes')
print(f'\tint:\t\t{np.random.randint(0, 4, k, dtype=int).nbytes} bytes')
print(f'\tint32:\t\t{np.random.randint(0, 4, k, dtype=np.int32).nbytes} bytes')
print(f'\tint16:\t\t{np.random.randint(0, 4, k, dtype=np.int16).nbytes} bytes')
print(f'\tint8:\t\t{np.random.randint(0, 4, k, dtype=np.int8).nbytes} bytes')
```

```
DNA sequence length = 1000. Size in RAM:

default (int64):        8000 bytes
        int:            8000 bytes
        int32:          4000 bytes
        int16:          2000 bytes
        int8:           1000 bytes
```



Claude Shannon

- 4 x (0.25 * log2(0.25)) = 2.0 bits

Note: could use 2 bits to store nucleotides in theory, but 1 byte (8 bits) is practical in python.

Encoding scheme:

A = 0 = 00
C = 1 = 01
G = 2 = 10
T = 3 = 11

Information entropy

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log \mathrm{P}(x_i)$$

Recall: 8-bit Integers

Can store 256 values (0 – 255)

```
[[0 0 0 0 0 0 0 0]   0
 [0 0 0 0 0 0 0 1]   1
 [0 0 0 0 0 0 1 0]   2
 [0 0 0 0 0 0 1 1]   3
 [0 0 0 0 0 1 0 0]   4

       [...]

 [1 1 1 1 1 1 0 0]   252
 [1 1 1 1 1 1 0 1]   253
 [1 1 1 1 1 1 1 0]   254
 [1 1 1 1 1 1 1 1]]  255
```

Overflow example

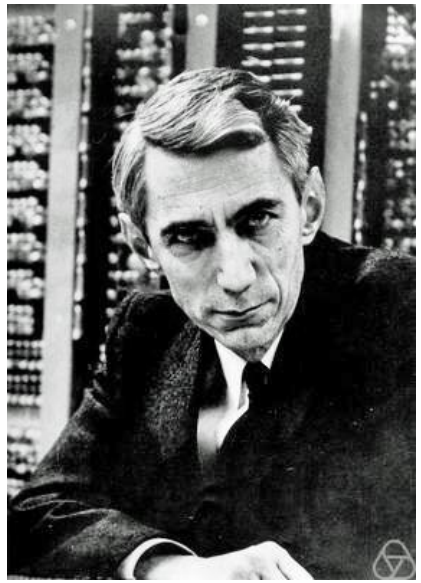
```python
# start with int arrays of both data types
a = np.arange(5, dtype=int)
b = np.arange(5, dtype=np.uint8)

# 8-bit overflow demo
print(f'{a} (a)')
print(f'{b} (b)\n')
print(f'{a * 100} (a * 100)')
print(f'{b * 100} (b * 100)')
```

```
[0 1 2 3 4] (a)
[0 1 2 3 4] (b)

[  0 100 200 300 400] (a * 100)
[  0 100 200  44 144] (b * 100)
```

Images & Bit-depth

8-bit Image

- 1 byte per pixel
- Intensity range (0..255)



Claude Shannon

16-bit Image

- 2 byte per pixel
- Intensity range (0..65,535)

$$-\sum_{i=0}^{255}\left(\frac{1}{256}\right)\log_2\left(\frac{1}{256}\right) = 8.0\ bits$$

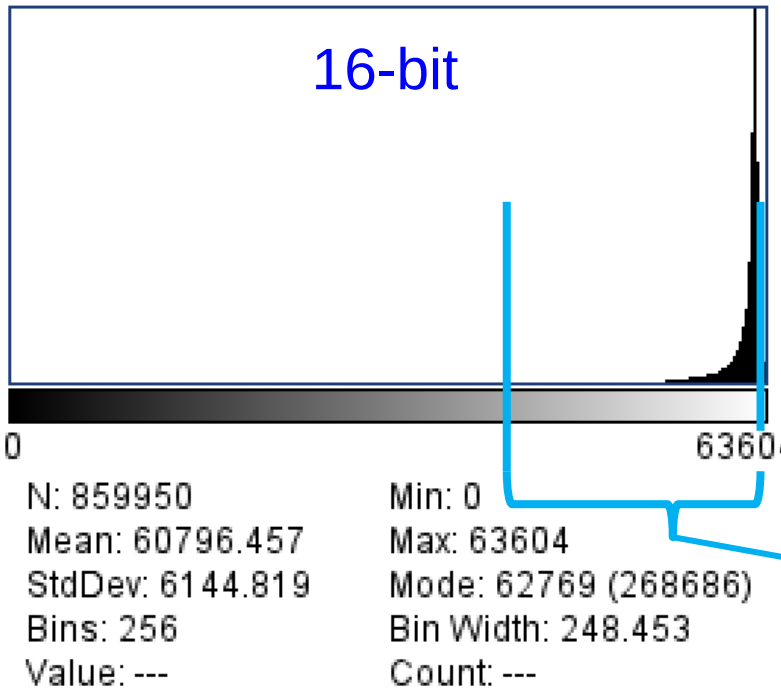
$$-\sum_{i=0}^{65535}\left(\frac{1}{65536}\right)\log_2\left(\frac{1}{65536}\right) = 16.0\ bits$$

Information entropy

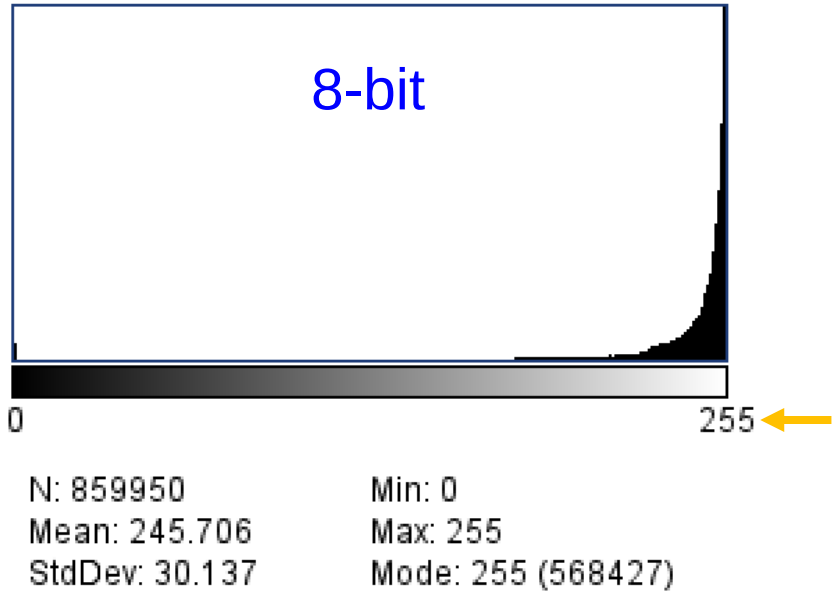
$$\mathrm{H}(X) = -\sum_{i=1}^{n}\mathrm{P}(x_i)\log\mathrm{P}(x_i)$$

Images & Bit-depth

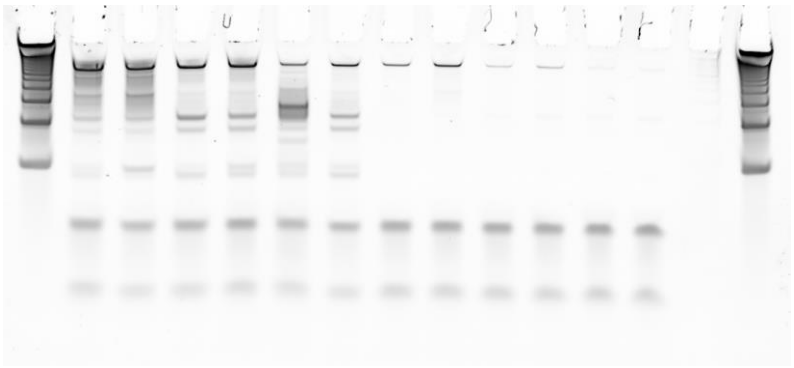
Pixel intensity hist. for image



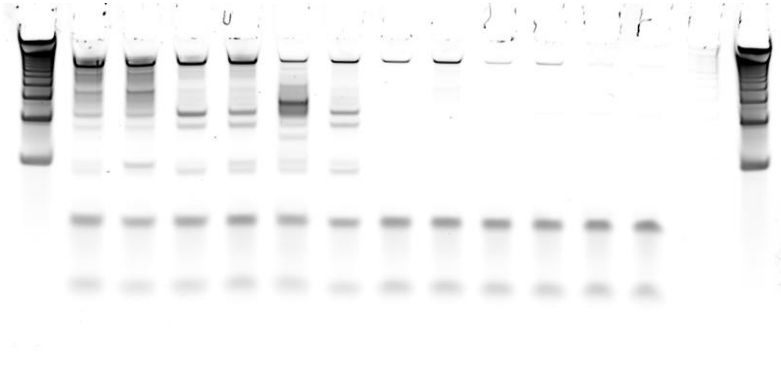
- Acquire with 16-bit dynamic range (0..65,535)
  - “cast a wide net”, large dynamic range, robust to bright/dim signals across samples/exps/colors
- Useful info in only a subset of dynamic range
  - If {subset} size >= 1/256th of hist, you have >= 8-bit info throughout {subset}, truncate, convert to 8-bit
  - End result: signal information fills the entire 8-bit value space (hard to imagine if 8-bit acquisition!)



16-bit raw image



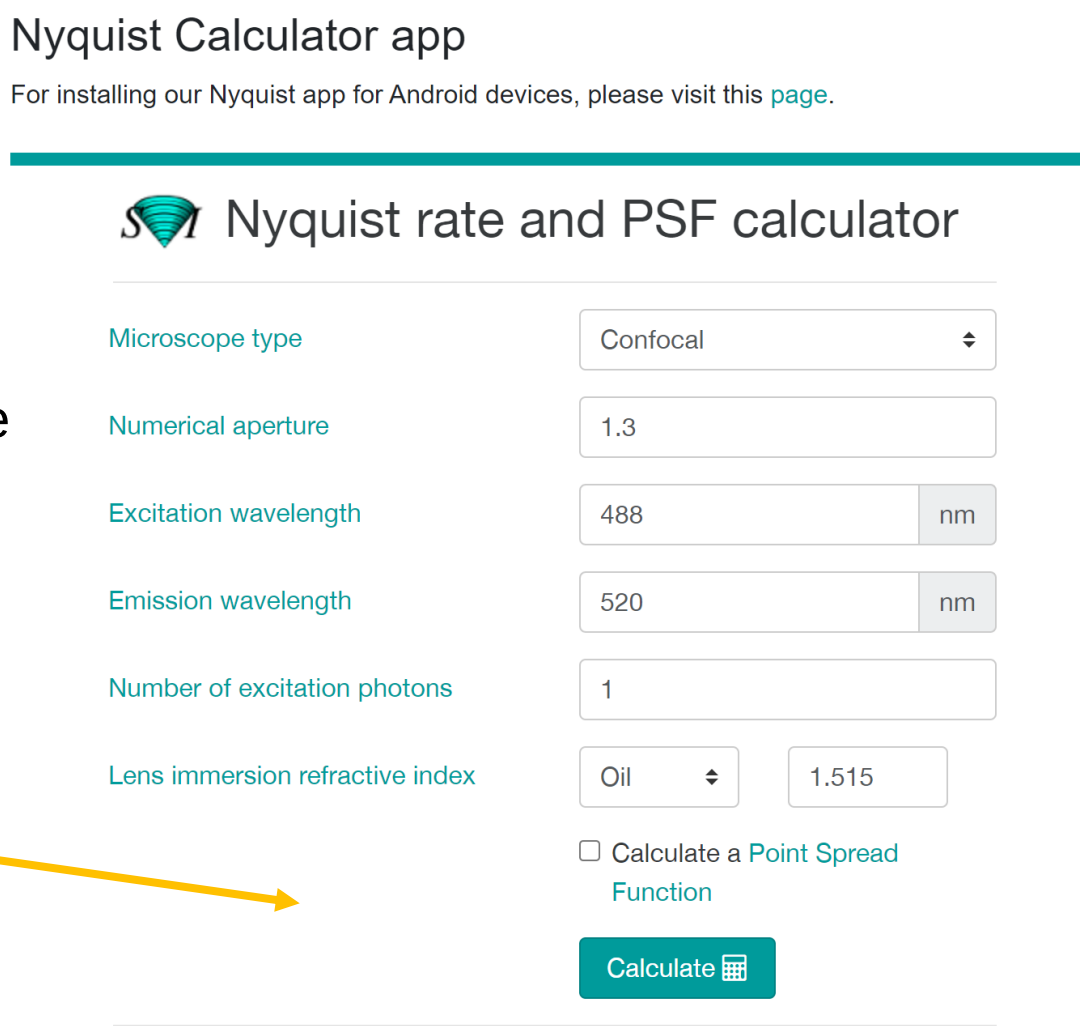
8-bit subset of hist.



Note: visual difference due to contrast adjustment (histogram truncation), not bit depth!

Summary

- Segmentation & object detection problems
  - Parallels in 540 psets
- Models
  - Machine learning models for segmentation
  - Point spread functions (PSFs) for modeling noise
    - Use PSF model to deconvolve images
  - Gaussian model(s) for fluorescent puncta
- Information theory informs experiment design
  - Nyquist sampling in z-stacks
  - Bit-depth during acquisition, processing



https://svi.nl/NyquistCalculator

Outline

- Homework 6 wrap-up

- Related topics:
  - Image processing concepts, algorithms
    - Image segmentation, de-convolution, object detection
  - Information theory → imaging experiments
    - Hyperstacks, sampling, bit-depth

- Homework 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
      [ ... ]
```

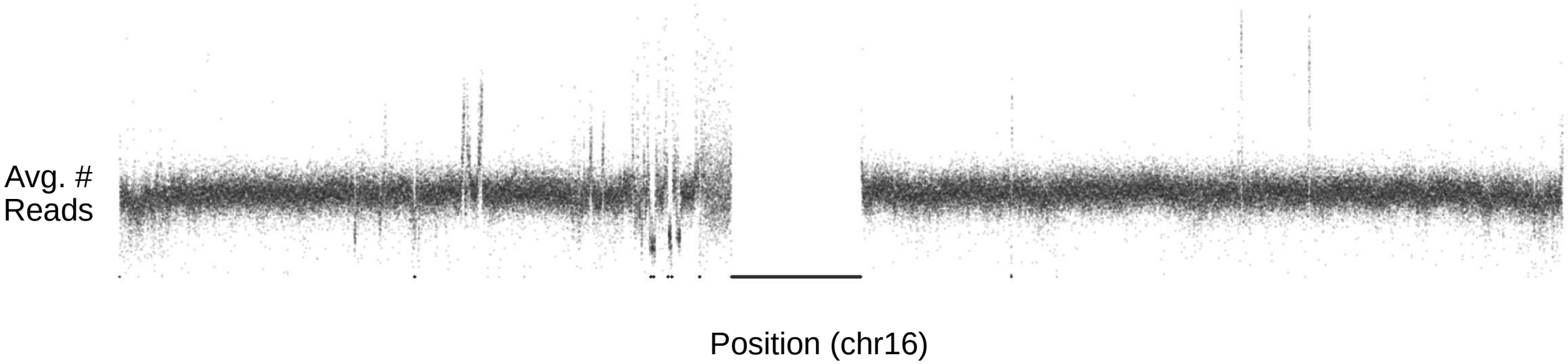
Data: next-gen read alignments to genome, CHM13 chr16

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)

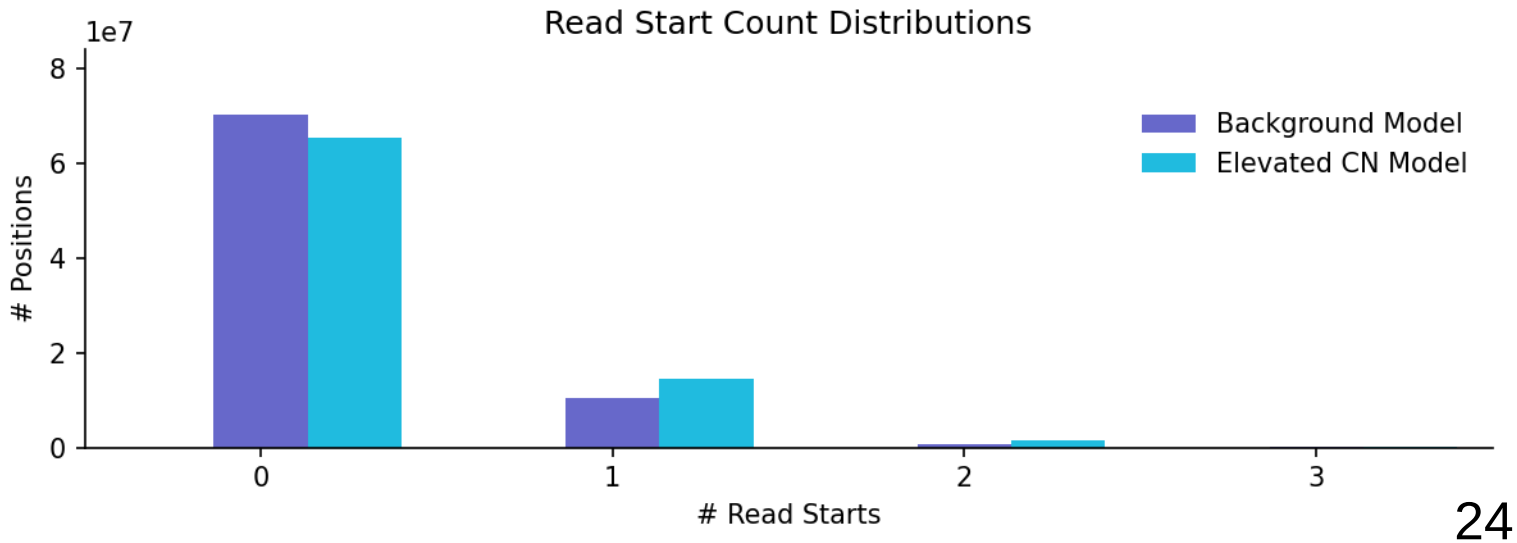
```python
# compute mean read count, adjusting for N's in denominator
N_CORRECTION = 8422401
ADJ_CHROM_SIZE = len(df) - N_CORRECTION
ADJ_MEAN_COUNT = df['num_reads'].sum() / ADJ_CHROM_SIZE

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)

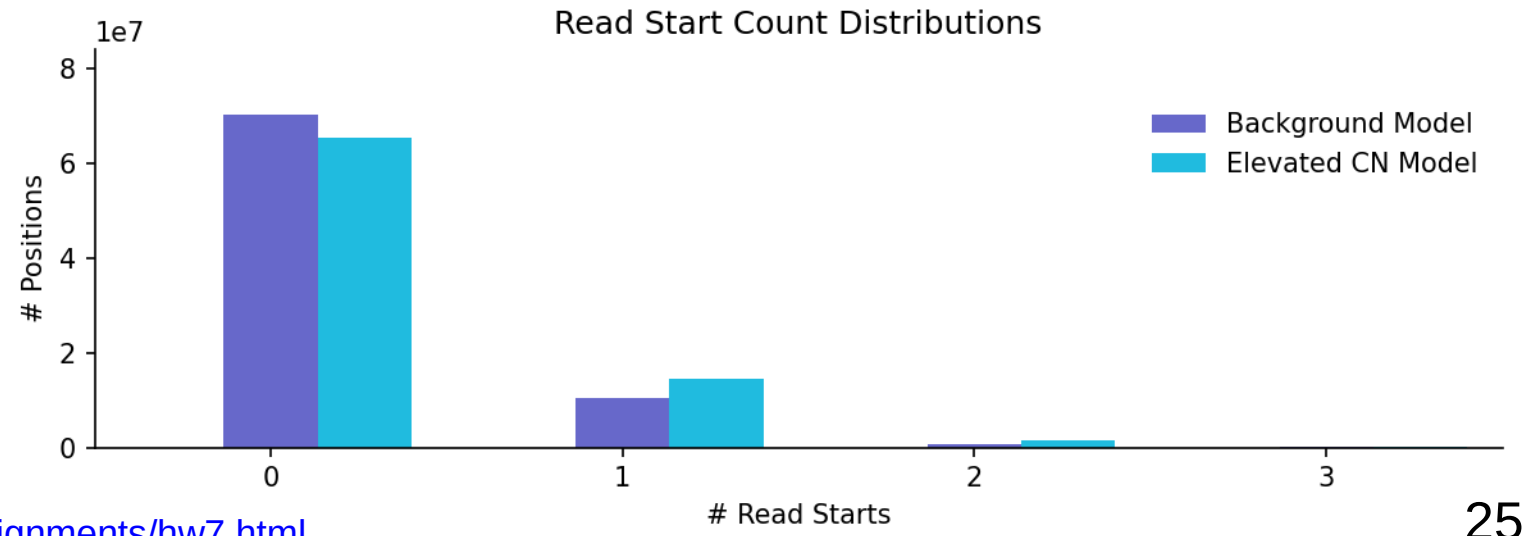
Created LLR Scoring Scheme

```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):\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)

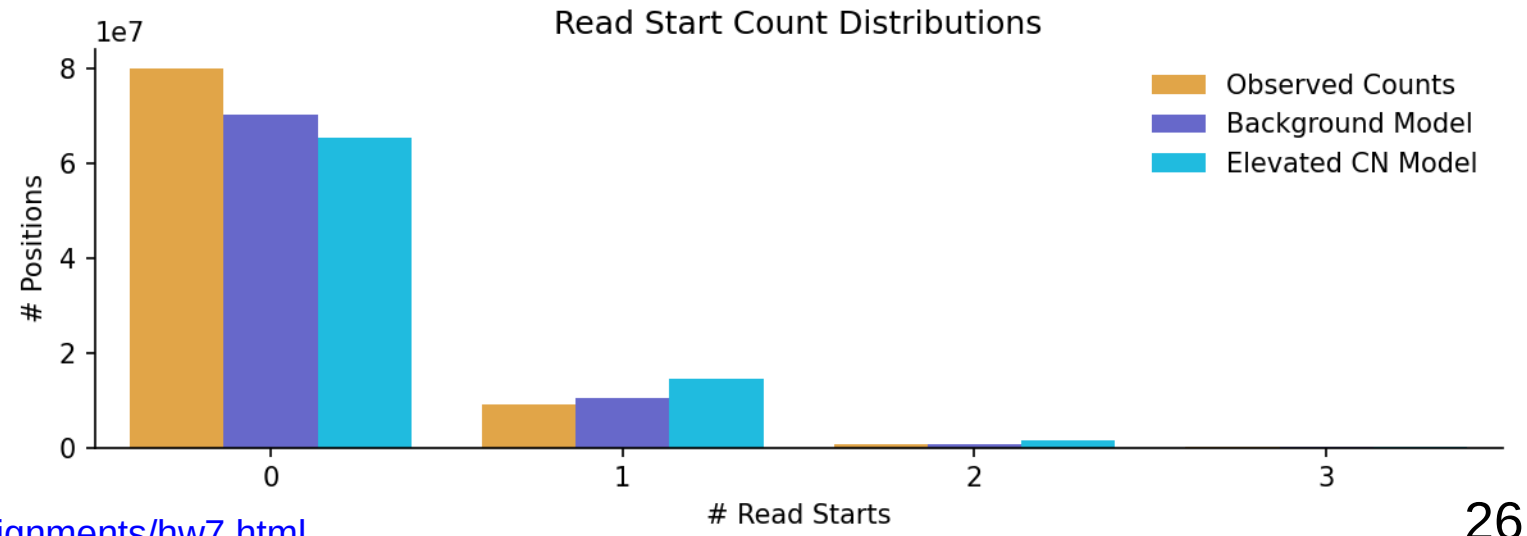
Created LLR Scoring Scheme

```
# 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.

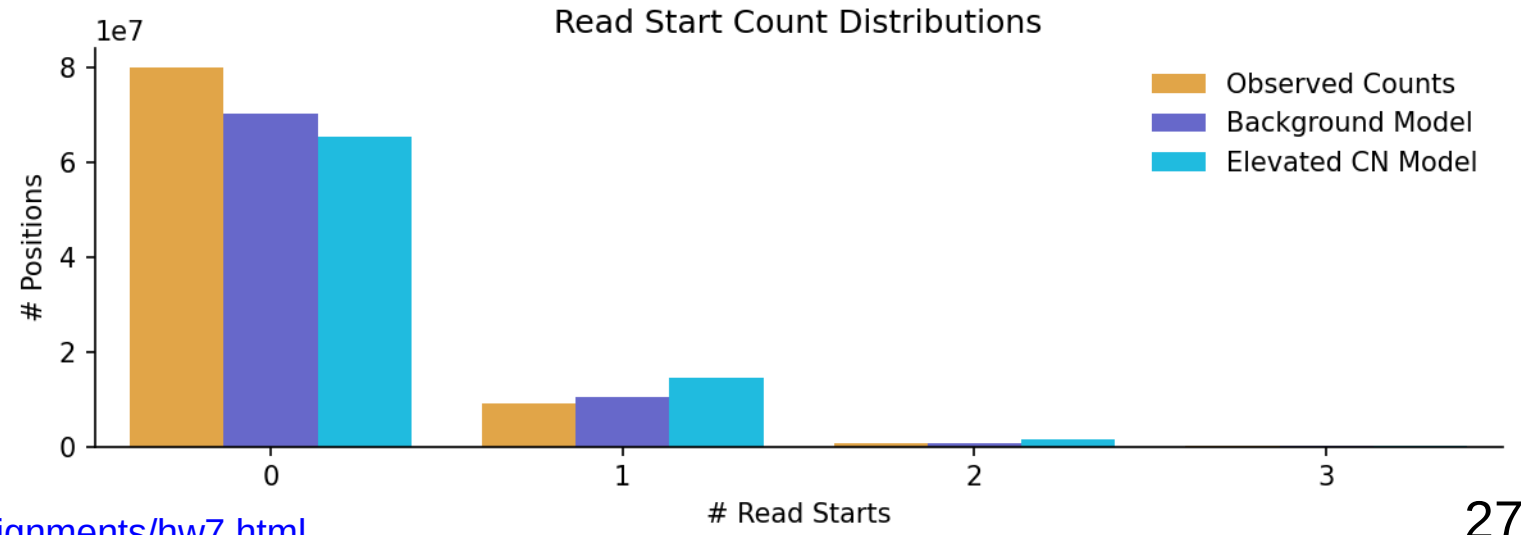
- Use HW6 results to refine our model

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

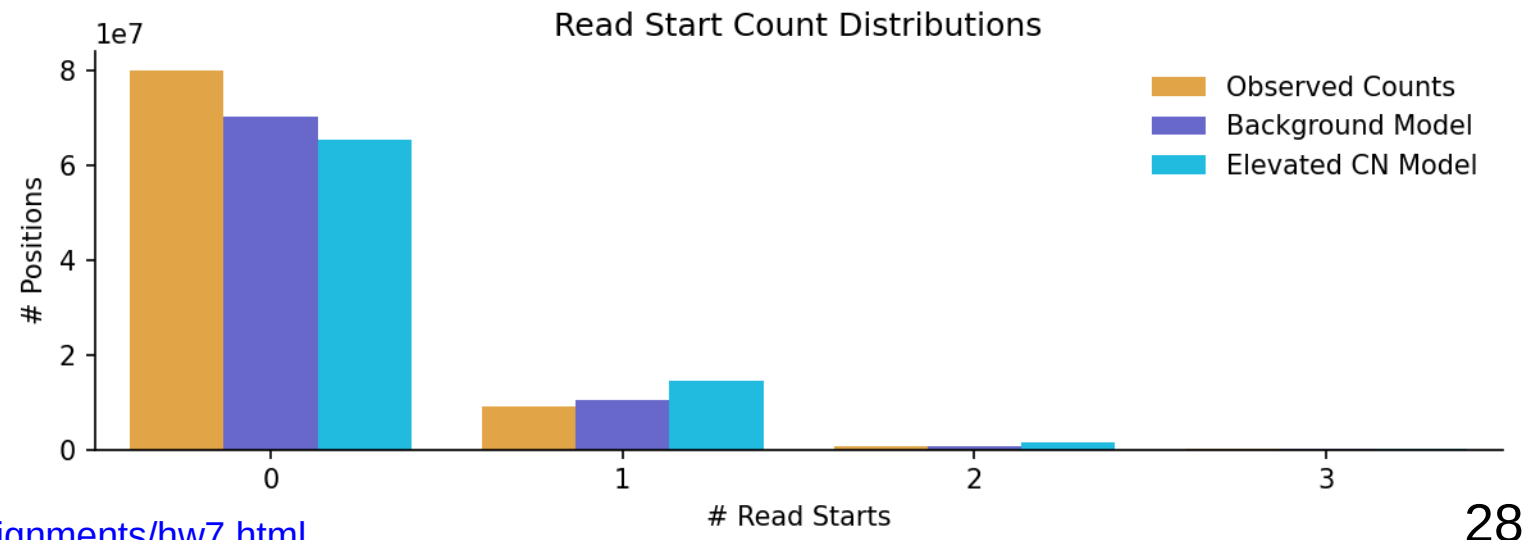
Homework 7 Overview

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

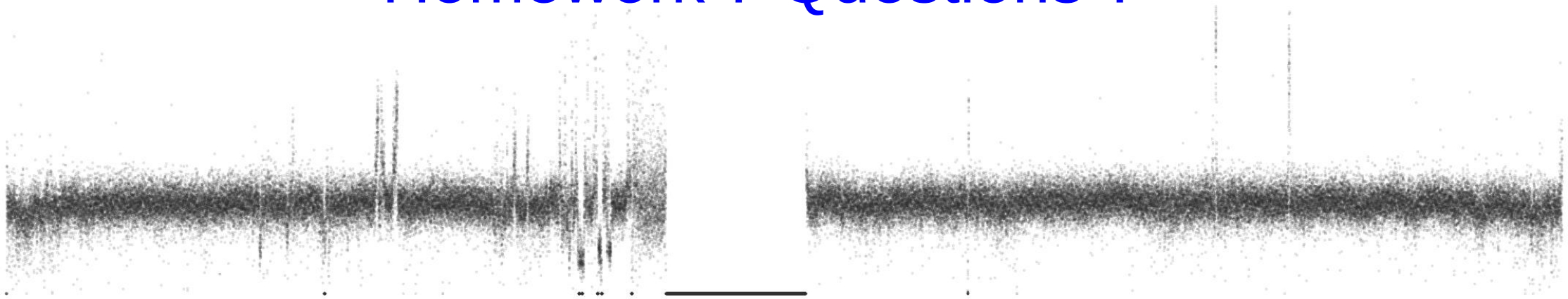
```
Ratios of simulated data:
N_seg(5)/N_seg(6) {# of segments with score >= 5 / # of segments with score >= 6}
N_seg(6)/N_seg(7) {# of segments with score >= 6 / # of segments with score >= 7}
N_seg(7)/N_seg(8) {# of segments with score >= 7 / # of segments with score >= 8}
.
.
.
list all ratios
(only 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?

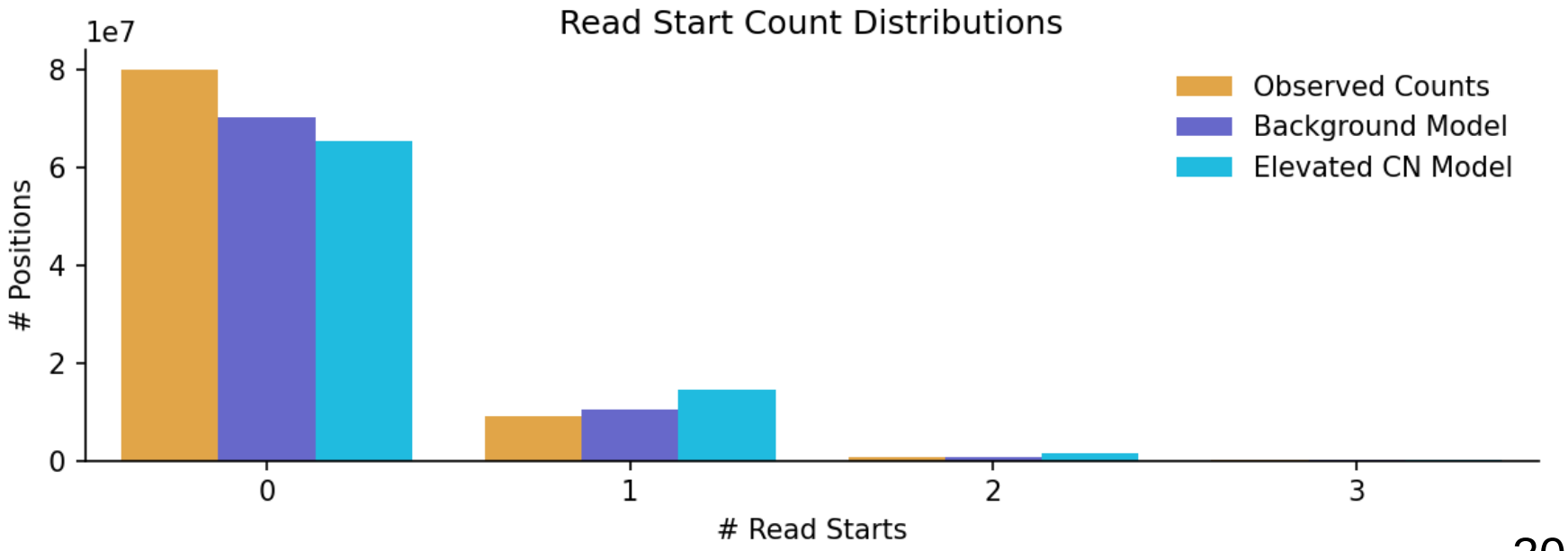
http://bozeman.mbt.washington.edu/compbio/mbt599/assignments/hw7.html

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

- Homework 8 will be posted tomorrow

. . .