

Genome Sequencing

Michael Schatz

August 27 2025

Lecture 2: Applied Comparative Genomics



Course Webpage

A screenshot of a GitHub repository page for "schatzlab / appliedgenomics2025". The repository is public and contains 1 branch and 0 tags. The most recent commit was made by mschatz 6 minutes ago, adding a syllabus. Other commits include "add syllabus" by mschatz, "Initial commit" by mschatz, and "fix text" by mschatz. The README file is present, along with a CC0-1.0 license link.

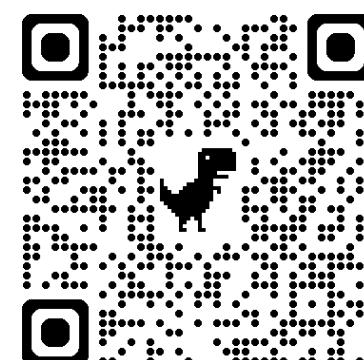
The repository page includes an "About" section with the following details:

- Materials for EN.601.449/649
- Computational Genomics: Applied Comparative Genomics

Links in the "About" section include:

- Readme
- CC0-1.0 license
- Activity
- Custom properties

Star count: 1 star



<https://github.com/schatzlab/appliedgenomics2025>

<https://github.com/schatzlab/appliedgenomics2025>

TA: Mahler Revsine



Office Hours: Mondays at 4:30pm
@ Malone 216 and Zoom

Assignment I

The screenshot shows a GitHub repository interface for 'assignment1'. On the left, there's a sidebar with a 'Files' section containing various genome size files like TAIR10.chrom.sizes, ce10.chrom.sizes, and hg38.chrom.sizes. The main content area has a title 'Assignment 1: Chromosome Structures' and details about the assignment date (Wednesday, August 27, 2025) and due date (Wednesday, Sept. 3, 2025 @ 11:59pm). It includes an 'Assignment Overview' section with a note about Piazza and a 'Question 1: Chromosome structures [10 pts]' section listing eight species with links to their genome size files.

Assignment 1: Chromosome Structures

Assignment Date: Wednesday, August 27, 2025
Due Date: Wednesday, Sept. 3, 2025 @ 11:59pm

Assignment Overview

In this assignment you will profile the overall structure of the genomes of several important species and then study human chromosome 22 in more detail. As a reminder, any questions about the assignment should be posted to [Piazza](#).

Question 1: Chromosome structures [10 pts]

Download the chromosome size files for the following genomes (Note these have been preprocessed to only include main chromosomes):

1. [E. coli \(Escherichia coli K12\)](#) - One of the most commonly studied bacteria [\[info\]](#)
2. [Yeast \(Saccharomyces cerevisiae, sacCer3\)](#) - an important eukaryotic model species, also good for bread and beer [\[info\]](#)
3. [Worm \(Caenorhabditis elegans, ce10\)](#) - One of the most important animal model species [\[info\]](#)
4. [Fruit Fly \(Drosophila melanogaster, dm6\)](#) - One of the most important model species for genetics [\[info\]](#)
5. [Arabidopsis thaliana \(TAIR10\)](#) - An important plant model species [\[info\]](#)
6. [Tomato \(Solanum lycopersicum v4.00\)](#) - One of the most important food crops [\[info\]](#)
7. [Human \(hg38\)](#) - us :) [\[info\]](#)
8. [Wheat \(Triticum aestivum, IWGSC\)](#) - The food crop which takes up the largest land area [\[info\]](#)

Using these files, make a table with the following information per species:

- Question 1.1. Total genome size
- Question 1.2. Number of chromosomes
- Question 1.3. Largest chromosome size and name
- Question 1.4. Smallest chromosome size and name
- Question 1.5. Mean chromosome length

Question 2: Coverage simulator [20 pts]

<https://github.com/schatzlab/appliedgenomics2025/tree/main/assignments/assignment1>
Due end of day on Wednesday Sept 3 (right before midnight)

Plotting in Python

The screenshot shows the Matplotlib gallery index page (<https://matplotlib.org/stable/gallery/index.html>). The page features a navigation bar with links to Plot types, Examples, Tutorials, Reference, User guide, Develop, and Releases. A search bar and a "stable" dropdown are also present. The main content area is titled "Examples" and contains a brief description: "This page contains example plots. Click on any image to see the full image and source code. For longer tutorials, see our tutorials page. You can also find external resources and a FAQ in our user guide." Below this, a section titled "Lines, bars and markers" displays a grid of 15 small plots illustrating various plotting techniques. To the left is a "Section Navigation" sidebar with a hierarchical list of Matplotlib topics. To the right is a "On this page" sidebar listing additional categories.

Section Navigation

- Lines, bars and markers
- Images, contours and fields
- Subplots, axes and figures
- Statistics
- Pie and polar charts
- Text, labels and annotations
- The pyplot module
- Color
- Shapes and collections
- Style sheets
- The axes_grid1 module
- The axisartist module
- Showcase
- Animation
- Event handling
- Miscellaneous
- 3D plotting
- Scales
- Specialty plots
- Spines
- Ticks
- Units
- Embedding Matplotlib in graphical user interfaces
- Widgets
- Userdemo

Examples

This page contains example plots. Click on any image to see the full image and source code. For longer tutorials, see our tutorials page. You can also find external resources and a FAQ in our user guide.

Lines, bars and markers

- Bar color demo
- Bar Label Demo
- Stacked bar chart
- Grouped bar chart with labels
- Horizontal bar chart
- Broken Barh
- CapStyle
- Plotting categorical variables
- Plotting the coherence of two signals
- CSD Demo
- Curve with error band
- Errorbar limit selection
- Errorbar subsampling
- EventCollection Demo
- Eventplot demo

On this page

- Lines, bars and markers
- Images, contours and fields
- Subplots, axes and figures
- Statistics
- Pie and polar charts
- Text, labels and annotations
- The pyplot module
- Color
- Shapes and collections
- Style sheets
- The axes_grid1 module
- The axisartist module
- Showcase
- Animation
- Event handling
- Miscellaneous
- 3D plotting
- Scales
- Specialty plots
- Spines
- Ticks
- Units
- Embedding Matplotlib in graphical user interfaces
- Widgets
- Userdemo

<https://matplotlib.org/>

Plotting in R / ggplot2

Data visualization with ggplot2 :: CHEATSHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(<mapping = aes(<MAPPINGS>),  
  stat = <STAT>, position = <POSITION>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>
```

ggplot(data = mpg, aes(x = cyl, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Aes

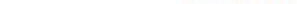
Common aesthetic values.

color and fill - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotdash", 5 = "longdash", 6 = "twodash")

size - integer (line width in mm)

shape - integer/shape name or a single character ("a")



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables.
Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))  
b <- ggplot(seals, aes(x = long, y = lat))  
  
a + geom_blank() and a + expand_limits()  
Ensure limits include values across all plots.  
  
b + geom_curve(aes(yend = lat + 1,  
xend = long + 1), curvature = 1) - x, y, label, alpha, angle, color,  
family, fontface, hjust, lineheight, size, vjust  
  
a + geom_path(linewidth = 2)  
linejoin = "round", linemetre = 1)  
x, y, alpha, color, group, linetype, size  
  
a + geom_polygon(aes(alpha = 50)) - x, y, alpha,  
color, fill, group, subgroup, linetype, size  
  
b + geom_rect(aes(xmin = long, ymin = lat,  
xmax = long + 1, ymax = lat + 1)) - x, y, alpha,  
color, fill, group, linetype, size  
  
a + geom_ribbon(aes(ymin = unemploy - 900,  
ymax = unemploy + 900)) - x, y, alpha, fill,  
group, linetype, size, weight
```

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

```
b + geom_abline(aes(intercept = 0, slope = 1))  
b + geom_hline(aes(yintercept = lat))  
b + geom_vline(aes(xintercept = long))  
  
b + geom_segment(aes(yend = lat + 1, xend = long + 1))  
b + geom_spoke(aes(angle = 1:1155, radius = 1))
```

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)  
c + geom_area(stat = "bin")  
x, y, alpha, color, fill, linetype, size  
  
c + geom_density(kernel = "gaussian")  
x, y, alpha, color, fill, group, linetype, size, weight  
  
c + geom_dotplot()  
x, y, alpha, color, fill  
  
c + geom_freqpoly()  
x, y, alpha, color, group, linetype, size  
  
c + geom_histogram(binwidth = 5)  
x, y, alpha, color, fill, linetype, size, weight  
  
c2 + geom_qq(aes(sample = hwy))  
x, y, alpha, color, fill, linetype, size, weight
```

discrete

```
d <- ggplot(mpg, aes(f))  
d + geom_bar()  
x, alpha, color, fill, linetype, size, weight
```

TWO VARIABLES both continuous

```
e <- ggplot(mpg, aes(cty, hwy))  
  
e + geom_label(aes(label = cyl), nudge_x = 1,  
nudge_y = 1) - x, y, label, alpha, angle, color,  
family, fontface, hjust, lineheight, size, vjust  
  
e + geom_point()  
x, y, alpha, color, fill, shape, size, stroke  
  
e + geom_quantile()  
x, y, alpha, color, group, linetype, size, weight  
  
e + geom_rug(sides = "bl")  
x, y, alpha, color, linetype, size  
  
e + geom_smooth(method = lm)  
x, y, alpha, color, fill, group, linetype, size, weight  
  
e + geom_text(aes(label = cyl), nudge_x = 1,  
nudge_y = 1) - x, y, label, alpha, angle, color,  
family, fontface, hjust, lineheight, size, vjust
```

one discrete, one continuous

```
f <- ggplot(mpg, aes(class, hwy))  
  
f + geom_col()  
x, y, alpha, color, fill, group, linetype, size  
  
f + geom_boxplot()  
x, y, lower, middle, upper, yname, ymin, alpha,  
color, fill, group, linetype, shape, size, weight  
  
f + geom_dotplot(binaxis = "y", stackdir = "center")  
x, y, alpha, color, fill, group  
  
f + geom_violin(scale = "area")  
x, y, alpha, color, fill, group, linetype, size, weight
```

both discrete

```
g <- ggplot(diamonds, aes(cut, color))  
g + geom_count()  
x, y, alpha, color, fill, shape, size, stroke  
  
e + geom_jitter(height = 2, width = 2)  
x, y, alpha, color, fill, shape, size
```

THREE VARIABLES

```
sealsSz <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))  
  
l + geom_contour(aes(z = z))  
x, y, z, alpha, color, group, linetype, size, weight  
  
l + geom_contour_filled(aes(fill = z))  
x, y, alpha, color, fill, group, linetype, size, subgroup  
  
l + geom_raster(aes(fill = z), hijst = 0.5,  
vjust = 0.5, interpolate = FALSE)  
x, y, alpha, fill  
  
l + geom_tile(aes(fill = z))  
x, y, alpha, color, fill, linetype, size, width
```



continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))  
  
h + geom_bin2d(binwidth = c(0.25, 500))  
x, y, alpha, color, fill, linetype, size, weight  
  
h + geom_density_2d()  
x, y, alpha, color, group, linetype, size  
  
h + geom_hex()  
x, y, alpha, color, fill, size
```

continuous function

```
i <- ggplot(economics, aes(date, unemploy))  
  
i + geom_area()  
x, y, alpha, color, fill, linetype, size  
  
i + geom_line()  
x, y, alpha, color, group, linetype, size  
  
i + geom_step(direction = "hv")  
x, y, alpha, color, group, linetype, size
```

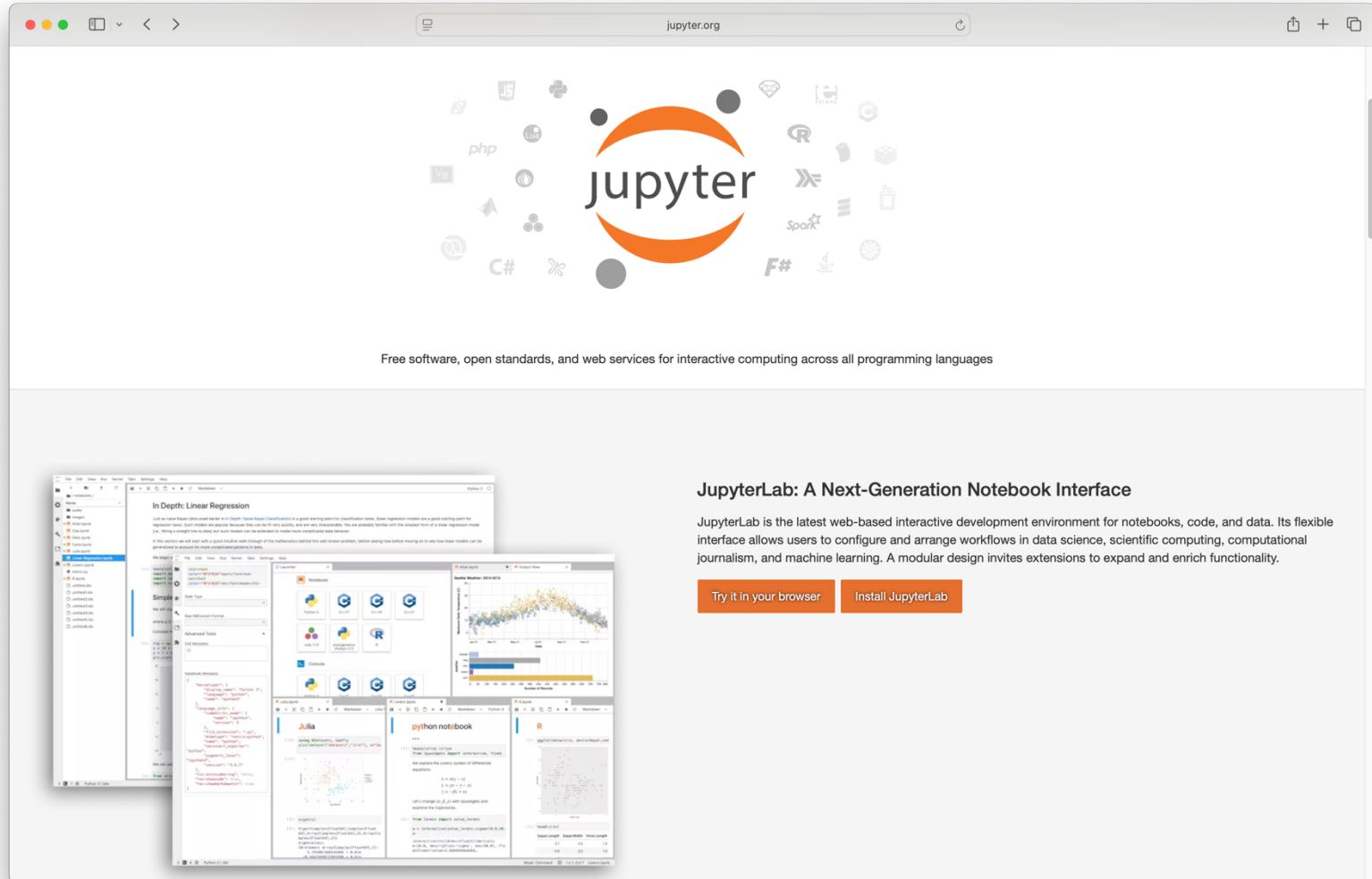
visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)  
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))  
  
j + geom_crossbar(fatten = 2) - x, y, ymax,  
ymin, alpha, color, fill, group, linetype, size  
  
j + geom_errorbar() - x, y, max, ymin,  
alpha, color, group, linetype, size, width  
Also geom_errorbarh().  
  
j + geom_linerange()  
x, ymin, ymax, alpha, color, group, linetype, size  
  
j + geom_pointrange() - x, y, ymin, ymax,  
alpha, color, fill, group, linetype, shape, size
```

maps

```
maps <- data.frame(murder = USAArrests$Murder,  
state = tolower(rownames(USA$Arrests)))  
map <- map_data("state")  
k <- ggplot(data, aes(fill = murder))  
  
k + geom_map(aes(map_id = state), map = map)  
+ expand_limits(x = map$long, y = map$lat)  
map_id, alpha, color, fill, linetype, size
```

Jupyter Notebooks



JupyterLab: A Next-Generation Notebook Interface

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

[Try it in your browser](#)

[Install JupyterLab](#)

<https://jupyter.org>

VS Code

The screenshot shows the Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar with icons for files, folders, and other workspace items. The main editor area displays a Python script named `kmer_counter.py`. The script is a simple k-mer counter that reads a FASTA file, counts all k-mers, and outputs the top N most frequent kmers. It includes command-line usage instructions and imports for `argparse`, `sys`, and `Counter` from `collections`. The code uses a generator function `parse_fasta` to yield sequences from a FASTA file as uppercase strings.

To the right of the editor is the Chat pane, which contains an AI-generated todo list:

- ✓ Todos (0/3)
- Create files
- Run test
- Update todo statuses

A callout box highlights the first item: "write a python program that reads in a fasta file of a genome, counts all of the kmers, and outputs the top X most frequent kmers. allow the input file name, kmer length, and number of kmers to report to be set of the command line."

Below the Chat pane, the AI provides progress updates:

- I'll create a todo list, then add a small, well-documented Python script `kmer_counter.py`, a README, and a sample FASTA, then run the script on the sample to verify output. I'll mark progress as I go.
- ✓ Created 3 todos
- I'll create the script, README, and a small test FASTA file now. Expect three file creations and a checkpoint after they're added.
- ✓ Created `kmer_counter.py`

The bottom right corner shows the Git Changes panel, indicating 3 files changed: `kmer_counter.py`, `README.md`, and `test.fasta`. A context menu is open over the `kmer_counter.py` file, with options like "Add Context..." and "kmer_counter.py +".

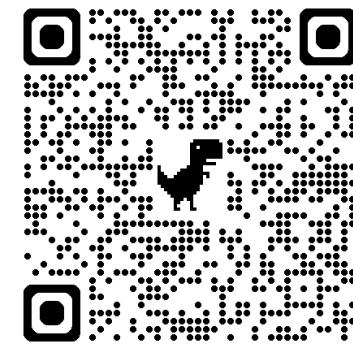
"write a python program that reads in a fasta file of a genome, counts all of the kmers, and outputs the top X most frequent kmers. allow the input file name, kmer length, and number of kmers to report to be set of the command line."

Github Copilot Pro

Screenshot of the GitHub Docs page for "Getting free access to GitHub Copilot Pro as a student, teacher, or maintainer".

The page shows the following content:

- Section Header:** Getting free access to GitHub Copilot Pro as a student, teacher, or maintainer
- Text:** Learn how to use Copilot Pro for free as a student, teacher, or open-source maintainer.
- Section Header:** About free GitHub Copilot Pro access
- Text:** There are three ways to qualify for free access to Copilot Pro:
 - As a verified student on GitHub Education. To learn about becoming a verified student, see [Apply to GitHub Education as a student](#).
 - As a verified teacher on GitHub Education. To learn about becoming a verified teacher, see [Apply to GitHub Education as a teacher](#).
 - As a maintainer of a popular open-source repository. To determine if you are an eligible maintainer, see [Accessing Copilot Pro for free](#).
- Text:** GitHub reevaluates your eligibility every month.
- Section Header:** What if I don't qualify for free access to Copilot Pro?
- Text:** If you do not meet the previous criteria, you can either:
 - Try [Copilot Pro for free](#) with a one-time 30-day trial. After the free trial, you will need a paid plan for continued use.
 - Set up Copilot Free to get a limited experience of Copilot without a paid plan. See [About individual GitHub Copilot plans and benefits](#).
- Section Header:** Accessing Copilot Pro for free
- List:**
 - In the upper-right corner of any page, click your profile picture, then click [Your Copilot](#).
 - If you qualify for free access to Copilot, you will see a page titled "GitHub Copilot Pro".



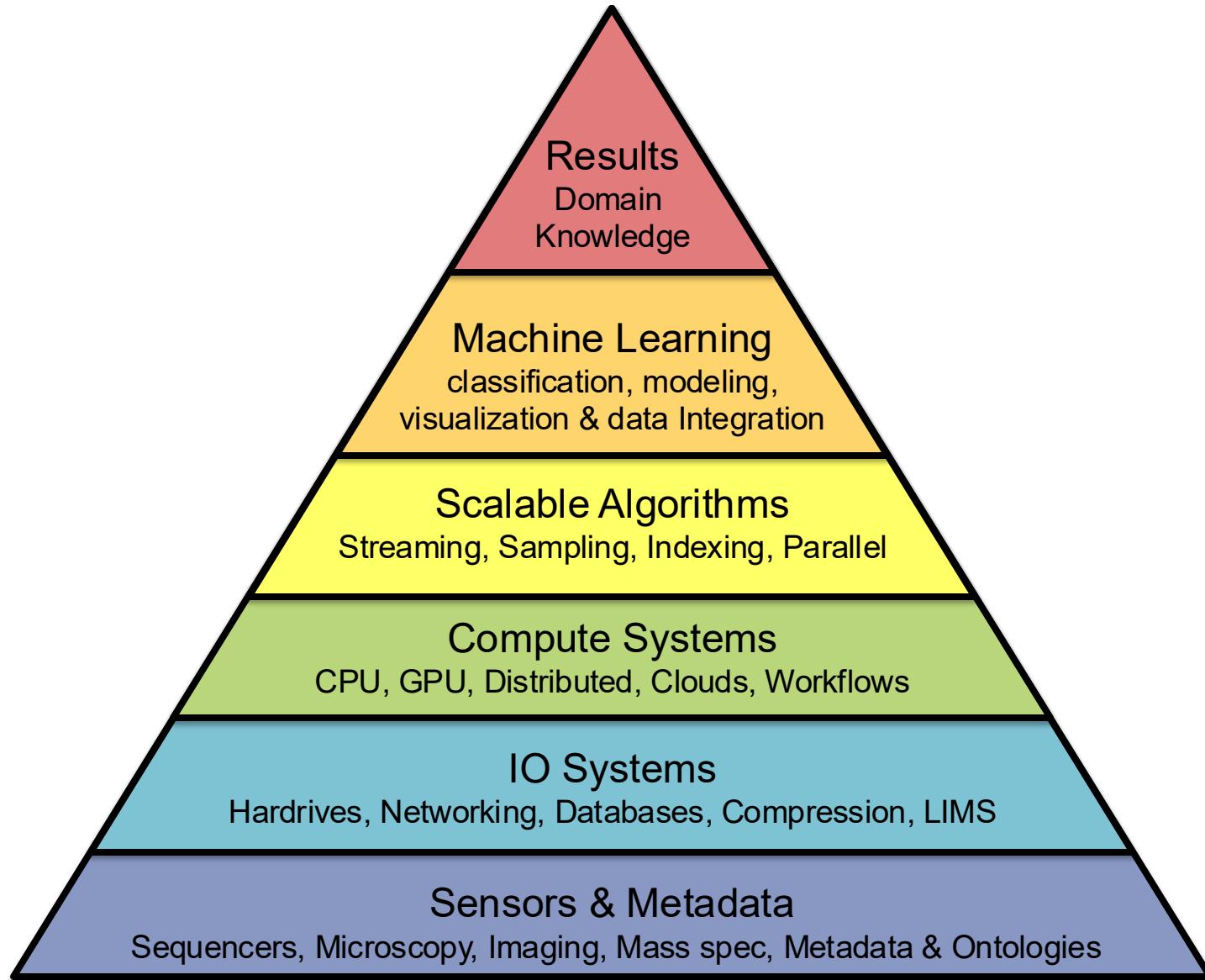
<https://docs.github.com/en/copilot/how-tos/manage-your-account/get-free-access-to-copilot-pro>

Unsolved Questions in Biology

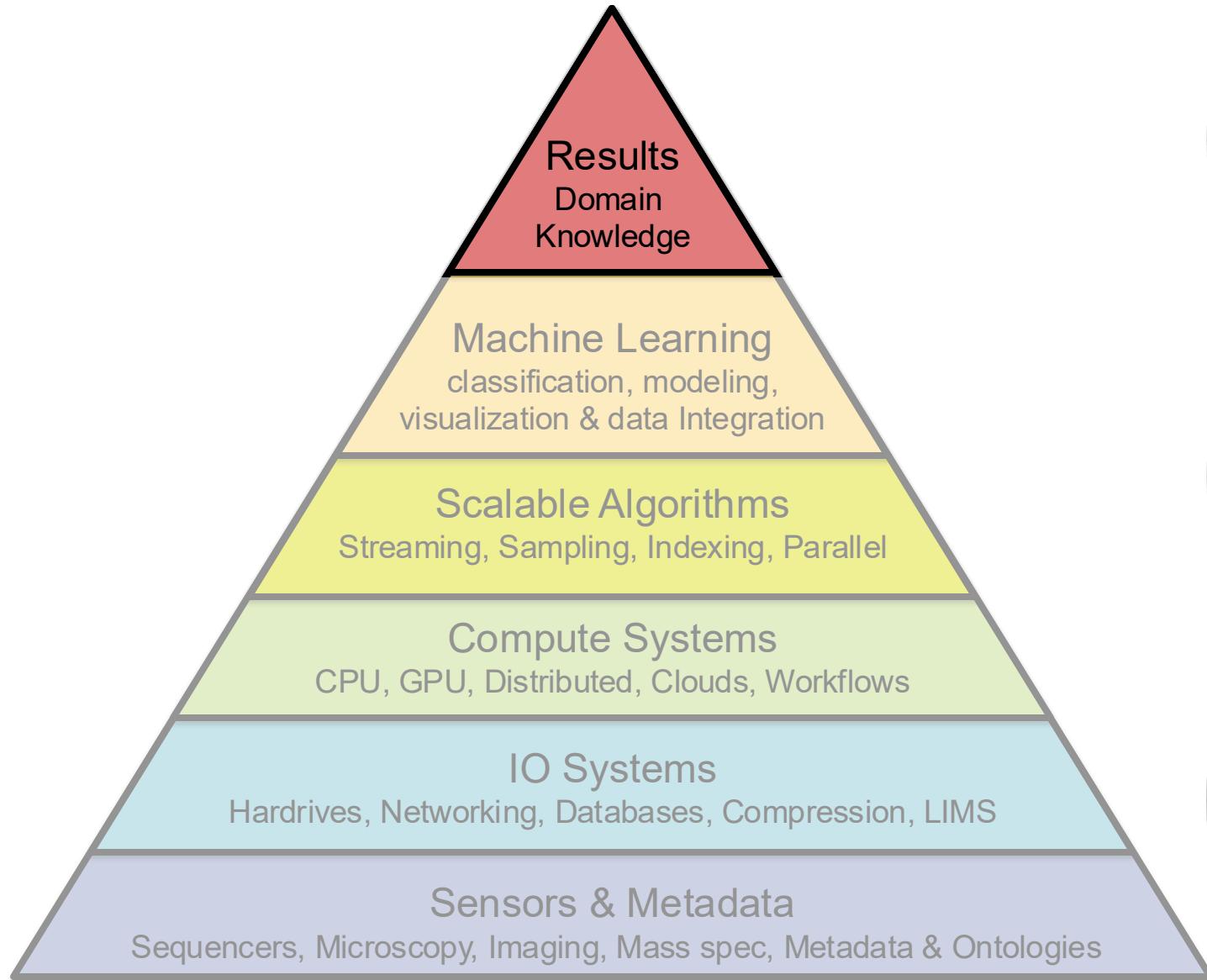
- What is your genome sequence?
- How does your genome compare to my genome?
- Where are the genes and how active are they?
- How does gene activity change during development?
- How does splicing change during development?
- How does methylation change during development?
- How does chromatin change during development?
- How does your genome folded in the cell?
- Where do proteins bind and regulate genes?
- What virus and microbes are living inside you?
- How do your mutations relate to disease?
- What drugs and treatments should we give you?
- ***Plus thousands and thousands more***



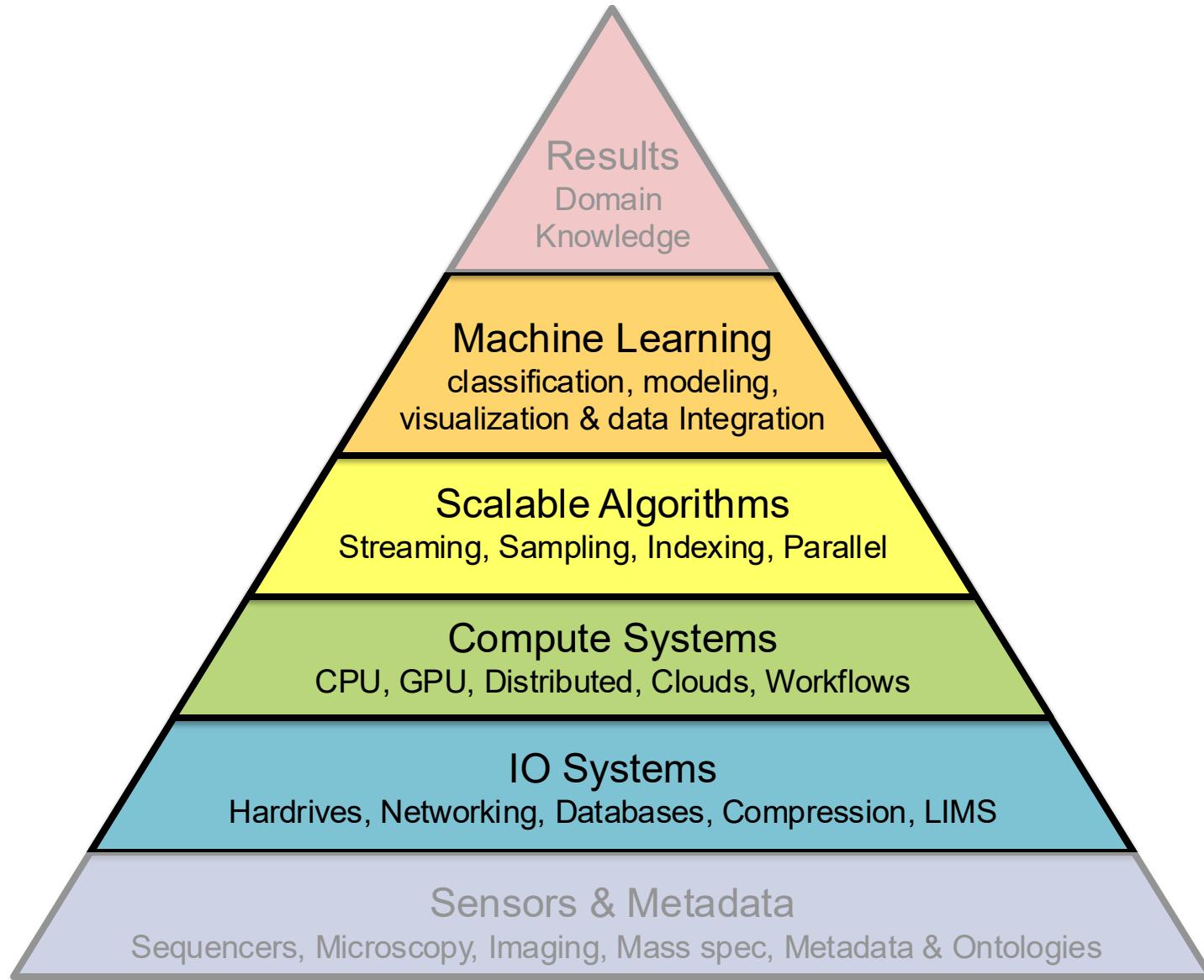
Comparative Genomics Technologies



Comparative Genomics Technologies



Comparative Genomics Technologies

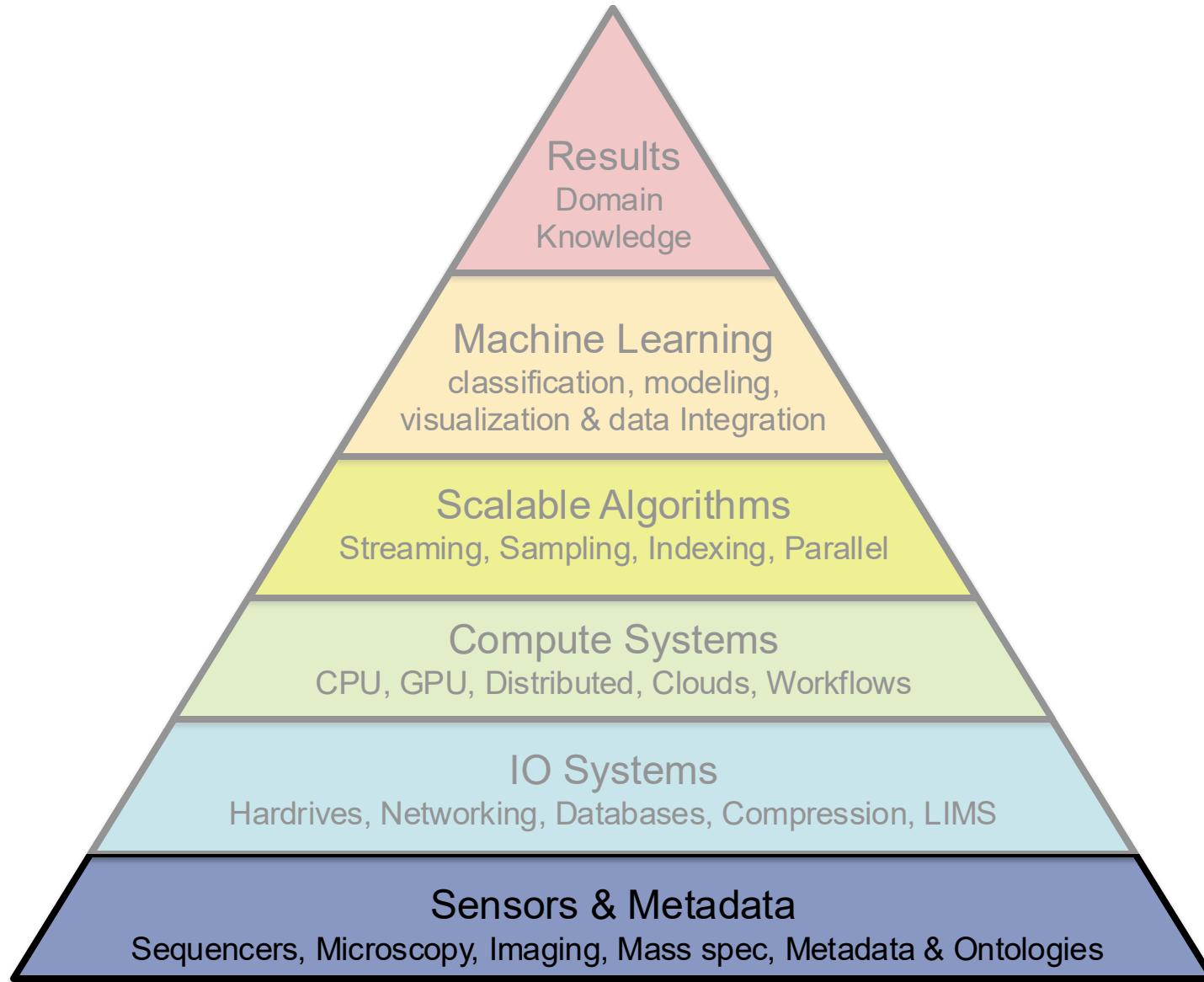


Selected Topics

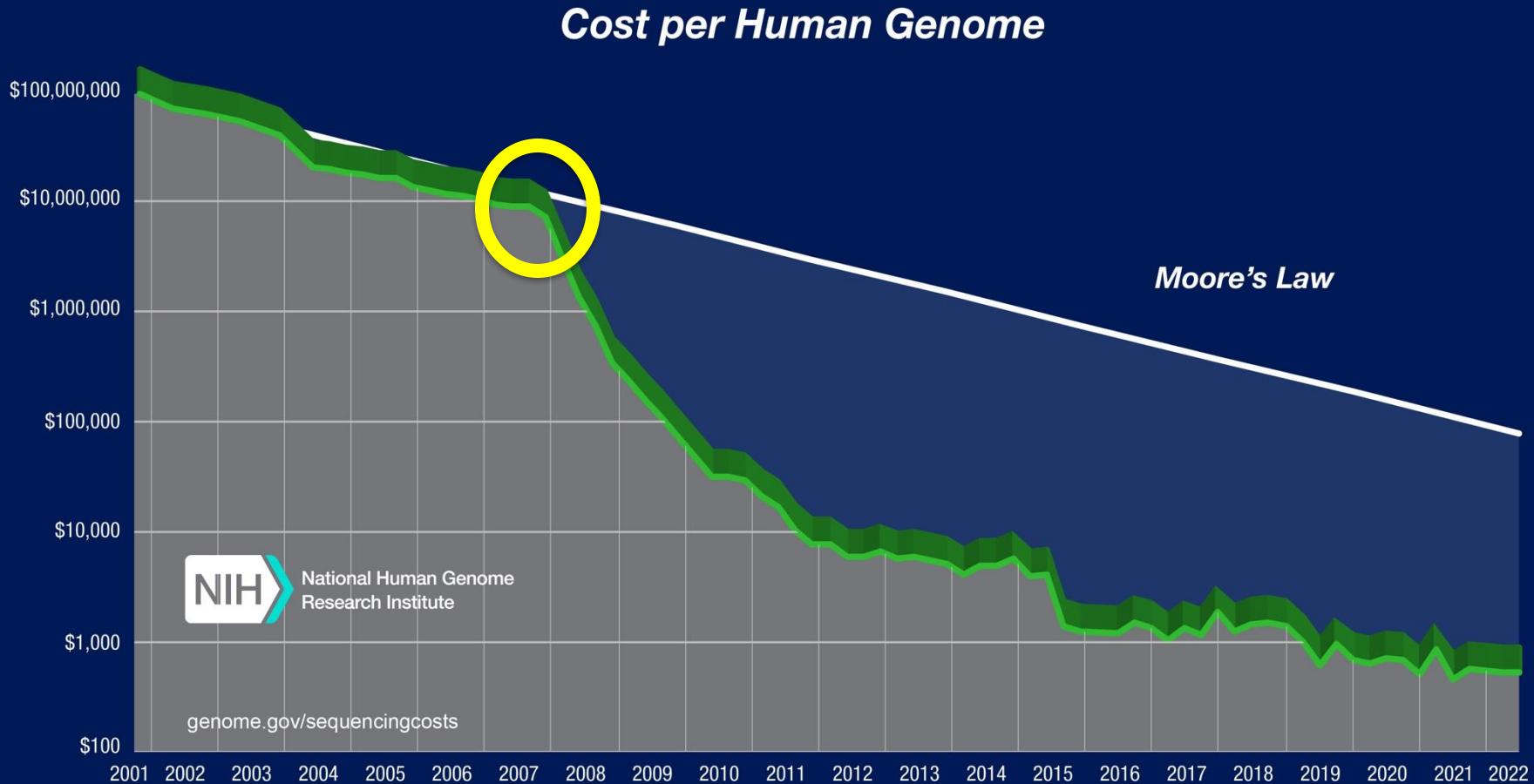
- Genome assembly, whole genome alignment
- Full text indexing: Suffix Trees, Suffix Arrays, FM-index
- Dynamic Programming: Edit Distance, sequence similarity
- Read mapping & Variant identification
- Gene Finding: HMMs, Plane-sweep algorithms
- RNA-seq: mapping, assembly, quantification
- ChIP-seq: Peak finding, motif finding
- Methylation-seq: Mapping, CpG island detection
- HiC: Domain identification, scaffolding
- Chromatin state analysis: ChromHMM, Enformer, etc
- Scalable genomics: Cloud computing, scalable data structures
- Population & single cell analysis: clustering, pseudotime
- Disease analysis, cancer genomics, Metagenomics
- Deep learning in genomics



Comparative Genomics Technologies



Cost per Genome

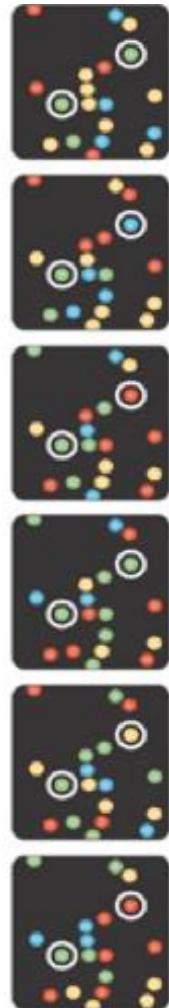
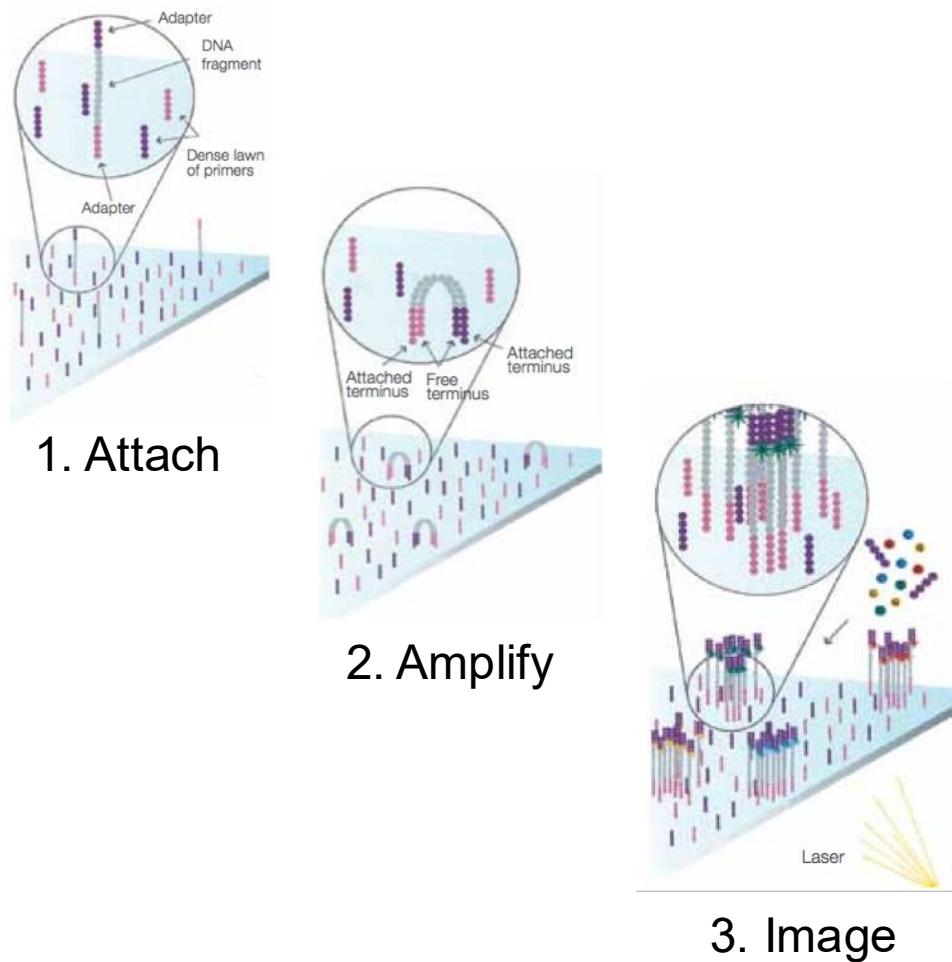


Second Generation Sequencing

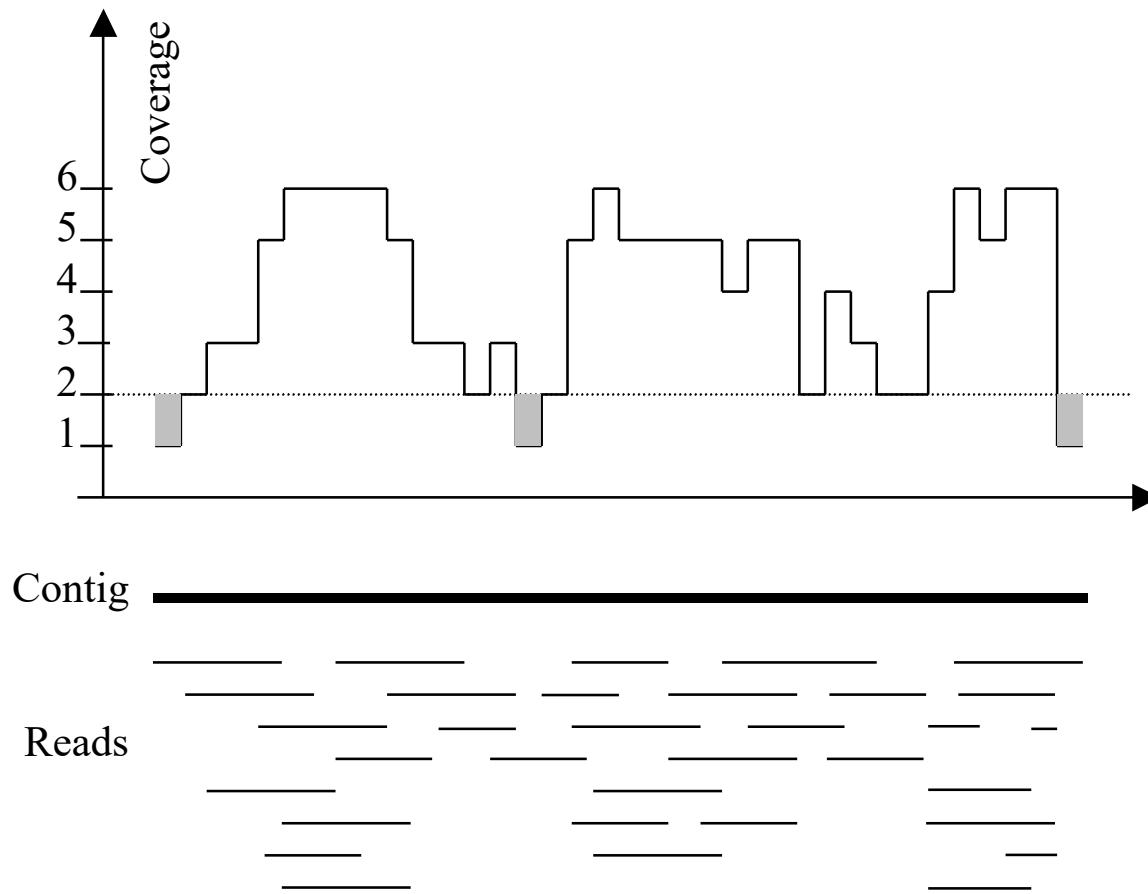


Illumina NovaSeq 6000
Sequencing by Synthesis

>3Tbp / day
(JHU has 4 of these!)



Typical sequencing coverage

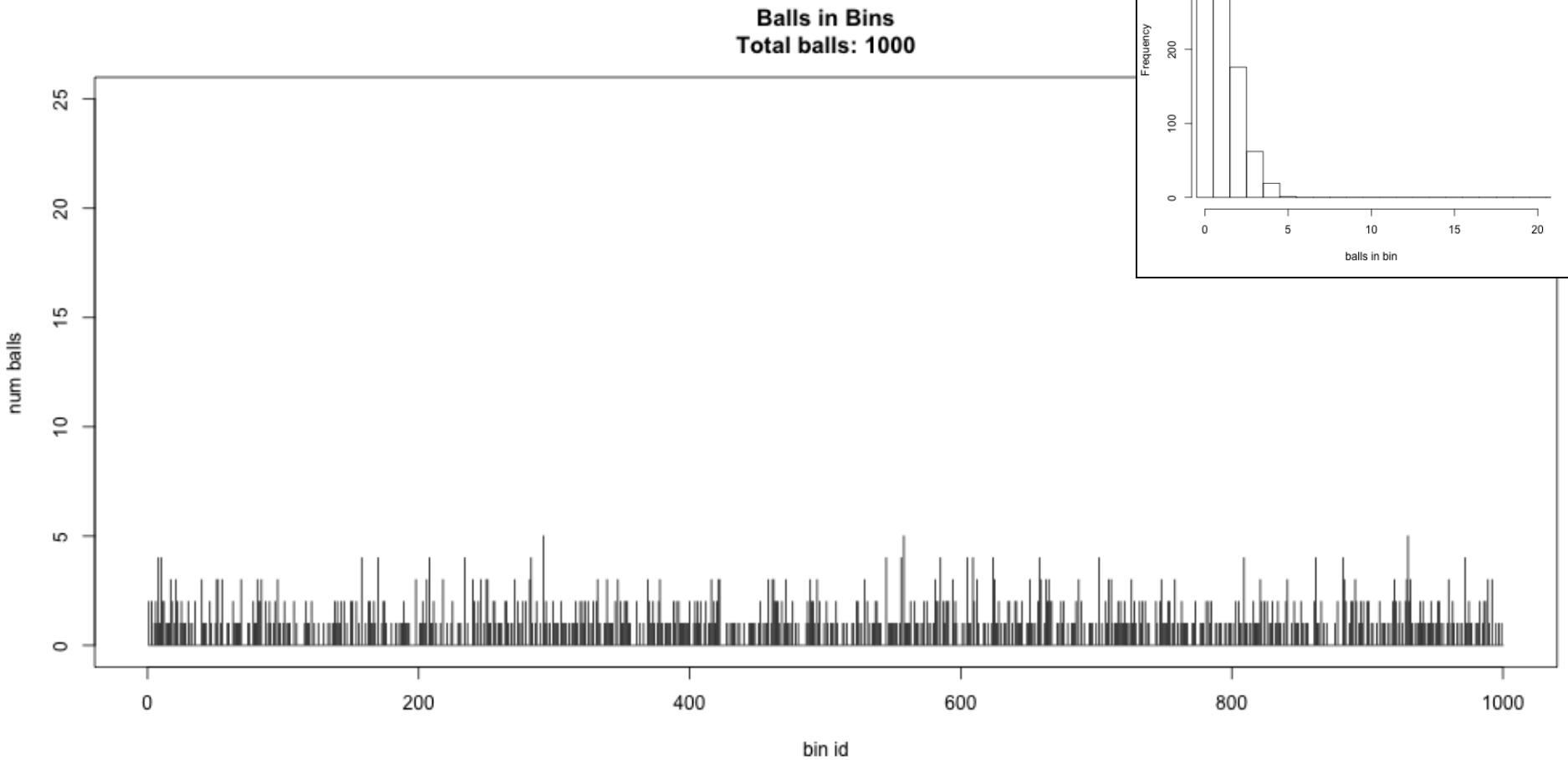


Imagine raindrops on a sidewalk

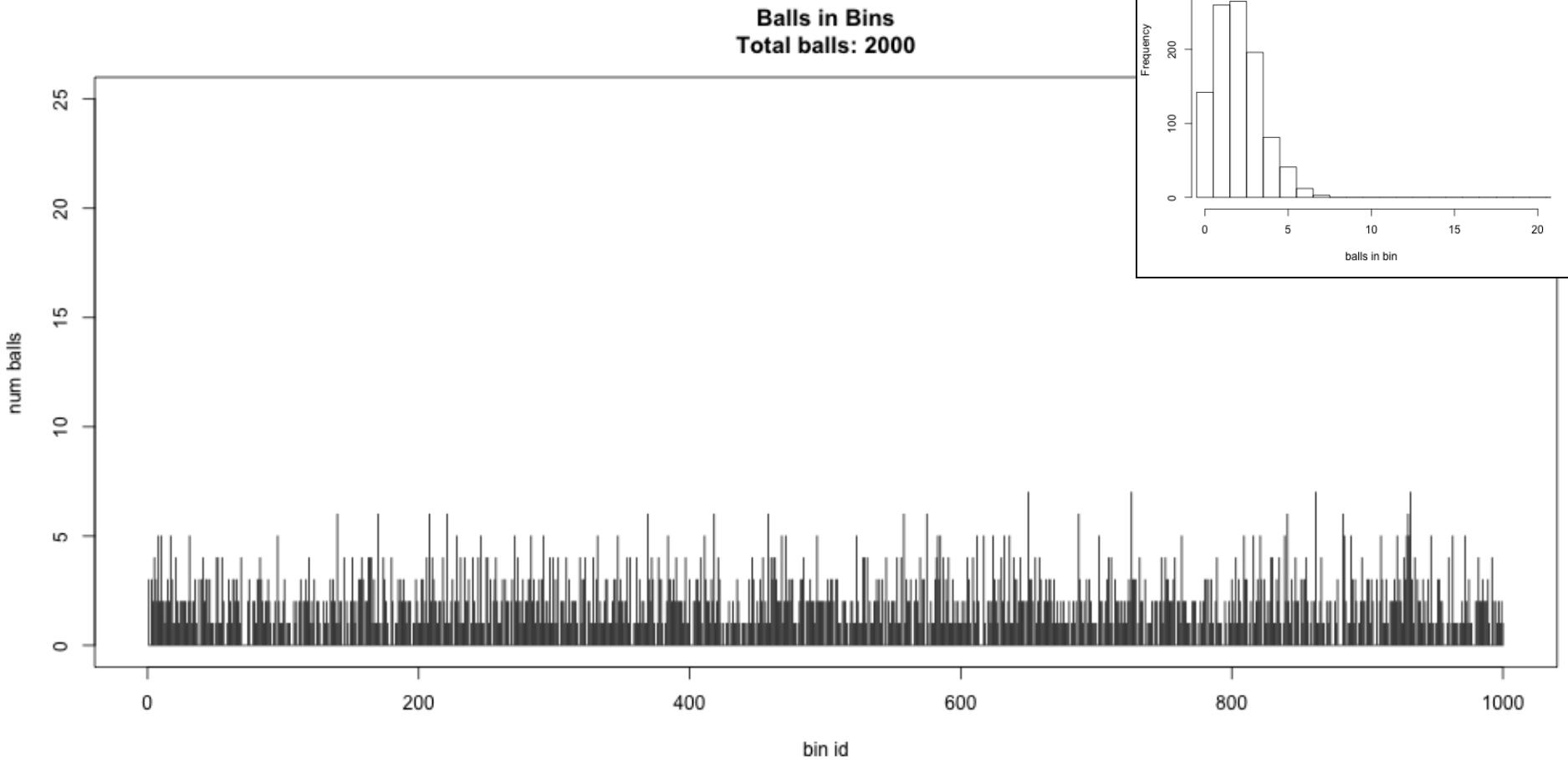
We want to cover the entire sidewalk but each drop costs \$1

If the genome is 10 Mbp, should we sequence 100k 100bp reads?

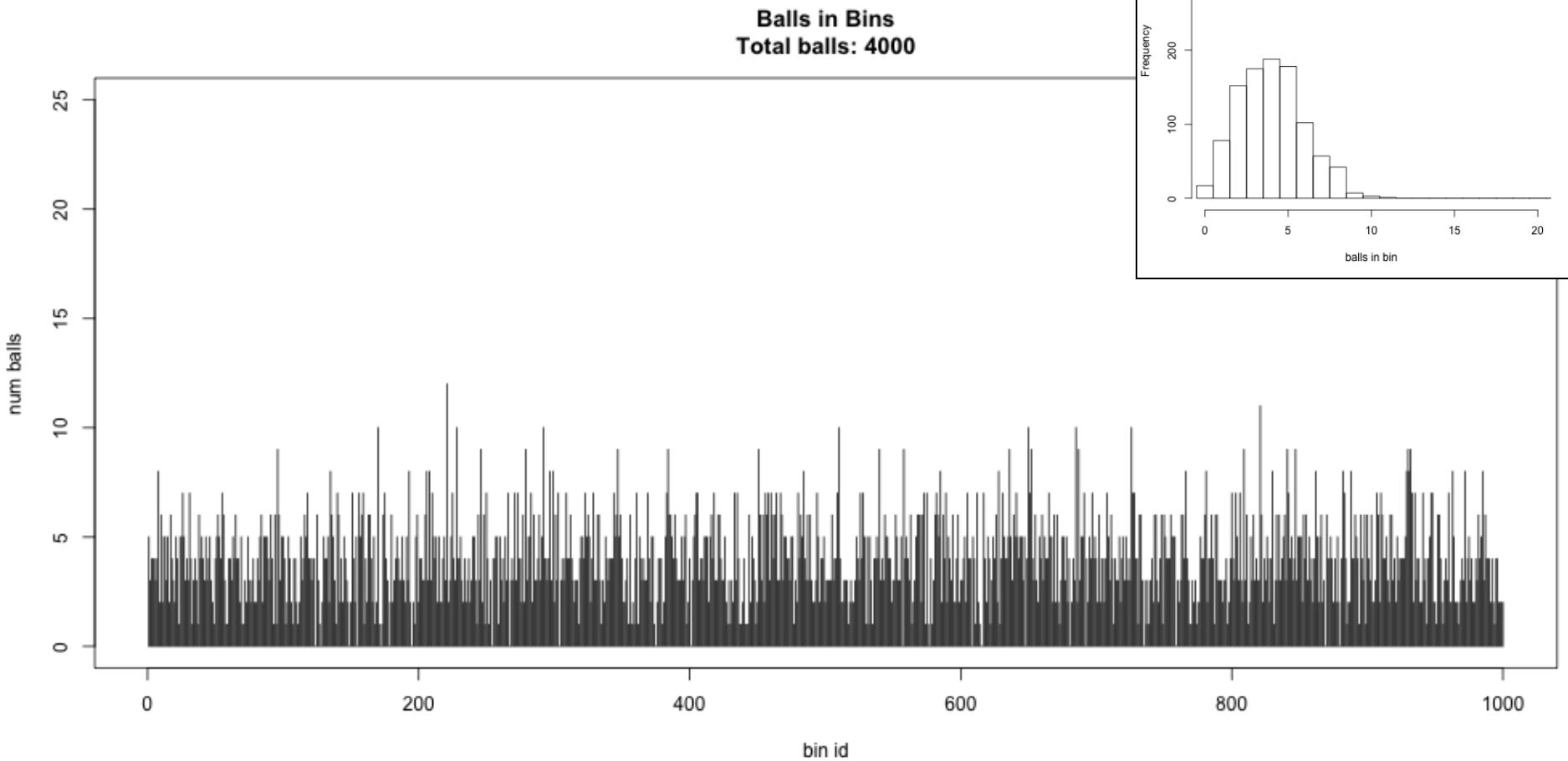
Ix sequencing



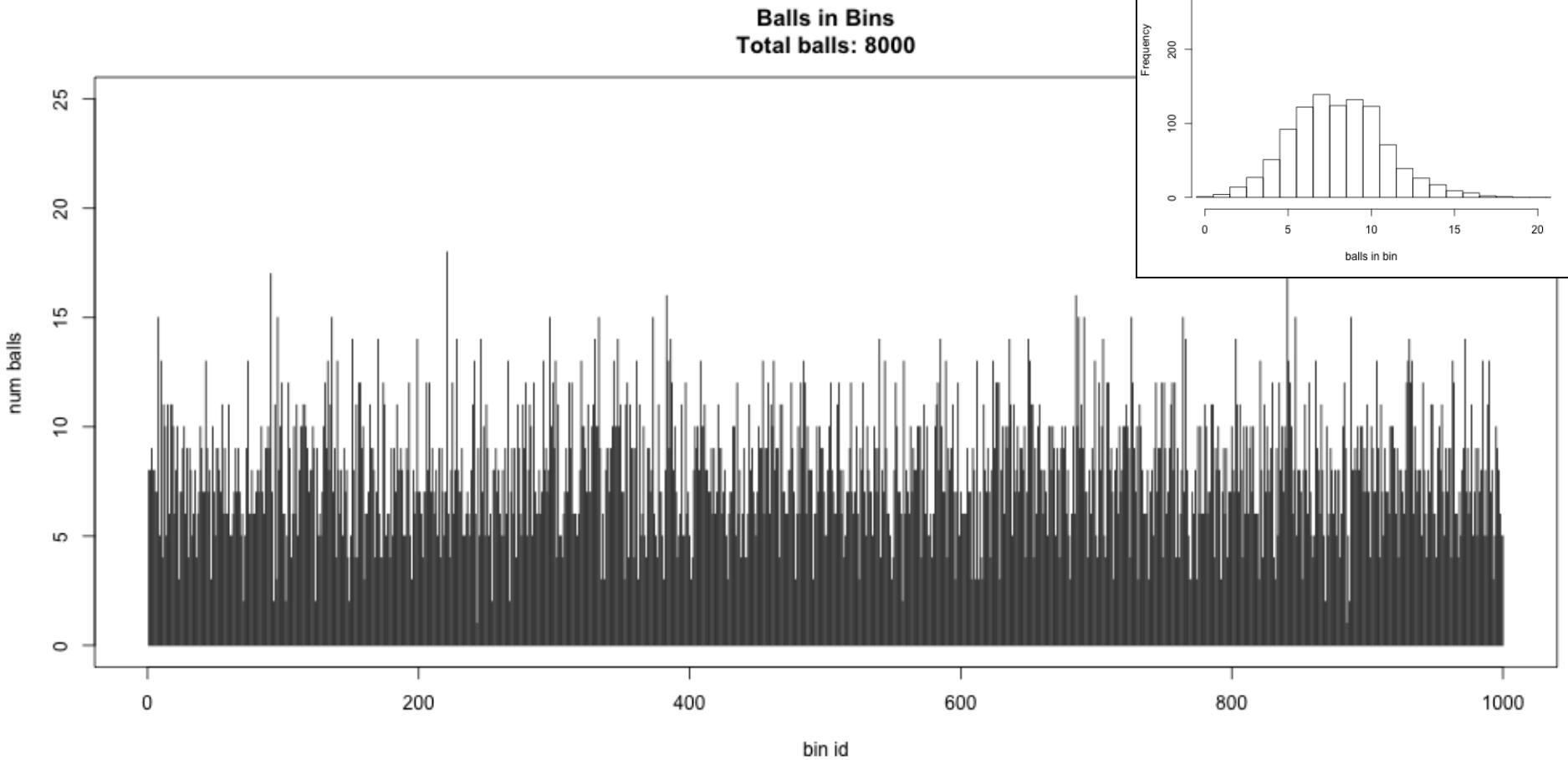
2x sequencing



4x sequencing



8x sequencing



Poisson Distribution

The probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event.

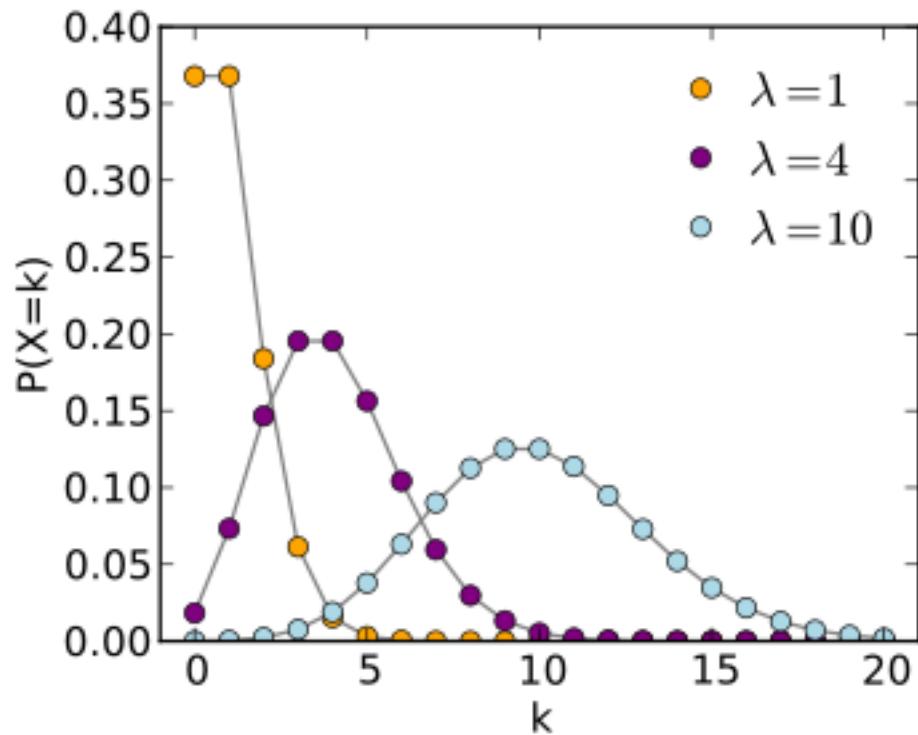
Formulation comes from the limit of the binomial equation

Resembles a normal distribution, but over the positive values, and with only a single parameter.

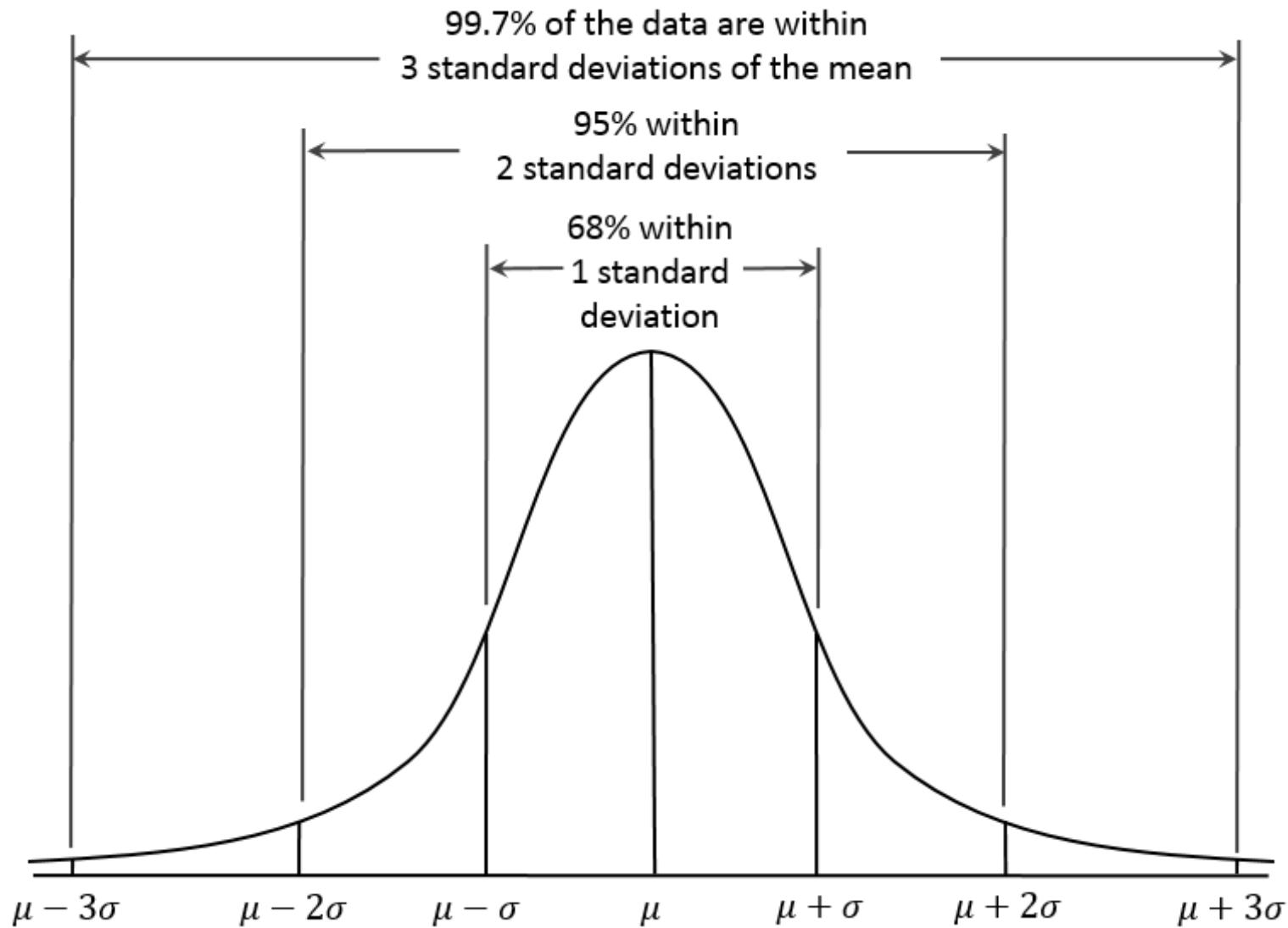
Key properties:

- ***The standard deviation is the square root of the mean.***
- ***For mean > 5, well approximated by a normal distribution***

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$



Normal Approximation



Can estimate Poisson distribution as a normal distribution when $\lambda > 10$