Big data

Rayan Chikhi
Institut Pasteur
Workshop on Genomics 2024
Hello again again!

- I'm Rayan and I do bioinformatics!

@RayanChikhi on Twitter

http://rayan.chikhi.name
High expectations from last year - This won’t be the greatest big data talk, just a tribute
Part 1: Intro
Founding members of biological big data

Early Eras of Bioinformatics, Representative Leaders

- Generation -1: E.O. Wilson (compatibility aka perfect-phylogeny - 1965)
- Generation 0: Margret Dayhoff, Russ Doolittle, Joe Felsenstein
- Generation 1: Mike Waterman, David Sankoff (Era of algorithms, pre-data)
- Generation 2: Gene Myers, Russ Altman, Richard Durbin, Sean Eddy

Dayhoff-Eck

- Worked out the theoretical basis of "shotgun-sequencing" of protein (1970)
- Published the first "Atlas of protein sequence and structure" (1966) with 65 sequences. Really the first comprehensive database in bioinformatics. Continued with several additional editions.

Margaret Oakley Dayhoff

Born
Margaret Belle Oakley
March 11, 1925
Philadelphia, Pennsylvania

Died
February 5, 1983
Big data is the natural flow of biology

1972: single gene sequenced
2000: 1 high-quality human genome
2013: many low-quality human genomes
2021: 10 petabases of reads analyzed
2022: 1 million humans VCFs
2022: 50 high-quality human genomes
2024+: ?

Is big data just a technical matter?!
“Informatics is to biology, what mathematics is to physics”
Informatics?
Big data in biology: NCBI GenBank & WGS

**Type:** genome assemblies of >500,000 species

**Size:** 1.2 terabytes (TB) (2022)

All sequences are annotated

**Type:** genome assemblies

**Size:** 16 TB (2022)

Unannotated
Genome issues for comparative analyses

ALL EUKARYOTIC GENOMES (Cumulative: Dec 2023):

- GenBank genomes (all): 36,593 (15,453 species)
- GenBank (with annotation): 6,817 (3,801 species)

GenBank eukaryotic genome submissions (2021):

- 55% are contaminated
- 80% lack annotation
- 20% have annotation
- 58% have >50% proteins annotated as “HYPOTHETICAL”
NCBI SRA

Type: reads
Size: 50 PB
## Units

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Symbol</th>
<th>Power</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>yotta</td>
<td>Y</td>
<td>$10^{24}$</td>
<td>1,000,000,000,000,000,000,000,000,000</td>
</tr>
<tr>
<td>zetta</td>
<td>Z</td>
<td>$10^{21}$</td>
<td>1,000,000,000,000,000,000,000,000</td>
</tr>
<tr>
<td>exa</td>
<td>E</td>
<td>$10^{18}$</td>
<td>1,000,000,000,000,000,000,000</td>
</tr>
<tr>
<td>peta</td>
<td>P</td>
<td>$10^{15}$</td>
<td>1,000,000,000,000,000,000</td>
</tr>
<tr>
<td>tera</td>
<td>T</td>
<td>$10^{12}$</td>
<td>1,000,000,000,000,000</td>
</tr>
<tr>
<td>giga</td>
<td>G</td>
<td>$10^{9}$</td>
<td>1,000,000,000</td>
</tr>
<tr>
<td>mega</td>
<td>M</td>
<td>$10^{6}$</td>
<td>1,000,000</td>
</tr>
<tr>
<td>kilo</td>
<td>k</td>
<td>$10^{3}$</td>
<td>1,000</td>
</tr>
<tr>
<td>hecto</td>
<td>h</td>
<td>$10^{2}$</td>
<td>100</td>
</tr>
<tr>
<td>deca</td>
<td>da</td>
<td>$10^{1}$</td>
<td>10</td>
</tr>
</tbody>
</table>
UK Biobank

Size: 25+ PB
Source: https://twitter.com/uk_biobank/status/1578023831578427393
Type: reads*
* but many use just the SNPs

GTEx

Size: 150 TB
Source: https://www.genomeweb.com/informatics/anvil-platform-makes-popular-nhgri-gtex-database-free-download
Type: reads*
* but many use just the expression data
Your laptop: 0.001 PB

Institut Pasteur: 10 PB

(Youtube: 300 PB)
State of Data Archives (2024):
What if I told you with big data and big computers, one could perform wonderful, ground-breaking genomics... But how?
People at the leading edge of a rapidly changing field "live in the future."

- Paul Buchheit (GMail creator)
“Living in the future” in biology?

- Have a lab technique only a few know
- Have data that will only be public later
- Hold a belief that isn’t established yet
- Discover for the first time that [some phenomenon] happens
- Work on “sci-fi” projects (e.g. create a cell from scratch, genome editing, ..)
- ...


“Living in the future” in biology bioinformatics

- Have a lab computational technique only a few know
- Have data that will only be public later
- Work on “sci-fi” projects (e.g. quantum computing, AI, big data, ..)
Some people living in the future

- George Church, Craig Venter
- Karen Miga & T2T team*
- Evan Eichler, Erik Garrison
- All researchers**

* While the rest of the world still used GRCh38/hg19

** Generally ~months ahead, with your papers to be published
Part 2: Big Data Toolbox

Computation
- Big computers, Cloud, Cluster
- Storage management
- Galaxy
- Knowledge of scaling limits
- Knowledge of cloud costs
- GNU parallel

Data mining
- Pebblescout, branchwater
- ORA
- deCOM
- SRA metadata
No such thing as ‘big data’, only ‘small computers’
Cloud

= A collection of computers owned by a single organization and accessible from the Internet
Recap of last year’s talk

Live: Demo of mapping human 10x coverage HiFi reads using mapquik in <20 seconds, including FASTA conversion using seqkit and chatgpt
Galaxy Project

- Versatile and reproducible workflows
- Web platform
- Open source under Academic Free License

- If you do not have a cluster
- ..or the will to install tools..
- Galaxy offers free computation on pre-installed workflows
Cluster

Acquire knowledge about it:

- Queues:
  - How many CPUs/RAM per job, what timelimit
  - Can your group access any ✨special queue✨

- Storage:
  - Your quota
  - Is “scratch” quota-free? Do files expire?

My scripts:

```bash
srun -q seqbio -p seqbio --mem 100G -c 10 --pty bash
```
Quickly allocates a terminal on any machine

```bash
squeue -o "%.18i %.9P %.8j %.8u %.2t %.10M %.6D %R cores:%c mem:%m cmd:%o " | grep seqbio
```
See what machines are currently being used
Storage management

- How to never run out of storage space:
  - Have 2 folders:
    - ~/archive
    - ~/scratch
  - Rules:
    - Archive is backed up, contains command lines and final results
    - Scratch is fast, but may be deleted at any time
    - Keep the list of files for both, somewhere
  - Keep a dummy 100 GB file ready to be deleted

- Data compression
  - BAM => CRAM => delete it
  - FASTQ => gzip => delete it
  - VCF => BCF
  - GFF/GTF => don’t annotate
Part 2: Big Data Toolbox

Computation
- Big computers, Cloud, Cluster
- Galaxy
- Storage management
- Knowledge of scaling limits
- Knowledge of cloud costs
- GNU parallel

Data mining
- Pebblescout, branchwater
- ORA
- deCOM
- SRA metadata
Knowledge of scaling limits

In order of difficulty:

1. **Estimate** how long an analysis will take
2. Reasons **why** some analyses are slower than expected
3. **How** to reduce that time
Do 200 CPUs always go 200x faster?

Amdahl’s law: NO
Connect the dots from left to right

1) Access data from a SSD disk

2) Access data in memory

3) Access [http://www.evomics.org](http://www.evomics.org) in Australia

4) Human cell cycle

5) Align 1 million short reads

- 100 nanoseconds
- 100 microseconds
- 200 milliseconds
- 10 seconds
- 24 hours

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>nano</td>
<td>n</td>
<td>$10^{-9}$</td>
</tr>
<tr>
<td>micro</td>
<td>μ</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>milli</td>
<td>m</td>
<td>$10^{-3}$</td>
</tr>
</tbody>
</table>
Connect the dots from left to right

- Access data from a SSD disk • 100 nanoseconds
- Access data in memory • 100 microseconds
- Access [http://www.evomics.org](http://www.evomics.org) in Australia • 200 milliseconds
- Human cell cycle • 10 seconds
- Align 1 million short reads • 24 hours

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>nano</td>
<td>$10^{-9}$</td>
</tr>
<tr>
<td>μ</td>
<td>micro</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>m</td>
<td>milli</td>
<td>$10^{-3}$</td>
</tr>
</tbody>
</table>
Knowledge of scaling limits

In order of difficulty:

1. **Estimate** how long an analysis will take
   - Look at performance table in tool paper
   - Try on smaller data and extrapolate

2. **Reasons why** some analyses are slower than expected
   - Limited number of CPUs
   - Limited RAM
   - Slow disk (HDD < Cluster network drives < SSD < NVMe)

3. **How** to reduce that time
   - Most analyses go fast enough on a big cloud/cluster and the right tools
Knowledge of cloud costs

Your workshop instance: 
\texttt{t3a.large}: 2 CPU cores, 8 GB memory
15 cents per hour, 3$/day

💕 \texttt{c6a.48xlarge} 💕 : 192 cores, 384 GB mem, 7$/hour

All costs: https://instances.vantage.sh/
Knowledge of cloud costs

Storage costs!

- EBS (instances hard drive): $0.08/GB/month
- S3 ("Dropbox"): $0.023/GB/month

- If an instance is stopped: EBS costs occur
- If you create an instance snapshot: EBS costs occur too

How to avoid these costs? Terminate instances, delete snapshots, don’t store too much on your S3
General scaling considerations

- **Alignment**
  - Highly parallel, low memory, scales well with number of CPUs

- **Assembly**
  - Moderately parallel, high memory, typically requires a single big machine

- **Annotation**
  - Don’t! (jk), but moderately parallel. Single machine too?

- **Phylogenomics**
  - Can be made parallel (RAxML, Iq-Tree)
GNU parallel

Allows to run the same job on multiple files, simultaneously. Circumvents SLURM.

To count number of lines across many FASTQ files:

```
find . -name *.fastq | parallel -j10 "wc -l {} > {}_.nb_lines"
```

To run many jobs defined by CSV data:

```
cat data.csv | parallel --colsep ',' "./myprogram {1} {2}""
Part 2: Big Data Toolbox

**Computation**
- Big computers, Cloud, Cluster
- Storage management
- Galaxy
- Knowledge of scaling limits
- Knowledge of cloud costs
- GNU parallel

**Data mining**
- Pebblescout, branchwater
- ORA
- deCOM
- SRA metadata
Exploring metagenomes: Pebblescout and Branchwater

- Cutting-edge sequence database search tools
- Think BLAST, but the database is no longer “nr”; it’s all metagenomes.
- All metagenomes, all assemblies (WGS), all human RNAseq, RefSeq

- Search for any sequence > 42 nt
Pebblescout usage example

Collaborator needs to search SRA for all samples containing Wolbachia

We did exactly this in our paper!

- 36 host species were known for Wolbachia
  - Found by searching SRA metadata (2,545 runs)
- Pebblescout search for 3 genes (ftsZ, groE, wsp)
  - Found 16 more hosts (35 runs)
Compared to Pebblescout:
- Only support long queries (> 10 kbp)
- More verbose output/visualizations
We gathered a collection of 360 samples (including contaminants and non contaminants) and obtained a k-mer matrix.
SRA metadata

.. will be presented in the next part
Wrapping up of Part 2: Big Data Toolbox

**Computation**
- Big computers, Cloud, Cluster
- Galaxy
- Storage management
- Knowledge of scaling limits
- Knowledge of cloud costs
- GNU parallel

**Data mining**
- Pebblescout, branchwater
- ORA
- deCOM
- SRA metadata
Part 3

SRA-scale sequence exploration
NCBI SRA

All public sequencing reads

Size: 47 PB as of late 2023
What to do with the entire SRA?
Serratus: all public RNA-seqs analyzed for viral discovery

Discovery of 130,000 new RNA viral species. One-off analysis, 20,000 CPUs (Nature, 2022)
All RNA-seqs pre-2020 (10 petabases)

Serratus download & align (bowtie2) to all viral reference genomes

56,000 CoV+ samples including 9 novel coronavirus species discovered
All RNA-seqs pre-2020

Serratus download & sensitive align (DIAMOND2) to all known versions of RNA virus universal gene

aligned reads (.bam files)
130k novel species discovered
Toolbox used in Serratus

Part 2: Big Data Toolbox

**Computation**
- Big computers, Cloud
- Galaxy
- Knowledge of scaling limits
- Knowledge of cloud costs
- GNU parallel

**Data mining**
- Pebblescout, branchwater
- ORA
- deCOM
- SRA metadata

Didn’t exist
Some follow-ups to Serratus

**Viral reactivation** (Nature 2023)

Discovered HHV-6 reactivation in CAR-T cells

**Ambiviruses** (Nat Comm 2023)

50 known => 20,000 discovered viroids

**Independent use** of Serratus data

Analysis of **circular contigs** in Serratus assemblies
Diving into SRA’s data
What’s SRA metadata?

**SRX8451857**: Resequencing of *Vicugna vicugna* V_ss18
1 ILLUMINA (HiSeq X Ten) run: 111.2M spots, 33.4G bases, 11.8Gb downloads

**Design**: Resequencing

**Submitted by**: Universidad Austral de Chile

**Study**: Resequencing of Genomes of South American Camelids

  PRJNA612032 • SRP265528 • All experiments • All runs

**Sample**: V_ss18

  SAMN14360346 • SRS6753932 • All experiments • All runs

**Organism**: *Vicugna vicugna mensalis*

**Library**:

  *Name*: Vss18  
  *Instrument*: HiSeq X Ten  
  *Strategy*: WGS  
  *Source*: GENOMIC  
  *Selection*: RANDOM  
  *Layout*: PAIRED

**Runs**: 1 run, 111.2M spots, 33.4G bases, 11.8Gb

<table>
<thead>
<tr>
<th>Run</th>
<th># of Spots</th>
<th># of Bases</th>
<th>Size</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRR11905265</td>
<td>111,191,160</td>
<td>33.4G</td>
<td>11.8Gb</td>
<td>2020-06-08</td>
</tr>
</tbody>
</table>

All this information
Accessing SRA metadata

0. NCBI website

1. NCBI FTP metadata

2. SRA metadata on cloud SQL database
   (AWS Athena, GCP BigQuery)
SRA accessions sizes (2023)

Histogram of SRA Accessions Sizes

Count

$10^7$

$10^6$

$10^5$

$10^4$

$10^3$

Size (Gbases)

[0, 10)

[10, 20)

[20, 50)

[50, 100)

[100, 500)

[500, 1000)
Wondering: how many individual sequencing reads are in SRA?

@chris_osulliva is there an estimate of this someplace? 😏

7:46 PM · Nov 10, 2023

**Rayan Chikhi** @RayanChikhi · 3m

About 387 trillion as of today.

```sql
SELECT SUM(
    CASE
        WHEN avgspotlen > 0 THEN (CAST(mbases AS BIGINT) * 1000000) / avgspotlen
        ELSE 0
    END
) AS total_number_of_reads
FROM metadata;
```
SRA accessions types (2023)

- Genomic: 8 M
- Viral RNA: 4 M
- Transcriptomic: 4 M
- Metagenomic: 4 M
- Transcriptomic Single Cell: 1 M
- Other: 1 M
- Metatranscriptomic: 1 M
- Genomic Single Cell: 1 M
- Synthetic: 1 M

Total Megabases (Mb):

- Genomic: 20 G
- Viral RNA: 5 G
- Transcriptomic: 15 G
- Metagenomic: 5 G
- Transcriptomic Single Cell: 1 G
- Other: 1 G
- Metatranscriptomic: 1 G
- Genomic Single Cell: 1 G
- Synthetic: 1 G
"we have processed more than 27.9 Peta base pairs from runs"
How to analyze the entire SRA?
● How much time to download 40 petabytes at 200 MB/sec?
How much time to download 40 petabytes at 200 MB/sec?

~ 6 years
Serratus infrastructure
Alignment: high **speed** or high **sensitivity**, choose one

Credit: RC Edgar
SRA-scale alignment

State of the art (ordered by sensitivity/speed):

1. **Sourmash branchwater** (sketches)
   - Metagenomes, long sequences

2. **NCBI Pebblescout** (k-mers, no alignment)
   - Metagenomes, > 42 bp sequences

3. **Bowtie2, STAR** (k-mers, alignment)
   - Serratus1 (all RNAseqs)
   - Recount3 (750k human/mouse RNAseqs)

4. **DIAMOND** (AA-mers)
   - Serratus1.5 (all RNAseqs)

5. **HMMs**? (profile)
An apparté on unitigs

Many dedicated construction methods:

- **BCALM** (2014), **BCALM2** (2016), .., **Cuttlefish2** (2022), **GGCAT** (2023)
Summary

- Exploring all of Life’s sequencing data
- Tools:
  - SRA metadata
  - SRA data on cloud
  - Alignment algorithms (fast+sensitive)
  - Short read assembly (fast+lowmem+contiguous)
  - Indexing algorithms (fast+sensitive)
Part 4: pangenomics into the wild

Species: Gallus gallus
Vocabulary

Kmer:
A “fun” experiment..

Let’s study why Kmer has this bright yellow color.

1) First, what is the gene responsible for feather color in Gallus gallus? (let’s ask Kmer itself)
2) Then we’ll gather sequencing data from chickens
MC1R

Where does this gene appear in the wild?

First we need to get its sequence, or a chunk of it:

Gathering MC1R genes

I don't have a farm of chicken, but I have big public data.
“If you don’t have data, download it”
Genomics is big
Searching for MC1R

>MC1R
CTTCCCATCTACCGCCGCTGAGCCCTCTCTGAGCCGATACCGGAGCCGCGAGGCAGTGCCGGTGGGGAGGGCGGCCGAGACAGCGGAGTCCCCGCGCTGCTGCCCAGAGGGCTCCCGGGTGGGGGACCGCTTCCCCATCCTTGTGCCTGGGGTGCAGAGGTGCCCACATCCCCTCTGCCTCGTGACCGCGTGCTGCGGGAGCACTGGTGGGGCTGGTTGGGCGCACGGGGGCTTTGGAGGTGCTGCAGTTGTGCTCGGGGCCACGGCCTCCAGCCAGGGGGTCCCTGGGGGCTGAGGCCGGGGCCATGTCGATGCTGGCCCCCCTGCGCCTGCTGCCGCGAGCCCTGGGACGCAGTGAGGGCAACCAGAGCAATGCCACGGCCGGGGCCGGAGGTGCCTGGTGCCAGGGGCTGGACATCCCCAATGAGCTCTTCCTGACGCTGGGGCTGGTGAGCCTGGTGGAGAACCAGCTGGTGGTGGCCGCCATCCTCAAGAACAGGAATCTGCACTCGCCCACGTACTACTTCATCTGCTGCCTGGCCGTCTCCGACATGCTGGTGAGCGTCAGCAACCTGGCCAAGACGCTCTTCATGCTGCTGATGGAGCACGGCGTGCTGGTGATCCGCGCCAGCATCGTCCGCCACATGGACAATGTCATCGACATGCTCATCTGCAGCTCCGTCGTGTCCTCCCTCTCCTTCCTGGGGGTCATCGCCGTGGACCGCTACATCACCATCTTCTATGCGCTGCGCTACCACAGCATCATGACGCTGCAGCGCGCCGTGGTCACCATGGCCAGCGTCTGGCTGGCCAGCACCGTCTCCAGCACCGTCTTAATCACCTACTACCCCAACAACGCCCACATCGCTCTCTCCATGCTGCCTCTCCTCTCTCATGCTGGGGGCCAGAGCTCCGGCGGACGCTGCGGGAGGTGGTGCTGTGCTCCTGGTAGGAGGCGGCACAGACAGGAGGATGGATGGATGGATGGATGGACGGATGGACGATGGATGGATGGACAAACAGATGGGTGGATGGACAGATGGGTGGATGGACAGACAGACGCACCGCGGGGTGTCCCCTGGGTGCCCCAGTGCAGCTGGGGTTGGGCTGCCTGGCCTCGCGCTCCCAAATAAAGGCTCTTTGCAGTGA

Three routes:

1) **https://pebblescout.ncbi.nlm.nih.gov/**

2) SRA metadata query

3) SRA taxonomy query
Pebblescout query

Finds 2000+ hits.

Some of doubtful quality. “chicken” stops appearing in titles after hit number 1750.
SRA metadata query

https://www.ncbi.nlm.nih.gov/sra/?term=%22yellow+chicken%22

Overlap with Pebblescout: 0 😨


SRX4478521: DNA-seq of Gallus gallus: Wuhua yellow chicken
1 ILLUMINA (HiSeq X Ten) run: 38M spots, 11G bases, 3.9Gb downloads
SRA Athena STAT query

```
SELECT *
FROM "sra"."tax_analysis"
WHERE name = 'Gallus gallus' AND total_count > 100
```

In retrospect this is probably way too low, many false hits

Results (317,949)

Contains 72% of the Pebblescout hits

Contains 83% of the “yellow chicken” metadata hits
Getting data from the SRA

**TL;DR:** state of the art is **prefetch + fasterq-dump**

**prefetch:** downloads .sra file locally

**fasterq-dump:** transforms .sra to .fastq or .fasta

Example:

```
prefetch [accession] && fasterq-dump [accession].sra
```
Getting data from the SRA, easily

```
aws s3 cp s3://sra-pub-run-odp/sra/{accession}/{accession} \
  {accession}.sra \
  --no-sign-request

cq-dump --fasta-unsorted --stdout {accession}.sra
```

NIH NCBI Sequence Read Archive (SRA) on AWS

**Description**
The Sequence Read Archive (SRA), produced by the National Center for Biotechnology Information (NCBI) at the National Library of Medicine (NLM) at the National Institutes of Health (NIH), stores raw DNA sequencing data and alignment information from high-throughput sequencing platforms. The SRA provides open access to these biological sequence data to support the research community's efforts to enhance reproducibility and make new discoveries by comparing data sets. Buckets in this registry contain public SRA data in the original (user submitted) format from select high value and newly-released studies as well as all public-access SRA formatted ETL+BIQS data. Also included is all SRA metadata that can be leveraged for attribute-based data discovery.

**Resources on AWS**

**Description**
.s3 buckets for source submissions from sequencing methodologies such as PacBio, Oxford Nanopore Technologies, and 10X Genomics.

**Resource type**
S3 Bucket

**Amazon Resource Name (ARN)**
arn:aws:s3:::sra-pub-src-1

**AWS Region**
us-east-1

**AWS CLI Access (No AWS account required)**
aws s3 ls --no-sign-request s3://sra-pub-src-1/
Big data genomics:)

```bash
$ cat download_and_map_accession.sh

set -e
accession=$1

aws s3 cp s3://sra-pub-run-odp/sra/$accession/$accession $accession.sra --no-sign-request

minimap2 -t20 -x sr mclr.fa <(fasterq-dump --fasta-unsorted $accession.sra) -o mapping/$accession.minimap2_output

rm -f $accession.sra

Parallelize processing:

cat accessions.txt | parallel -j 10 "./download_and_map_accession.sh {}"
```
Running.. *(htop)*

On c6a.32xlarge (128 threads, 256 GB mem):

```
Mem

Swp

Tasks: 92, 383 thr ; 128 running
Load average: 21.41 30.84 45.33
Uptime: 01:39:13
```

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PRI</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>CPU%</th>
<th>MEM%</th>
<th>TIME+</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>70507 ec2-user 20 1676M 235M 2544 889.2 0.1 1:14.58 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71216 ec2-user 20 1906M 274M 2524 882.6 0.1 0:43.88 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70747 ec2-user 20 1806M 273M 2264 856.8 0.1 0:56.76 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70544 ec2-user 20 1842M 242M 2276 856.1 0.1 0:07.39 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70904 ec2-user 20 1730M 266M 2264 833.2 0.1 0:31.07 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71072 ec2-user 20 1672M 242M 2254 833.0 0.1 0:46.12 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73054 ec2-user 20 1842M 249M 2321 815.8 0.1 0:24.28 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70527 ec2-user 20 1906M 267M 2492 793.3 0.1 0:58.48 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70556 ec2-user 20 1906M 274M 2544 781.4 0.1 0:58.39 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71433 ec2-user 20 1736M 266M 2284 773.4 0.1 0:40.49 minimap2 -t 20 -a -x sr mclrla.fa /dev/fd/63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71522 ec2-user 20 1900M 826M 4876 303.0 0.3 0:17.61 fasterq-dump --fasta-unsorted --stdout SRR10315070.sra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70766 ec2-user 20 1900M 829M 4880 300.4 0.3 0:23.55 fasterq-dump --fasta-unsorted --stdout SRR10058581.sra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70536 ec2-user 20 1900M 829M 4872 298.4 0.3 0:28.64 fasterq-dump --fasta-unsorted --stdout SRR10058584.sra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71305 ec2-user 20 1900M 826M 4876 298.4 0.3 0:18.30 fasterq-dump --fasta-unsorted --stdout SRR10315066.sra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70685 ec2-user 20 1900M 829M 4880 296.4 0.3 0:23.57 fasterq-dump --fasta-unsorted --stdout SRR10058582.sra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70973 ec2-user 20 1900M 827M 4888 296.4 0.3 0:20.93 fasterq-dump --fasta-unsorted --stdout SRR10315069.sra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
But then a bit later..

<table>
<thead>
<tr>
<th>Tasks: 66, 185 thr ; 15 running</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load average: 24.14 21.70 29.99</td>
</tr>
<tr>
<td>Uptime: 02:23:01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PRI</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>CPU%</th>
<th>MEM%</th>
<th>TIME+</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>80288</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1736M</td>
<td>259M</td>
<td>2508</td>
<td>S</td>
<td>944.7</td>
<td>0.1</td>
<td>0:00.66</td>
<td>minimap2 -t 20 -a -x sr mcl.r.fa /dev/fd/63</td>
</tr>
<tr>
<td>802917</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>305.6</td>
<td>0.3</td>
<td>0:56.21</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>802914</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1736M</td>
<td>259M</td>
<td>2508</td>
<td>S</td>
<td>73.6</td>
<td>0.1</td>
<td>0:11.68</td>
<td>minimap2 -t 20 -a -x sr mcl.r.fa /dev/fd/63</td>
</tr>
<tr>
<td>802915</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1736M</td>
<td>259M</td>
<td>2508</td>
<td>R</td>
<td>59.0</td>
<td>0.1</td>
<td>0:10.75</td>
<td>minimap2 -t 20 -a -x sr mcl.r.fa /dev/fd/63</td>
</tr>
<tr>
<td>802916</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1736M</td>
<td>259M</td>
<td>2508</td>
<td>R</td>
<td>57.0</td>
<td>0.1</td>
<td>0:11.10</td>
<td>minimap2 -t 20 -a -x sr mcl.r.fa /dev/fd/63</td>
</tr>
<tr>
<td>802919</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>51.0</td>
<td>0.3</td>
<td>0:08.86</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>802924</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>51.0</td>
<td>0.3</td>
<td>0:08.85</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>802922</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>50.4</td>
<td>0.3</td>
<td>0:08.99</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>802889</td>
<td>ec2-user</td>
<td>20</td>
<td>5340</td>
<td>1312</td>
<td>1260</td>
<td>R</td>
<td>49.7</td>
<td>0.3</td>
<td>0:08.56</td>
<td>samtools view -Hf4 -</td>
<td></td>
</tr>
<tr>
<td>802923</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>49.7</td>
<td>0.3</td>
<td>0:08.57</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>802920</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>45.1</td>
<td>0.3</td>
<td>0:08.77</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>802921</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>38.5</td>
<td>0.3</td>
<td>0:08.55</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>802918</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1906M</td>
<td>828M</td>
<td>4868</td>
<td>R</td>
<td>14.6</td>
<td>0.3</td>
<td>0:08.51</td>
<td>fasterq-dump --fasta-untersorted --stdout SRR6490215.sra</td>
</tr>
<tr>
<td>868759</td>
<td>ec2-user</td>
<td>20</td>
<td>0</td>
<td>1338M</td>
<td>370M</td>
<td>16652</td>
<td>R</td>
<td>8.0</td>
<td>0.1</td>
<td>1:55.79</td>
<td>/usr/bin/python3 -s /usr/bin/aws s3 cp s3://sra-pub-run-odp/sra/SRR4897316/SRR4897316.fq.gz s3://sra-pub-run-odp/sra/SRR6446456.SRR4897316.fq.gz</td>
</tr>
</tbody>
</table>
Remember Part 2..

Knowledge of scaling limits

In order of difficulty:

1. **Estimate** how long an analysis will take
   - Look at performance table in tool paper
   - Try on smaller data and extrapolate

2. **Reasons why** some analyses are slower than expected
   - Limited number of CPUs
   - Limited RAM
   - Slow disk (HDD < Cluster network drives < SSD < NVMe)

3. **How to reduce that time**
   - Most analyses go fast enough on a big cloud/cluster and the right tools
What's happening? see `iotop`

```
<table>
<thead>
<tr>
<th>TID</th>
<th>PRIO</th>
<th>USER</th>
<th>DISK READ</th>
<th>DISK WRITE</th>
<th>SWAPIN</th>
<th>IO&gt;</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>618212</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>12.02 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>618393</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>11.79 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>687135</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>12.48 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>688754</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>12.59 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>778885</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>11.81 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>791289</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>12.40 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>798802</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>11.40 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>802884</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>12.67 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>807479</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>11.98 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>809481</td>
<td>be/4</td>
<td>ec2-user</td>
<td>0.00 B/s</td>
<td>12.34 M/s</td>
<td>?unavailable?</td>
<td>python3 -s /usr/bin</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>be/4</td>
<td>root</td>
<td>0.00 B/s</td>
<td>0.00 B/s</td>
<td>?unavailable?</td>
<td>systemd --switched</td>
<td></td>
</tr>
</tbody>
</table>
```

Disk speed limit around 125 MB/sec
Workaround

-> Use more machines, smaller ones.

# setup 10 machines, make a list of different accessions on each, then run:

```bash
cat ips.txt | parallel -j 10 \
    ssh ec2-user@{} ./run_previous_parallel_cmd.sh
```

Didn’t do that here 😞 (lazy instructor)
Analyzed so far

3 terabases from “yellow chicken” SRA accessions downloaded and mapped to MC1R

Took around 1.5 hours, 6$/hour
Making mini-assemblies of MC1R

$ sgpt --model gpt-4 "setup a variant calling script that takes a SAM file as input, mapped to a reference, and outputs a FASTA consensus of the reference"

```
  samtools view -S -b $SAM_FILE > $BAM_FILE
  samtools sort $BAM_FILE -o $SORTED_BAM_FILE
  samtools index $SORTED_BAM_FILE
  bcftools mpileup -O b -o $BCF_FILE -f reference.fasta $SORTED_BAM_FILE
  bcftools call --ploidy 1 -m -v -o $VCF_FILE $BCF_FILE
  cat reference.fasta | bcftools consensus $VCF_FILE > $FASTA_FILE
```
Making mini-assembly of MC1R

$ sgpt --model gpt-4 "can bcftools consensus mask regions where it doesn't have confidence in the base call? how to get mask.bed?"

```
samtools depth -a alignment.bam | awk '$3 < 5' | awk '{print $1"\t"$2"\t"$2+1}' > mask.bed
```
Analysis of variation

- Filtered consensuses with too many N’s (chatgpt again)
- Constructed DB graph with BCALM of all the MC1R genes
So, where’s the “yellow chicken” allele?

- Remember the Wuhua yellow chicken accession?
- BLASTed the consensus gene against the pangenome graph

Several hits to low-frequency SNPs, could be any/some of those..
We need more data
Need color metadata
Breed information given for some of the chicken. How to extract?
Python script calling chatGPT

```python
from openai import OpenAI

def determine_chicken_color(line):
    query = f"Determine the color of the chicken based on the following data: {line}. You may reply only: yellow, orange, or other. [...]. Do not guess."

    response = client.completions.create(
        model="gpt-3.5-turbo-instruct",
        prompt=query, max_tokens=50)

    return response.choices[0].text.strip()
```
Result of chicken coloring

$ tail chicken_color.txt
SRR2917304,other
SRR8490109,other
SRR25338401,Other
SRR13193600,other
ERR5036744,yellow
SRR24605477,other
SRR12228200,other
ERR4351384,other
ERR3505973,other
Those chatGPT color predictions..

$ tail chicken_color.txt
[..]
ERR5036744,yellow
[..]

$ grep ERR5036744 *.csv
[..], {k=common_name_sam, v=chicken}, [..],
{k=insdc_center_alias_sam, v=QUEEN MARY UNIVERSITY OF LONDON}, [..]

No breed information! ChatGPT hallucinated that yellow color.
This was a failed analysis

- This is OK
- Not all analyses are successes
- Move on to the next one
- We learned along the way, right?
Outro
What we’ve seen today

● Some elements of big data bioinformatics
● Toolbox for Big Data
   ○ Cloud, parallelism, storage handling, knowledge of limitations
● SRA primer
   ○ Mining metadata
   ○ Mining sequences
   ○ Serratus
● Chicken Pop-Pan
   ○ Mining 1 gene for 1 species across the SRA
   ○ Using metadata search and taxonomy search
WE'RE GONNA NEED

A BIGGER INSTANCE TYPE
This talk was first dedicated to the **Workshop on Genomics 2023** in Cesky Krumlov: Guy, Janina, Milos, Kartik, Alena, Madee, Joan, Mercè, and Josie.

Fourthly to Workshop on Genomics 2024!

.. and secondly dedicated to JOBIM’23 organizers..
Side note: all microbes can fit onto an SD, carrier pigeons are faster than Internet

Testing this still on our todo list

K. Brinda

WHEN YOU LIVE IN THE 1920S

BUT CAN ALREADY SEE THE FUTURE IN YOUR HAND

ME EVERY DAY AFTER THE WORKSHOP

WHEN YOU ONLY LIVE IN THE 1920S
Genomes & metagenomes assembly

Algorithms and data structures on k-mers

Sequence search in very large datasets

Pangenomics
Thank you for your attention!