Applied Bioinformatics

The Reference Genome
November 2015, Belgrade

Goran Rakočević, PhD
goran.rakocevic@sbgenomics.com

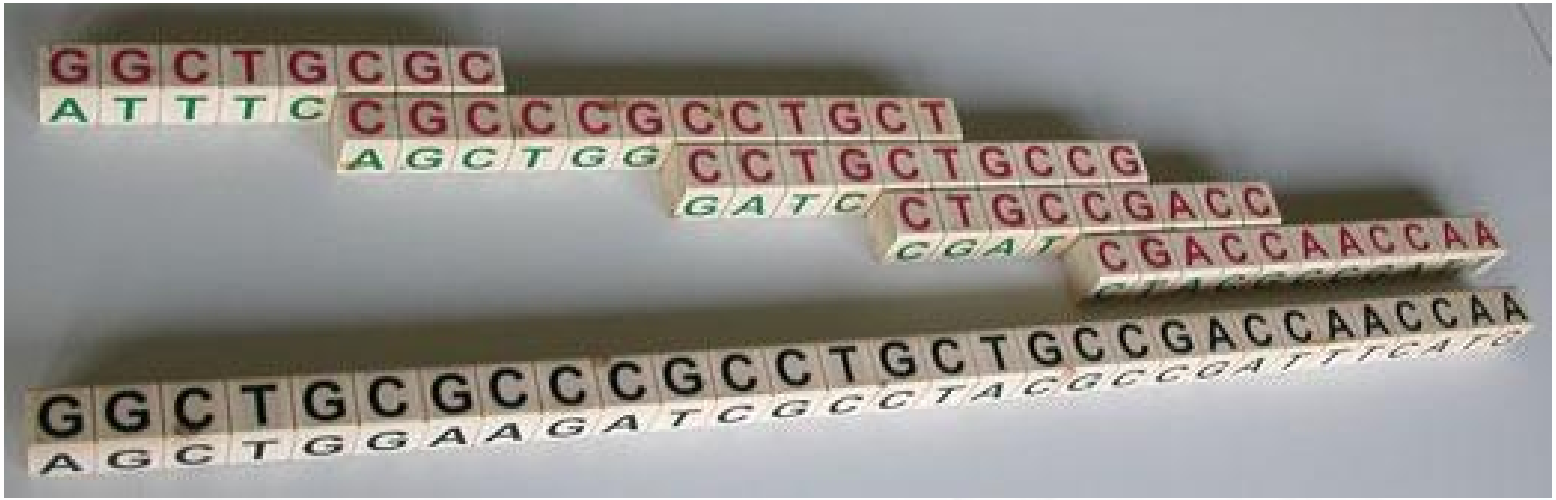
SevenBridges
genomics

DNA Sequencing - Reminder

- We got a FASTQ file with the “reads” - little pieces of the genome

What to do with the sequencing reads?

- How do we reconstruct the genome that went into the “shredder”?
- We could try “assembly” - connecting the reads into longer sequences



SevenBridges genomics

Genome Assembly

- Greedy algorithm (suboptimal solution):
  1. Calculate pairwise alignments of all fragments.
  2. Choose two fragments with the largest overlap.
  3. Merge chosen fragments.
  4. Repeat step 2 and 3 until only one fragment is left.
- Even the more practical solutions have problems:
  - High computational cost
  - High memory consumption (100s of GB or RAM)
  - Difficult to connect the genome with single library preparation
- Won’t be covering in depth in this course
  - Those interested can look up “de novo genome assembly” for more info

What do with the sequencing reads? (an alternative)

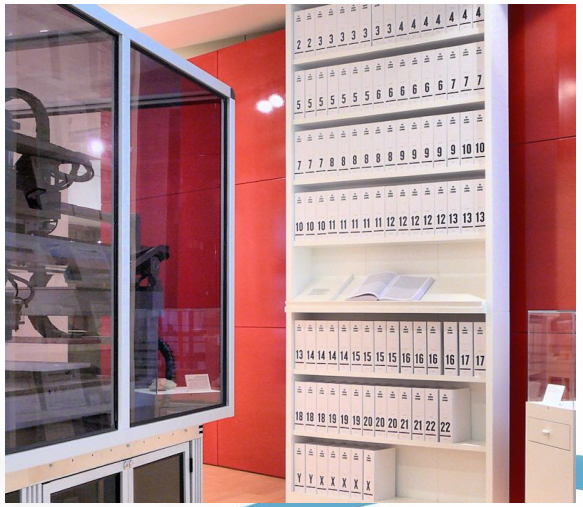
- We do have an alternative approach to recover that genome (the genome that went into the “shredder”)
- This way is faster and more practical than assembly
- It is based around a *Reference genome, Alignment,* and *Variant Calling*
- But first we will need to define these terms and procedures

It will take 3 weeks before we can connect all the dots, so bear with me!

SevenBridges genomics

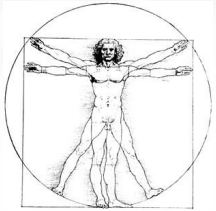
The reference genome

- A reference genome is representative example of a species' genome
- It is often built from genomes of multiple individuals
- Every individual differs from this genome in some places
- We can think of *genomic variation* as differences from the reference (genome)
- Reference genomes provide a coordinate system for communicating genomic data
- And, they make analyses easier!



SevenBridges genomics

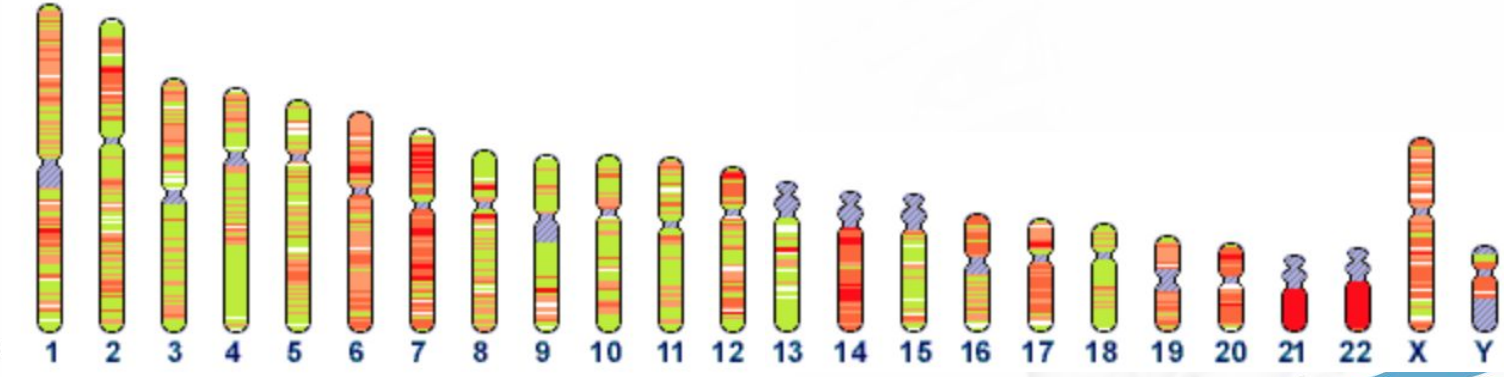
Human Genome Project



- International scientific initiative
  to create a reference human genome
- Active in years 1990 to 2003, at a cost over 3B$
- In parallel, Celera Corporation launched a privately funded project,
  with the same goal, but had the intention to patent the sequence
- This caused a race, leading up the release of the first draft
  of the public version of the human genome on July 7, 2000,
  by the UCSC Genome Bioinformatics Group

Human genome stats

- The human genome has 23 pairs of chromosomes:
  - Males have 22 autosomes, one X and one Y chromosome
  - Females have 22 autosomes and two X chromosomes
- There is also a separate Mitochondrial DNA contig
- Total length is ~3 Gigabases (3B basepairs)
- About 20 000 genes covering about 30 Megabases



genomics

Human genome HG38

- Current version of the chromosomes is *HG38* (though 37 is still widely used)
- Released December 24th 2013
- Additional sequences are added to the genome:
  - Unplaced sequences (some genomes contain them, somewhere)
  - Unlocalized sequences (chromosome known, but coordinates are not)
  - Alternate sequences (some genomes contain them, instead of something parts)
  - Human Herpesvirus 4 type 1
- Patches are commonly added
- http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/human/

FASTA file format

- A simple format for storing reference files
- Two “types” of lines:
  - Description lines, start with ‘>’, contain sequence name and optional description
  - Sequence lines follow a description line and contain the actual nucleotide sequence
- Sequences are represented by a single description line, followed by one or more sequence lines (usually 70 bases or less in a line)

```
>chr 1
ACTGCCCTCCGTACCTACTGCCCTCCGTACCTACTGCCCTCCGTACCTACTGCCCTCCGTACCT
ACTGCCCTCCGTACCTACTGCCCTCCGTACCTACTGCCCTCCGTACCTACTGCCCTCCGTACCT
ACTGCCCTCCGTACCTACTGCCCTCCGTACCTACTGCCCTCCGTACCTACTGCCCTCCGTACCT
ACTGCCCTCCGTACCTACTGCCCTCCGTACCTACTGCCCTCCG
```

Pysam - Python fasta interface

- Pysam - a Python toolkit for working with genomic files
- pysam.Fastafile:
  - Create a fasta file parser: fasta = pysam.Fastafile(path_to_file)
  - Get the sequence names in the file: fasta.references
  - Get the lengths of the sequences: fasta.lengths
  - Retrieve a (part of) sequence: fasta.(sequence_name, [start], [stop])
- Fasta coordinates in pysam are zero-based
  - Not true for all filetypes in pysam!
- Other fasta interfaces for Python exist
  - pyfasta - works only on fasta files
  - biopython - a much larger toolkit that complement sequences, etc.
  - We are describing pysam, as it covers all the file types covered in the course

Exercise: FASTA file format (15 minutes)

- An example FASTA file is found under data/example_reference.fasta
- View the contents of the file
  - You can use “less file”, or “!less” file in the Notebook to invoke less
- Create a pysam Fastafile parser
- Get and print sequence names
  - How many sequences are there?
- Fetch the entire sequence
  - How long is it?
  - Print the first 100 bases

```
import pysam

fasta = pysam.Fastafile(file_path)

fasta.sequences() # sequence names
fasta.lengths() # length of each sequence
fasta.fetch(sequence_name, [start, [stop]]) #fetch a sequence

len(my_string/list) #get the length of list or string

my_list[0:10], my_list[0:], my_list[:10] # list slicing
my_list[:-3], my_list[0:10:4], my_list[::4] # more slicing
```

Exercise: FASTA file format (30 minutes)

- Full hg38 FASTA file is located under in data/GRCh38.fasta
- Create a pysam Fastafile parser
- Get sequence names for all contigs
  - How many contigs are there?
  - How long is chromosome 5?
- Fetch section chromosome 17:43044295-43125370
  - What is the “GC content” on this region? (percent of G and C bases)
  - What is the most common 3-mer?
- Fetch base at chromosome 1:248755121
  - What is the base?
- Fetch region at chromosome 1:5000-5100
  - What is the base composition?

FASTA File IUPAC Codes (and contig names)

- A, C, T, G, U - Nucleotides (Adenine, Cytosine, Guanine, Thymine, Uracil)
- Ambiguous bases:
  - R - A or G
  - Y - C, T, or U
  - N - Any base
  - K, M, S, W, B, D, H, V, X, - https://en.wikipedia.org/wiki/FASTA_format

- Often two versions of the Human Reference Genome are found
  - Chromosomes labeled 1, 2, 3… (Human Genome Consortium style)
  - Chromosomes labeled chr1, chr2, chr3… (UCSC style)

FASTA index (fai)

- Some tools build or require indices of the fasta file
  - Needs to be in the same folder, and have the same name as the fasta + .fai extension
- Fai file structure:
  1. The name of the sequence
  2. The length of the sequence
  3. The offset of the first base in the file
  4. The number of bases in each fasta line
  5. The number of bytes in each fasta line

Genes in HG38 (UCSC Genome Browser)

- A FASTA file contains raw sequences of nucleotide from a genome
- Some sections of these sequences have biological meanings attached
- Examples:
  - Chromosome 17, 43.04 Mb - 43.13 Mb is the BRCA1 Gene, implicated in breast cancer
  - Chromosome X, 73.82 Mb - 73.85 Mb is the XIST lncRNA, implicated in X inactivation
  - Chromosome 1, 121.1 Mb - 124.3 Mb is the Chromosome 1 centromere
- USCS Genome browser is a place where different annotation tracks can be explored in depth
- USCS Genome browser is located at https://genome.ucsc.edu/

UCSC Genome Browser exercise (25 minutes)

- Google: UCSC Genome Browser
- Turn off all tracks, but: *Base Position*, *RefSeq Genes*, *Gencode*, and *Common SNPs*
  - RefSeq and Gencode are two alternative gene location packs
  - Common SNPs is a set of common known mutations in the genome
- Go to the BRCA1 gene in the genome browser
- Switch gene tracks between full and dense views
  - Why are there multiple lines in the full view?
  - How many exons are visible in the dense view?
- Add *7 way cons* full view track
  - Open and explore track description in a new tab
  - How conserved is BRCA1? Which parts are most conserved
- Explore other tracks and options

Annotations file formats

- Annotations are stored in few (similar) file types
  - **BED**
  - GTF
  - GFF
  - …
- **BED file format** (tab-separated):

CHROM START END NAME score ticks blocks

- Chrom, start and end are required
- Start is 0-based
- End is open interval (not included)

PyBedTools - Python Bed Interface

- pybedtools.BedTool is the primary interface
- Parses bed files, allows for iteration (for line in b)
- Built-in functions for sorting, intersecting, merging bed files, etc.
- pybedtools.BedTool:
  - Create a BedTool: *b = BedTool(file_path*)
  - Iterate through a BedTools: *for line in b: …*
  - Intersect two bed files: a.intersect(b)
- More detailed documentaion available at: https://pythonhosted.org/pybedtools/

Exercise: PyBedTools (30 minutes)

- Create a BedTool from the file (**Illumina.TruSeq.b37.bed**)
  - This BED files is from an actual exome capture kit!
  - How many regions are present in the BED file?
- Print out the few regions from this BED files
  - **CAREFUL! Printing the whole BED file WILL kill your Notebook!**
  - Which fields are present?
- Place all regions whose name contains ‘BRCA1’
  - In python you can do: *if ‘a’ in ‘string’*
  - Use a list comprehension for the task *[x for x in iterator]*
- What does BedTool.merge() function do?