

The human genome & cloud computing

Michael Schatz

Sept 19, 2023

Lecture 7: Applied Comparative Genomics



Assignment 3: Genome Assembly

Due Wednesday Sept 27 by 11:59pm

The screenshot shows a GitHub repository page for 'appliedgenomics2023' with the path 'main/assignments/assignment3'. The 'README.md' file is open, displaying assignment details and instructions.

Assignment Date: Wednesday, September 20, 2023
Due Date: Wednesday, September 27, 2023 @ 11:59pm

Assignment Overview

In this assignment you will explore WDLs as a workflow language to orchestrate a variety of read mapping tasks. You can execute your WDLs using `miniwdl` after running `pip install miniwdl`. You may also find `learn-wdl` to be very helpful.

If you are using a Mac, make sure to disable [gRPC FUSE for file sharing](#). There is a currently a very weird bug in Docker desktop 4.12.0 (docker engine 20.10.17) on Apple M1 where if you try to disable "Use gRPC FUSE for file sharing" via the GUI it will turn itself back on when you try to activate it. On M1 you will also need to set the DOCKER_HOST environment variable [Notes](#). Docker is aware of the problem and are working on a fix. In the meantime there is a workaround available: [docker-for-mac#6467](#)

The bioinformatics tools you will need for the assignment (bowtie, samtools, etc) are bundled into a [Docker](#) container so you wont need to install other software packages. This will get you ready to run your code in the cloud for future assignments. Instructions for running the container are available here: <https://github.com/mschatz/wga-essentials>. This is known to work on new macs (M1) and older macs (intel chip). It should also run on Linux and Windows but let us know if you have any issues.

As a reminder, any questions about the assignment should be posted to [Piazza](#).

Question 1. de Bruijn Graph construction [10 pts]

- Q1a. Draw (by hand or by code) the de Bruijn graph for the following reads using k=3 (assume all reads are from the forward strand, no sequencing errors, complete coverage of the genome). You may find [graphviz](#) to be helpful (see below).

ATTCA
ATTGA
CATTG
CTTAT
GATTG
TATTT
TCATT
TCTTA
TGATT
TTATT

<https://github.com/schatzlab/appliedgenomics2023/tree/main/assignments/assignment3>

Check Piazza for questions!

The screenshot shows the Docker website homepage. At the top, there's a banner for "DOCKERCON LIVE IS BACK". Below it, the Docker logo is on the left, followed by navigation links: Products, Developers, Pricing, Blog, About Us, and Partners. To the right are a search bar, a "Sign In" button, and a prominent blue "Get Started" button. The main feature is a large, bold, dark blue headline: "Develop faster. Run anywhere." Below the headline, a subtext reads: "The most-used tool in Stack Overflow's [2023 Developer Survey](#)". Underneath are two buttons: "Download for Mac - Intel Chip" and "Get started". A modal window titled "Images" is partially visible at the bottom, showing a list of images including "whale-api" with its tags and details.

<http://docker.com>

The screenshot shows a web browser window displaying the OpenWDL website at <http://openwdl.org>. The page has a dark blue header bar with the URL and a magnifying glass icon. The main content area features a large logo with three blue hexagons followed by the text '{wdl}'. To the right is a sidebar with a teal header containing the text 'OpenWDL' and 'Community driven open-development workflow language'. The sidebar includes links for 'About', 'How to participate', 'More information', and 'Core group'. The main content area contains sections for 'About OpenWDL', 'How to participate', and a footer with social media icons.

{wdl}

About OpenWDL

The **Workflow Description Language** (WDL) is a way to specify data processing workflows with a human-readable and -writeable syntax. WDL makes it straightforward to define analysis tasks, chain them together in workflows, and parallelize their execution. The language makes common patterns simple to express, while also admitting uncommon or complicated behavior; and strives to achieve portability not only across execution platforms, but also different types of users. Whether one is an analyst, a programmer, an operator of a production system, or any other sort of user, WDL should be accessible and understandable.

WDL was originally developed for genome analysis pipelines by the Broad Institute. As its community grew, both end users as well as other organizations using WDL for their own software, it became clear that there was a need to allow WDL to become a true community driven standard. The **OpenWDL** community has thus been formed to steward the WDL language specification and advocate its adoption.

How to participate

As OpenWDL is looking to put the active development of WDL into the hands of the community we have set up multiple ways to participate.

 Mailing list for general conversation with the WDL community

 Github issues for suggestions and discussions on the WDL specification

<http://openwdl.org>

github.com/openwdl/learn-wdl#readme

learn-wdl

What is WDL?

- The Workflow Description Language (WDL or 'widdle') is an [open source scripting language](#) which allows you to specify data processing workflows with a human-readable and -writeable syntax.
- WDL was originally developed for genomics, but can be extended to other domains.
- This repository contains educational materials for learning to read and write WDL scripts.

Learn WDL in 3 Steps

1. Set up dev env
 - Verify dev env by running 'hello.wdl'
2. Review/run WDL scripts in folders
 - /hello-worlds (can watch screencasts)
 - /language-patterns (can watch screencasts)
 - /pipeline-examples
 - /genomic-tool-pipelines (optional – uses GATK...)
3. Review reference material
 - WDL language concepts
 - External links, includes nf-to-wdl ref

About WDL script execution

WDL scripts are not executable in and of themselves, but they require an execution engine and environment to be runnable. Compliant executions engines should support the features of a specific version of the WDL specification, i.e. WDL 1.0, etc...

Please see the linked engine documentation for information on available execution options and support. Some of the environments (shown linked below) include WDL parsing, linting and DAG (Directed Acyclic Graph or workflow) visualization tools as well.

Most of these environments request one or more language runtimes (such as Java, Python). Please read the documentation for the execution environment documentation for the particular environment for details.

Environments

- cromwell** - IMPORTANT This Repo uses `cromwell-50.jar` for testing all WDL example scripts
- miniwdl** - WDL script local runner & developer toolkit for Python 3.6+ - [docs here](#) & [course here](#)
- dxWDL** - takes a pipeline written in WDL and compiles it to an equivalent workflow on the DNAexus platform. WDL draft-2, version 1.0, and the development version are supported
- wdLTools** - Scala programming language library for parsing WDL, and command-line tools for type-checking, code formatting, and more.

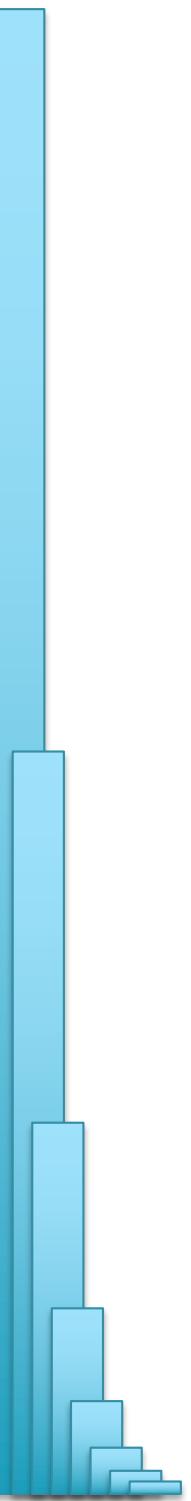
 Click below to WATCH 'Course Intro Video' (10 min) on YouTube

Contributors 5

Languages

WDL 92.2% Shell 7.8%

<https://github.com/openwdl/learn-wdl>



Part I: The human genome



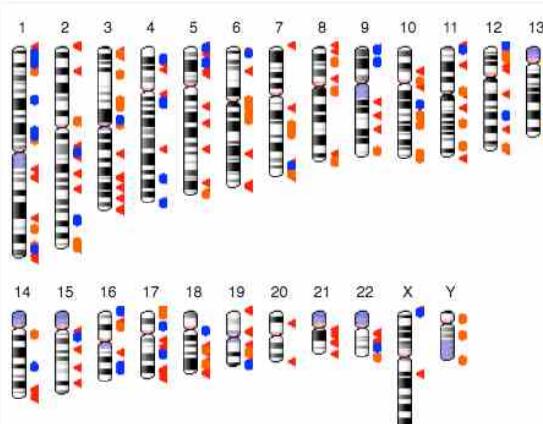
The Sequence of the Human Genome
Venter et al.
Science 291, pp 1304–1351 (2001)



Initial sequencing and analysis of the human genome
International Human Genome Sequencing Consortium
Nature 409, pp 860–921 (2001)

Human Genome Overview

Information about the continuing improvement of the human genome



- ◀ Region containing alternate loci
- Region containing fix patches
- Region containing novel patches

Ideogram of the latest human assembly, GRCh38.p11

GRCh38.p11

GRCh37.p13

GRCh37

GRCh38.p11

Release date: June 14, 2017

Release type: minor

Release notes: GRCh38.p11 is the eleventh patch release for the GRCh38 reference assembly. No chromosome coordinate changes were made. The total number of patch scaffolds is now: 64 FIX and 59 NOVEL.

Assembly accessions: GenBank: [GCA_000001405.26](#), RefSeq: [GCF_000001405.37](#)

Pseudoautosomal regions

Name	Chr	Start	Stop
PAR#1	X	10,001	2,781,479
PAR#2	X	155,701,383	156,030,895
PAR#1	Y	10,001	2,781,479
PAR#2	Y	56,887,903	57,217,415

The GRC is working hard to provide the best possible assembly by both generating multiple representations (alternative paths) for each chromosome, represented by a single path. Additionally, we are reassembling the genome to allow users who are interested in a specific locus to do so without affecting users who need chromosome coordinate sets.

Download data:

- GRCh38.p11 (latest minor release) FTP
- GRCh38 (latest major release) FTP
- Genomic regions under review FTP
- Current Tiling Path Files (TPFs)

Transitioning to GRCh38? Try the [NCBI Remapper](#) to convert your assembly alignments used by the GRC.

Next assembly update

The next assembly update (GRCh38.p12) will be released in July 2017.

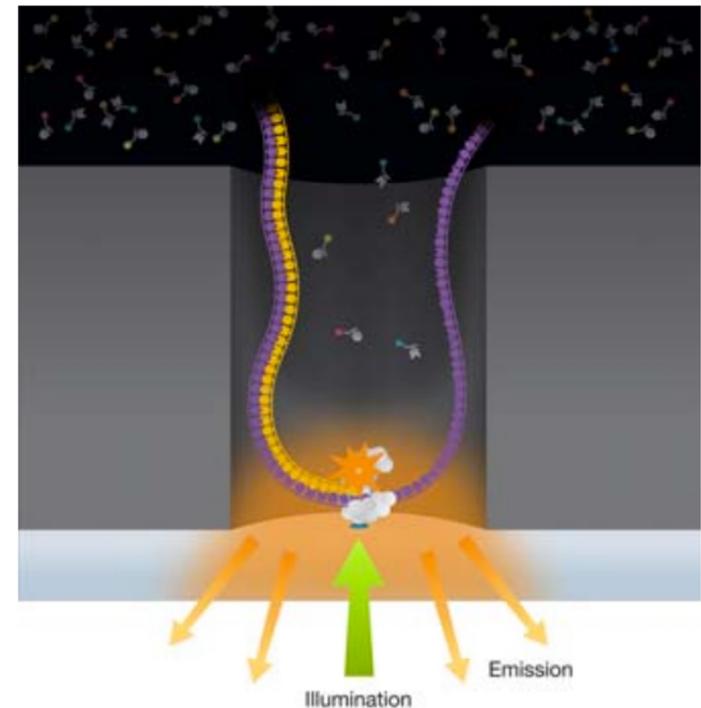
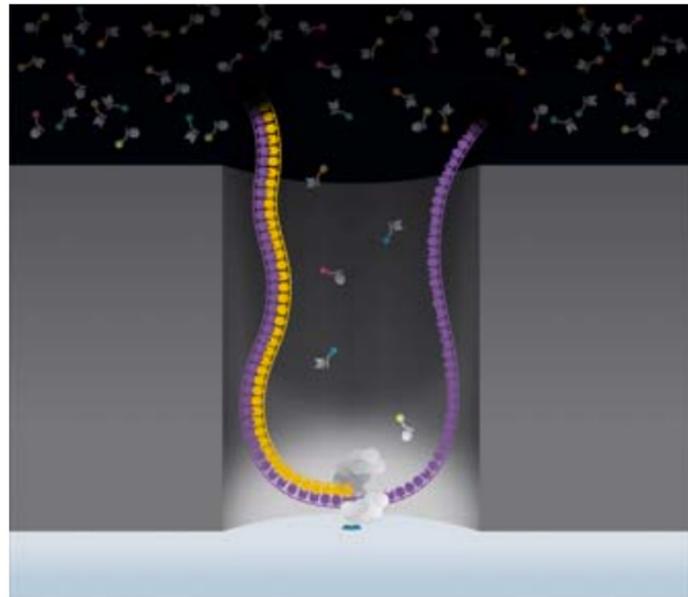




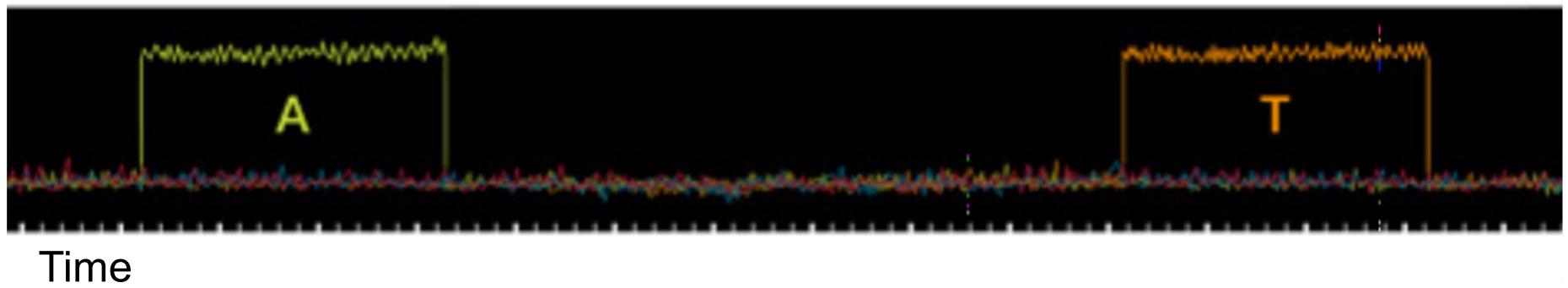
PacBio Single Molecule Real Time Sequencing (SMRT-sequencing)

PacBio: SMRT Sequencing

Imaging of fluorescent phospholinked labeled nucleotides as they are incorporated by a polymerase anchored to a Zero-Mode Waveguide (ZMW).



Intensity



“HiFi” Circular Consensus Reads

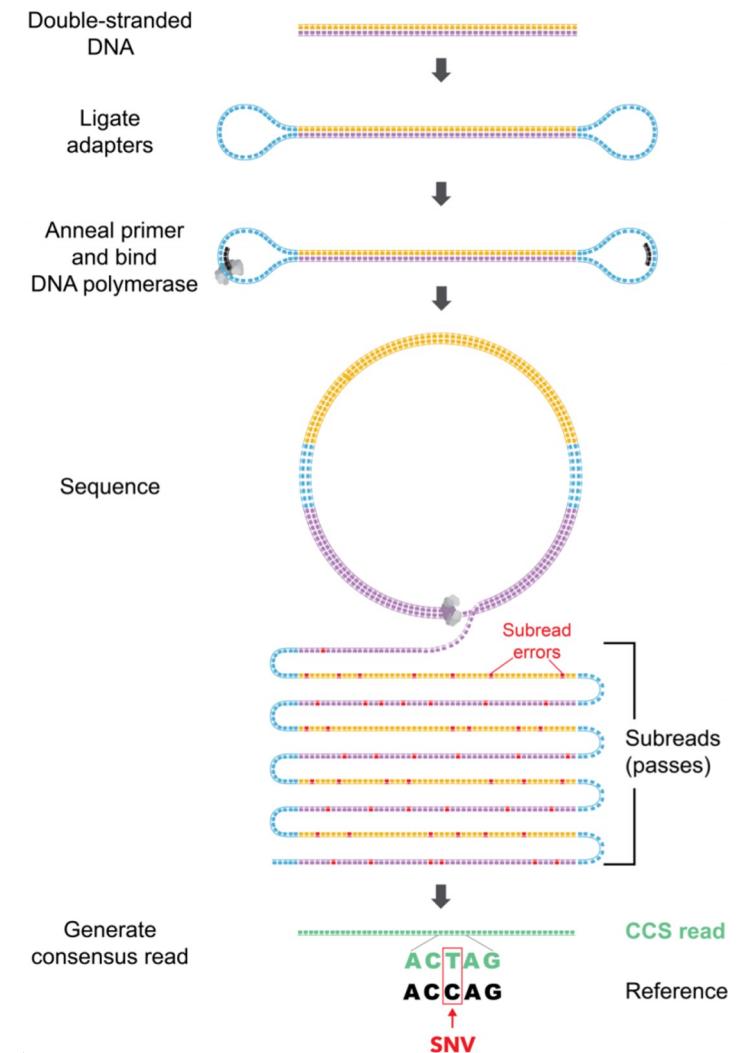
High-quality reads produced by sequencing the same molecule multiple times

Higher accuracy for low-coverage sequences like somatic variants or lowly expressed transcripts in RNA-seq, more interpretable alignments, better & faster assembly

Limits read length, used to be very expensive but more manageable now

Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome

Wenger et al (2019) Nature Biotechnology doi:10.1038/s41587-019-0217-9

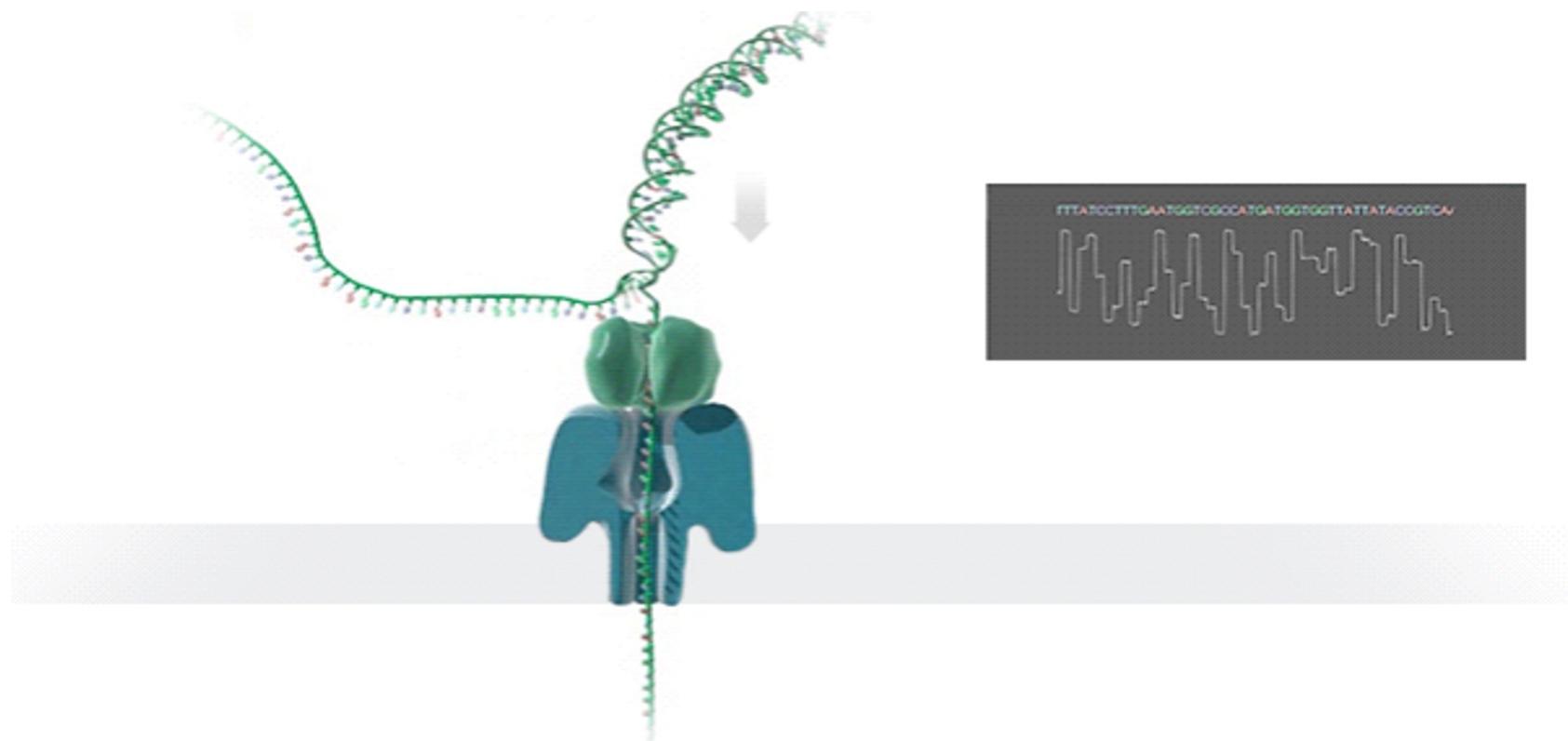


Oxford Nanopore Technologies (ONT)

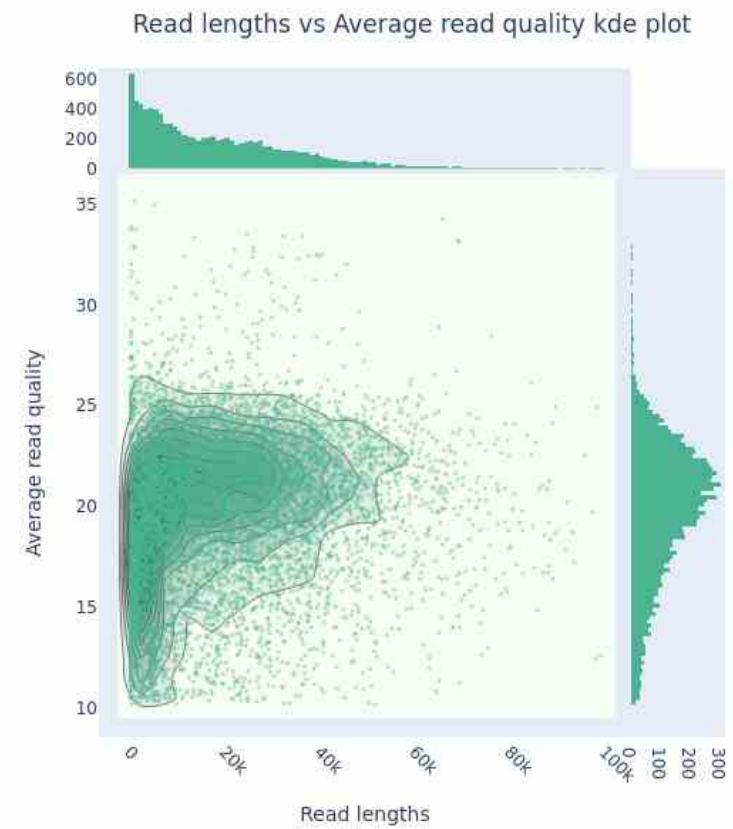
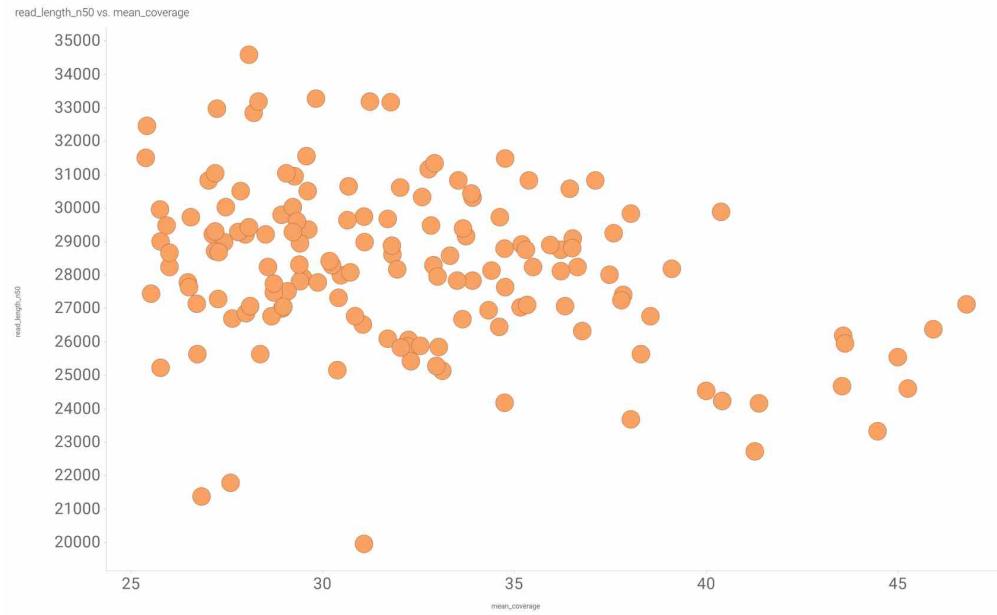


Nanopore Sequencing

Sequences DNA/RNA by measuring changes in ionic current as nucleotide strand passes through a pore

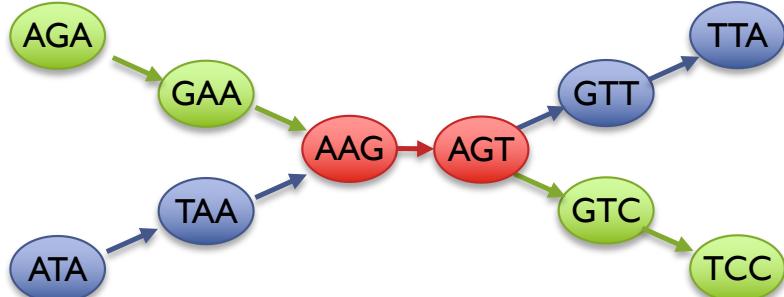


ONT at JHU



Two Paradigms for Assembly

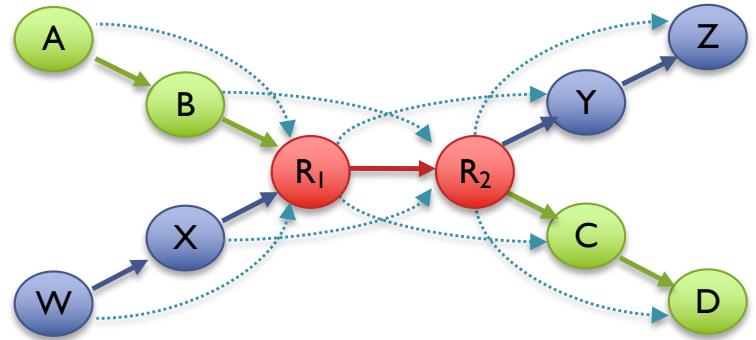
de Bruijn Graph



Short read assemblers

- Repeats depends on word length
- Read coherency, placements lost
- Robust to high coverage

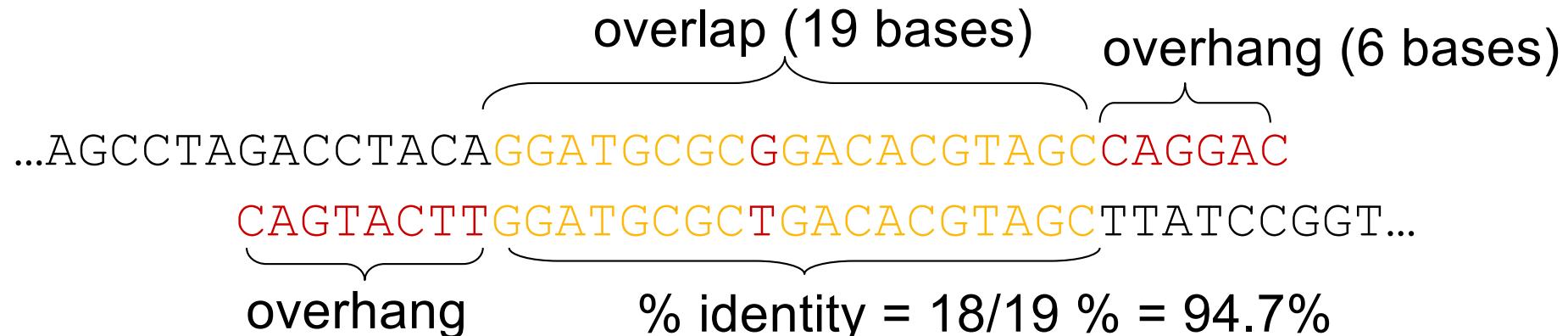
Overlap Graph



Long read assemblers

- Repeats depends on read length
- Read coherency, placements kept
- Tangled by high coverage

Overlap between two sequences



overlap - region of similarity between regions
overhang - un-aligned ends of the sequences

The assembler screens merges based on:

- length of overlap
- % identity in overlap region
- maximum overhang size.

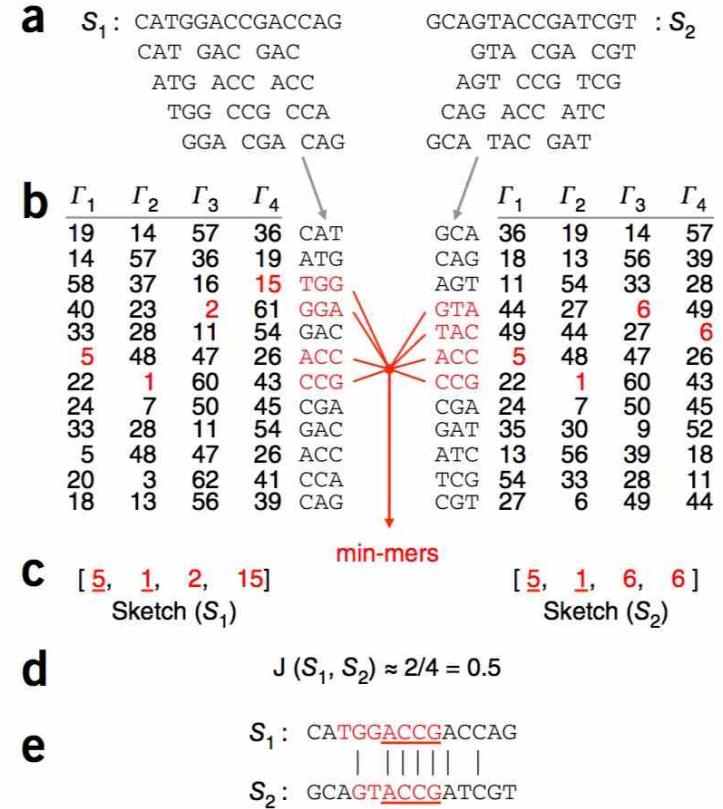
[How do we compute the overlap?]

[Do we really want to do all-vs-all?]

Very fast approximate overlapping

Maybe we don't need to compute the exact identity of the overlap region, just approximate it

- If two reads overlap, they should share many of the same kmers: Their Jaccard coefficient should be high: $|\text{intersection}| / |\text{union}|$
- But tracking all of the kmers for a read is a lot of overhead
- Instead, compare the “sketch” of the reads: a small fraction of kmers carefully chosen
- LSH: Find the sketch by applying N hash functions to the kmers, and keeping the minimum hash values reported from each ($N=4$ in example)
- This forms a nice “random” sample of the reads, and the Jaccard coefficient is a good approximation of the sequence similarity



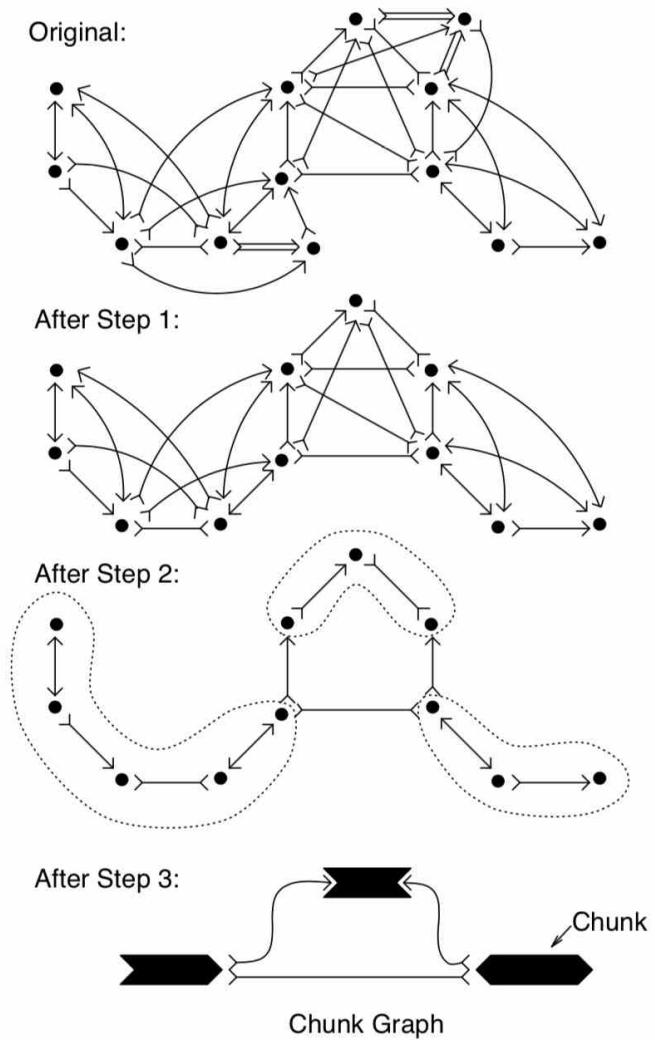
Unitigging: Pruning the Overlap Graph

The overlap graph has many redundant edges:

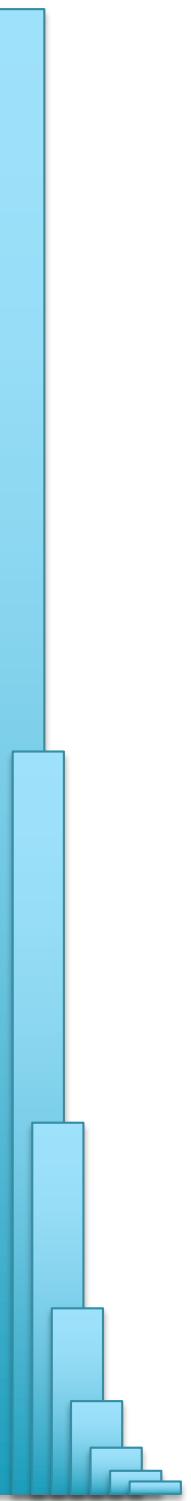
- If the average coverage is D, we should expect D overlaps at the beginning of the read, and D at the end

Transform the graph to simplify the assembly problem (without changing the valid solutions):

1. **Contained reads removal:** Short reads that are substrings of longer reads don't advance the assembly, remove those nodes and all of the edges
2. **Transitive edge removal:** If A \rightarrow B, and B \rightarrow C, remove the transitive edge A \rightarrow C
3. **“Chunkification”:** Linear subgraphs define uniquely assemblable segments: “unitigs”



Towards Simplifying and Accurately Formulating Fragment Assembly
Myers (1995) J Comput Biol. Summer;2(2):275-90.

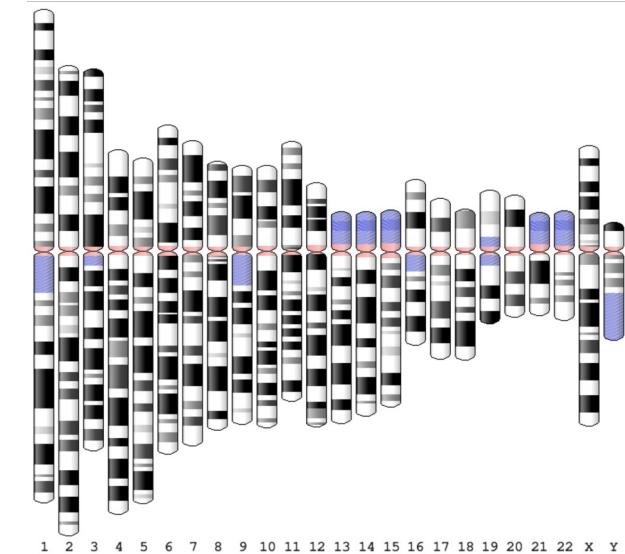


Part II: The T2T human genome

Finishing the human genome

238Mbp is missing or incorrect

- Centromeres and telomeres
- Segmentally duplicated genes
- Tandem gene arrays (e.g. rDNAs)
- And an unknown number of errors...



Why does it matter?

Variation in these regions is unexplored

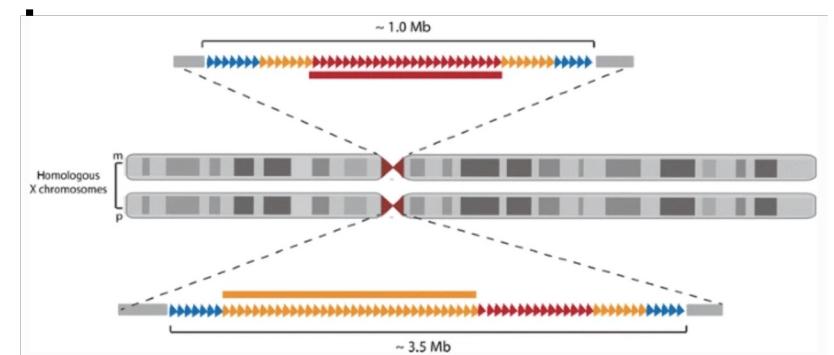
Functional studies need sequence

Reference gaps lead to artifacts

We don't know what we don't

Why has it taken so long?

Repeats, repeats, repeats...



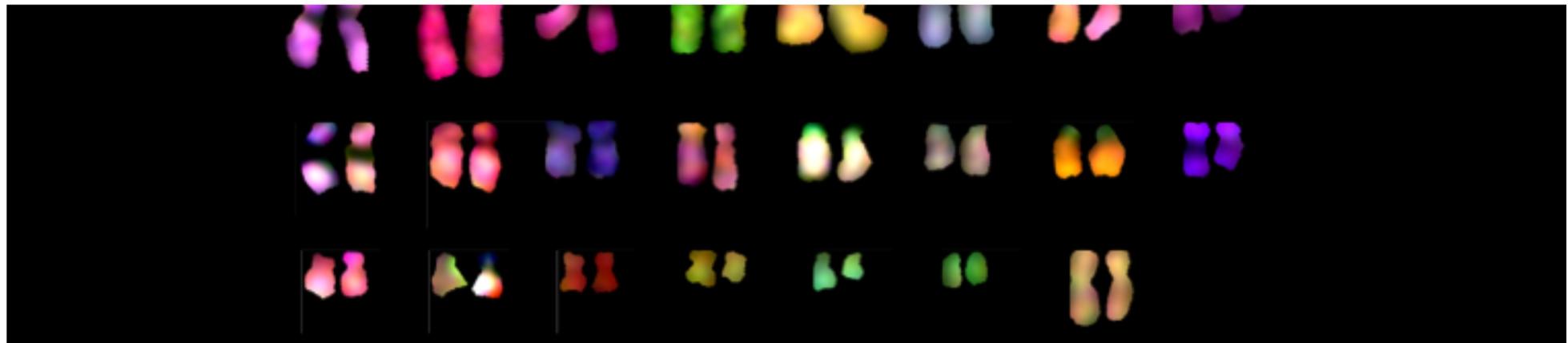
Miga 2015

Let's finish a human genome



T2T Working Group

[Home](#) · [Technology](#) · [Data](#) · [CHM13 Cell Line](#) · [Remaining Challenges](#) ▾ · [Who We Are](#) · [Join Us](#) 



The Telomere-to-Telomere (T2T) consortium is an open, community-based effort to generate the first complete assembly of a human genome.

CHM13 homozygous 46,XX cell line from Urvashi Surti, Pitt; SKY karyotype from Jennifer Gerton, Stowers



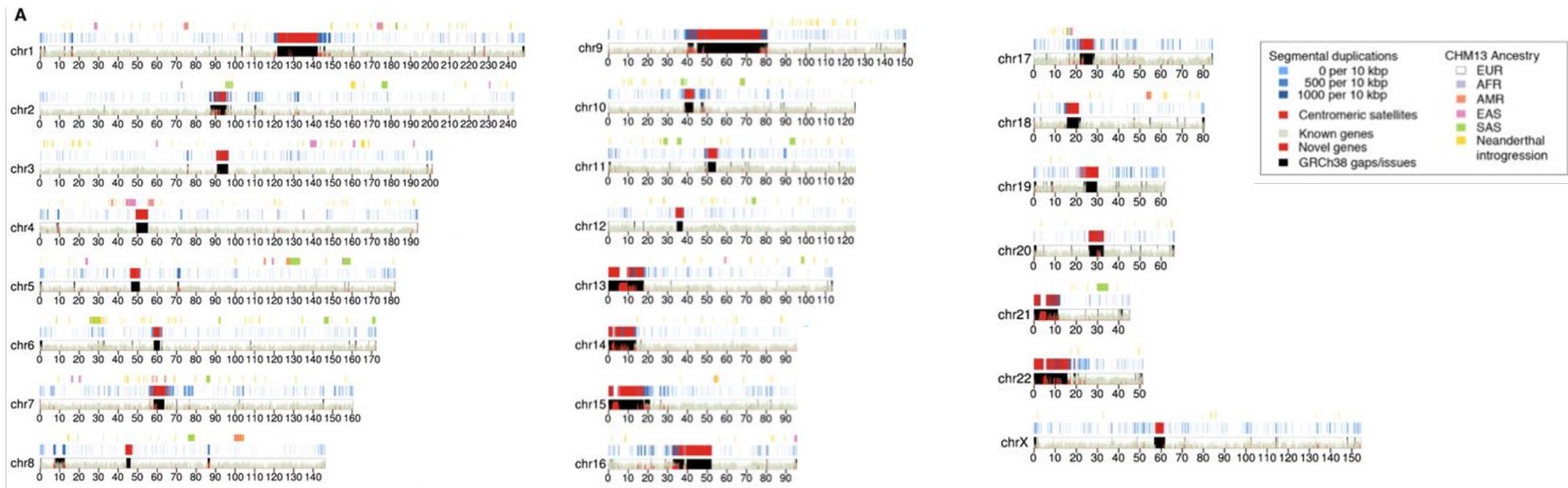
CHM13 assembly graph



HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads.

Nurk et al. *Genome Research* (2020)

The complete sequence of a human genome

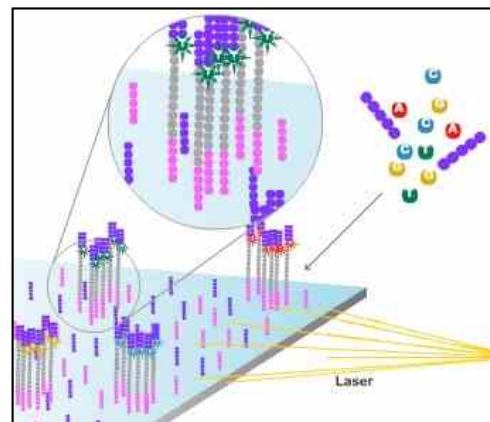
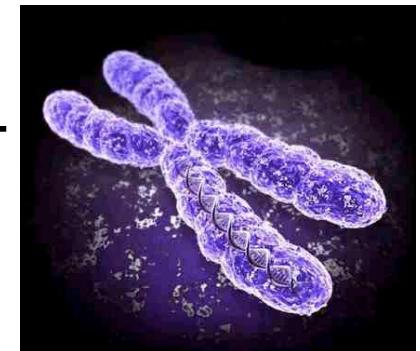
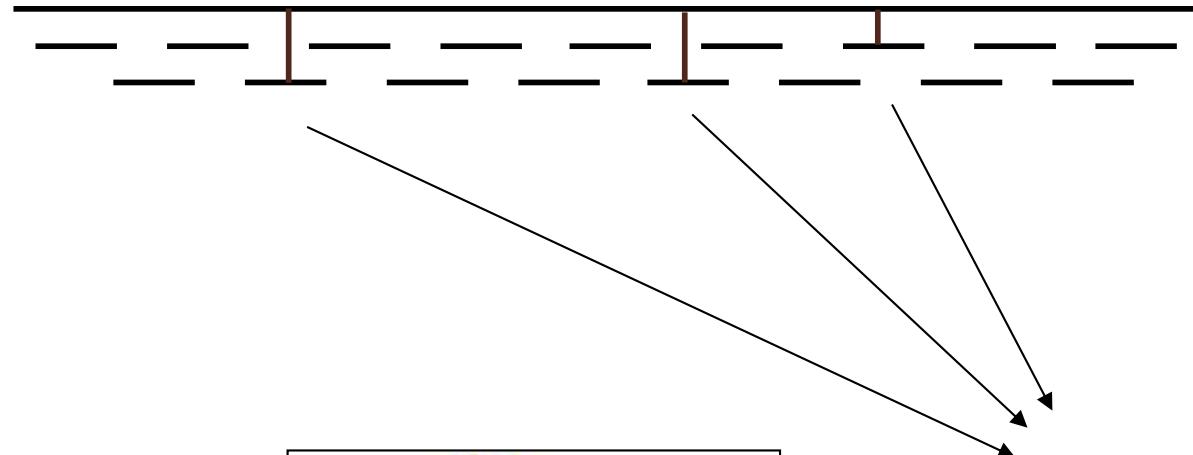


CHM13v1.1 genome size is **3.057 Gbp with zero Ns**
Every chromosome is telomere-to-telomere, quality estimated >Q70
~190 Mbp (~8%) of new sequence vs. GRCh38, fixes thousands of errors

(Nurk et al. Science, 2022)

Personal Genomics

How does your genome compare to the reference?

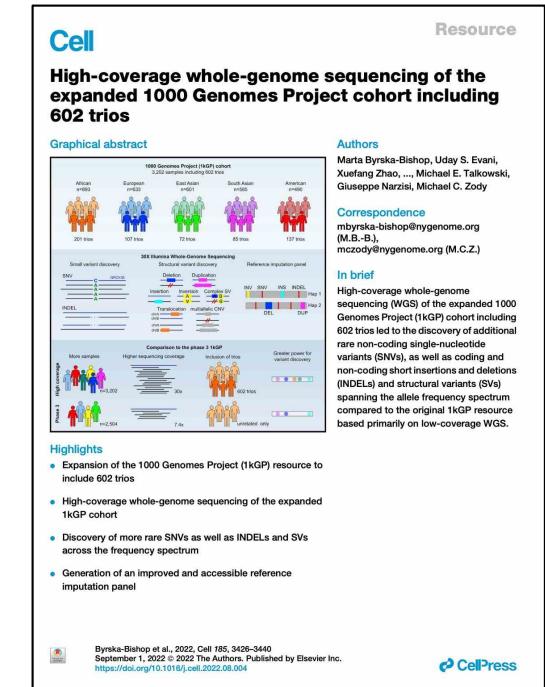


Heart Disease

Cancer

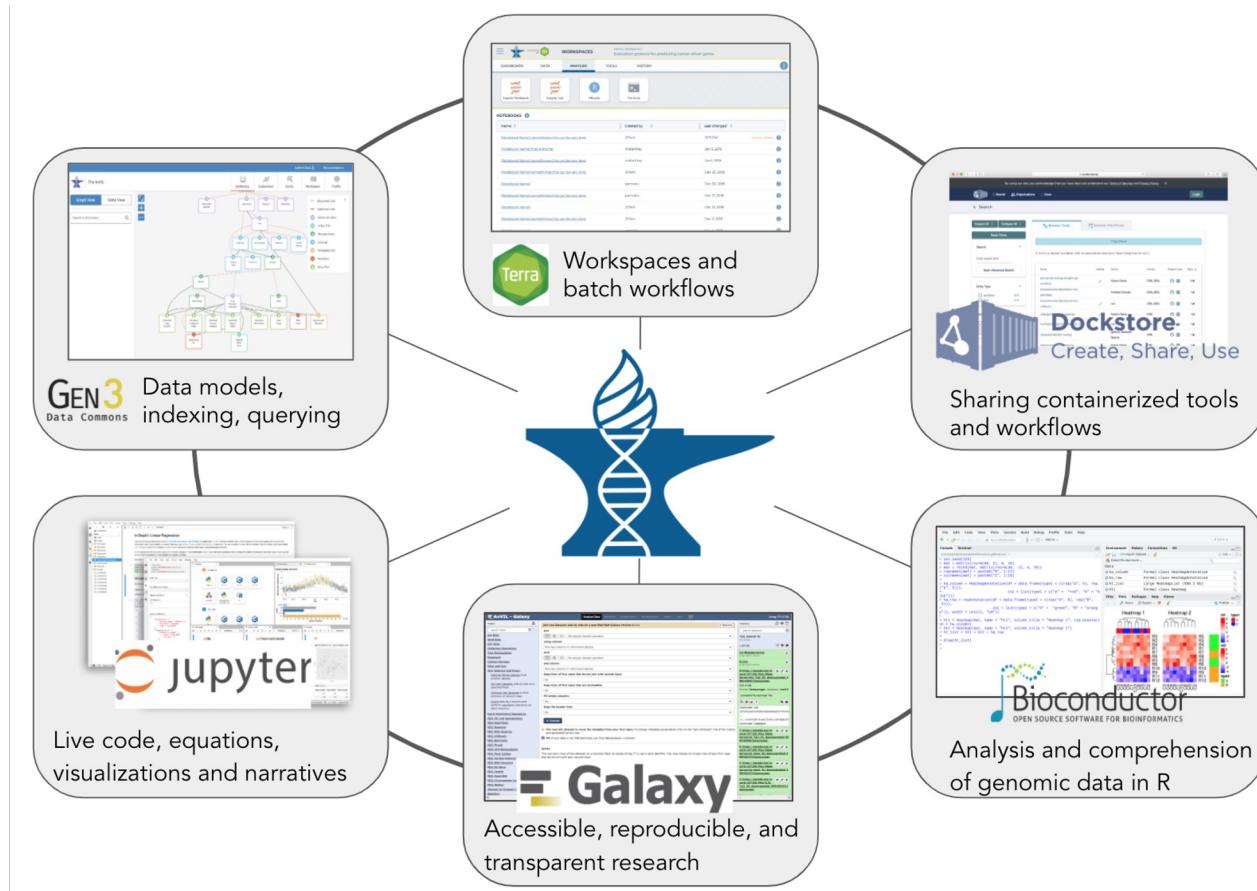
Presidential smile

T2T Variants: 1000 Genomes Project



26 populations from 5 superpopulations (continental regions)
3202 samples (2504 core genomes + 698 offspring)

3202 samples x 30Gb = 96Tb input data | >5Pb of intermediate data | >>1M core hours

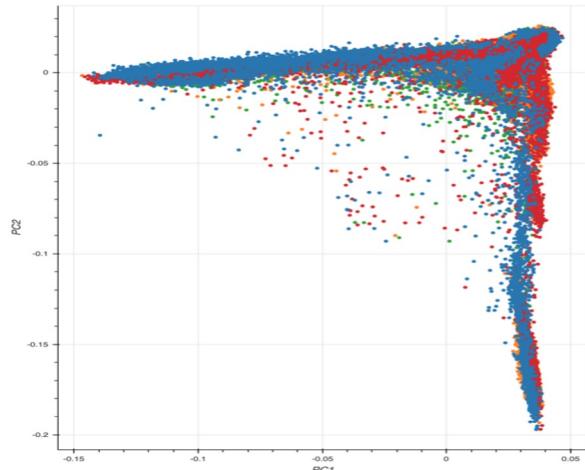


**Inverting the model of genomics data sharing with the
NHGRI Genomic Data Science Analysis, Visualization, and Informatics Lab-space (AnVIL)**
 Schatz, Philippakis et al. (2022) *Cell Genomics*. doi: <https://doi.org/10.1016/j.xgen.2021.100085>

Powered by AnVIL



Centers for
Common Disease Genomics



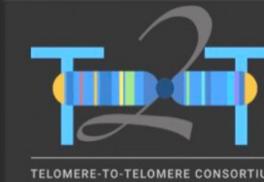
136,959 WGS samples

Broad Institute: 16,828

Baylor College: 35,493

NY Genome Center: 40,975

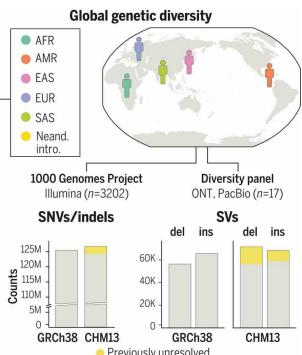
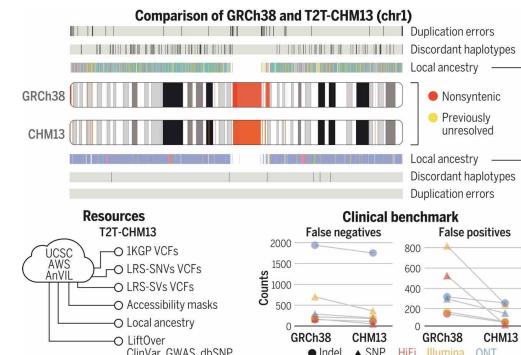
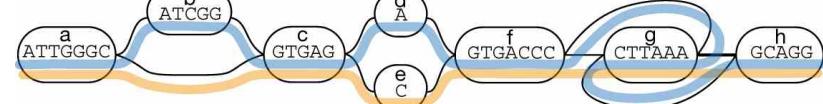
WUSTL: 43,620



TOWARDS A
COMPLETE
REFERENCE OF
HUMAN GENOME
DIVERSITY



ATTGGGCATCGGGTGAGAGTGACCCCTTAAGGCAGG
ATTGGGC - - - GTGAGCCTGACCCCTTAAAGCAGG



Nurk et al. (2022) Science
Liao et al. (2023) Nature

WDL 101: Reproducible genomics at scale



```
task samtoolsStats {
    input {
        File inputCram
        File cramIndex
        File targetRef
        String sampleName
    }

    command <<<
        samtools stats -r "~{targetRef}" \
            --reference "~{targetRef}" \
            -@ "$(nproc)" \
            "~{inputCram}" > "~{sampleName}.samtools.stats.txt"
    >>>

    Int diskGb = ceil(2.0 * size(inputCram, "G"))

    runtime {
        docker : "szarate/t2t_variants:v0.0.2"
        disks : "local-disk ${diskGb} SSD"
        memory: "12G"
        cpu : 16
        preemptible: 3
        maxRetries: 3
    }

    output {
        File stats = "~{sampleName}.samtools.stats.txt"
    }
}
```

What are the inputs?

Automatically copy from buckets to VMs

What should we run?

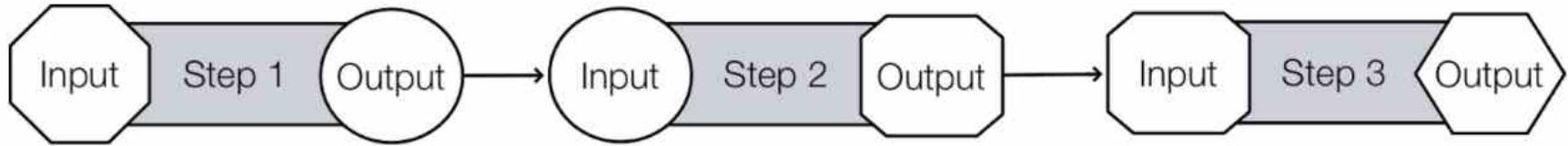
Essentially any command line tool!

What type of computers should we use?

Match resources (cores, RAM, disk) to needs

What are the outputs?

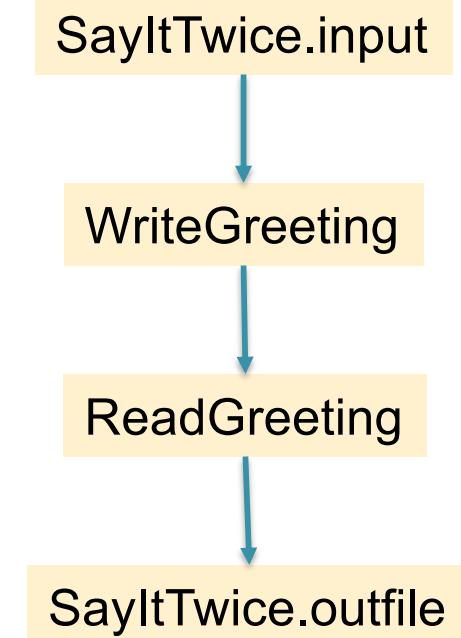
Automatically copy to buckets for next stage



```

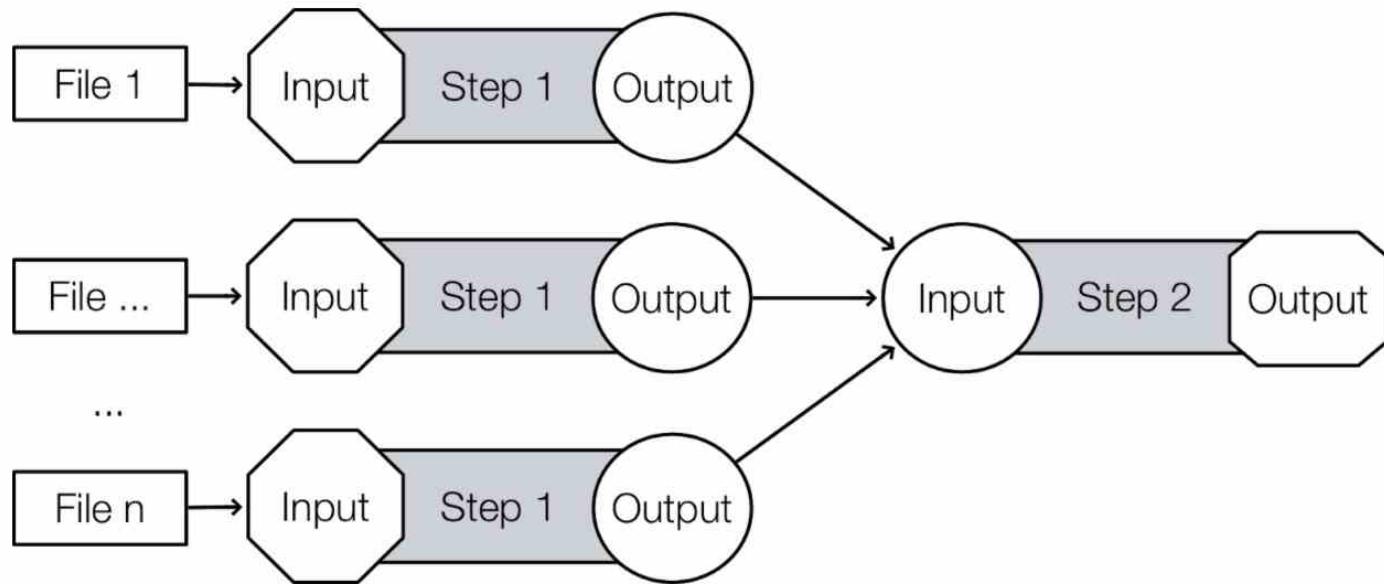
version 1.0
workflow SayItTwice {
    input {
        String name
    }
    call WriteGreeting {
        input:
            name
    }
    call ReadItBackToMe {
        input:
            written_name = WriteGreeting.output.name
    }
    output {
        File outfile = ReadItBackToMe.repeated_name
    }
}
task WriteGreeting {
    input {
        String name
    }
    command {
        echo "${name}"
    }
    output {
        File output_name = stdout()
    }
    runtime {
        docker: "ubuntu:latest"
    }
}
task ReadItBackToMe {
    input {
        File written_name
    }
    String original_name = read_string(written_name)
    command {
        echo "${original_name} to you too"
    }
    output {
        File repeated_name = stdout()
    }
    runtime {
        docker: "ubuntu:latest"
    }
}

```



https://github.com/openwdl/learn-wdl/blob/master/1_script_examples/2_language_patterns/1_linear_two_task/linear-docker.wdl

Scatter-Gather



17 lines (14 sloc) | 220 Bytes

```
1 version 1.0
2
3 workflow HelloWorld {
4     scatter(i in range(15)) {
5         call WriteGreeting
6     }
7
8 }
9
10 task WriteGreeting {
11     command {
12         echo "Hello Scattered World"
13     }
14     output {
15         File output_greeting = stdout()
16     }
17 }
```

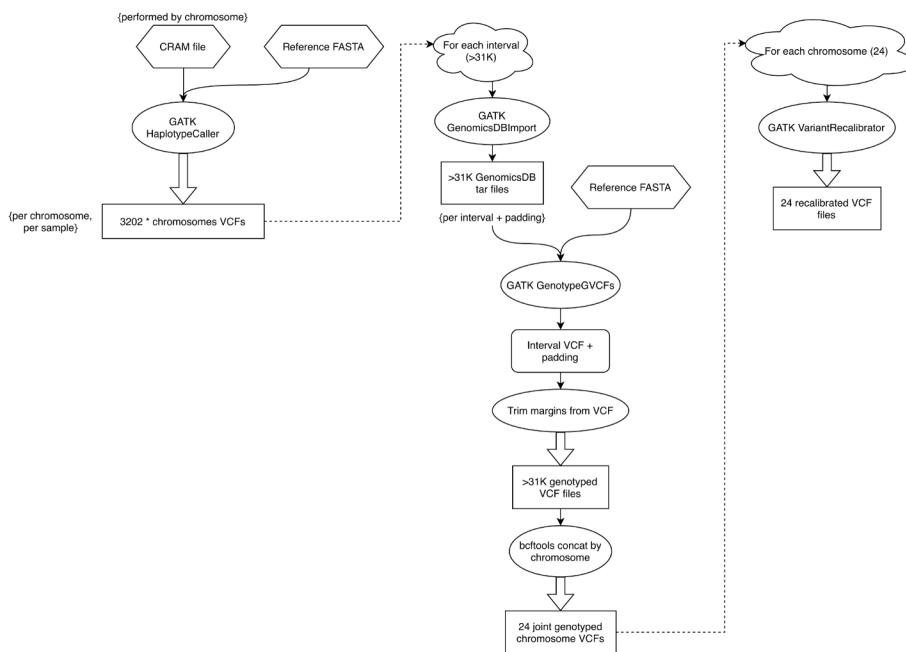
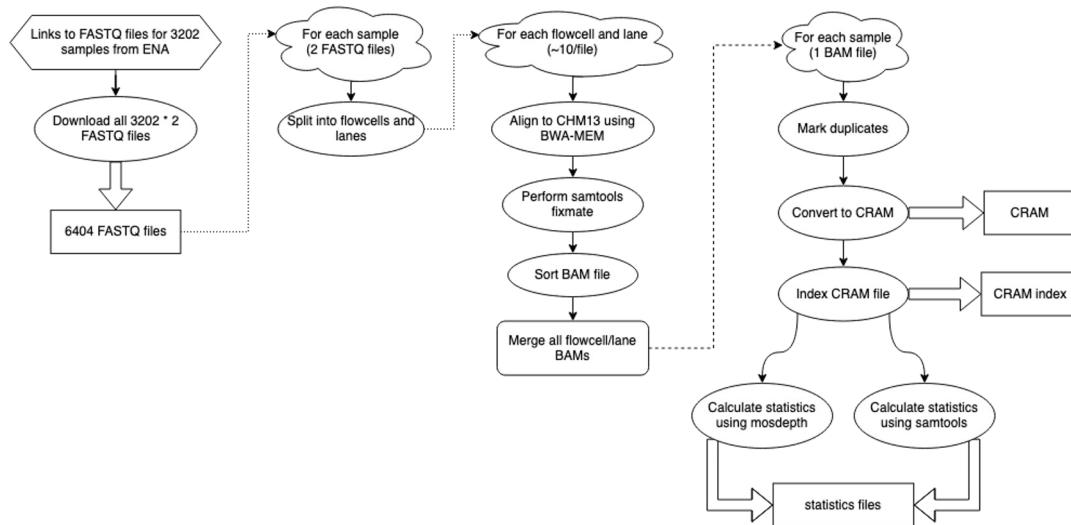
26 lines (22 sloc) | 369 Bytes

```
1 version 1.0
2
3 workflow TimingWorkflow {
4     input {
5         Int sleep_time
6     }
7     # scatter keyword parallelizes this Sleep task execution
8     scatter(i in range(15)) {
9         call Sleep { input: sleep_time = i }
10    }
11 }
12
13 task Sleep {
14     input {
15         Int sleep_time
16     }
17     command {
18         echo "I slept for ${sleep_time}"
19         Sleep ${sleep_time}
20     }
21     output {
22         String out = read_string(stdout())
23     }
24 }
25
26
```

3202 Genomes to go...



Samantha Zarate



```

task samtoolsStats {
    input {
        File inputCram
        File cramIndex
        File targetRef
        String sampleName
    }

    command <<<
        samtools stats -r "~{targetRef}" \
            --reference "~{targetRef}" \
            -@ "${nproc}" \
            "~{inputCram}" > "~{sampleName}.samtools.stats.txt"
    >>>

    Int diskGb = ceil(2.0 * size(inputCram, "G"))

    runtime {
        docker : "szarate/t2t_variants:v0.0.2"
        disks : "local-disk ${diskGb} SSD"
        memory: "12G"
        cpu : 16
        preemptible: 3
        maxRetries: 3
    }

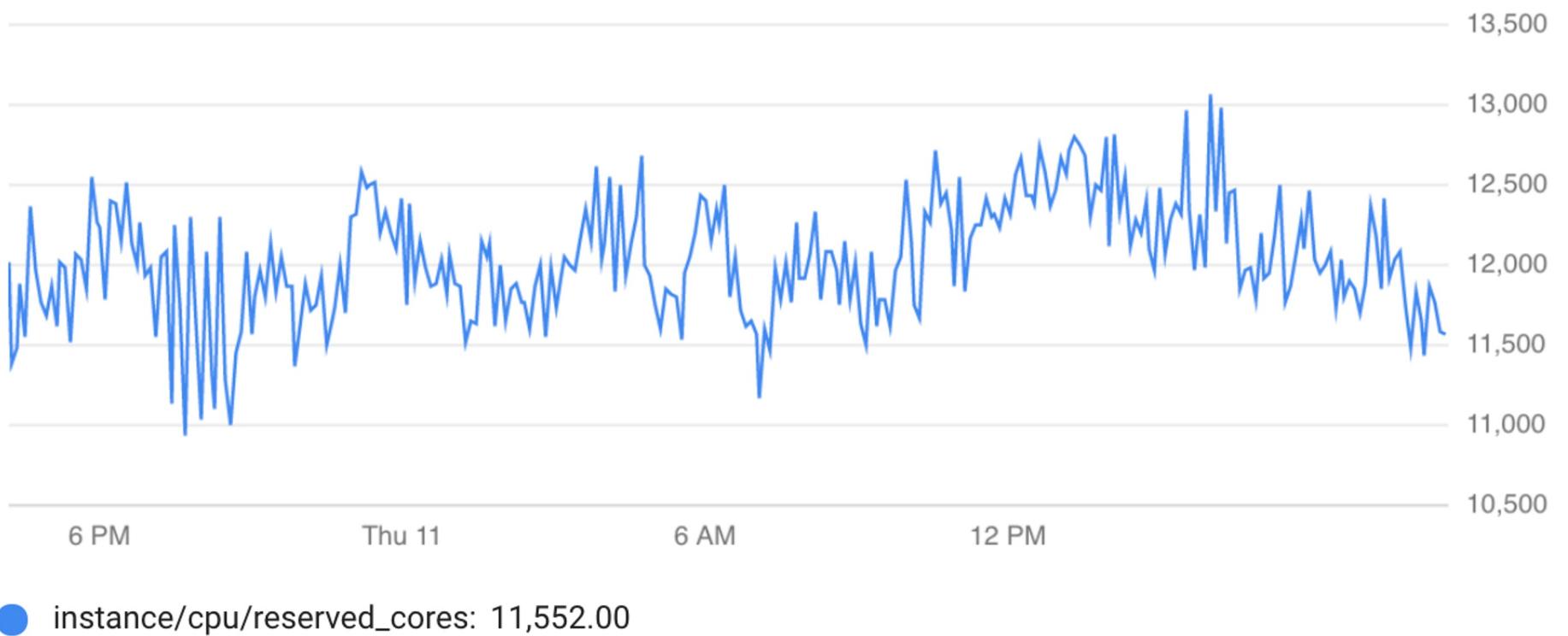
    output {
        File stats = "~{sampleName}.samtools.stats.txt"
    }
}
  
```



Core usage over 24 hours

Preview

1 hour 4 hours **1 day**





AnVIL Data Table

Screenshot of the AnVIL Data Table interface on a web browser.

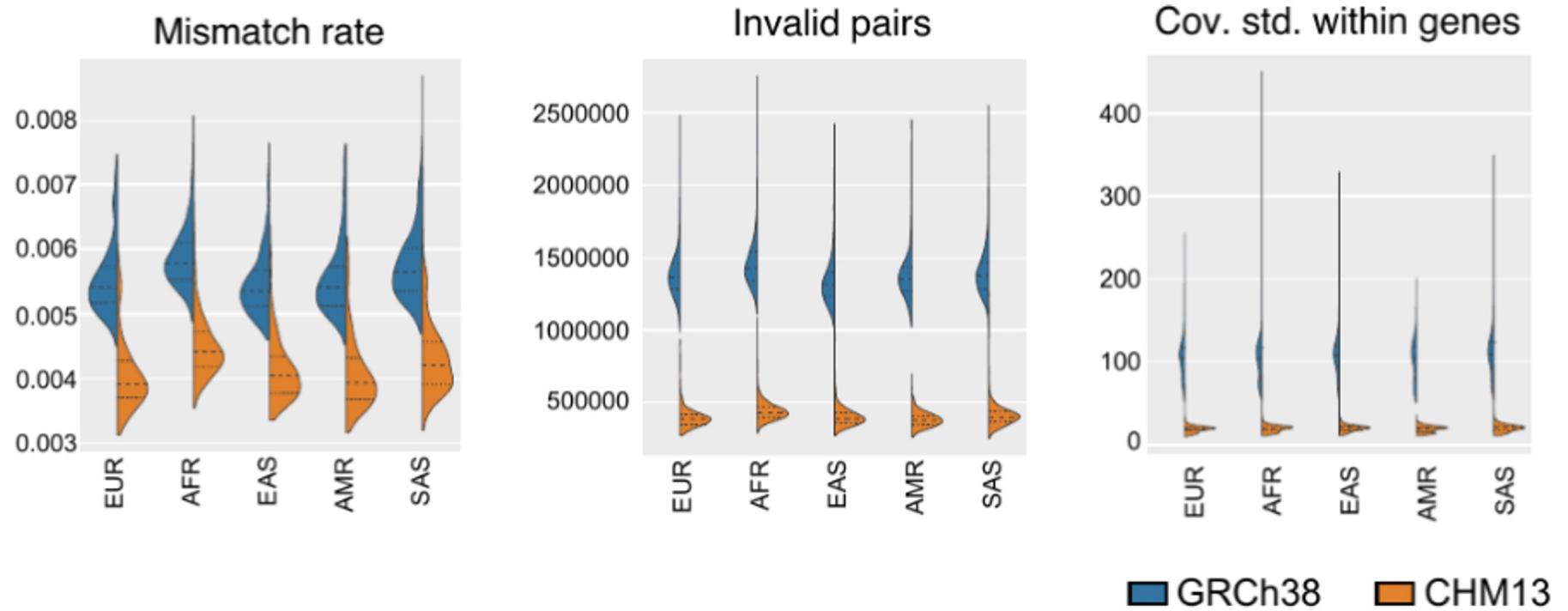
The browser address bar shows: `anvil.terra.bio/#workspaces/anvil-datastorage/AnVIL_T2T/data`

The page title is: `Workspaces > anvil-datastorage/AnVIL_T2T > Data`

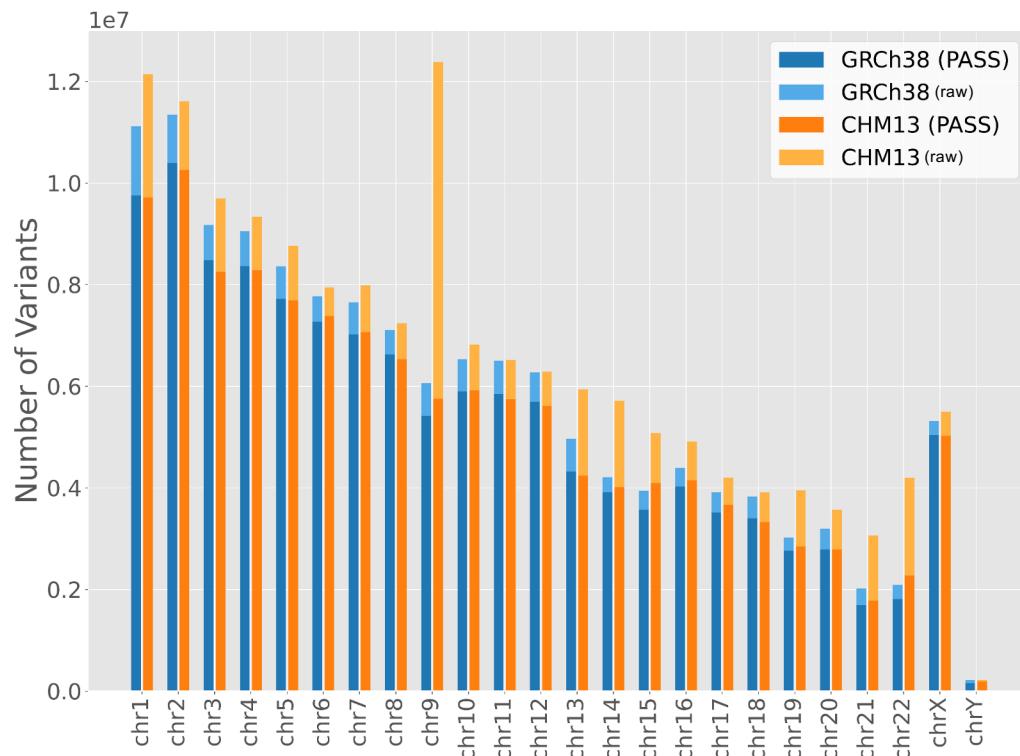
A message in the top right corner states: "Your access to NHGRI AnVIL Data Commons Framework Services has expired. Log in to restore your access or unlink your account."

The main content area displays a table titled "TABLES". The table has 18 columns and 32 rows. The columns are labeled: ena_sample..., cram, cram_index, mosdepth_global_dist, mosdepth_regions_bed, mosdepth_regions_bed_idx, mosdepth_regions_dist, mosdepth_su, HG00096, HG00096.cram.crai, HG00096.mosdepth.global.dist.txt, HG00096.regions.bed.gz, HG00096.regions.bed.gz.cs1, HG00096.mosdepth.region.dist.txt, HG00096.mosde, HG00097, HG00097.cram.crai, HG00097.mosdepth.global.dist.txt, HG00097.regions.bed.gz, HG00097.regions.bed.gz.cs1, HG00097.mosdepth.region.dist.txt, HG00097.mosde, HG00099, HG00099.cram.crai, HG00099.mosdepth.global.dist.txt, HG00099.regions.bed.gz, HG00099.regions.bed.gz.cs1, HG00099.mosdepth.region.dist.txt, HG00099.mosde, HG00100, HG00100.cram.crai, HG00100.mosdepth.global.dist.txt, HG00100.regions.bed.gz, HG00100.regions.bed.gz.cs1, HG00100.mosdepth.region.dist.txt, HG00100.mosde, HG00101, HG00101.cram.crai, HG00101.mosdepth.global.dist.txt, HG00101.regions.bed.gz, HG00101.regions.bed.gz.cs1, HG00101.mosdepth.region.dist.txt, HG00101.mosde, HG00102, HG00102.cram.crai, HG00102.mosdepth.global.dist.txt, HG00102.regions.bed.gz, HG00102.regions.bed.gz.cs1, HG00102.mosdepth.region.dist.txt, HG00102.mosde, HG00103, HG00103.cram.crai, HG00103.mosdepth.global.dist.txt, HG00103.regions.bed.gz, HG00103.regions.bed.gz.cs1, HG00103.mosdepth.region.dist.txt, HG00103.mosde, HG00105, HG00105.cram.crai, HG00105.mosdepth.global.dist.txt, HG00105.regions.bed.gz, HG00105.regions.bed.gz.cs1, HG00105.mosdepth.region.dist.txt, HG00105.mosde, HG00106, HG00106.cram.crai, HG00106.mosdepth.global.dist.txt, HG00106.regions.bed.gz, HG00106.regions.bed.gz.cs1, HG00106.mosdepth.region.dist.txt, HG00106.mosde, HG00107, HG00107.cram.crai, HG00107.mosdepth.global.dist.txt, HG00107.regions.bed.gz, HG00107.regions.bed.gz.cs1, HG00107.mosdepth.region.dist.txt, HG00107.mosde, HG00108, HG00108.cram.crai, HG00108.mosdepth.global.dist.txt, HG00108.regions.bed.gz, HG00108.regions.bed.gz.cs1, HG00108.mosdepth.region.dist.txt, HG00108.mosde, HG00109, HG00109.cram.crai, HG00109.mosdepth.global.dist.txt, HG00109.regions.bed.gz, HG00109.regions.bed.gz.cs1, HG00109.mosdepth.region.dist.txt, HG00109.mosde, HG00110, HG00110.cram.crai, HG00110.mosdepth.global.dist.txt, HG00110.regions.bed.gz, HG00110.regions.bed.gz.cs1, HG00110.mosdepth.region.dist.txt, HG00110.mosde, HG00111, HG00111.cram.crai, HG00111.mosdepth.global.dist.txt, HG00111.regions.bed.gz, HG00111.regions.bed.gz.cs1, HG00111.mosdepth.region.dist.txt, HG00111.mosde, HG00112, HG00112.cram.crai, HG00112.mosdepth.global.dist.txt, HG00112.regions.bed.gz, HG00112.regions.bed.gz.cs1, HG00112.mosdepth.region.dist.txt, HG00112.mosde, HG00113, HG00113.cram.crai, HG00113.mosdepth.global.dist.txt, HG00113.regions.bed.gz, HG00113.regions.bed.gz.cs1, HG00113.mosdepth.region.dist.txt, HG00113.mosde, HG00114, HG00114.cram.crai, HG00114.mosdepth.global.dist.txt, HG00114.regions.bed.gz, HG00114.regions.bed.gz.cs1, HG00114.mosdepth.region.dist.txt, HG00114.mosde, HG00115, HG00115.cram.crai, HG00115.mosdepth.global.dist.txt, HG00115.regions.bed.gz, HG00115.regions.bed.gz.cs1, HG00115.mosdepth.region.dist.txt, HG00115.mosde, HG00116, HG00116.cram.crai, HG00116.mosdepth.global.dist.txt, HG00116.regions.bed.gz, HG00116.regions.bed.gz.cs1, HG00116.mosdepth.region.dist.txt, HG00116.mosde, HG00117, HG00117.cram.crai, HG00117.mosdepth.global.dist.txt, HG00117.regions.bed.gz, HG00117.regions.bed.gz.cs1, HG00117.mosdepth.region.dist.txt, HG00117.mosde, HG00118, HG00118.cram.crai, HG00118.mosdepth.global.dist.txt, HG00118.regions.bed.gz, HG00118.regions.bed.gz.cs1, HG00118.mosdepth.region.dist.txt, HG00118.mosde.

1000G Mapping on T2T-CHM13

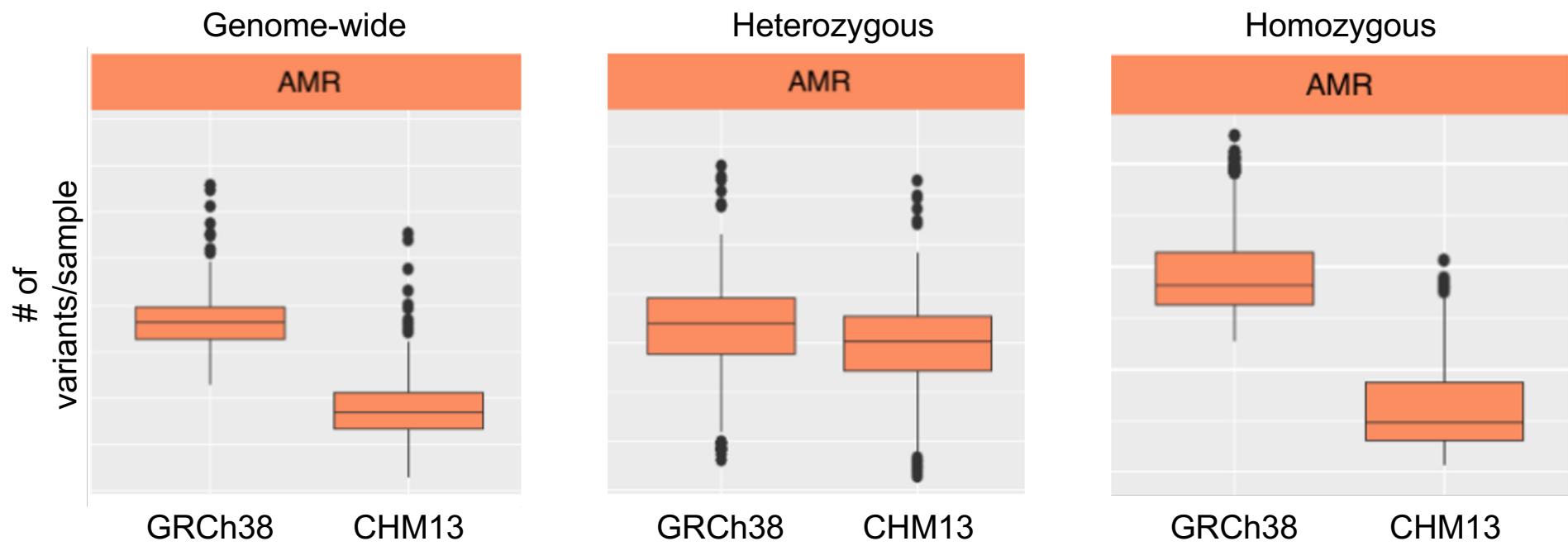


Across 1000G, Many More Variants Found Using CHM13

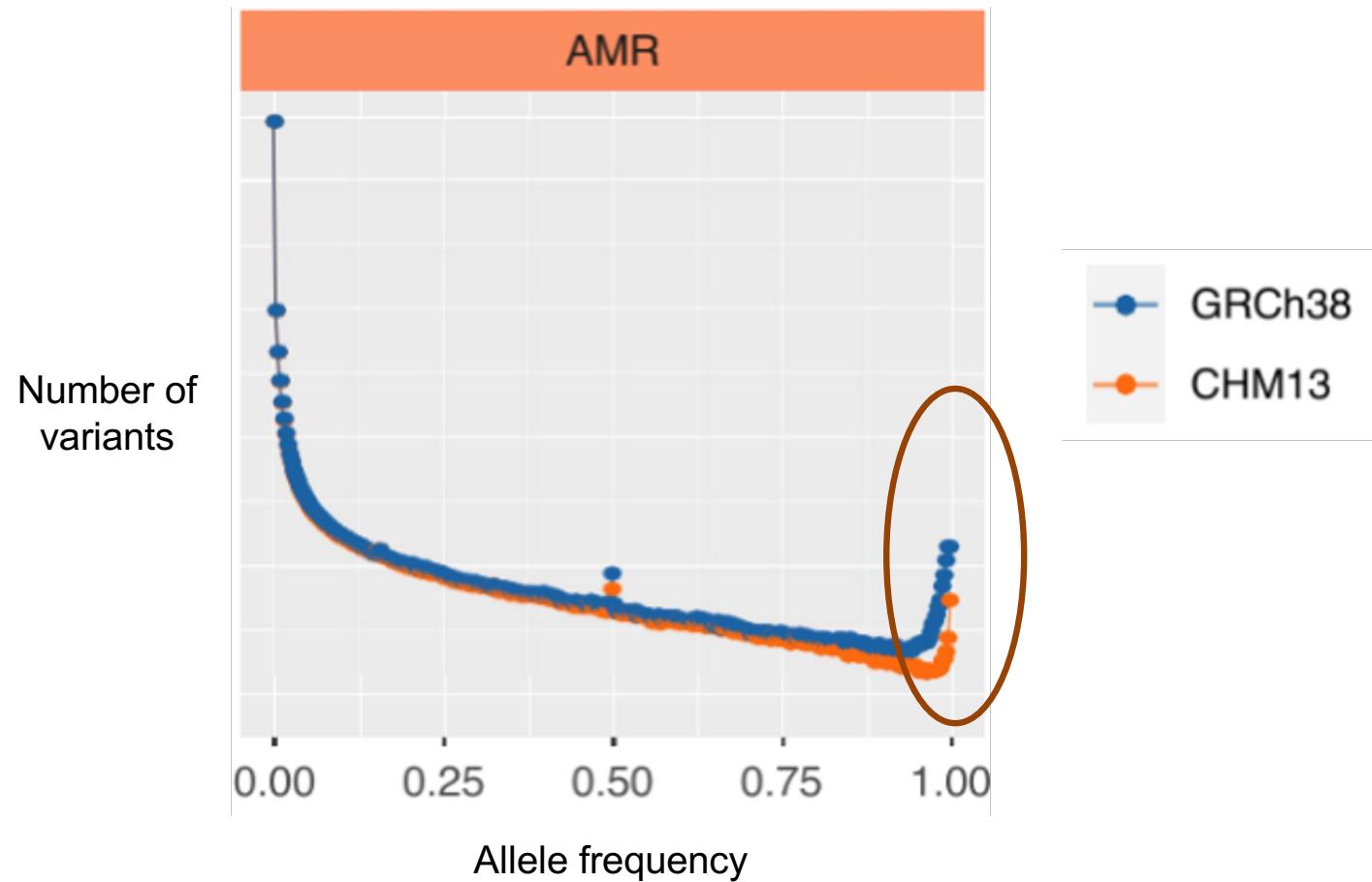


	GRCh38	CHM13
# PASS variants	125,484,020	126,591,489

1000G Per-Sample Variant Counts on T2T-CHM13



Explaining Decreased Per-Sample Count

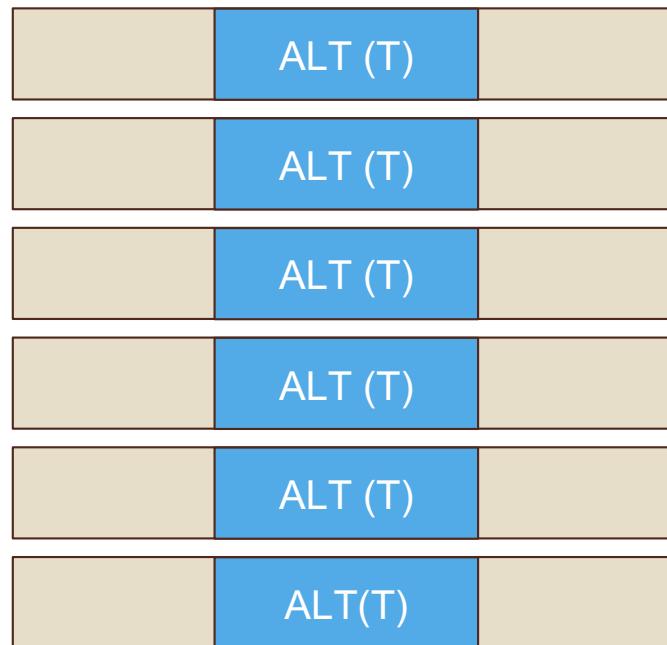


Allele Frequency = I: Reference error / private variant

GRCh38



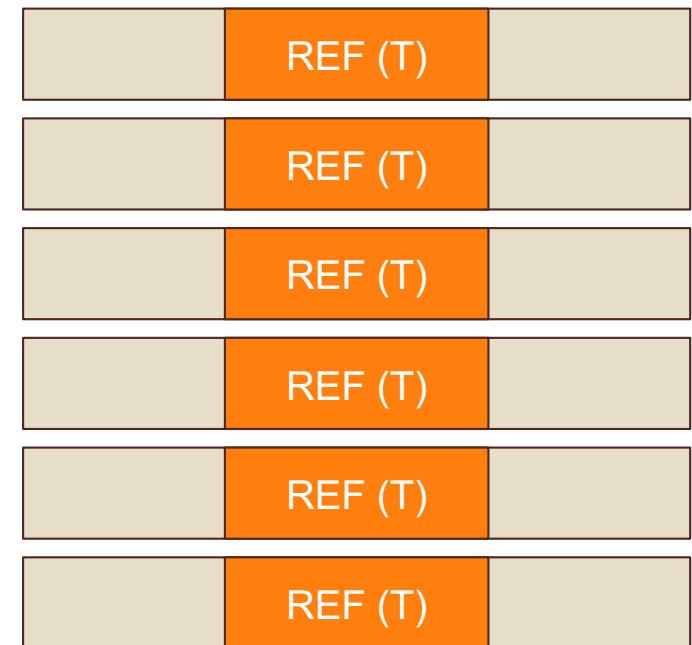
Samples



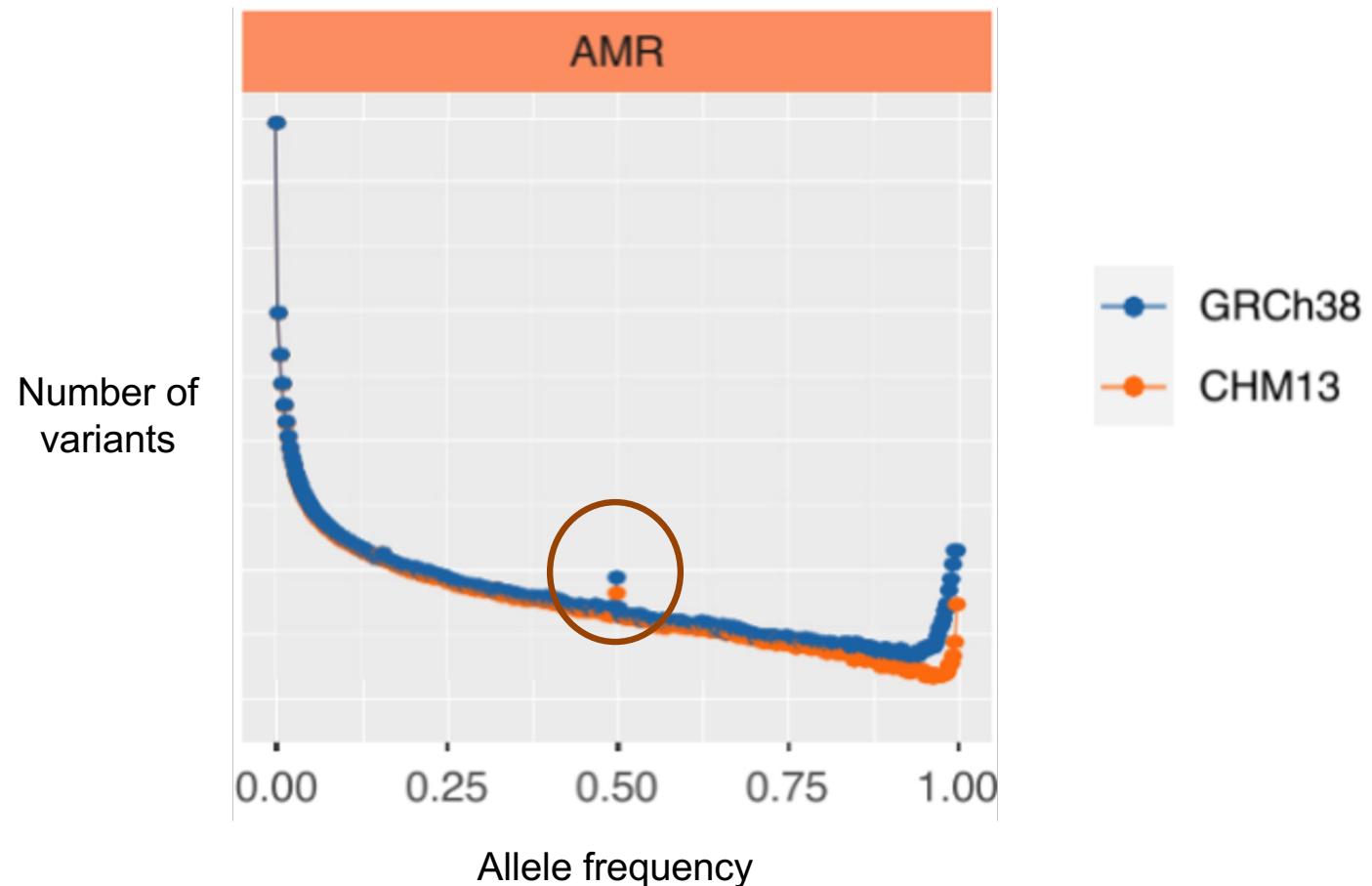
CHM13



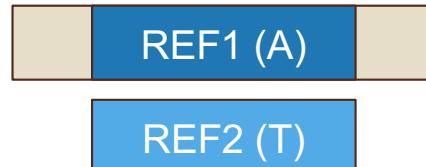
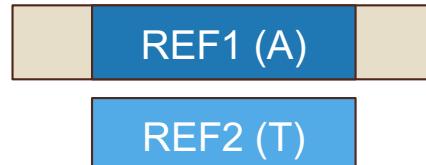
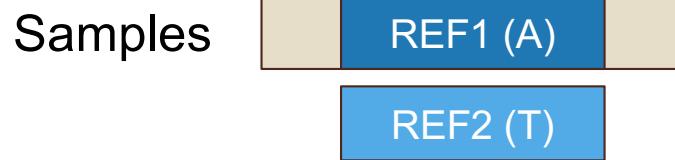
Samples



Explaining Decreased Per-Sample Count

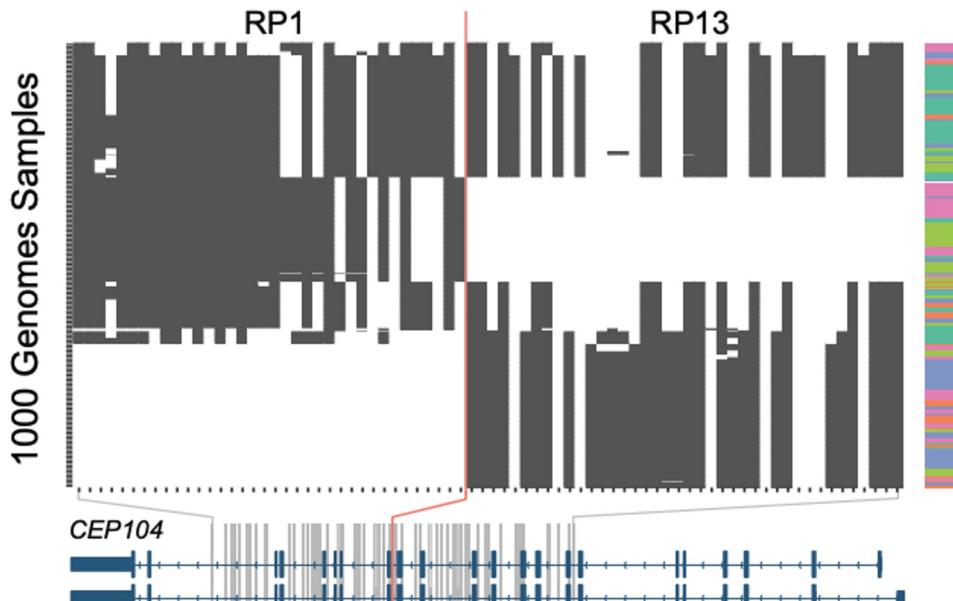


Allele Frequency \approx 0.5: Collapsed duplication

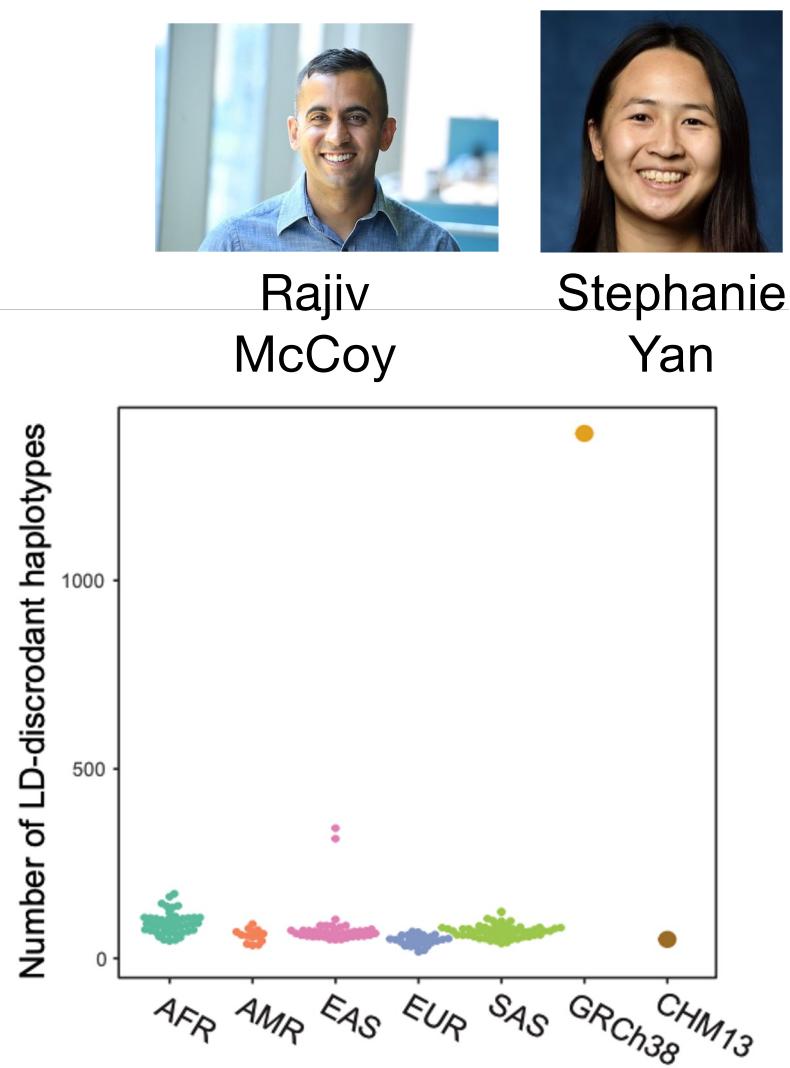


Local Ancestry Comparisons

C



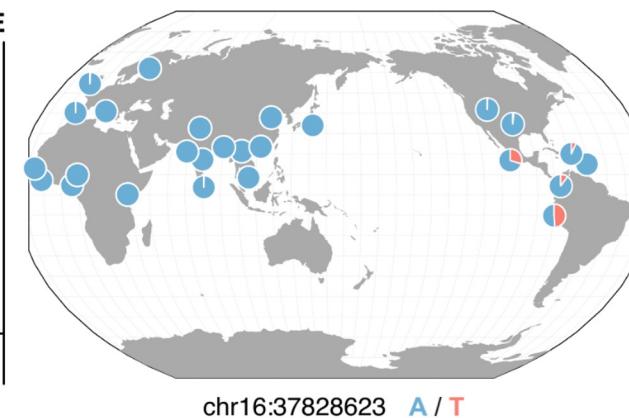
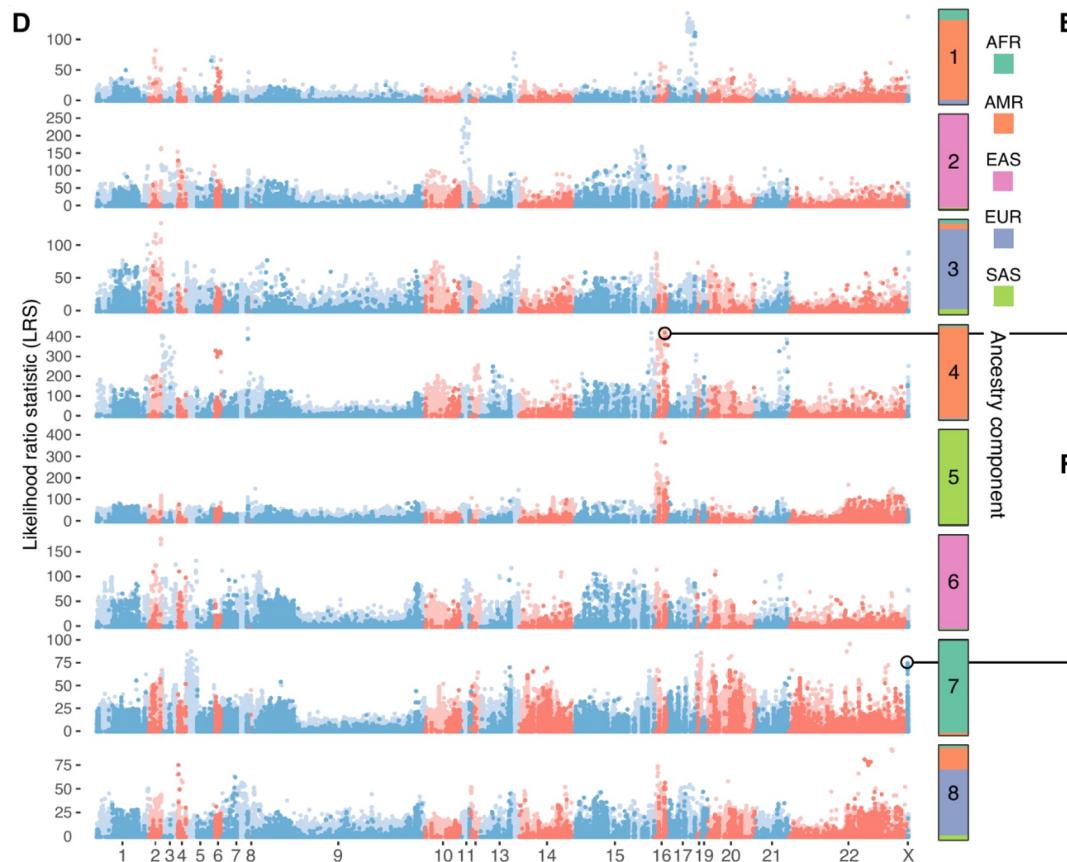
D



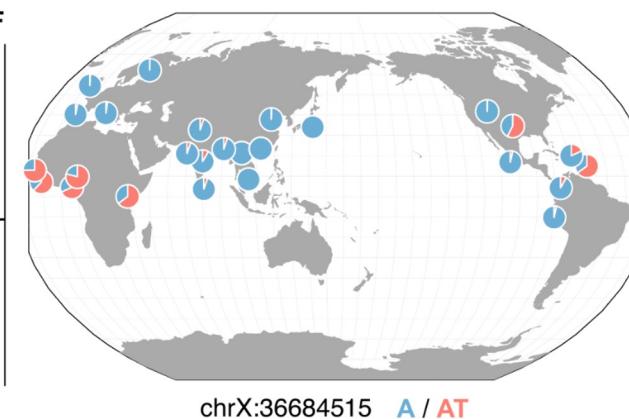
Rajiv
McCoy

Stephanie
Yan

Variants in Newly Resolved Regions

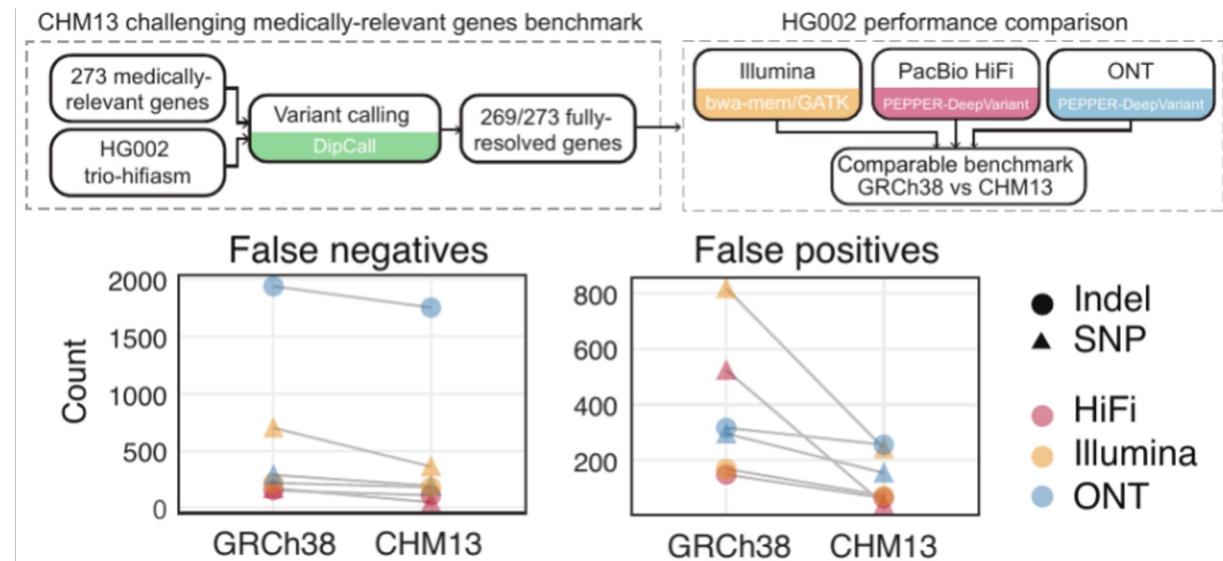


Stephanie
Yan



Rajiv McCoy

T2T-CHM13 Improves Clinical Genomics Variant Calling



Danny Miller



Daniela Soto



Megan Dennis



Justin Zook



Fritz Sedlazeck

For more information, see: (Wagner, J et al., NBT, 2022)



Learning More about WDL

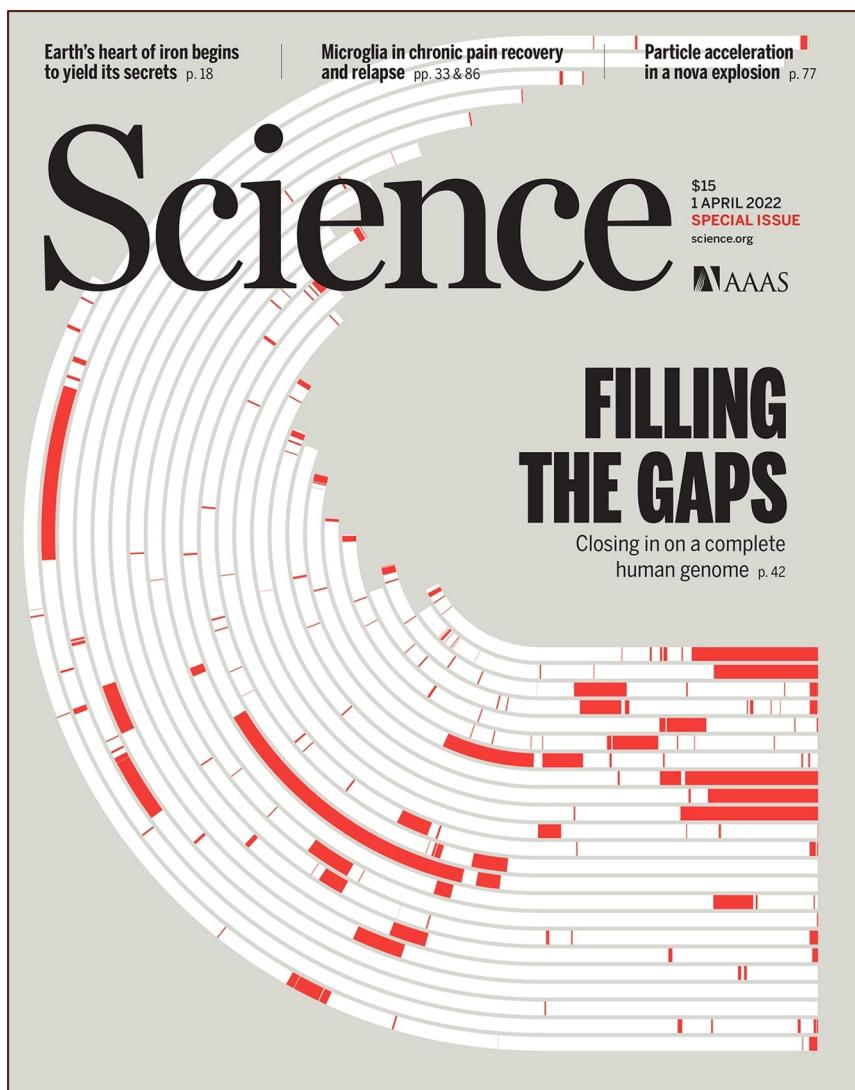
The image displays three screenshots related to WDL (Workflow Description Language) learning and tooling:

- learn-wdl GitHub README:** Shows the main README.md file with a title "learn-wdl" and a section "What is WDL?" containing a bulleted list of facts and a logo featuring a hexagon and the letters "wd".
- miniwdl GitHub README:** Shows the main README.md file for the miniwdl project, which is a local runner and developer toolkit for Python 3.6+. It includes sections for "Install miniwdl" and "About WDL script execution", along with a diagram illustrating the workflow components: client, miniwdl, Docker Hub, and optional Task Container.
- Dockstore search results:** A screenshot of the Dockstore search interface showing results for "Search: the language is WDL". The results list various WDL workflows, such as "DataBiosphere/ncap:workflows/UM_variant_caller_wdl", "DataBiosphere/ncap:workflows/UM_aligner_wdl", and "ICGC-TCGA-PanCancer/wdl-pcawg-sanger:codeWorkflow". The results are filtered by "WDL" and show details like author, verified status, and stars.

<https://github.com/openwdl/learn-wdl/>

<https://github.com/chanzuckerberg/miniwdl/>

<https://dockstore.org/>



NEWS CAREERS COMMENTARY JOURNALS ▾

HOME > COLLECTIONS > COMPLETING THE HUMAN GENOME

COMPLETING THE HUMAN GENOME

A fully sequenced human genome was announced more than 20 years ago. However, owing to technological limitations, some genomic regions remained unresolved. Here, *Science* and other journals present research by the Telomere-to-Telomere (T2T) Consortium, reporting on the endeavor to complete a comprehensive human reference genome.

FILTERS

6 RESULTS FOUND

SPECIAL ISSUE RESEARCH ARTICLE

Segmental duplications and their variation in a complete human genome

BY MITCHELL R. VOLLMER, XAVI GUITART, PHILIP C. DISHUCK, LUDOVICA MERCURI, WILLIAM T. HARVEY, ARIEL GERSHMAN, MARK DIEKHANS, ARVIS SULOVARI, KATHERINE M. MUNSON, ALEXANDRA P. LEWIS, [...] EVAN E. EICHLER

SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022

SPECIAL ISSUE RESEARCH ARTICLE

Complete genomic and epigenetic maps of human centromeres

BY NICOLAS ALTEMOSSE, GLENNIS A. LOGSDON, ANDREY V. BZIKADZE, PRAGYA SIDHWANI, SASHA A. LANGLEY, GINA V. CALDAS, SAVANNAH J. HOYT, LEV URALSKY, FEDOR D. RYABOV, COLIN J. SHEW, [...] KAREN H. MIGA

SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022

SPECIAL ISSUE RESEARCH ARTICLE

From telomere to telomere: The transcriptional and epigenetic state of human repeat elements

BY SAVANNAH J. HOYT, JESSICA M. STORER, GABRIELLE A. HARTLEY, PATRICK G. S. GRADY, ARIEL GERSHMAN, LEONARD G. DE LIMA, CHARLES LIMOUSE, REZA HALABIAN, LUKE WOJENSKI, MATIAS RODRIGUEZ, [...] RACHEL J.

+16 authors SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022

SPECIAL ISSUE RESEARCH ARTICLE

A complete reference genome improves analysis of human genetic variation

BY SERGEY AGANEZOV, STEPHANIE M. YAN, DANIELA C. SOTO, MELANIE KIRSCH, SAMANTHA ZARATE, PAVEL AVDEYEV, DYLAN J. TAYLOR, KISHWAR SHAFIN, ALAINA SHUMATE, CHUNLIN XIAO, [...] MICHAEL C. SCHATZ

SCIENCE • VOL. 376, NO. 6588 • 01 APR 2022

SPECIAL ISSUE RESEARCH ARTICLE

Epigenetic patterns in a complete human genome

BY ARIEL GERSHMAN, MICHAEL E. G. SAURIA, XAVI GUITART, MITCHELL R. VOLLMER, PAUL W. HOOK, SAVANNAH J. HOYT, MITEN JAIN, ALAINA SHUMATE, ROHAM RAZAGHI, SERGEY KOREN, [...] WINSTON TIMP

VOL. 376, NO. 6588 • 01 APR 2022

SPECIAL ISSUE RESEARCH ARTICLE

The complete sequence of a human genome

BY SERGEY NURK, SERGEY KOREN, ARANG RHIE, MIKKO RAUTIAINEN, ANDREY V. BZIKADZE, ALLA MIKHEENKO, MITCHELL R. VOLLMER, NICOLAS ALTEMOSSE, LEV URALSKY, ARIEL GERSHMAN, [...] ADAM M. PHILLIPPI

SCIENCE • VOL. 376, NO. 6588 • 31 MAR 2022 • 44:53

LOG IN BECOME A MEMBER

T2T Team @ Santa Cruz, August 2022

