Human Genome Variation Lab

Table of Contents

# 1 Homepage

This is the course homepage for Human Genome Variation with Computational Lab (AS.020.321).

#### 1.0.0.1 Instructor

Rajiv McCoy, rajiv.mccoy[at]jhu.edu

#### 1.0.0.2 Schedule & Logistics

Class is **Tuesdays from 3-3:50PM**, in **UTL G89**.

Please bring your laptop with you to every class. If you don’t have a working laptop, contact me so that we can find a solution.

| Session | Content |
| --- | --- |
| **Session 1:** *January 24, 2023* | The reference genome Genome browsers |
| **Session 2:** *January 31, 2023* | De novo mutations Data exploration and plotting in R |
| **Session 3:** *February 7, 2023* | De novo mutations Data exploration and plotting in R |
| **Session 4:** *February 14, 2023* | Linkage disequilibrium |
| **Session 5:** *February 21, 2023* | Population structure – part I |
| **Session 6:** *February 28, 2023* | Population structure – part II |
| **Session 7:** *March 7, 2023* | Genome-wide association studies – part I |
| **Session 8:** *March 14, 2023* | Genome-wide association studies – part II |
|  | *Spring break – no class* |
| **Session 9:** *March 28, 2023* | Scans for selection – part I |
| **Session 10:** *April 4, 2023* | Scans for selection – part II |
| **Session 11:** *April 11, 2023* | Simulating evolution – part I |
| **Session 12:** *April 18, 2023* | Simulating evolution – part II |
| **Session 13:** *April 25, 2023* | Archaic introgression |

## 1.1 Course Description

The course on Human Genome Variation has exposed you to the power of genomic studies for understanding human evolutionary history as well as revealing the genetic basis of human traits and disease.

What does real human genomic data look like? How are these data analyzed in practice? Supplementing the main course, this lab module will explore public datasets and computational tools used to analyze human genomic data to better understand how patterns in these data can be used to test hypotheses about evolution and human phenotypes.

#### 1.1.0.1 Educational Objectives

* Explore the ways in which human genomic data is generated, encoded, summarized, and visualized.
* Develop an awareness of potential confounding factors and approaches by which they can be overcome.
* Establish familiarity working with summarized forms of genomic data in R, as well as resources for further independent learning.

## 1.2 Assessment & Grading

All students will conduct mid-term and final self-evaluations and use these evaluations for self-grading. The instructor will provide individual feedback on these evaluations.

Final grades will be determined either through:

1. A comprehensive final exam that is graded by the instructors
2. A self-grade that is based upon criteria set forth in your self-evaluations.

To qualify for self-grading, students must demonstrate participation in 12 out of 13 class sessions. Participation can be demonstrated by in-person class attendance or, if absent from class, via evidence of work on the Posit Cloud workspace associated with a particular week.

#### 1.2.0.1 Homework

There are required and optional homework assignments for each topic we cover.

## 1.3 Posit Cloud

All the coding and lectures for this class are conducted on Posit Cloud.

Posit is a computing environment for **R**, one of the most widely used coding languages in genomics and statistics.

**Posit Cloud** is an online version of the Posit environment. This cloud workspace avoids us having to install R on everyone’s computers, and also allows us to upload and share data files for class.

#### 1.3.0.1 Making a Posit Cloud account

Go to the [Posit Cloud website](https://posit.cloud/) and sign up for an account, using your JHU email.

# 2 Genome browsers

In this module, we’ll learn how to use the **UC Santa Cruz (UCSC) Genome Browser** and the **Integrative Genomics Viewer (IGV)**, two extremely popular tools for visualizing genomic data.

#### 2.0.0.1 Learning objectives

After completing this chapter, you’ll be able to:

1. Explain why a reference genome is an important resource for genomics research.
2. Use the UCSC genome browser to find genomic features in a region of interest.
3. Describe the data contained in a file of sequencing reads.
4. Load and interpret sequencing data in IGV.

## 2.1 DNA sequencing data

These days, the vast majority of genomic data is generated through **high-throughput Illumina short-read sequencing**. The broad steps of this sequencing process are:

* Extract DNA
* Fragment DNA
* Prepare for sequencer (add adapters, etc.)
* Amplify DNA
* Sequencing (add fluorescently labeled nucleotides that are read by a digital camera)

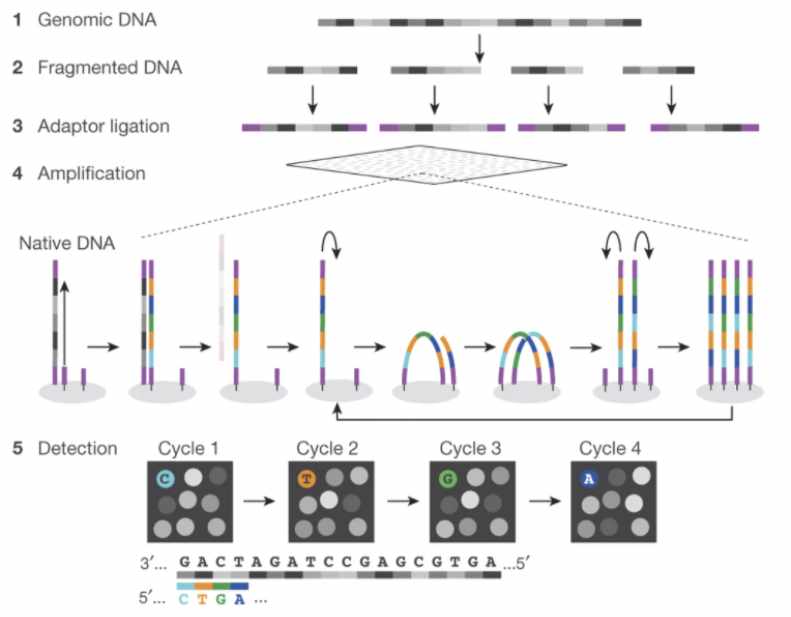


Figure : **Fig. 1 (**[**source**](https://www.researchgate.net/figure/Schematics-of-the-Illumina-and-ONT-sequencing-procedure-A-Illumina-sequencing-1-2_fig1_364792307)**).** Schematic of Illumina short-read sequencing.

This sequencing approach is fast and cost efficient, but introduces two main limitations.

1. Because of the fragmentation step, the resulting sequencing reads are extremely short (~150 bp).
2. We don’t know where in the genome the sequencing reads came from. (This is a limitation common to nearly every sequencing experiment.)

## 2.2 Assembling a genome

When the human genome is 3 billion base pairs long, assembling short sequencing reads into a full genome is a major computational challenge.

How is genome assembly performed?

We can combine sequencing reads that partially overlap with each other into longer sequences.

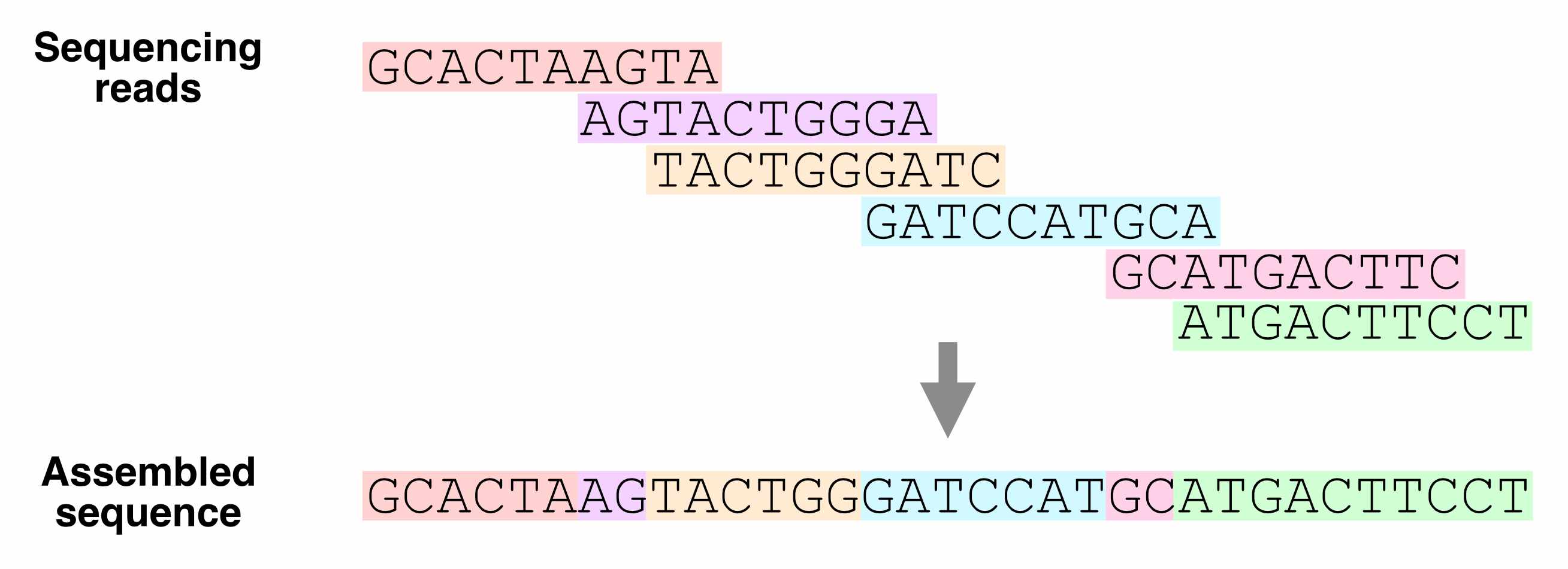


Figure : **Fig. 2.** Using overlapping sequencing reads to assemble a genome.

Which regions of the genome are hardest to assemble?

Ideally, with enough sequencing data, we would be able to reconstruct an entire genome from overlapping reads. In practice, genome assembly is complicated by **repetitive DNA** – sequences in different regions of the genome that are completely or nearly identical.

These repeats make it difficult (or impossible) to determine the order of the sequences around them, or how many copies of the repeat there are.

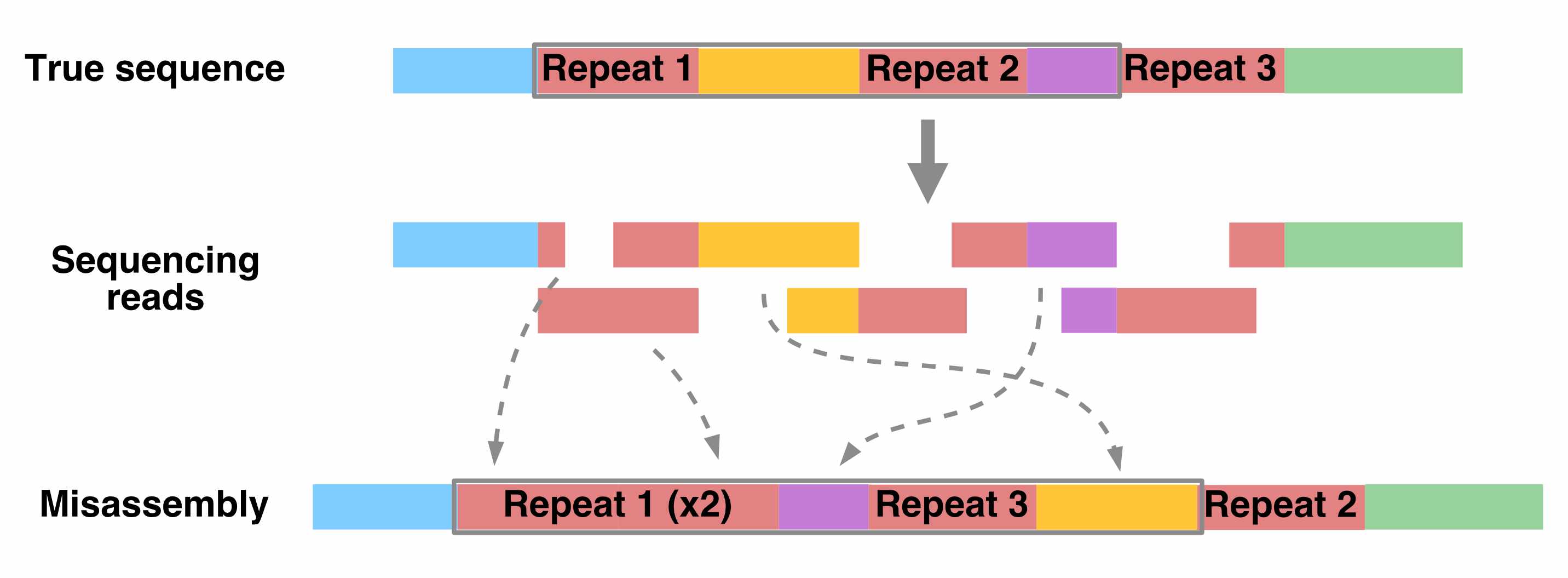


Figure : **Fig. 3.** How repetitive sequences affect genome assembly.

Resolving repetitive regions requires sequencing reads that are longer than the repeat itself, which allow us to determine the flanking sequences on the sides of the repeat.

Using such long-read sequencing technology (i.e., [PacBio](https://www.pacb.com/technology/hifi-sequencing/) and [Nanopore](https://nanoporetech.com/applications/dna-nanopore-sequencing) sequencing), the [Telomere-to-Telomere consortium](https://www.science.org/doi/10.1126/science.abj6987) was able to create a complete, ungapped assembly of the human genome in 2021.

## 2.3 The human reference genome

Having to assemble an entire genome every time you sequence a new individual is a hassle (and often infeasible, if you don’t have enough sequencing data). Instead, we typically align sequencing reads to a **reference genome** – a high-quality genome assembly for that species, which we use to guide our analysis.

The human reference genome was initially assembled in 2000 by the [Human Genome Project](https://www.genome.gov/human-genome-project), and has undergone decades of refinement since. The current version of this reference, which we’ll be using, is **hg38**.

Whose DNA was sequenced for the human reference genome?

DNA from multiple individuals was sequenced to construct the reference genome. Its sequence is a mosaic of these individuals’ DNA.

You can classify the ancestry of different parts of hg38 by comparing its sequence to DNA from different populations. From this, we know that around 70% of hg38 comes from one individual, called RP11, who likely had African American ancestry.

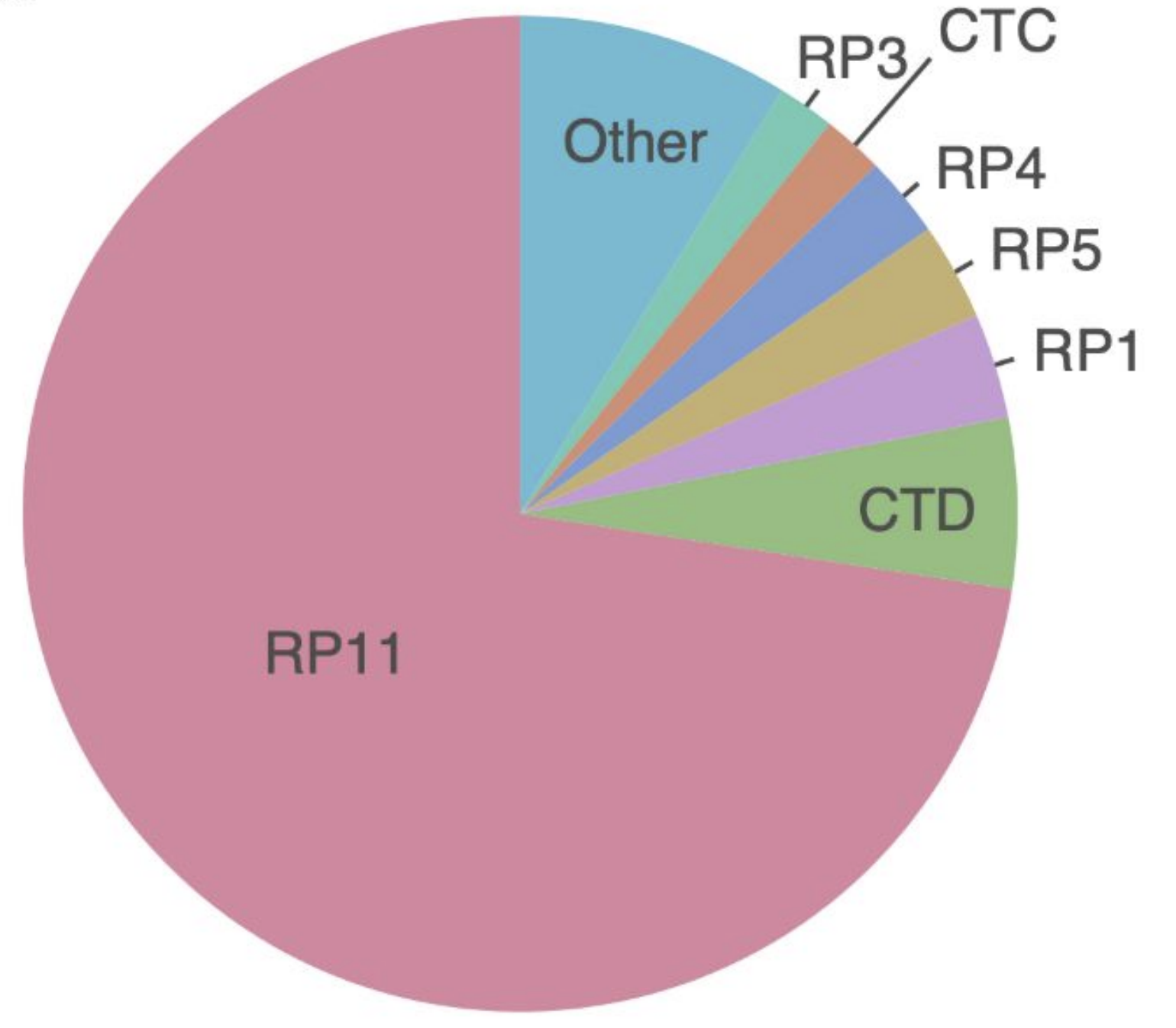


Figure : **Fig. 4.** Sample composition of the human reference genome.

## 2.4 UCSC genome browser

What does the human reference genome actually look like? We can view it in the **UC Santa Cruz (UCSC) genome browser**, an interactive website for viewing genomes – both the human reference and reference genomes for several other species.

The browser also displays **genome annotations**, such as the locations of genes and clinically relevant genetic variants.

Go to the [**UCSC genome browser**](https://genome.ucsc.edu/cgi-bin/hgGateway?hgsid=1537534853_Fxba671lul6Qu1chkXaoY68Kr3lP).

### 2.4.1 Homepage

There are a few key areas of this page:

* Browse/Select Species – choose the species
* Human Assembly – choose the version of the human reference genome
* Position/Search Term – type in a specific position (ex: chr2:25160915), region (ex: chr1:100000-200000), or gene name (ex: HLA-A)

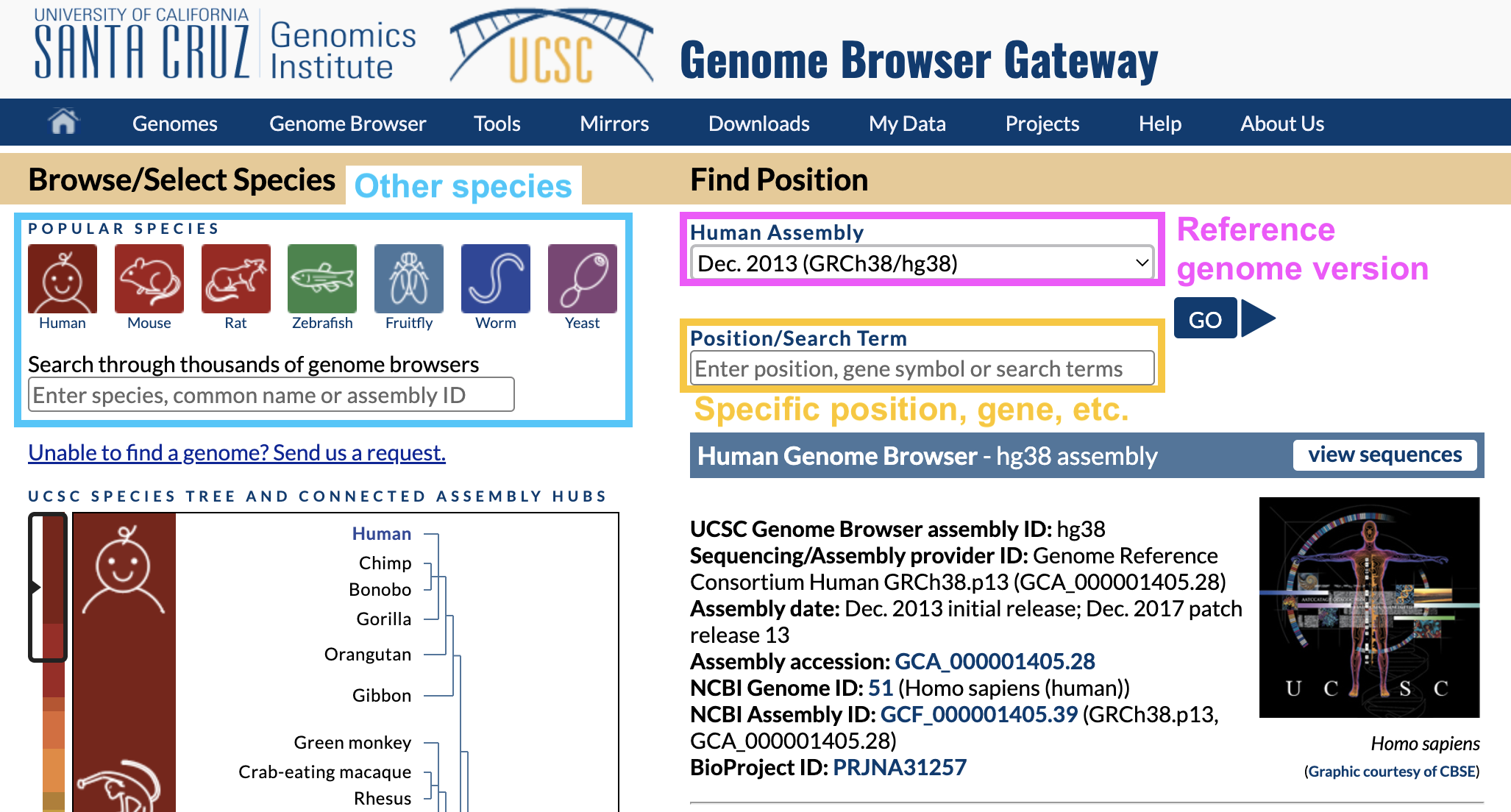


Figure : **Fig. 5.** UCSC genome browser homepage.

In Position/Search Term, type in a gene you’re interested in and hit enter.

Why are there different versions of the reference genome?

You may have noticed that the name of the reference genome we’re using is **GRCh38** or **hg38**, which stands for Genome Reference Consortium Human Build 38 – version 38 of the reference genome.

Over time, the Genome Reference Consortium makes improvements to the reference genome by closing gaps, fixing problems, and resolving repetitive regions. hg38, the most recent version, was released in 2013.

## 2.5 Viewing one region of the genome

Once you hit enter, you should end up on a page like this:

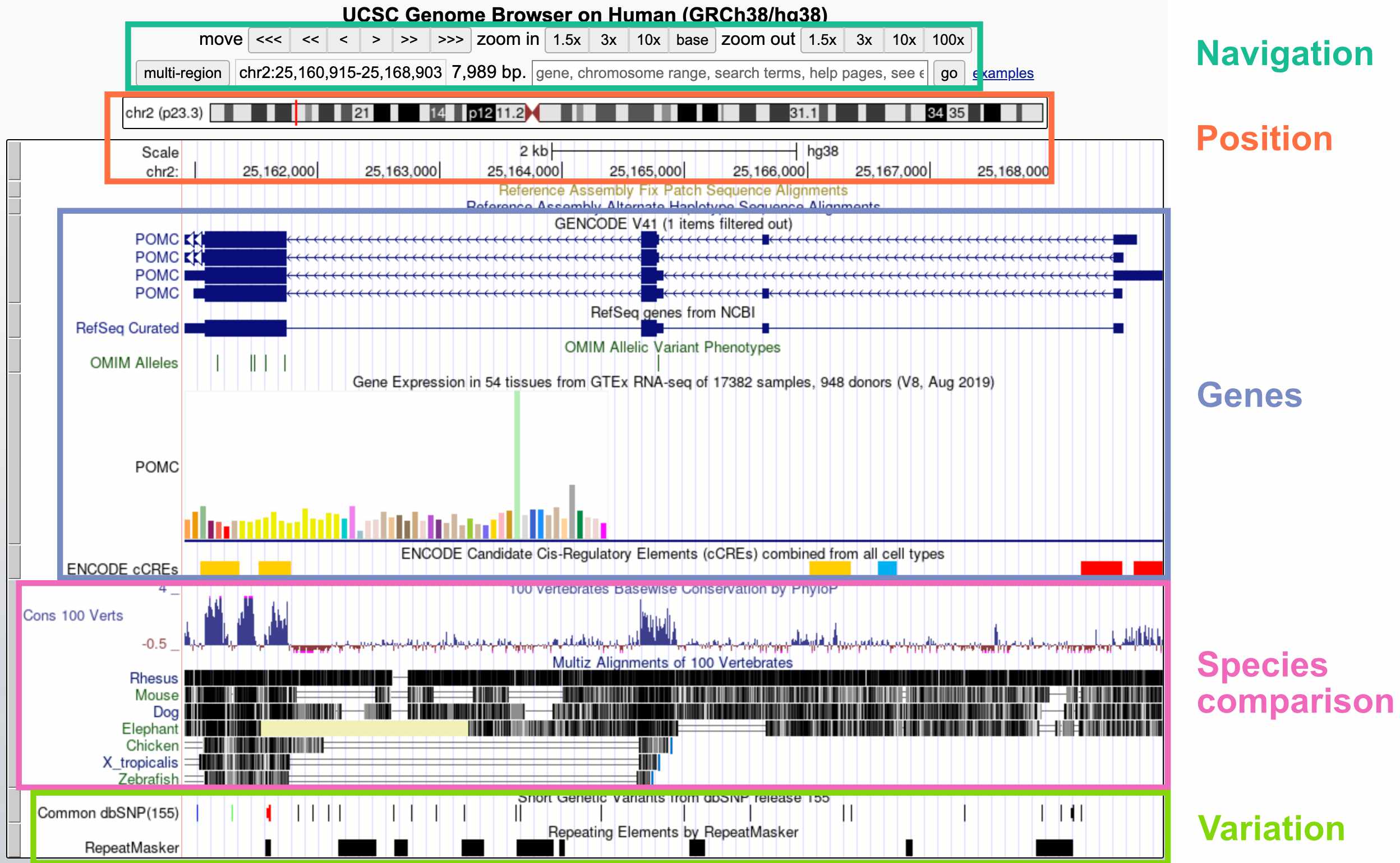


Figure : **Fig. 6.** Viewing one genomic region in the UCSC genome browser.

The default display includes these broad groups of annotations:

* **Navigation**: Buttons for zooming and moving around (you can also move by clicking the display, holding, and moving your mouse); current region; search bar
* **Position**: Current position on the chromosome; current base pair position
* **Genes**: Gene annotations; gene expression by tissue; gene regulatory elements (CREs)
* **Species comparison**: DNA sequence conservation across vertebrates; regions that align with the genomes of other vertebrates
* **Variation**: Genetic variants in the dbSNP database; repeat elements

Inspecting a specific track

If you’re interested in more information about a specific track – for example, the *POMC* gene annotation – you can click on that element to go to a webpage with more details.

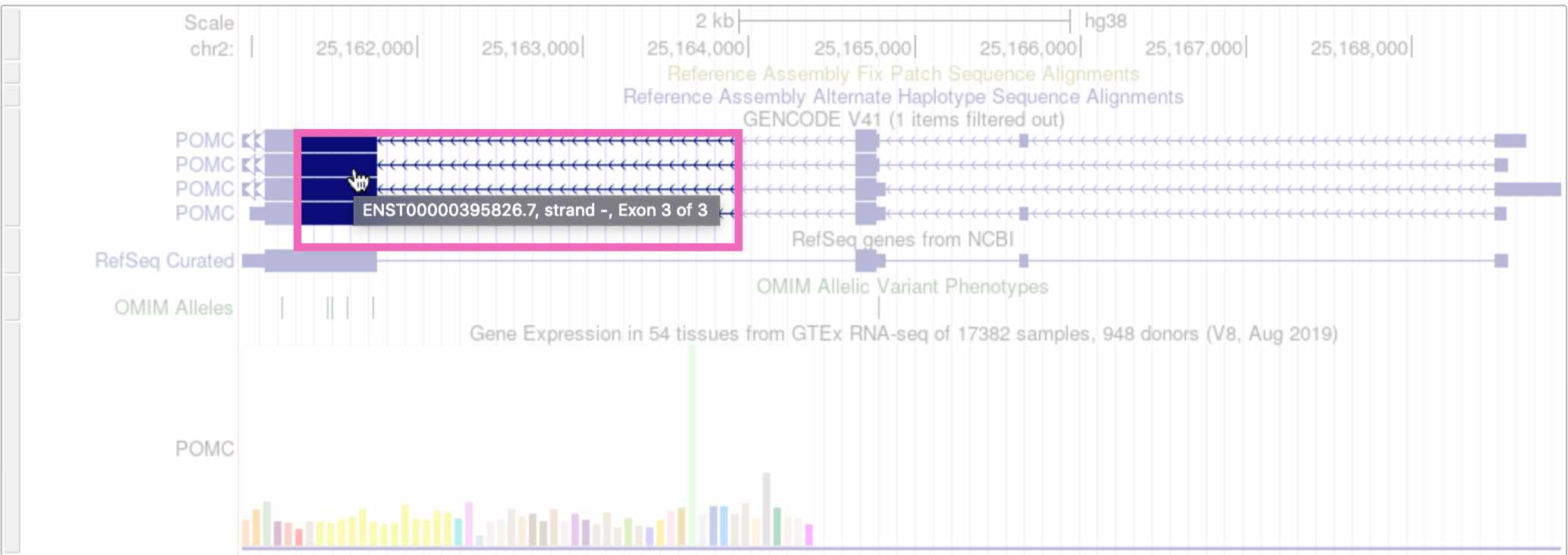


Figure : **Fig. 7.** Clicking on the *POMC* gene track.

(Note that if you click on whitespace instead of an annotation element, it will change the track’s display density instead.)



Figure : **Fig. 8.** Clicking on whitespace to expand the RefSeq genes track.

Customizing the display tracks

The tracks that are automatically displayed are just a small subset of what’s available. You can select which tracks you want to see, and set their display density, by scrolling down on the page.

To add a new track to your browser view, click the drop-down menu below that track and select any of the options besides hide. Here we’re viewing the “Clone Ends” track, which shows the different individuals that were sequenced to create this section of the reference genome.

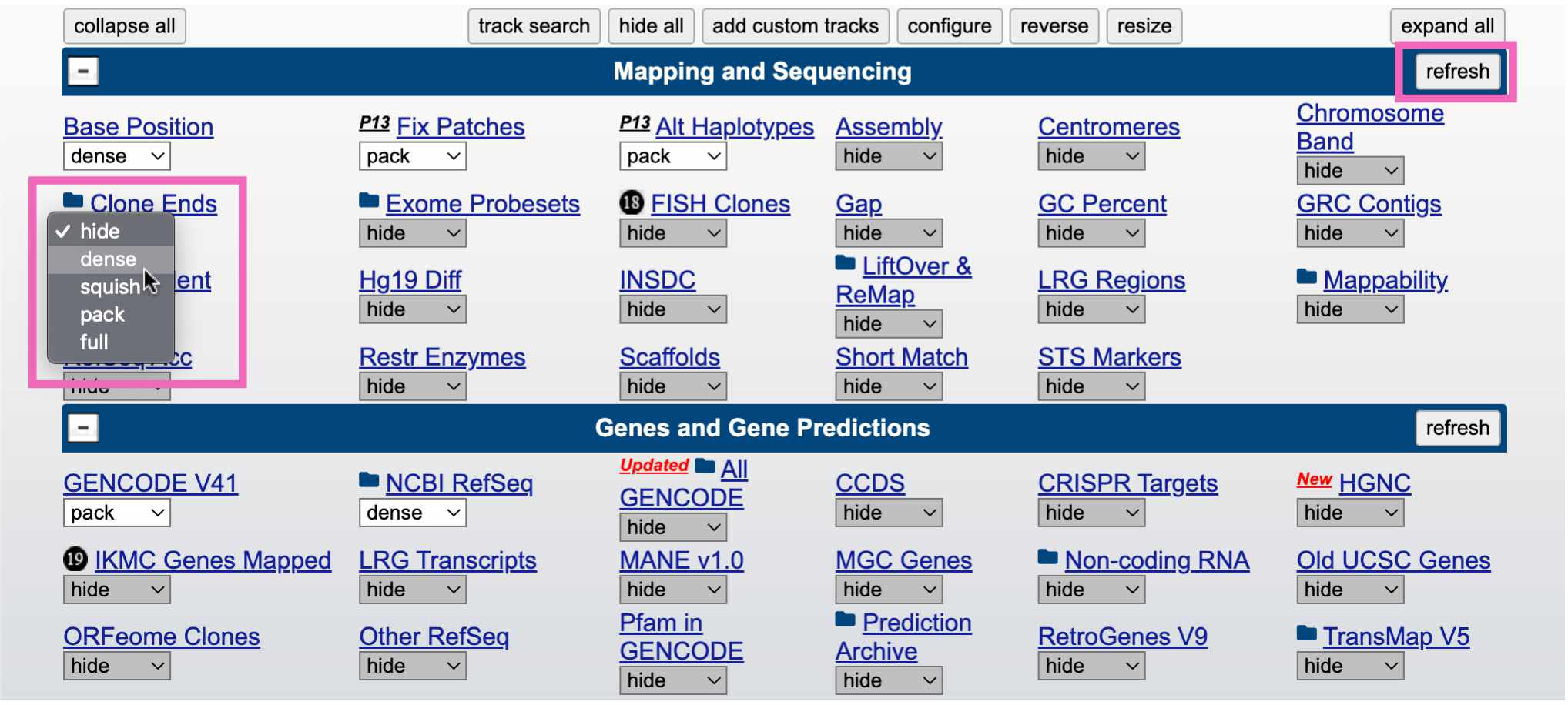


Figure : **Fig. 9.** Adding the “clone ends” track to the browser.

Click the refresh button in the upper right to reload the genome view. You should see something like this, showing that this region of the reference genome was sequenced in three individuals (CH17, CTD, and RP11):

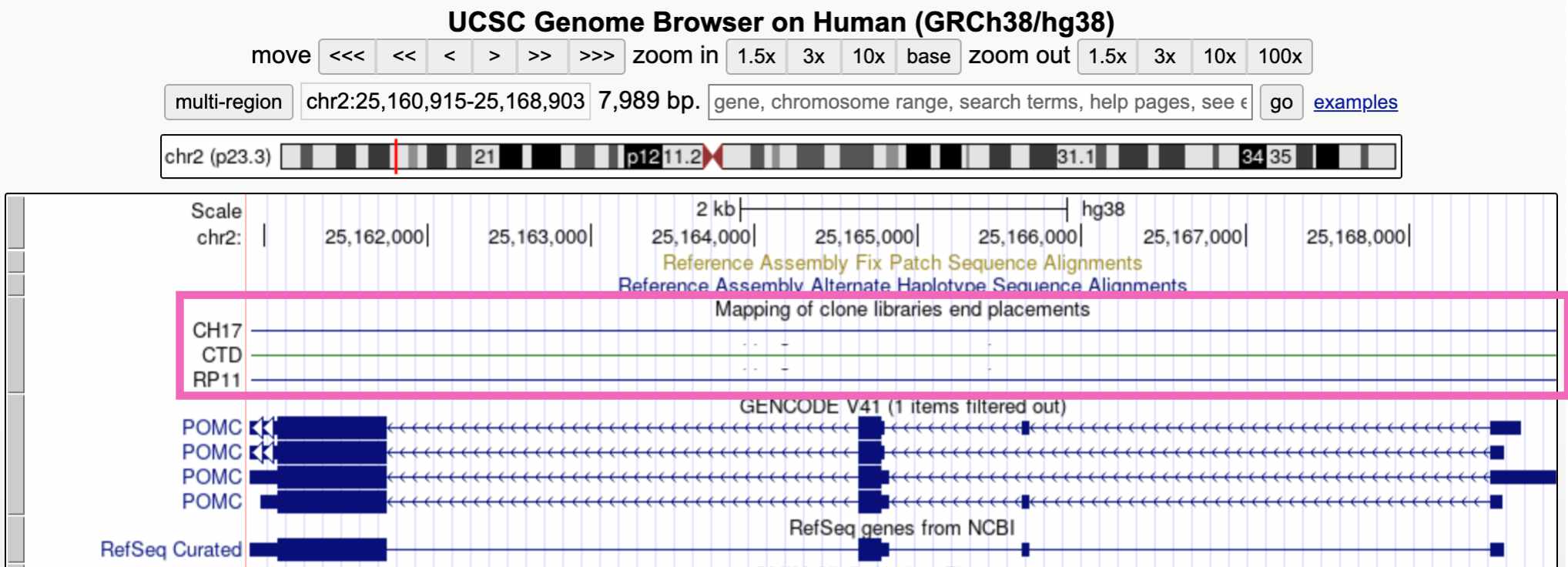


Figure : **Fig. 10.** Viewing the “clone ends” track.

## 2.6 IGV

While the UCSC genome browser lets you view the reference genome itself, we’re often interested in looking at sequencing data – sequencing reads that are aligned to the reference genome. For this we use the **Integrative Genomics Viewer (IGV)**.

Go to the [IGV web app](https://igv.org/app/).

### 2.6.1 Homepage

The IGV homepage is fairly empty because we haven’t loaded any sequencing reads to look at, and also because we’re zoomed too far out to see anything.

* The Genome tab in the upper right lets you choose which reference genome to work in. The default is **hg38**
* A drop-down menu and search bar below the header allow you to pick a chromosome and genomic position

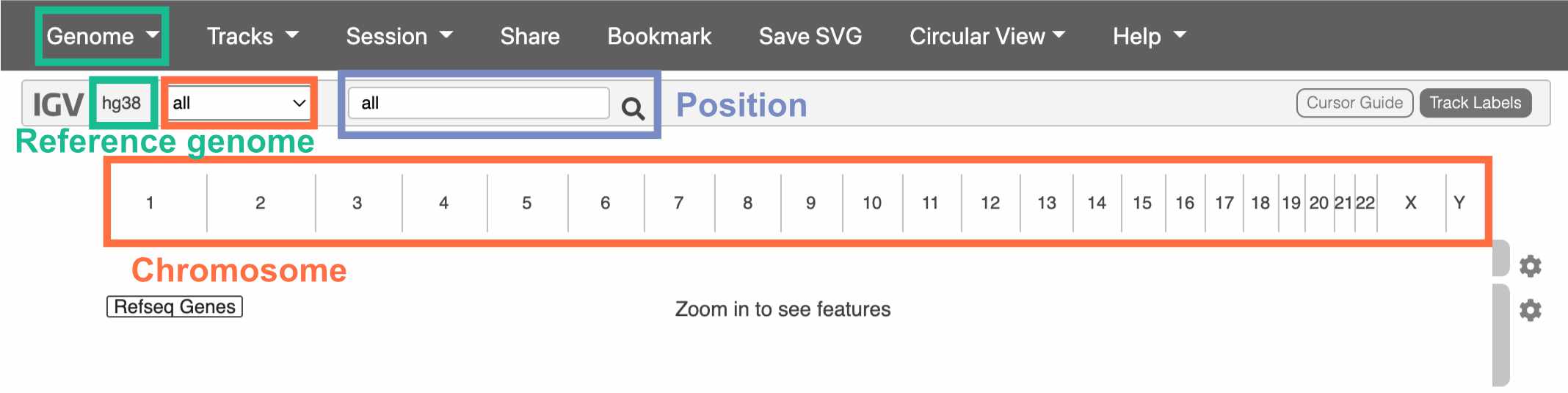


Figure : **Fig. 11.** The IGV homepage.

We haven’t chosen a chromosome yet, so all of them are displayed below the drop-down menu. Click on one to go to a zoomed-in view of that particular chromosome.

## 2.7 Navigating IGV

Once you’ve clicked on a chromosome, zoom in until you can see colors on the top track. This track displays the DNA sequence, colored by nucleotide.

The track below the DNA sequence has gene annotations from RefSeq.

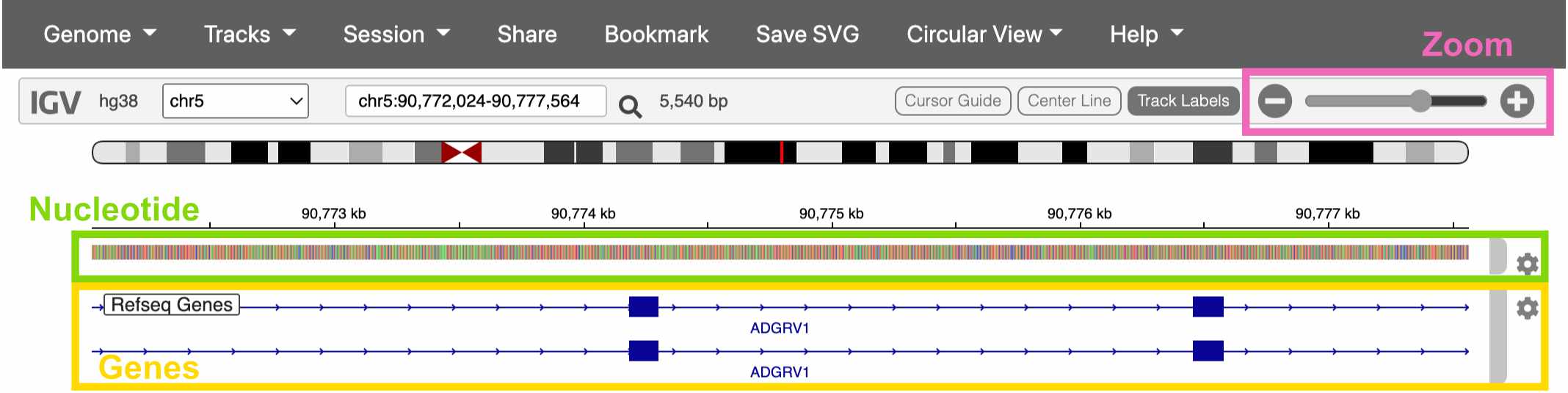


Figure : **Fig. 12.** Viewing a gene in IGV.

## 2.8 Loading sequencing data

Click on the Genome drop-down menu and switch to the Human (hg38 1kg/GATK) reference genome. This version of the hg38 reference has sequencing data already loaded into the IGV web app.

Once you’ve switched references, click Tracks -> 1KG Low Coverage Alignments. This gives you a list of sample to load sequencing data from. Click any sample and then OK.

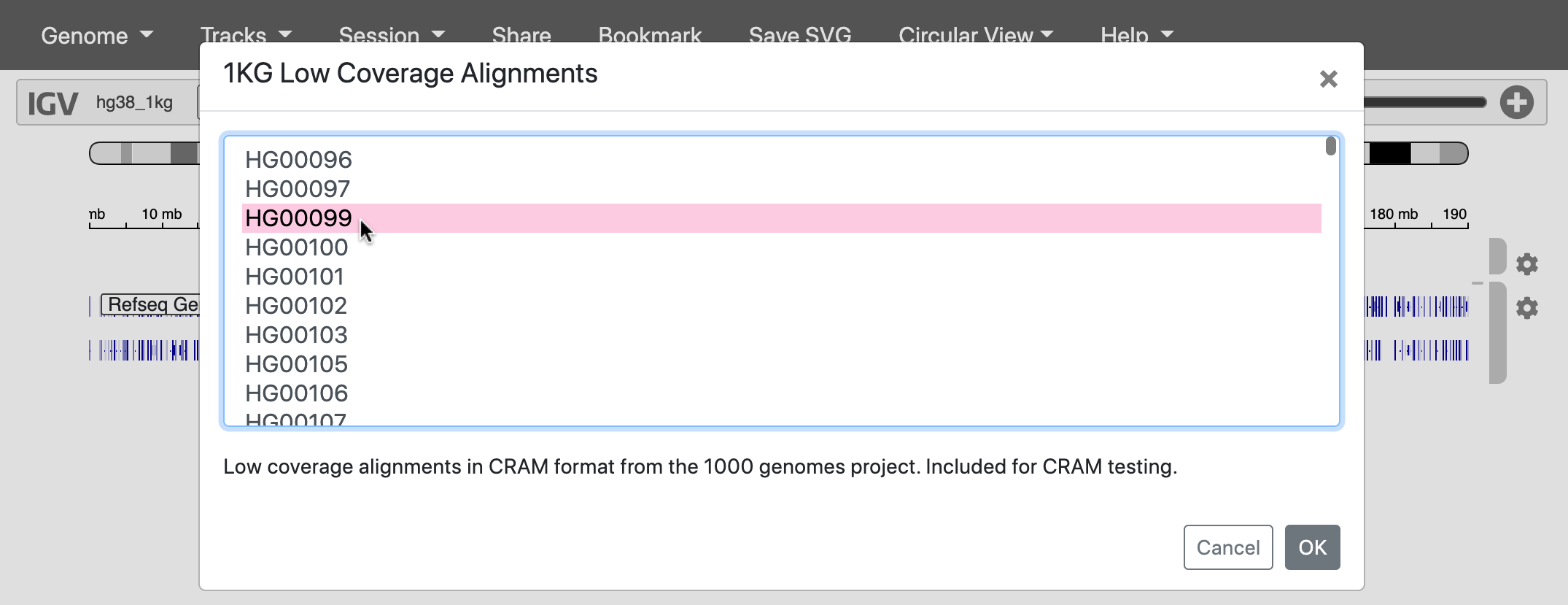


Figure : **Fig. 13.** Loading reads from a 1000 Genomes sample.

## 2.9 The 1000 Genomes Project

Where did this sequencing data come from?

In 2015, a study called the [**1000 Genomes Project**](https://www.nature.com/articles/nature15393) **(1KG or 1KGP)** sequenced 3,202 individuals from 26 globally diverse populations. Because this data is publicly available, it’s become one of the most widely used datasets in human genetics.

Notably, 1KGP still excludes key regions of the world – such as Oceania, the Middle East, native American populations in North America, and many populations within Africa.

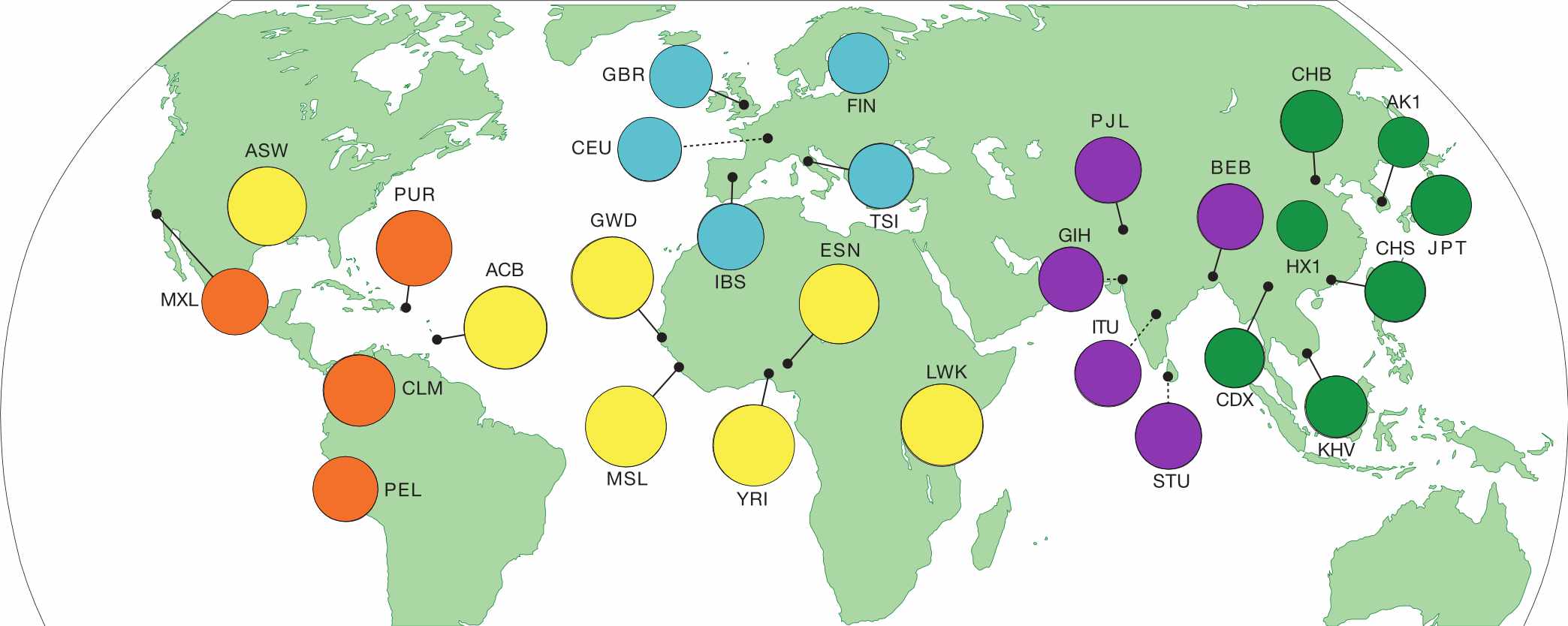


Figure : **Fig. 14.** Regions sampled by the 1000 Genomes Project.

Go to the [**1000 Genomes Project** website](https://www.internationalgenome.org/home) and click the Data tab. Then click the link to the data portal.

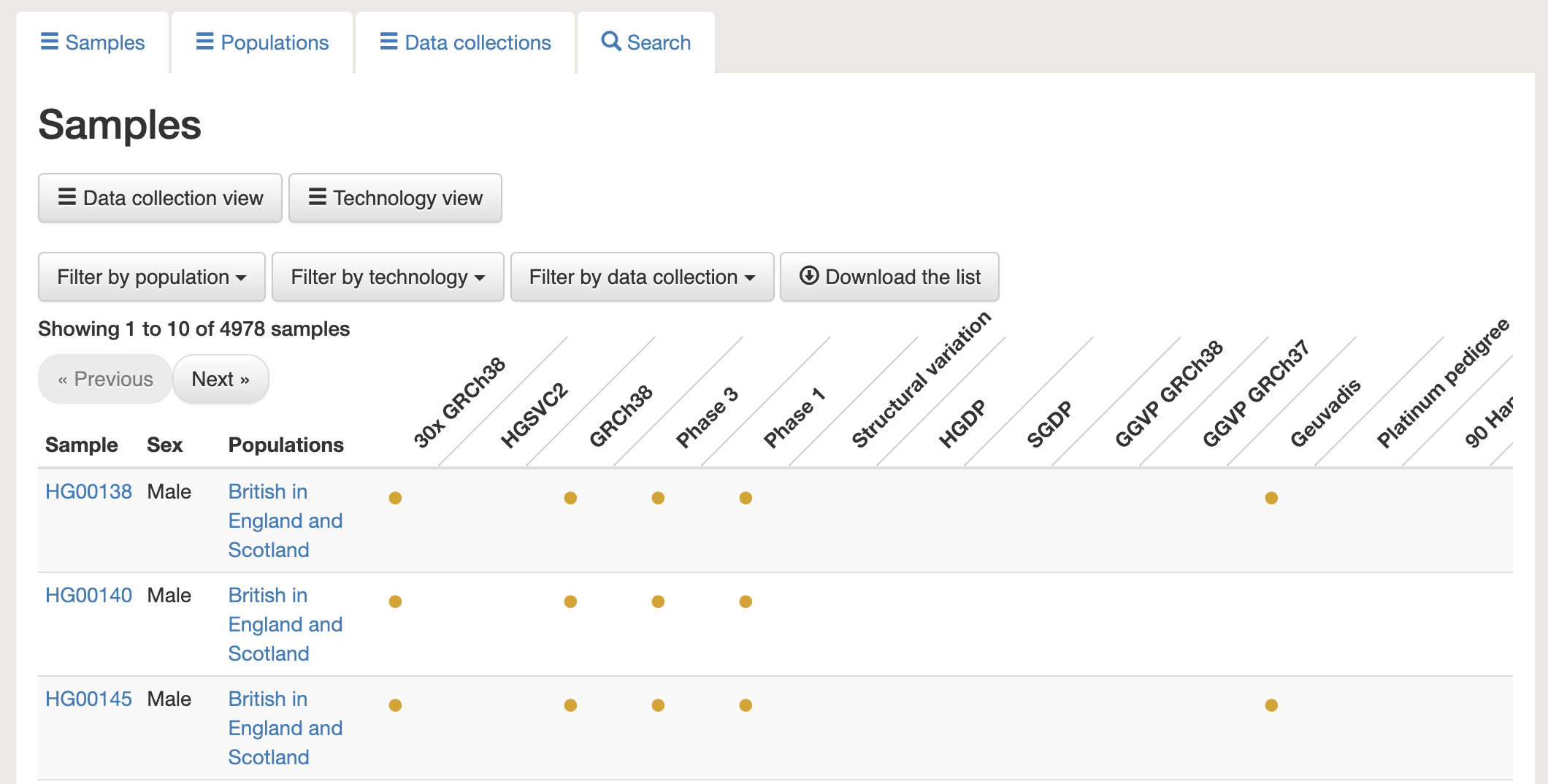


Figure : **Fig. 15.** The data portal includes information about the samples in this dataset.

Choose any individual and copy their sample ID (ex: **HG00138**). We can use this sample ID to find this individual’s raw sequencing data in the [**Sequence Read Archive (SRA)**](https://www.ncbi.nlm.nih.gov/sra).

## 2.10 SRA

Search for the sample ID you chose in SRA. You should see something like this, where every item is a sequencing dataset generated for this sample.

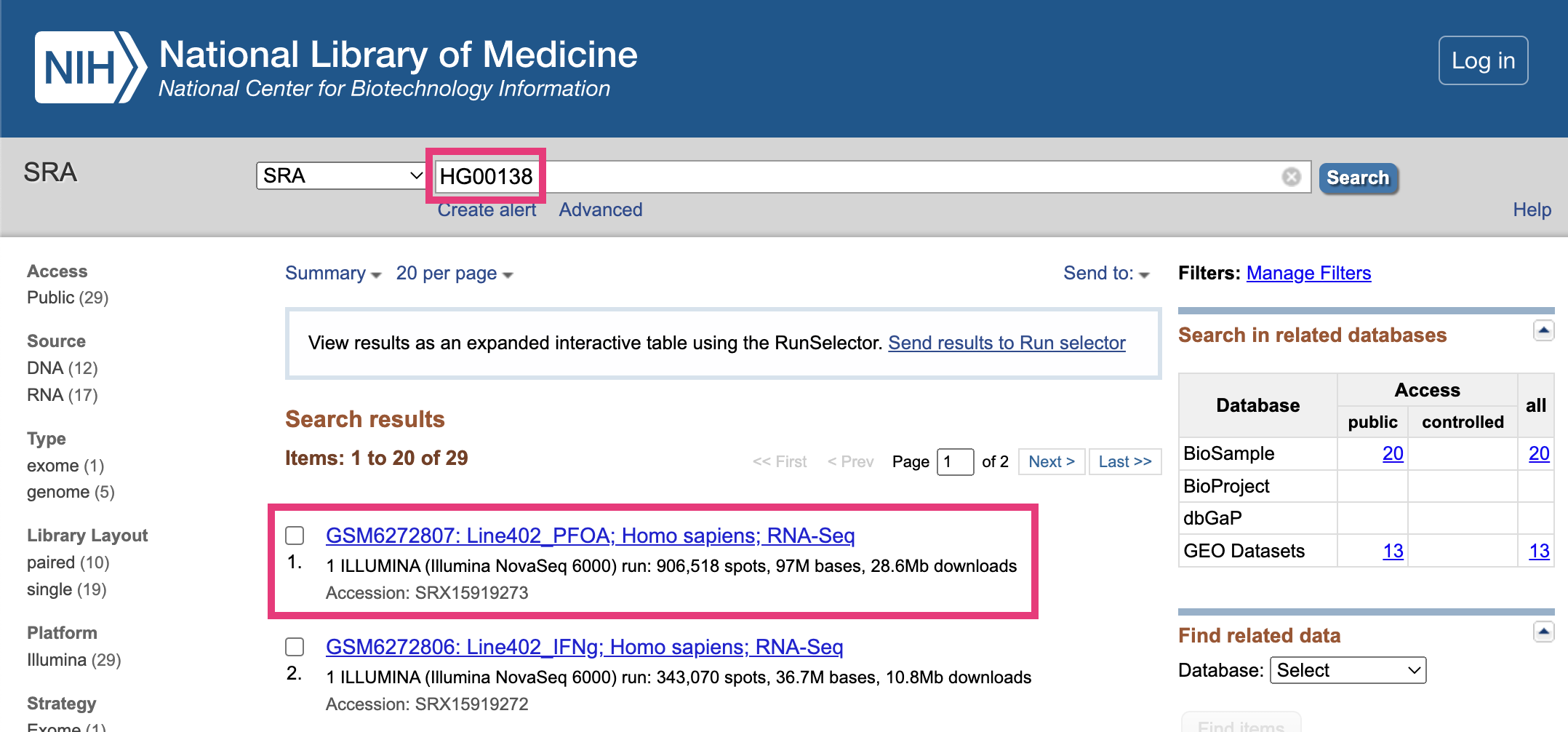


Figure : **Fig. 16.** Finding sequencing data in SRA.

### 2.10.1 Previewing sequencing data

Choose any sequencing dataset, and then click on any item in the **Run** table at the bottom.

This takes you to a page that displays a specific sequencing run (i.e., one use of a sequencing machine). Go to the **Reads** tab.

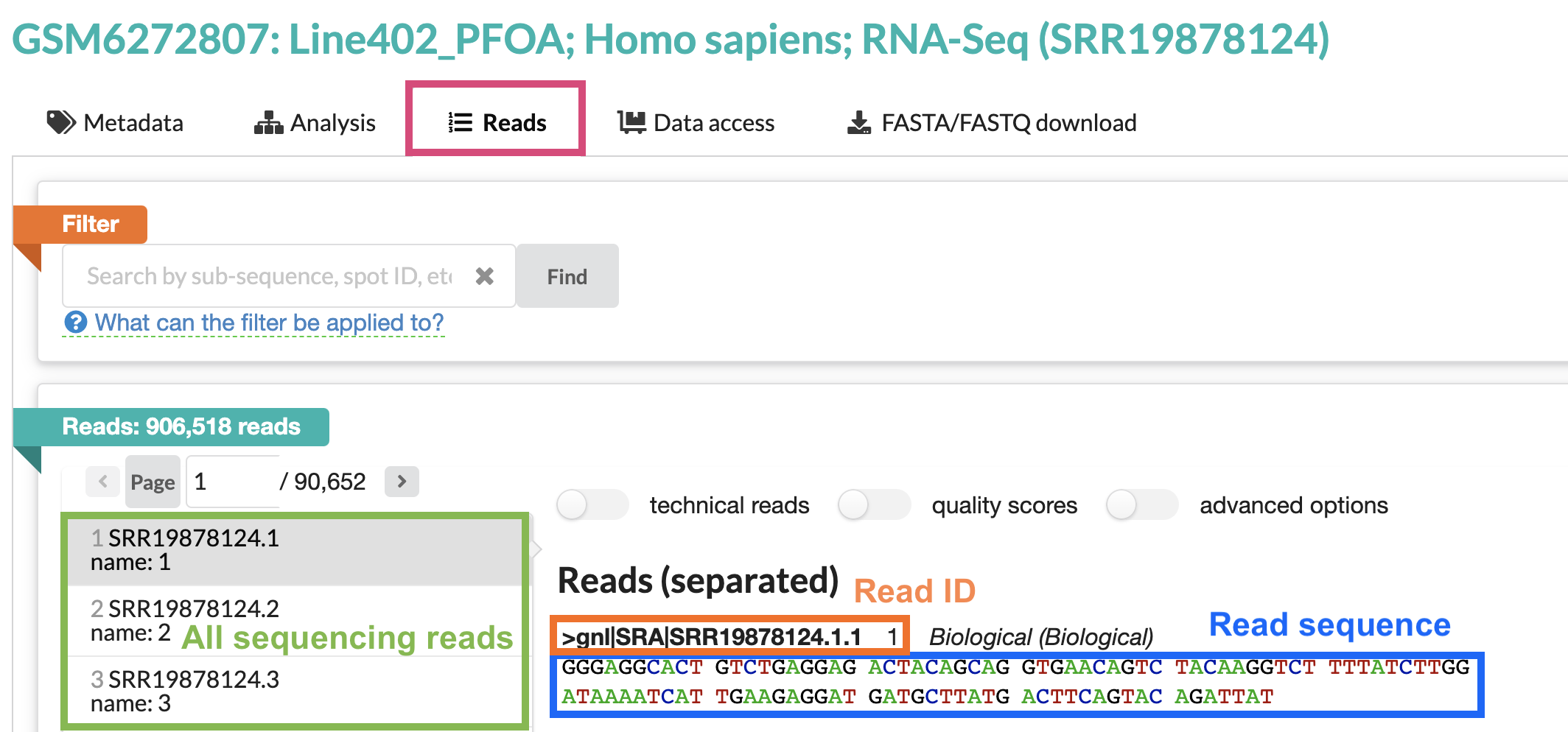


Figure : **Fig. 17.** The Reads tab in SRA.

The right-hand panel shows one sequencing read from this run. Note that the sequence of this read is around 100bp long – the average length for short-read Illumina sequencing.

All sequencing data looks like this! It’s just a text file filled with the IDs and DNA/RNA sequences of your reads.

## 2.11 Viewing sequencing reads in IGV

Now that we’ve seen what raw sequencing data looks like, let’s look at it in IGV.

Return to your IGV tab, where you should have one sample’s DNA sequencing data loaded. Make sure you’re zoomed in enough for the data to display.



Figure : **Fig. 18.** The Reads tab in SRA.

The top track is a histogram of sequencing coverage (i.e., how many reads there are at that position in the genome). The bottom track shows the reads themselves.

How do we know where in the genome each read belongs?

We match the sequence of the read to the sequence of the reference genome (called **alignment**). With 100bp reads, the probability that a match occurs by chance is , or .

Extracting alignment information in IGV

If you click on a specific read, IGV will display additional information about it, including:

* The exact position it aligns to
* The mapping quality (a score indicating how uniquely it aligns to this position)
* If you’re working with paired-end sequencing data, where its paired read is

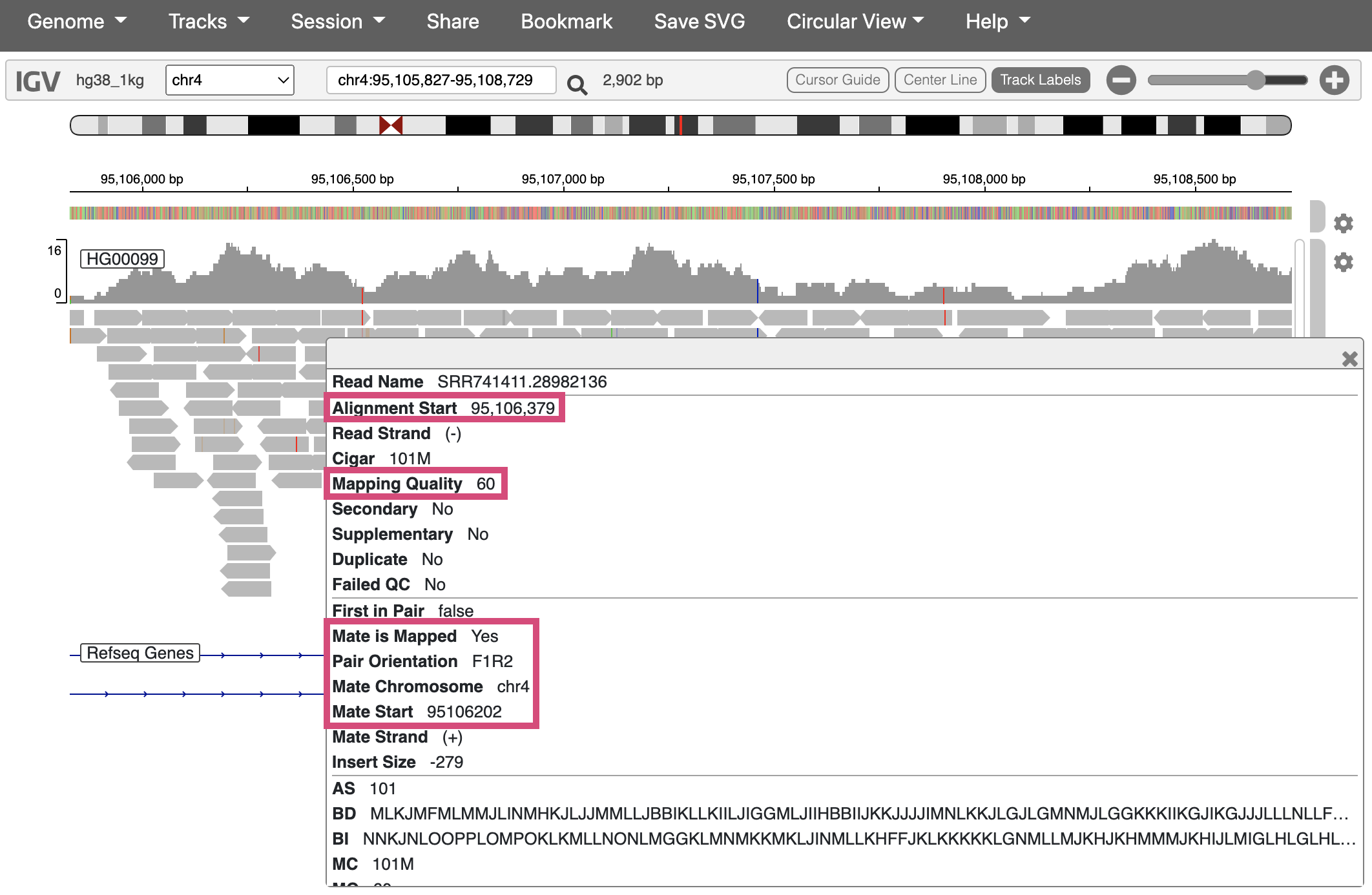


Figure : **Fig. 19.** Viewing additional info for one sequencing read.

## 2.12 Interpreting IGV alignments

Sequencing reads in IGV are colored at bases where they differ from the reference genome. These differences can be caused by either real genetic variation or sequencing error. **How would you distinguish these two?**



Figure : **Fig. 20.** Two of these colored bases are probably real SNPs, and two are probably errors.

The sequencing coverage track also colors the positions that it thinks are real variants.

In the screenshot above, which spans about 2kb, there are two SNPsin the coverage track. This pattern holds more broadly through the genome – humans carry about **one SNP every 1,000 bases**.

Is one SNP every 1,000bp a lot or a little?

Humans actually have much lower amounts of genetic variation than many species, [including many of the great apes](https://www.nature.com/articles/nature12228).

This is mostly the result of human evolutionary history. Because the effective size of human populations has historically been low, with only very recent expansion, the gene pool is still fairly homogenous, with many rare variants and few common ones.

## 2.13 Conclusion

In this lab, we explored several of the most commonly used websites in genomics:

#### 2.13.0.1 Genome browsers

* [**UCSC genome browser**](https://genome.ucsc.edu/): Used to explore features of the human genome
  + If you discover an interesting SNP in your research, you might look it up in the UCSC browser to see which genes it’s in/near, if it overlaps with any repetitive elements, etc.
* [**IGV**](https://igv.org/app/): Used to visualize sequencing data
  + It’s common practice to look at your sequencing reads in IGV to check alignment quality, verify that SNPs look like real variants and not errors, etc.

#### 2.13.0.2 Data repositories

* [**1000 Genomes Project**](https://www.internationalgenome.org/home): One of the largest and most diverse datasets of human sequencing data
  + Data from 1000 Genomes is frequently used in human genetics studies
* [**SRA**](https://www.ncbi.nlm.nih.gov/sra): A repository for publicly available sequencing data
  + Genetics studies deposit their data in SRA if it can be made publicly available (i.e., if it has no identifiable information)

## 2.14 Homework

#### 2.14.0.1 Goals & Learning Objectives

The goal of this homework is to make an account in Posit Cloud to use for the rest of the semester.

### 2.14.1 Required homework

Follow the instructions [here](https://mccoy-lab.github.io/hgv_modules/posit-cloud.html) to create a Posit Cloud account and join the HGV workspace.

# 3 Discovering mutations

In this module, we’ll use DNA sequencing data from human families to explore the relationship between parental age and *de novo* mutations in their children.

#### 3.0.0.1 Learning objectives

After completing this chapter, you’ll be able to:

1. Create plots to visualize the relationship between two variables.
2. Interpret the results of a linear model.
3. Compare the impact of maternal vs. paternal age on *de novo* mutation counts.
4. Explain what a confidence interval is and why it’s useful.

## 3.1 *De novo* mutations

Mutation and recombination are two biological processes that generate genetic variation. When these phenomena occur during gametogenesis, the changes that they make to DNA are passed down to the next generation through germline cells (i.e., sperm and oocyte).

***De novo* mutations (DNMs)** arise from errors in DNA replication or repair. These mutations can be single-nucleotide polymorphisms (SNPs) or insertions and deletions of DNA. Every individual typically carries around 70 *de novo* SNPs that were not present in either of their parents.

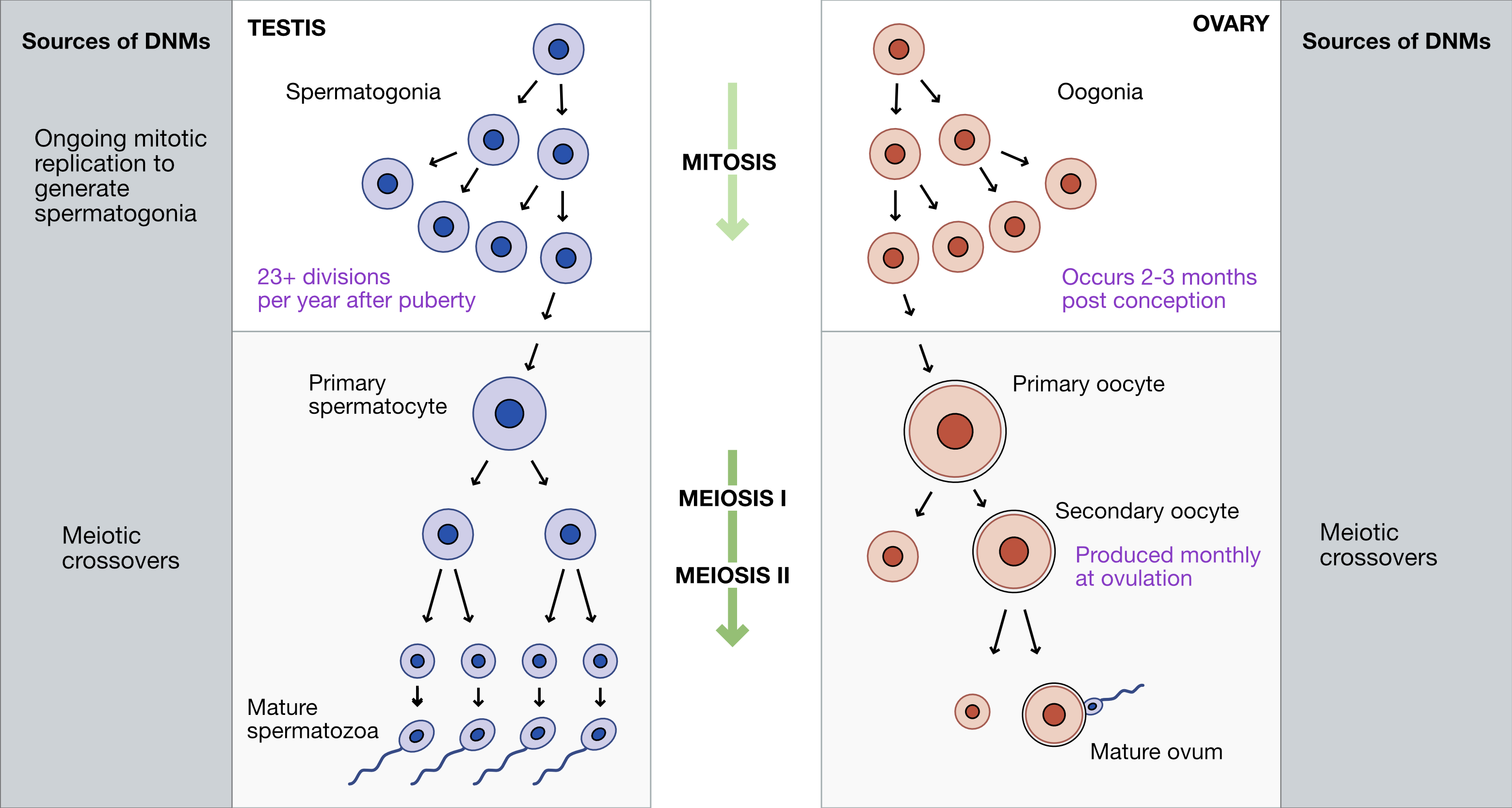


Figure : **Fig. 1.** Sources of DNMs in gametogenesis.

## 3.2 Recombination

**Crossovers**, or meiotic **recombination**, occur during prophase of meiosis I, when homologous chromosomes pair with each other. Double-strand breaks are deliberately generated in the DNA, and are then cut back and repaired based on the sequence of the homologous chromosome. These repairs can sometimes resolve in a crossover event, where sections of DNA are swapped between chromosomes.

Because the sequences of homologous chromosomes differ at sites where they carry different alleles, recombination generates genetic diversity by creating new haplotypes, or combinations of alleles.

Crossovers are required for meiosis because they ensure proper homologous chromosome pairing and segregation (although there are exceptions in some organisms, like male fruit flies). Humans experience 1-4 crossover events per chromosome, with longer chromosomes having more crossovers.

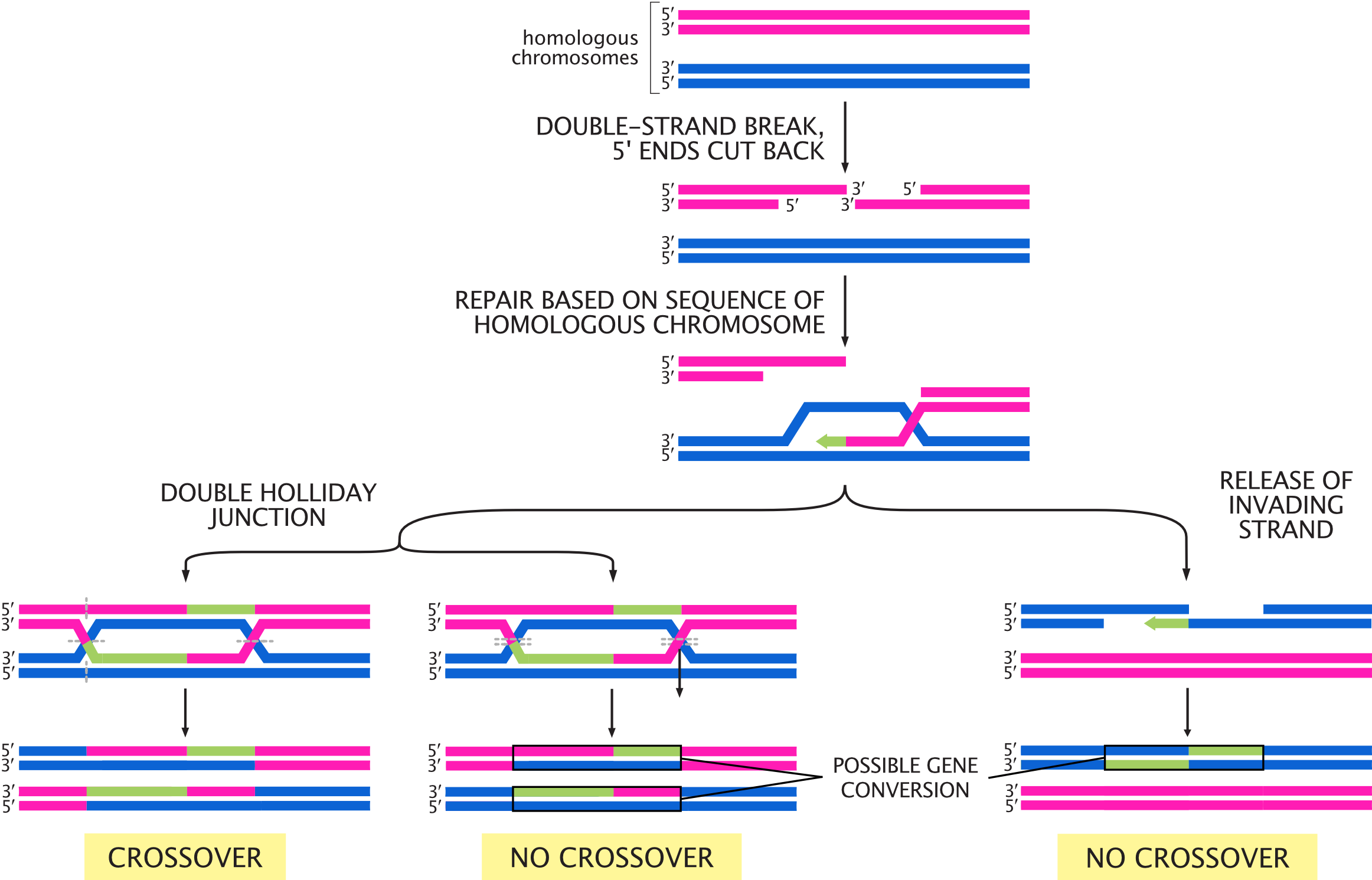


Figure : **Fig. 2.** Possible outcomes for double-strand breaks generated during meiosis I. Adapted from *Molecular Biology of the Cell, 6th Edition* (Alberts et al.)

## 3.3 Setup

In this module, we’ll use sequencing data from families to look at the relationship between DNMs, crossovers, and parental age.

### 3.3.1 R packages

We’re using R’s tidyverse library to analyze our data. You can load this R package by running:

library(tidyverse)

### 3.3.2 Data

Our data comes from the supplementary tables of [this paper by Halldorsson et al.](https://science.sciencemag.org/content/363/6425/eaau1043), which performed whole-genome sequencing on “trios” (two parents and one child) in Iceland. We’ve pre-processed the data to make it easier to work with.

Load the pre-processed data by running the code chunk below.

# read data  
dnm\_by\_age <- read.table("dnm\_by\_age\_tidy\_Halldorsson.tsv",  
 sep = "\t", header = TRUE)  
# preview data  
head(dnm\_by\_age)

## Proband\_id n\_paternal\_dnm n\_maternal\_dnm n\_na\_dnm Father\_age Mother\_age  
## 1 675 51 19 0 31 36  
## 2 1097 26 12 1 19 19  
## 3 1230 42 12 3 30 28  
## 4 1481 53 14 1 32 20  
## 5 1806 61 11 6 38 34  
## 6 2280 63 9 3 38 20

The columns in this table are:

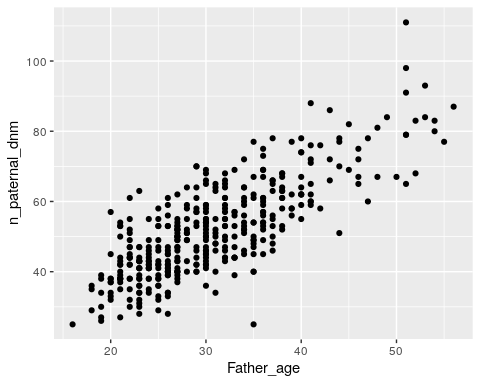
1. Proband\_id: ID of the child (i.e., “proband”)
2. n\_paternal\_dnm: Number of DNMs (carried by the child) that came from the father
3. n\_maternal\_dnm: Number of DNMs that came from the mother
4. n\_na\_dnm: Number of DNMs whose parental origin can’t be determined
5. Father\_age: Father’s age at proband’s birth
6. Mother\_age: Mother’s age at proband’s birth

## 3.4 Visualizing the data

We can use our tidied data to ask questions about the *de novo* mutation rate in these Icelandic individuals. How does parental age affect the number of DNMs for males and females?

Use the dnm\_by\_age data to plot this relationship for *males*.

ggplot(data = dnm\_by\_age,  
 # specify where ggplot should be getting the x location for each data point  
 aes(x = Father\_age,  
 # specify where ggplot should be getting the y location for each data point  
 y = n\_paternal\_dnm)) +  
 # specify that the data should be plotted as points  
 geom\_point()

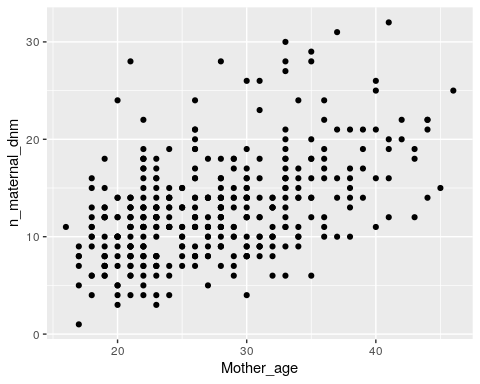


Based on your plot, would you say that there’s an association between paternal age and number of DNMs?

It looks like there’s a pretty strong association between paternal age and number of DNMs, where older males have more DNMs.

Modify your code to plot the relationship between age and number of DNMs for *females*. Does there seem to be an association between maternal age and number of DNMs?

ggplot(data = dnm\_by\_age,  
 aes(x = Mother\_age,  
 y = n\_maternal\_dnm)) +  
 geom\_point()



There’s also a strong positive association between maternal age and number of DNMs, although the slope (i.e., the increase in number of DNMs per year) is shallower.

## 3.5 Linear models

We can visually observe that age seems associated with number of DNMs in both males and females, but we need a way to ask if that this is a statistically meaningful association.

We can do this with a **linear model**. This model fits a line to the plots that we just made, and asks if the slope is significantly different from 0 (i.e., if there’s a significant increase in DNM count as age increases).

If this is a statistical test, what’s the null hypothesis?

The null hypothesis for this linear model is that the slope is 0 – i.e., that there’s no association between parental age and the number of DNMs from that parent.

If the slope is significantly different from 0, we can reject the null hypothesis.

We’ll fit a linear model using R’s lm function. Run the following code block to open a manual describing the function.

?lm

lm requires two arguments:

* The formula or equation it’s evaluating
* A table of data

The formula must be in the format response variable ~ predictor variable(s), where each variable is the name of a column in our data table.

Is our predictor variable the parental age or the number of DNMs?

The predictor variable is parental age. We expect the number of DNMs to change as a *consequence* of parental age.

## 3.6 Fitting a linear model for DNMs

Run the following code to fit a model for the effect of age on paternal DNMs.

# fit linear model for paternal DNMs  
fit\_pat <- lm(formula = n\_paternal\_dnm ~ Father\_age,  
 data = dnm\_by\_age)  
  
# print results of model  
summary(fit\_pat)

##   
## Call:  
## lm(formula = n\_paternal\_dnm ~ Father\_age, data = dnm\_by\_age)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -32.785 -5.683 -0.581 5.071 31.639   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 10.58819 1.70402 6.214 1.34e-09 \*\*\*  
## Father\_age 1.34849 0.05359 25.161 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 8.426 on 388 degrees of freedom  
## Multiple R-squared: 0.62, Adjusted R-squared: 0.619   
## F-statistic: 633.1 on 1 and 388 DF, p-value: < 2.2e-16

How do you interpret results from a linear model?

For our purposes, the only part of the results you need to look at is the line under (Intercept) in the Coefficients section:

Estimate Std. Error t value Pr(>|t|)  
Father\_age 1.34849 0.05359 25.161 < 2e-16 \*\*\*

* The fourth columm, Pr(>|t|), is the **p-value**.

Because this p-value is < 2e-16, we can reject the null hypothesis and say that there is association between paternal age and the number of paternal DNMs.

* The first column, Estimate, is the **slope**, or **coefficient**.

Linear regression fits a line to our plot of paternal age vs. number of DNMs. The coefficient estimate is the **slope** of that line.

The slope for paternal age given by this linear model is 1.34849. We can interpret this number this way: **For every additional year of paternal age, we expect 1.35 additional paternal DNMs in the child.**

Modify your code to assess the relationship between *maternal* age and number of *maternal* DNMs. Is this relationship significant? How many maternal DNMs do we expect for every additional year of maternal age?

# fit linear model for maternal DNMs  
fit\_mat <- lm(formula = n\_maternal\_dnm ~ Mother\_age,  
 data = dnm\_by\_age)  
  
# print results of model  
summary(fit\_mat)

##   
## Call:  
## lm(formula = n\_maternal\_dnm ~ Mother\_age, data = dnm\_by\_age)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.8683 -3.1044 -0.2329 2.2394 17.5379   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.51442 0.98193 2.561 0.0108 \*   
## Mother\_age 0.37846 0.03509 10.785 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.503 on 388 degrees of freedom  
## Multiple R-squared: 0.2307, Adjusted R-squared: 0.2287   
## F-statistic: 116.3 on 1 and 388 DF, p-value: < 2.2e-16

The p-value is <2e-16 and the Mother\_age slope is 0.37846.

This relationship is significant, and we expect 0.38 more maternal DNMs for every additional year of maternal age.

## 3.7 Confidence intervals

Our models predict that there are 1.35 more DNMs for additional every year of paternal age, and 0.38 more DNMs for every additional year of maternal age. Does this mean that sperm and oocytes accumulate DNMs at different rates?

The maternal and paternal slopes look different, but we need statistical evidence that they actually are. (For example, what if there’s a lot of variability in the maternal DNM data, and the true maternal coefficient could be anywhere between -1 and 10?)

To do this, we compare the **confidence intervals** of our slope estimates.

What is a confidence interval?

We use confidence intervals when estimating a value – in this case, the Mother\_age and Father\_age slope parameters.

A **confidence interval (CI)** is a range of values for which, for some probability, the interval will contain the true value of the slope.

So, a 95% CI contains the true value of the slope 95% of the time.

In R, we get the confidence interval of a parameter from a linear model with the confint function.

?confint

confint requires three arguments:

* A fitted linear model (our fit\_pat variable)
* The parameter we want a CI for (Father\_age)
* The CI’s probability (typically 95%)

## 3.8 Calculate 95% CIs

Run the following code to calculate the 95% confidence interval for the Father\_age slope parameter.

confint(fit\_pat, 'Father\_age', level = 0.95)

## 2.5 % 97.5 %  
## Father\_age 1.243118 1.45386

So, 95% of the time, the number of additional DNMs per year of paternal age is between 1.24 and 1.45.

Modify your code to get the 95% CI for the *Mother\_age* slope. What’s the interpretation of this confidence interval?

confint(fit\_mat, 'Mother\_age', level = 0.95)

## 2.5 % 97.5 %  
## Mother\_age 0.3094713 0.4474528

95% of the time, the number of additional DNMs per year of maternal age is between 0.31 and 0.45.

Now that we have the confidence intervals for both slope parameters, we can finally compare them.

Our two CI ranges are non-overlapping. The paternal range is [1.24, 1.45] and the maternal range is [0.31, 0.45].

If the 95% CIs for two parameters *don’t* overlap, this strongly supports that the parameters are significantly different from one another. **So, it seems likely that paternal and maternal gametes experience different rates of *de novo* mutation.**

If the CIs for two parameters overlap, are they not significantly different?

Not necessarily. More analysis, like a hypothesis test, is needed to make a final decision.

## 3.9 Conclusion

In this lab, we explored the relationship between parental age and the number of *de novo* mutations in their gametes.

* We **plotted** the relationship between maternal/paternal age and DNM count. This visualization suggested that DNM count increases with age for both groups.
* We confirmed this hypothesis by using a **linear model**, which tests if additional years of age have a non-zero effect on the number of DNMs.
* The number of paternal DNMs seemed to increase more quickly with age than maternal DNMs. We confirmed this by comparing the **95% confidence intervals** of the slopes of the two models.

One final question – let’s assume that there really is a difference between the effect of age on DNMs in male and female gametes. What biological reasons might be causing this difference?

## 3.10 Homework

So far, we’ve only looked at the *de novo* mutation data from [the Halldorsson et al. paper](https://science.sciencemag.org/content/363/6425/eaau1043). Now we’ll use their data on the number of maternal and paternal origin crossovers (i.e., how many crossovers occurred across all chromosomes in the maternal and paternal gametes).

#### 3.10.0.1 Goals & Learning Objectives

The goal of this homework is to practice with ggplot.

**Learning Objectives**

* Required homework: Practice visualizing data with ggplot2
* Optional homework: Practice interpreting linear models

## 3.11 Required homework

The data from the paper has been pre-filtered for you. Run this code block to read it in:

# read data  
crossovers <- read.table("crossovers.tsv", header = TRUE)  
  
# preview data  
head(crossovers)

## Proband\_id n\_pat\_xover n\_mat\_xover Father\_age Mother\_age  
## 1 3 22 51 29 28  
## 2 10 26 50 26 26  
## 3 11 25 38 25 22  
## 4 15 24 50 31 26  
## 5 20 27 35 26 24  
## 6 22 28 40 39 31

The columns in this table are:

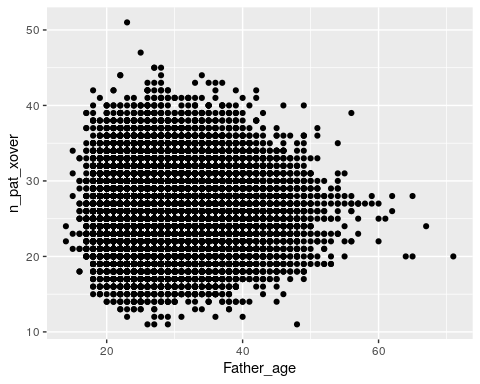
1. Proband\_id: ID of the child
2. n\_pat\_xover: Number of crossovers (carried by the child) that occurred in the paternal gametes
3. n\_mat\_xover: Number of crossovers that occurred in the maternal gametes
4. Father\_age: Father’s age at proband’s birth
5. Mother\_age: Mother’s age at proband’s birth

**Assignment:** Using the ggplot code from this module, plot the relationship between parental age and number of crossovers. As with the DNM data, make one plot for the maternal crossovers and one plot for the paternal. Do you think parental age impacts crossover number?

Solution

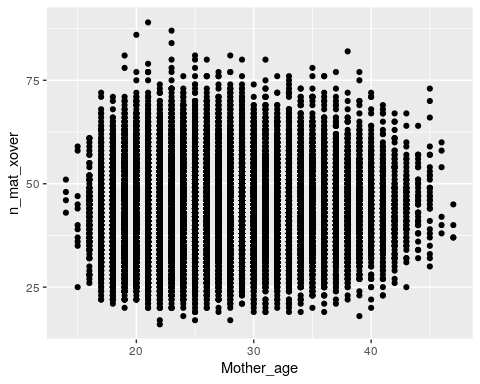
Plot paternal crossovers:

ggplot(data = crossovers,  
 # x axis is paternal age  
 aes(x = Father\_age,  
 # y axis is number of crossovers  
 y = n\_pat\_xover)) +  
 geom\_point()



Plot maternal crossovers:

ggplot(data = crossovers,  
 # x axis is maternal age  
 aes(x = Mother\_age,  
 # y axis is number of crossovers  
 y = n\_mat\_xover)) +  
 geom\_point()



Just by eye, it doesn’t really seem that age affects number of crossovers for either mothers or fathers.

## 3.12 Optional homework

**Assignment:** Fit *two* linear models (one paternal, one maternal) to ask if there is an association between the number of parental crossovers and parental age. If there is an association, how is the number of crossovers predicted to change with every year of maternal/paternal age?

Solution

# fit the model with paternal age  
fit\_pat <- lm(data = crossovers,  
 formula = n\_pat\_xover ~ Father\_age)  
summary(fit\_pat)

##   
## Call:  
## lm(formula = n\_pat\_xover ~ Father\_age, data = crossovers)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -15.2173 -3.1880 -0.1997 2.8061 24.7652   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 26.369432 0.102736 256.67 <2e-16 \*\*\*  
## Father\_age -0.005852 0.003462 -1.69 0.091 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 4.388 on 41090 degrees of freedom  
## Multiple R-squared: 6.953e-05, Adjusted R-squared: 4.519e-05   
## F-statistic: 2.857 on 1 and 41090 DF, p-value: 0.09098

There isn’t a significant association between paternal age and the number of paternal crossovers (p = 0.091).

# fit the model with maternal age  
fit\_mat <- lm(data = crossovers,  
 formula = n\_mat\_xover ~ Mother\_age)  
summary(fit\_mat)

##   
## Call:  
## lm(formula = n\_mat\_xover ~ Mother\_age, data = crossovers)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -27.161 -6.095 -0.425 5.641 45.905   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 41.709271 0.206238 202.24 <2e-16 \*\*\*  
## Mother\_age 0.065989 0.007576 8.71 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 8.685 on 41090 degrees of freedom  
## Multiple R-squared: 0.001843, Adjusted R-squared: 0.001819   
## F-statistic: 75.87 on 1 and 41090 DF, p-value: < 2.2e-16

Surprisingly, there *is* a significant association between maternal age and the number of maternal crossovers (p < 2e-16). For every year of maternal age, we expect the child to carry 0.07 additional maternal origin crossovers.

Although the maternal crossovers plot doesn’t look that impressive, our estimated slope is 0.07, which is probably too small to distinguish visually.

# 4 Linkage disequilibrium

In this module, we’ll use DNA sequencing data from human populations to assess linkage between two genetic variants.

#### 4.0.0.1 Learning objectives

After completing this chapter, you’ll be able to:

1. Define linkage disequilibrium.
2. Describe the data stored in a Variant Call Format (VCF) file.
3. Manually calculate , , and from genotype data.
4. Explain the differences in interpretation for different LD statistics.

## 4.1 What is linkage disequilibrium?

**Linkage disequlibrium (LD)** refers to correlation among genotypes at multiple sites in the genome. This is a consequence of the fact that chromosomes are transmitted in “chunks” through the generations.

When mutations arise, they arise on a single chromosome with a given set of alleles. The new mutation will continue to be associated with this genetic background until it is shuffled during the process of meiotic recombination. Together, a set of alleles that tend to occur together because of linkage disequilibrium is called a **haplotype**.

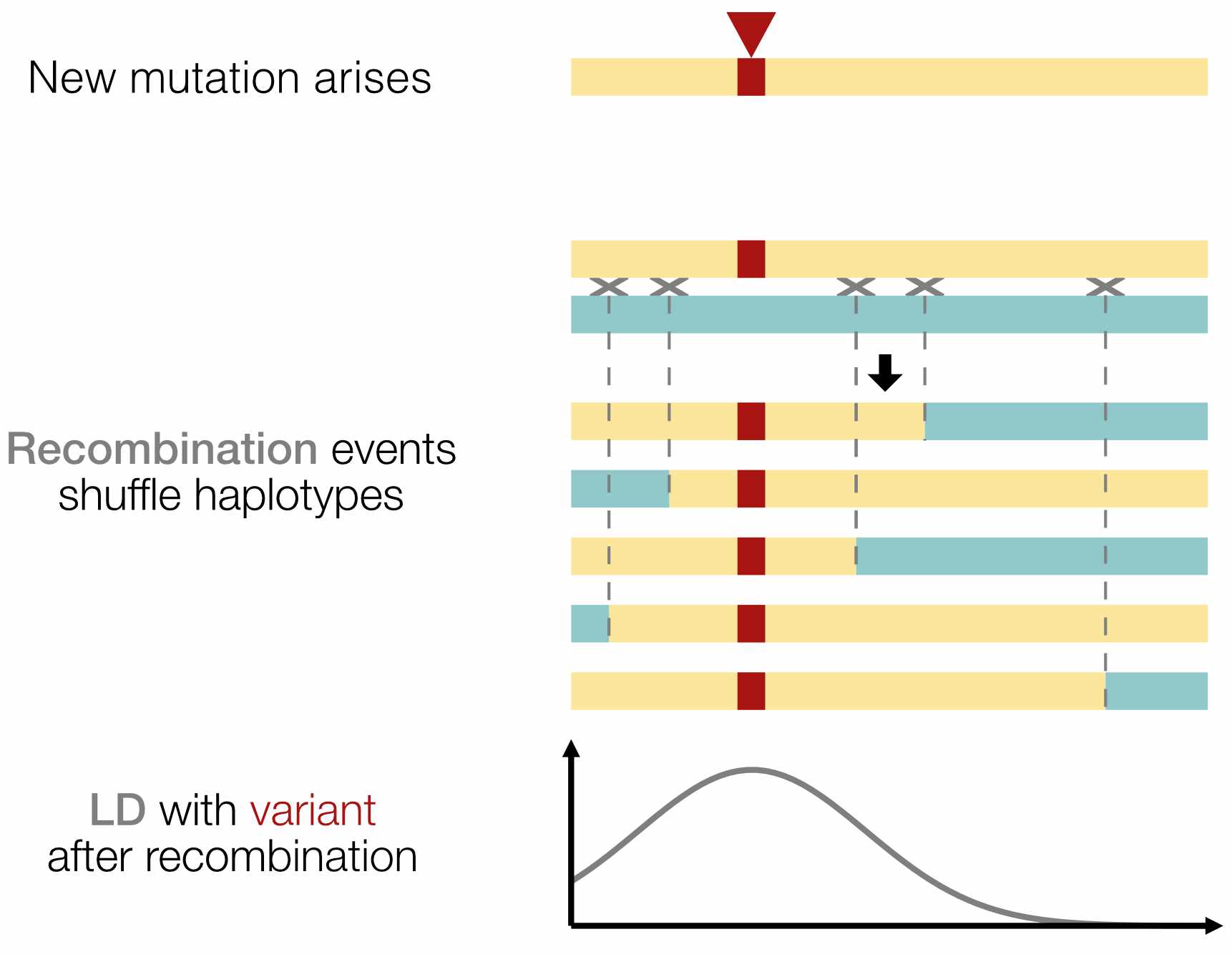


Figure : **Fig. 1.** After a new mutation arises, recombination over the course of many generations reduces the number of variants in LD with it.

## 4.2 Why do we care about LD?

As a result of linkage disequilibrium, knowledge of genotype at one site in the genome can be informative of genotype at another site, even if the second site was not actually genotyped. Using prior knowledge of LD to “fill in” missing genotype information is a process called **imputation**.

Linkage disequilibrium also means that correlation between genotype at a particular site and phenotype (e.g., disease outcome) does not imply causation. Even ignoring other possible confounders, any variant on the same haplotype could be driving the association.

Beyond mutation and recombination, other evolutionary forces such as gene flow, genetic drift, and natural selection can also influence patterns of LD observed in population genetic data. Measuring linkage disequilibrium is therefore important for both medical and evolutionary studies.

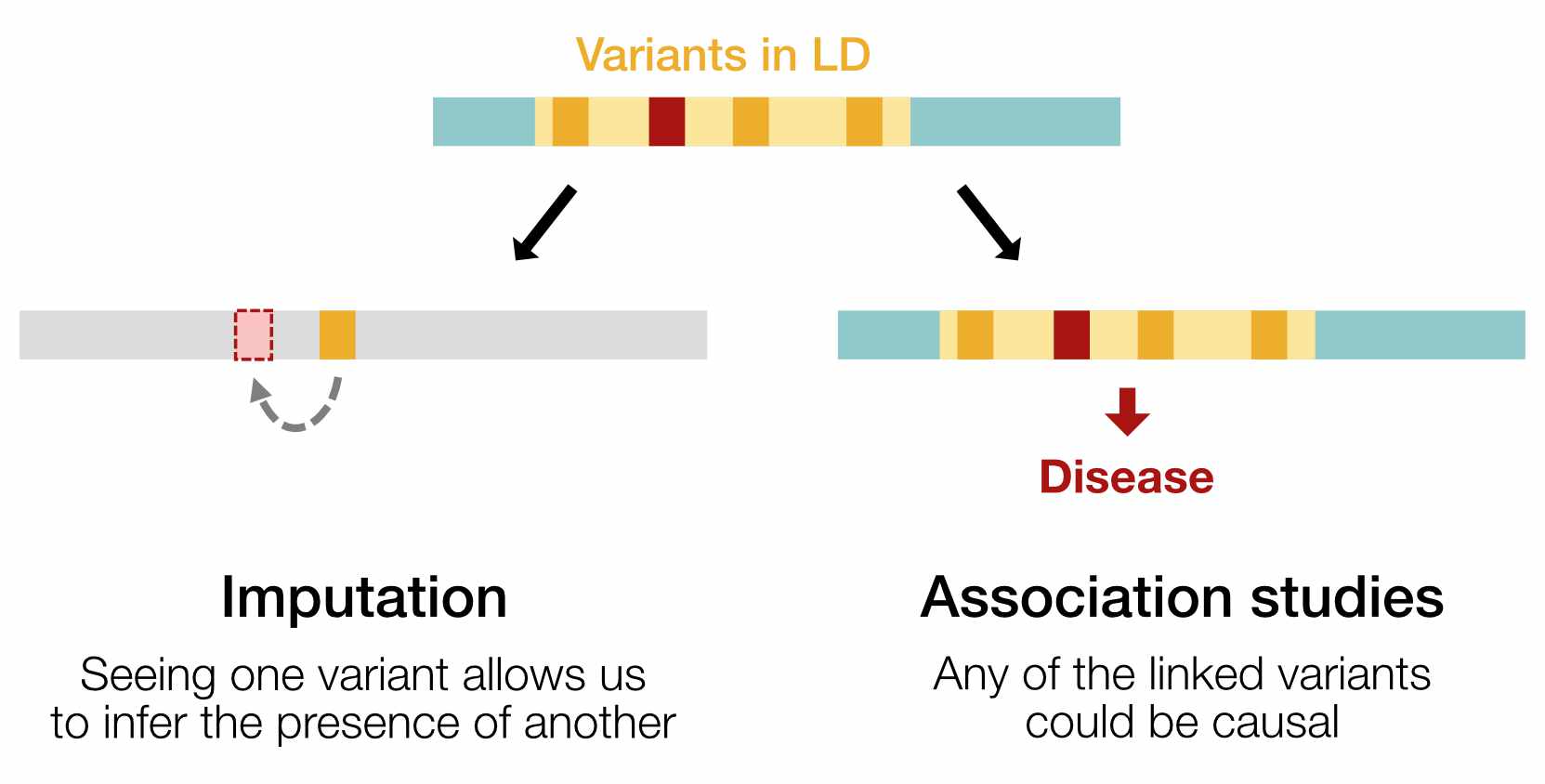


Figure : **Fig. 2.** LD can be used to impute missing genotypes, but also complicates genetic association studies (such as finding variants that cause disease).

## 4.3 Variant Call Format (VCF)

We’re investigating linkage disequilibrium in genotype data from the [1000 Genomes Project](https://mccoy-lab.github.io/hgv_modules/the-1000-genomes-project.html) – a sequencing dataset introduced in the genome browsers module.

We’ll eventually work with a pre-processed form of this genotype data, but first let’s look at the full file, which is in **Variant Call Format (VCF)**.

### 4.3.1 What is a VCF?

VCF files store **genotype** data at variable sites. Every line of a VCF represents a genetic variant, and contains information about what the variant is and which individuals carry it.

While these are just text files, they have a strange format that goes beyond a simple table. Run this code to read in and view the contiguous\_snippet.vcf.gz VCF file:

# use the `fread` function from `data.table` to read in the vcf  
vcf <- data.table::fread("contiguous\_snippet.vcf.gz",  
 skip = 19, header = TRUE)  
vcf

## #CHROM POS ID REF ALT QUAL FILTER  
## 1: chr21 15000072 . T C . PASS  
## 2: chr21 15000079 . G C . PASS  
## 3: chr21 15000082 . G A . PASS  
## 4: chr21 15000090 . C T . PASS  
## 5: chr21 15000091 . A G . PASS  
## 6: chr21 15000112 . T G . PASS  
## INFO  
## 1: AC=3;AN=5008;DP=16665;AF=0;EAS\_AF=0;EUR\_AF=0;AFR\_AF=0;AMR\_AF=0;SAS\_AF=0;VT=SNP;NS=2548  
## 2: AC=1;AN=5008;DP=18230;AF=0;EAS\_AF=0;EUR\_AF=0;AFR\_AF=0;AMR\_AF=0;SAS\_AF=0;VT=SNP;NS=2548  
## 3: AC=1;AN=5008;DP=18721;AF=0;EAS\_AF=0;EUR\_AF=0;AFR\_AF=0;AMR\_AF=0;SAS\_AF=0;VT=SNP;NS=2548  
## 4: AC=382;AN=5008;DP=20156;AF=0.08;EAS\_AF=0.27;EUR\_AF=0;AFR\_AF=0;AMR\_AF=0.01;SAS\_AF=0.1;VT=SNP;NS=2548  
## 5: AC=387;AN=5008;DP=20362;AF=0.08;EAS\_AF=0.27;EUR\_AF=0;AFR\_AF=0.01;AMR\_AF=0.01;SAS\_AF=0.1;VT=SNP;NS=2548  
## 6: AC=1;AN=5008;DP=21892;AF=0;EAS\_AF=0;EUR\_AF=0;AFR\_AF=0;AMR\_AF=0;SAS\_AF=0;VT=SNP;NS=2548  
## FORMAT HG00096 HG00097 HG00099 HG00100 HG00101 HG00102 HG00103 HG00104  
## 1: GT 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|0  
## 2: GT 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|0  
## 3: GT 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|0  
## 4: GT 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|0  
## 5: GT 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|0  
## 6: GT 0|0 0|0 0|0 0|0 0|0 0|0 0|0 0|0  
## HG00105 HG00106 HG00107  
## 1: 0|0 0|0 0|0  
## 2: 0|0 0|0 0|0  
## 3: 0|0 0|0 0|0  
## 4: 0|0 0|0 0|0  
## 5: 0|0 0|0 0|0  
## 6: 0|0 0|0 0|0

How do you read a VCF file?

The first 8 columns of the VCF are about the variant itself - its position, the reference/alternative alleles, etc. Note that column 8, the INFO column, also contains metadata such as the variant’s allele frequency (AF=).

The rest of the VCF columns contain the genotypes of all the samples we have data for. Here, every column is one individual from the 1000 Genomes Project (so there are 2,504 columns total).

## 4.4 Setup

We’ll use a subset of the VCF data to measure LD between two SNPs (randomly chosen for this exercise):

* [rs28574812](https://gnomad.broadinstitute.org/variant/21-15012619-A-G?dataset=gnomad_r3) (chr21:15012619)
* [rs2251399](https://gnomad.broadinstitute.org/variant/21-15013185-C-T?dataset=gnomad_r3) (chr21:15013185)

We’ve reformatted the VCF so that every line represents **one haplotype** in the 1000 Genomes database. Load the pre-processed data by running the code below.

# read data  
haplotypes <- read.table("snp\_haplotypes.txt", header = TRUE)  
  
# preview data  
head(haplotypes)

## sample haplotype snp1\_allele snp2\_allele  
## 1 HG00096 hap\_1 A C  
## 2 HG00096 hap\_2 A C  
## 3 HG00097 hap\_1 A C  
## 4 HG00097 hap\_2 A C  
## 5 HG00099 hap\_1 A C  
## 6 HG00099 hap\_2 A C

The columns in this table are:

1. sample: Name of the individual who was sequenced
2. haplotype: Haplotype (i.e., the maternal or paternal chromosome) that the SNP is on
3. snp1\_allele: Genotype at SNP1 on this haplotype
4. snp2\_allele: Genotype at SNP2 on this haplotype

Note that there are **2,504** samples in the 1000 Genomes Project but **5,008** total lines in the table. This is because there are two lines per individual – one for each of their maternal and paternal haplotypes.

Click for a biological representation of the data in the table

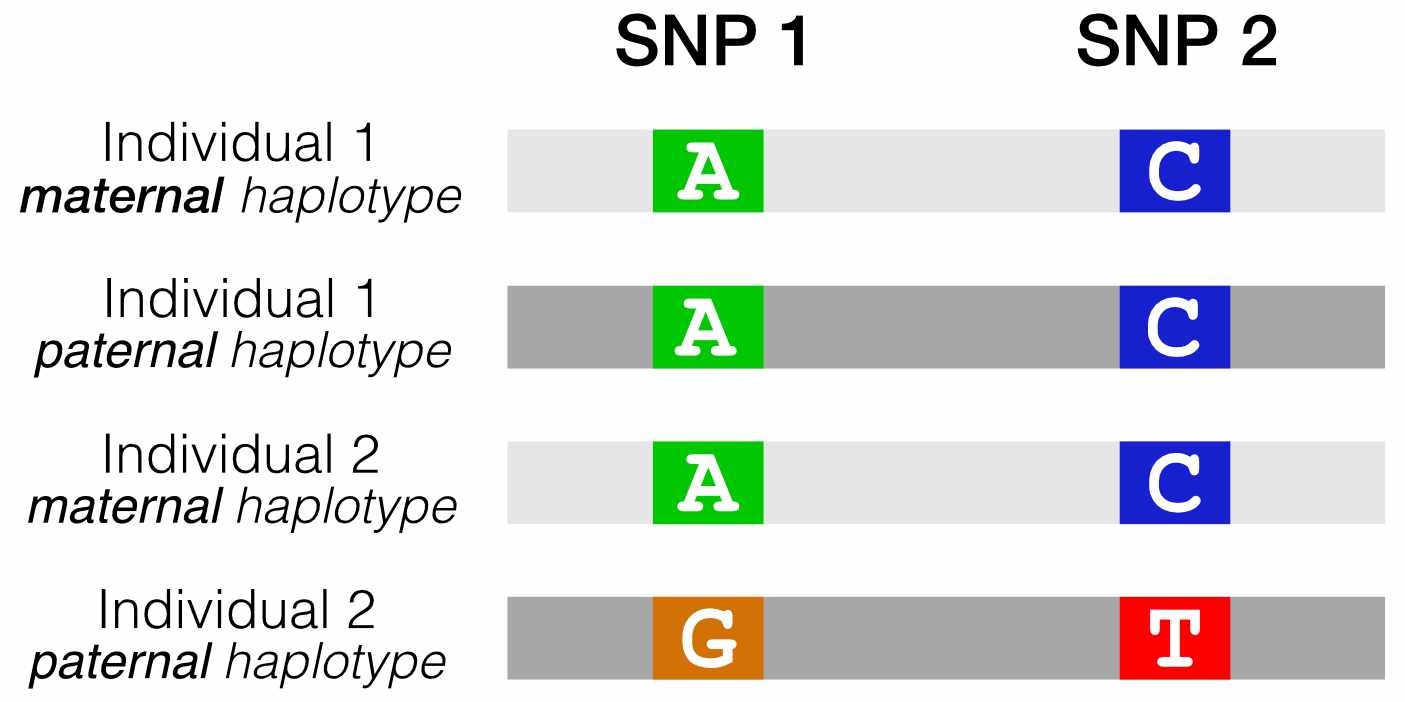


Figure : **Fig. 3.** Our reformatted VCF shows the combinations of alleles at two SNPs of interest, for all haplotypes in the 1000 Genomes dataset.

## 4.5 Are these SNPs in LD?

In the haplotypes table, we can see that there are two possible alleles at each SNP.

* SNP1 can be A or G
* SNP2 can be C or T

If these two SNPs are in **perfect LD**, we expect to see only two haplotypes in our data (**Fig. 2A**).

1. A C: If someone carries an A at SNP1, they will always carry a C at SNP2.
2. G T: If they carry a G at SNP1, they will always carry a T at SNP2.

If these two SNPs are in **linkage *equilibrium***, the allele at SNP1 gives us no information about SNP2. We would expect to see all four possible haplotypes, in amounts proportional to the component allele frequencies (**Fig. 2B**).

1. A C
2. A T
3. G C
4. G T

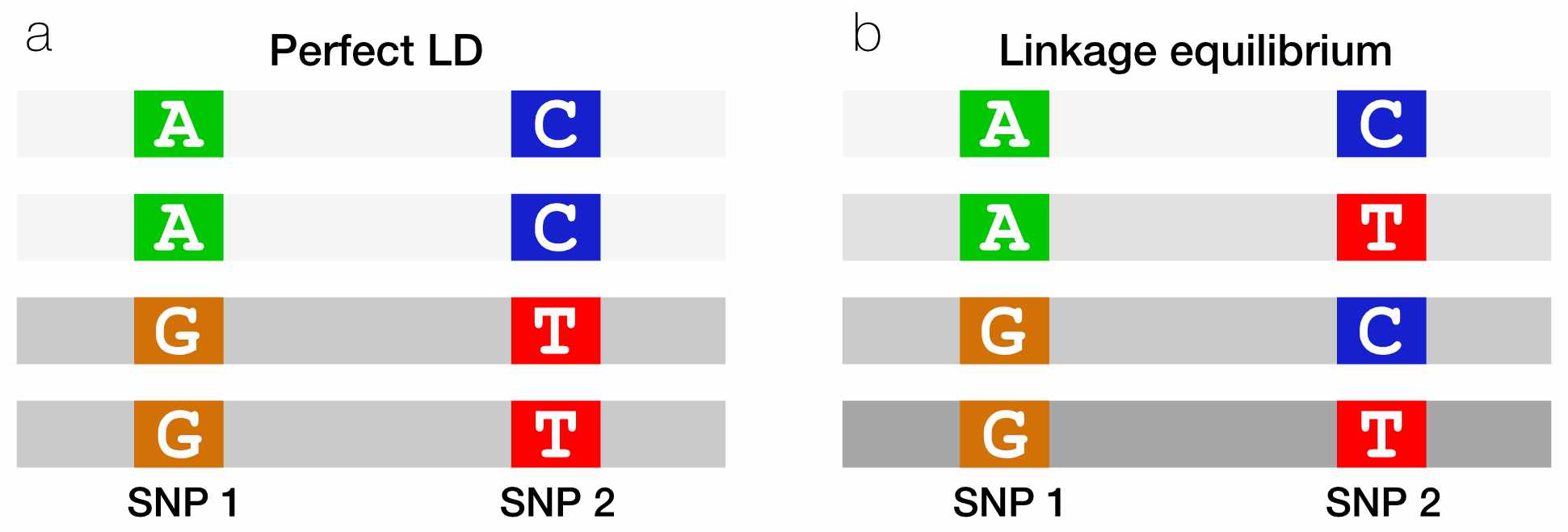


Figure : **Fig. 4.** When two SNPs are in perfect LD, seeing an allele on one haplotype perfectly predicts which allele is on the other haplotype.

## 4.6 Counting haplotypes with table

We can use the table function to count the occurrence of the four possible haplotypes.

table(haplotypes$snp1\_allele, haplotypes$snp2\_allele)

##   
## C T  
## A 2655 801  
## G 170 1382

The table tells us that there are 2655 A C haplotypes (A at SNP1 and C at SNP2), 170 G C haplotypes, etc.

Do these SNPs look like they’re in LD?

It looks like there are some haplotypes (A C and G T) that are overrepresented. However, it’s hard to tell whether that’s just because an A allele at SNP1 is much more common than T is.

## 4.7 Fisher’s exact test

We can wrap our table in the fisher.test function to perform a **Fisher’s exact test**. This test tells us whether there is a non-random association between any of the SNP alleles, while accounting for the relative proportions of each allele.

fisher.test(table(haplotypes$snp1\_allele, haplotypes$snp2\_allele))

##   
## Fisher's Exact Test for Count Data  
##   
## data: table(haplotypes$snp1\_allele, haplotypes$snp2\_allele)  
## p-value < 2.2e-16  
## alternative hypothesis: true odds ratio is not equal to 1  
## 95 percent confidence interval:  
## 22.49760 32.33934  
## sample estimates:  
## odds ratio   
## 26.9124

The **p-value** is very small (< 2.2e-16), so we can reject the null hypothesis that the two SNPs are associating independently of each other.

How do we interpret the odds ratio from the Fisher’s exact test?

In addition to the p-value, Fisher’s Exact Test also gives us an odds ratio of 26.9, with a 95% confidence interval of [22.5, 32.3]. In this context, the **odds ratio** reflects how much more likely you are to see an A at SNP1 if you also see an C at SNP2.

We can observe that the **95% confidence interval** doesn’t overlap with 1. An odds ratio of 1 would mean that seeing C at SNP2 doesn’t influence the probability of seeing an A at SNP1.

Together, the p-value and 95% confidence interval tell us that there is strong evidence of LD between these SNPs. Perhaps this isn’t surprising, since our two SNPs are common and close to one another on chromosome 21.

## 4.8 Measuring LD with

If SNP1 and SNP2 are in **linkage *equilibrium***, the probability of seeing an A C haplotype should be equal to the product of the allele frequencies of A and C. Otherwise, we should see A C either more or less often than expected from the allele frequencies.

This intuition is summarized in , a population genetics statistic for measuring LD between two SNPs.

* is the frequency of our haplotype of interest (A C).
* is the product of the frequencies of the two alleles on this haplotype (A at SNP1 and C at SNP2)

How do we interpret ?

If two SNPs are in **linkage equilibrium**, and should be the same, and we should get .

If two SNPs are in **linkage disequilibrium**, should be different from , so that .

## 4.9 Calculating

We can re-run our table code to find the probabilities we need for calculating .

table(haplotypes$snp1\_allele, haplotypes$snp2\_allele)

##   
## C T  
## A 2655 801  
## G 170 1382

What are , , and ?

is the probability of seeing the A C haplotype. This is equal to the number of A C haplotypes over the number of total haplotypes:

is the probability that SNP1 is A. We can get this by adding across the first **row** of the table (i.e., adding the number of A C and A T haplotypes):

is the probability that SNP2 is C. We can get this by adding across the first **column** of the table (i.e., adding the number of A C and G G haplotypes):

(Note that the denominator is always 5008 – the total number of haplotypes in our dataframe.)

Now we can plug in the corresponding probabilities to calculate D:

# define our probabilities of interest  
h <- 2655 / 5008  
p1 <- (2655 + 801) / 5008  
q1 <- (2655 + 170) / 5008  
  
# calculate D  
D <- h - (p1 \* q1)  
D

## [1] 0.1408705

, which is non-zero, suggesting that these SNPs are in LD.

## 4.10 Measuring LD with

Aside from being nonzero, what does the value of mean? This is surprisingly hard to interpret because the minimum and maximum value of is different for every pair of SNPs.

Why does the range of change?

The possible values of depend on the frequencies of the alleles at each SNP. For example:

* If and , then is between
* If and , then is between

The statistic fixes this issue by dividing by its theoretical maximum. is constrained between , where more extreme values denote stronger LD.

$$
D' = \frac{D}{\mathrm{max}(-p\_1 (1-p\_1), -q\_1 (1-q\_1))}, \mathrm{\:for\:} D < 0 \\
D' = \frac{D}{\mathrm{min}(p\_1 (1-p\_1), q\_1 (1-q\_1))}, \mathrm{\:for\:} D > 0
$$

and are the frequencies of the alleles at SNP1 and SNP2.

Use this formula to calculate for our two SNPs of interest.

Because is positive, we use the second formula for . First, we need to find the denominator, which is the minimum of and .

p1 \* (1-p1) # pmin

## [1] 0.2138636

q1 \* (1-q1) # qmin

## [1] 0.2458915

p1 \* (1-p1) is smaller, so we plug that into our formula:

Dprime <- D / (p1 \* (1-p1))  
Dprime

## [1] 0.6586931

This tells us that LD between these two SNPs is 65.9% of its theoretical maximum.

## 4.11 Measuring LD with

is the most common statistic for measuring LD. Its value ranges from [0, 1], where 1 indicates maximum LD.

Although it looks similar to the formulas for and , is actually derived from the **correlation coefficient** of the frequencies of SNP1 and SNP2, and has a slightly different interpretation:

* and measure whether recombination has occurred between two alleles
* measures how well we can predict the allele at one locus if given the allele at the other locus

Calculate for our two SNPs of interest.

r2 <- D^2 / (p1\*(1-p1)\*q1\*(1-q1))  
r2

## [1] 0.3773631

, indicating that these SNPs are in moderate LD.

## 4.12 Conclusion

In this lab, we used genotype data from the 1000 Genomes Project to ask whether there is **linkage disequilibrium** between two SNPs on chr21.

* We looked at the structure of a **VCF**, the file format that all genotype data is stored in.
* Using data from the VCF, we used table to count how often we observe combinations of alleles at these SNPs.
* We used the data in the table to calculate three LD statistics:
  + : the deviation of the observed haplotype frequency from the expected haplotype frequency
  + : a normalization of that ranges from
  + : how well the allele at one locus predicts the allele at another locus

#### 4.12.0.1 More LD resources

Check out the web application [**LDlink**](https://ldlink.nci.nih.gov/), which allows you to compute and visualize linkage disequilibrium using data from the 1000 Genomes Project (the same data you have been using).

The tool **LDproxy**, for example, can find *all* SNPs in strong LD with a SNP of interest. The tool **LDpair** can compute and between pairs of SNPs. (If you look up the two SNPs we used for class today, how do LDpair’s values compare to the ones we calculated by hand?)

## 4.13 Homework

#### 4.13.0.1 Goals & Learning Objectives

The goal of this homework is to calculate and interpret LD statistics for two new SNPs.

**Learning Objectives**

* Practice calculating and interpreting LD statistics

## 4.14 Required homework

We’ve subset the VCF from class to show haplotypes for a different pair of SNPs (chr21:15336586 and chr21:15336794). Run this code to read it in:

# read data  
haplotypes2 <- read.table("snp\_haplotypes\_hw.txt", header = TRUE)  
  
# preview data  
head(haplotypes2)

## sample haplotype snp1\_allele snp2\_allele  
## 1 HG00096 hap\_1 A A  
## 2 HG00096 hap\_2 G G  
## 3 HG00097 hap\_1 A A  
## 4 HG00097 hap\_2 A A  
## 5 HG00099 hap\_1 A A  
## 6 HG00099 hap\_2 A A

**Assignment:** Using the code from class, calculate , , and for these two SNPs. Which alleles are segregating together? What does each LD statistic indicate? (Feel free to check your work on [LDpair](https://ldlink.nci.nih.gov/?tab=ldpair), although the exact values may be slightly different.)

Solution

First use table to count the occurences of the four haplotypes.

table(haplotypes2$snp1\_allele, haplotypes2$snp2\_allele)

##   
## A G  
## A 3522 0  
## G 0 1486

It looks like the haplotypes that exist in this population are A A and G G.

#### 4.14.0.1

h <- 3522 / 5008  
p1 <- (3522 + 0)/5008  
q1 <- (3522 + 0)/5008  
  
D <- h - p1 \* q1  
D

## [1] 0.2086794

is non-zero, which suggests that these SNPs are in LD.

#### 4.14.0.2 (because ):

First we determine the denominator by calculating which of and is smaller:

p1 \* (1-p1)

## [1] 0.2086794

q1 \* (1-q1)

## [1] 0.2086794

The two values are exactly the same, so we can use either for the denominator. is:

Dprime <- D / (p1 \* (1-p1))  
Dprime

## [1] 1

! These SNPs are in maximum LD (no recombination has occured between them).

#### 4.14.0.3

r2 <- D^2 / (p1 \* (1-p1) \* q1 \* (1-q1))  
r2

## [1] 1

! These SNPs are in maximum LD (everyone who carries an A at SNP1 has an A at SNP2, and everyone with a G at SNP1 has a G at SNP2).

# Authors

| Credits | Names |
| --- | --- |
| **Pedagogy** |  |
| Instructor | [Rajiv McCoy](https://mccoy-lab.org/) |
| Content Author | [Stephanie Yan](https://stephaniemyan.github.io/) |
| Content Author | [Kate Weaver](https://kweav.github.io/) |
| **Website** |  |
| Template | [Jeff Leek](https://jtleek.com/) & [The Johns Hopkins Data Science Lab](https://jhudatascience.org/index.html) |
| Design Inspiration | [Ali Madooei](https://engineering.jhu.edu/faculty/ali-madooei/) & [JHU Data Structures](https://cs226sp22.github.io/) |
| **Funding** |  |
| JHU Center for Educational Resources | Techology Fellowship Grant |

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value   
## version R version 4.0.2 (2020-06-22)  
## os Ubuntu 20.04.3 LTS   
## system x86\_64, linux-gnu   
## ui X11   
## language (EN)   
## collate en\_US.UTF-8   
## ctype en\_US.UTF-8   
## tz Etc/UTC   
## date 2023-02-01   
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date lib source   
## assertthat 0.2.1 2019-03-21 [1] RSPM (R 4.0.3)   
## backports 1.1.10 2020-09-15 [1] RSPM (R 4.0.2)   
## blob 1.2.1 2020-01-20 [1] RSPM (R 4.0.3)   
## bookdown 0.24 2022-02-15 [1] Github (rstudio/bookdown@88bc4ea)   
## broom 0.7.1 2020-10-02 [1] RSPM (R 4.0.2)   
## callr 3.4.4 2020-09-07 [1] RSPM (R 4.0.2)   
## cellranger 1.1.0 2016-07-27 [1] RSPM (R 4.0.3)   
## cli 2.0.2 2020-02-28 [1] RSPM (R 4.0.0)   
## colorspace 1.4-1 2019-03-18 [1] RSPM (R 4.0.0)   
## crayon 1.3.4 2017-09-16 [1] RSPM (R 4.0.0)   
## data.table 1.13.0 2020-07-24 [1] RSPM (R 4.0.2)   
## DBI 1.1.0 2019-12-15 [1] RSPM (R 4.0.0)   
## dbplyr 1.4.4 2020-05-27 [1] RSPM (R 4.0.0)   
## desc 1.2.0 2018-05-01 [1] RSPM (R 4.0.3)   
## devtools 2.3.2 2020-09-18 [1] RSPM (R 4.0.3)   
## digest 0.6.25 2020-02-23 [1] RSPM (R 4.0.0)   
## dplyr \* 1.0.2 2020-08-18 [1] RSPM (R 4.0.2)   
## ellipsis 0.3.1 2020-05-15 [1] RSPM (R 4.0.3)   
## evaluate 0.14 2019-05-28 [1] RSPM (R 4.0.3)   
## fansi 0.4.1 2020-01-08 [1] RSPM (R 4.0.0)   
## farver 2.0.3 2020-01-16 [1] RSPM (R 4.0.3)   
## forcats \* 0.5.0 2020-03-01 [1] RSPM (R 4.0.0)   
## fs 1.5.0 2020-07-31 [1] RSPM (R 4.0.3)   
## generics 0.0.2 2018-11-29 [1] RSPM (R 4.0.0)   
## ggplot2 \* 3.3.2 2020-06-19 [1] RSPM (R 4.0.1)   
## glue 1.6.1 2022-01-22 [1] CRAN (R 4.0.2)   
## gtable 0.3.0 2019-03-25 [1] RSPM (R 4.0.3)   
## haven 2.3.1 2020-06-01 [1] RSPM (R 4.0.2)   
## highr 0.8 2019-03-20 [1] RSPM (R 4.0.3)   
## hms 0.5.3 2020-01-08 [1] RSPM (R 4.0.0)   
## htmltools 0.5.0 2020-06-16 [1] RSPM (R 4.0.1)   
## httr 1.4.2 2020-07-20 [1] RSPM (R 4.0.3)   
## jsonlite 1.7.1 2020-09-07 [1] RSPM (R 4.0.2)   
## knitr 1.33 2022-02-15 [1] Github (yihui/knitr@a1052d1)   
## labeling 0.3 2014-08-23 [1] RSPM (R 4.0.0)   
## lifecycle 1.0.0 2021-02-15 [1] CRAN (R 4.0.2)   
## lubridate 1.7.9 2020-06-08 [1] RSPM (R 4.0.2)   
## magrittr 2.0.2 2022-01-26 [1] CRAN (R 4.0.2)   
## memoise 1.1.0 2017-04-21 [1] RSPM (R 4.0.0)   
## modelr 0.1.8 2020-05-19 [1] RSPM (R 4.0.3)   
## munsell 0.5.0 2018-06-12 [1] RSPM (R 4.0.3)   
## ottrpal 0.1.2 2022-02-15 [1] Github (jhudsl/ottrpal@1018848)   
## pillar 1.4.6 2020-07-10 [1] RSPM (R 4.0.2)   
## pkgbuild 1.1.0 2020-07-13 [1] RSPM (R 4.0.2)   
## pkgconfig 2.0.3 2019-09-22 [1] RSPM (R 4.0.3)   
## pkgload 1.1.0 2020-05-29 [1] RSPM (R 4.0.3)   
## prettyunits 1.1.1 2020-01-24 [1] RSPM (R 4.0.3)   
## processx 3.4.4 2020-09-03 [1] RSPM (R 4.0.2)   
## ps 1.3.4 2020-08-11 [1] RSPM (R 4.0.2)   
## purrr \* 0.3.4 2020-04-17 [1] RSPM (R 4.0.3)   
## R.methodsS3 1.8.1 2020-08-26 [1] CRAN (R 4.0.2)   
## R.oo 1.24.0 2020-08-26 [1] CRAN (R 4.0.2)   
## R.utils 2.11.0 2021-09-26 [1] CRAN (R 4.0.2)   
## R6 2.4.1 2019-11-12 [1] RSPM (R 4.0.0)   
## Rcpp 1.0.8 2022-01-13 [1] CRAN (R 4.0.2)   
## readr \* 1.4.0 2020-10-05 [1] RSPM (R 4.0.2)   
## readxl 1.3.1 2019-03-13 [1] RSPM (R 4.0.2)   
## remotes 2.2.0 2020-07-21 [1] RSPM (R 4.0.3)   
## reprex 0.3.0 2019-05-16 [1] RSPM (R 4.0.0)   
## rlang 0.4.10 2022-02-15 [1] Github (r-lib/rlang@f0c9be5)   
## rmarkdown 2.10 2022-02-15 [1] Github (rstudio/rmarkdown@02d3c25)  
## rprojroot 2.0.2 2020-11-15 [1] CRAN (R 4.0.2)   
## rstudioapi 0.11 2020-02-07 [1] RSPM (R 4.0.0)   
## rvest 1.0.1 2022-02-15 [1] Github (tidyverse/rvest@4fe39fb)   
## scales 1.1.1 2020-05-11 [1] RSPM (R 4.0.3)   
## sessioninfo 1.1.1 2018-11-05 [1] RSPM (R 4.0.3)   
## stringi 1.5.3 2020-09-09 [1] RSPM (R 4.0.3)   
## stringr \* 1.4.0 2019-02-10 [1] RSPM (R 4.0.3)   
## testthat 3.0.1 2022-02-15 [1] Github (R-lib/testthat@e99155a)   
## tibble \* 3.0.3 2020-07-10 [1] RSPM (R 4.0.2)   
## tidyr \* 1.1.2 2020-08-27 [1] RSPM (R 4.0.2)   
## tidyselect 1.1.0 2020-05-11 [1] RSPM (R 4.0.3)   
## tidyverse \* 1.3.0 2019-11-21 [1] RSPM (R 4.0.3)   
## usethis 2.1.5.9000 2022-02-15 [1] Github (r-lib/usethis@57b109a)   
## vctrs 0.3.4 2020-08-29 [1] RSPM (R 4.0.2)   
## withr 2.3.0 2020-09-22 [1] RSPM (R 4.0.2)   
## xfun 0.26 2022-02-15 [1] Github (yihui/xfun@74c2a66)   
## xml2 1.3.2 2020-04-23 [1] RSPM (R 4.0.3)   
## yaml 2.2.1 2020-02-01 [1] RSPM (R 4.0.3)   
##   
## [1] /usr/local/lib/R/site-library  
## [2] /usr/local/lib/R/library