Analyzing sequencing data in the cloud

Class 16: Obtaining and processing SRA datasets on AWS

AUTHOR
Barry Grant

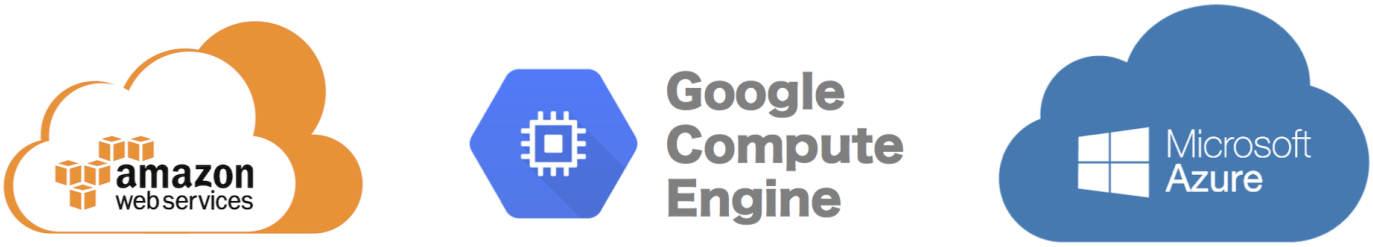
PUBLISHED
March 1, 2023

Background

In bioinformatics we often need to analyze datasets that are too large (or would take too long) to analyze on our local lab computers. The goal of this hands-on session is to show you how to obtain and work with large biomolecular sequencing datasets using **cloud computing** resources. In particular, we will query, download, decompress and analyze data from NCBI's main **Sequence Read Archive** (SRA) the worlds largest publicly available repository of high throughput sequencing data.

What is cloud computing?

Cloud computing allows access to arbitrary amounts of compute resources that are physically located elsewhere. Typically you pay for what you use rater than having to pay upfront for new hardware and all its associated maintenance costs. Cloud computing resources exist on servers managed by cloud providers. The most popular cloud providers include Amazon (**Amazon Web Services**, a.k.a. **AWS**), Google (**Google Compute Engine**) and Microsoft (**Azure**). For academic work in the US we also have NSF/XSEED (who manage access to JetStream and other supercomputers around the country).



Major cloud compute providers. At the time of writing Amazon's **AWS** is the market leader by a rather wide margin.

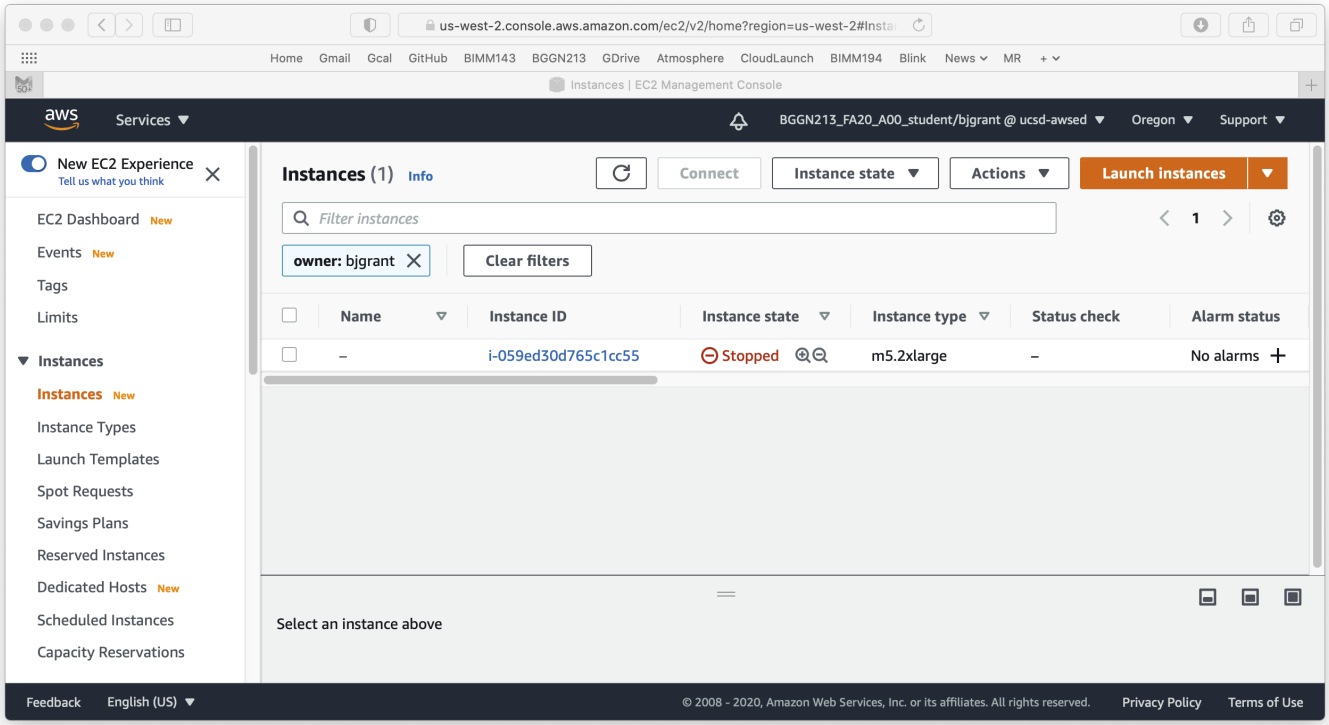
In our last hands-on session we introduced important cloud concepts and walked through configuring and launching **AWS EC2 instances**.

Accessing the AWS console

The AWS console is a password protected website where you can can configure, launch and control your EC2 instances. We introduced this console last day and we will not re-cover that material here beyond the

instructions for starting a new

If you are an enrolled student in this course you can access your own AWS console at https://awsed.ucsd.edu/ This will ask for your regular UCSD single-sign-on details and then you should be able to select our course and be re-directed to the AWS console:



Briefly, click the orange **Launch instances** button in the upper left and on the resulting launch page:

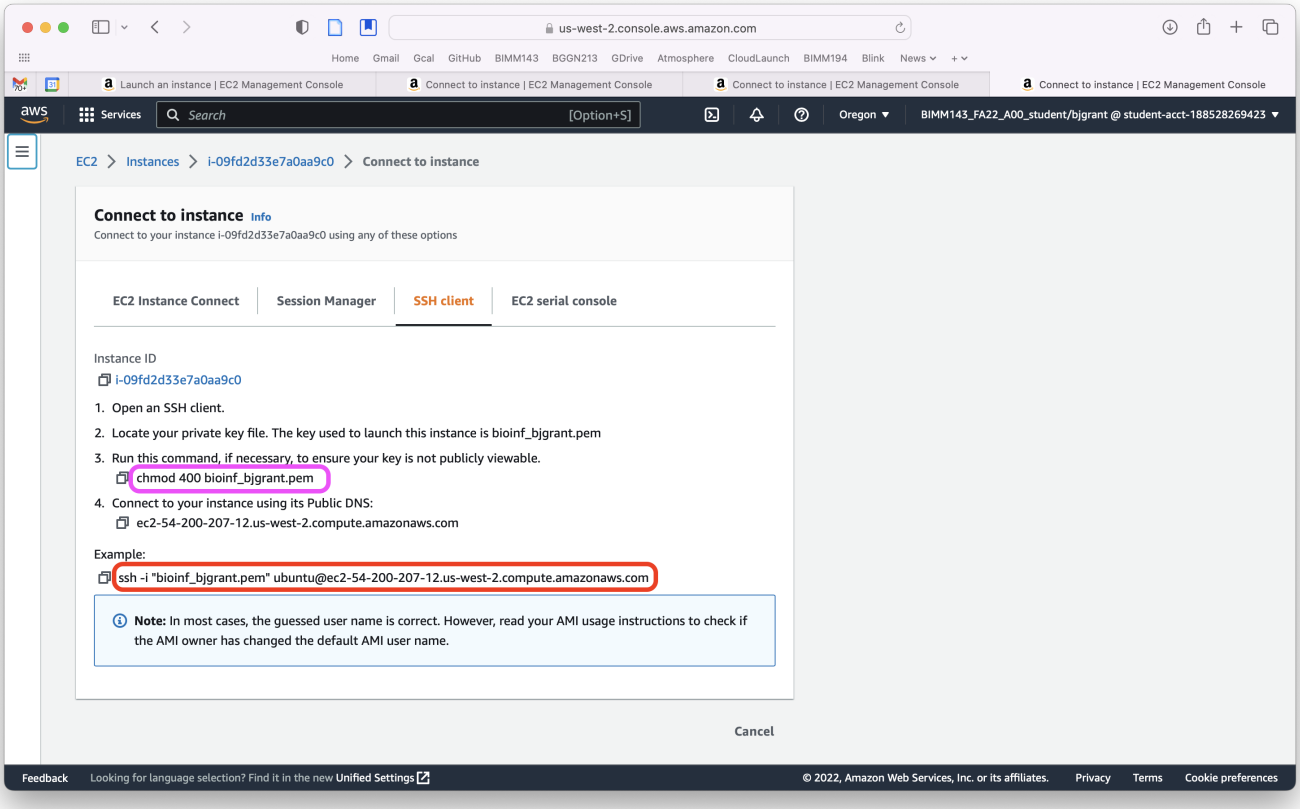
- Give a useful **name** for your instance (e.g. a combination of class number and your UCSD username, for example: `class16_bjgrant`);
- Select **Ubuntu** as the OS and the version **Ubuntu Server 20.04**
- Instance type **m5.2xlarge**
- You can use your previous **key pair** from the last class or (if in doubt about where to find it) *create a new one* and use it.
- Select **existing security group** and chose **BIMM143/BGGN213 FA20**
- Configure storage with 100Gib
- Verify all your settings in the right hand panel and click "**Launch instance**".

Conecting to your new instance

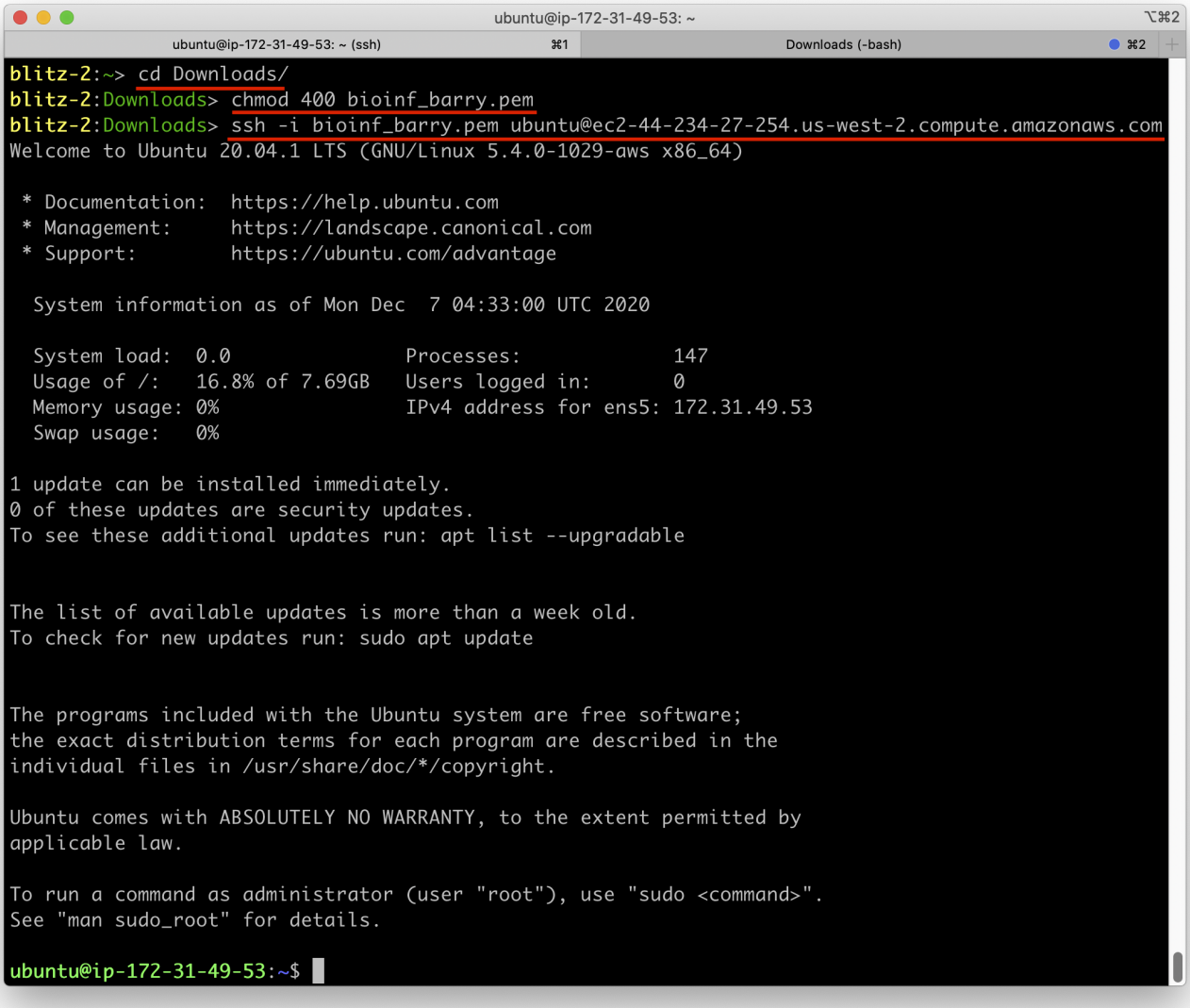
After configuring and launching your instance on the last page it will take a short amount of time to setup and become available for use.

Once available you can login via **SSH** in your Terminal as we did previously.

- Recall that you need to provide your **keyfile** along with your new instance IP address. See the "**Connect to instance**" link and "SSH client" link for instructions and details of your specific instance IP address (e.g. the boxed text in the example image below):



- You can use your favorite Terminal application or the RStudio Terminal to enter the appropriate SSH command. For example, in the below image I am using my keyfile `-i bioinf_barry.pem` along with my username `ubuntu` and the EC2 address provided from th AWS console:



Note that if it is your first time connecting to your instance it will ask if you are sure you want to connect to which you can answer `yes`.

Now that we have successfully gained access to a UNIX based machine with sufficient memory and cpu resources we can introduce the main Sequence Read Archive - curently the largest publicly available repository of high throughput sequencing data.

The Sequence Read Archive (SRA)

The **Sequence Read Archive** (SRA), https://www.ncbi.nlm.nih.gov/sra is NCBI's major repository for sequencing data and includes all types of high-throughput sequencing experiments.

Many other domain specific databases will link to entries in the SRA. For example the **Gene Expression Omnibus** (GEO) that focuses on gene expression data links to SRA for retrieval of raw data from sequencing-based experiments whose processed data is described in a *GEO record*. To get the actual raw data you need to go to the associated SRA entry.

Sequencing data stored on SRA is highly-compressed to enable efficient storage and downloading. The main consequences of this, beyond faster download speeds, is that *we must use special tools to decompress these records into usable FASTQ files*.

The SRA-toolkit package

The **SRA-toolkit** software package provides functionality to retrieve data from the SRA, and to extract the retrieved records to FASTQ files.

Instructions for obtaining the package on different platforms can be found on the projects GitHub repo. Note that precompiled binaries are available for Ubuntu that we will make use of here.

- On your AWS instance download the Ubuntu binaries with the `curl` command. Then unzip and untar to be able to use the software:

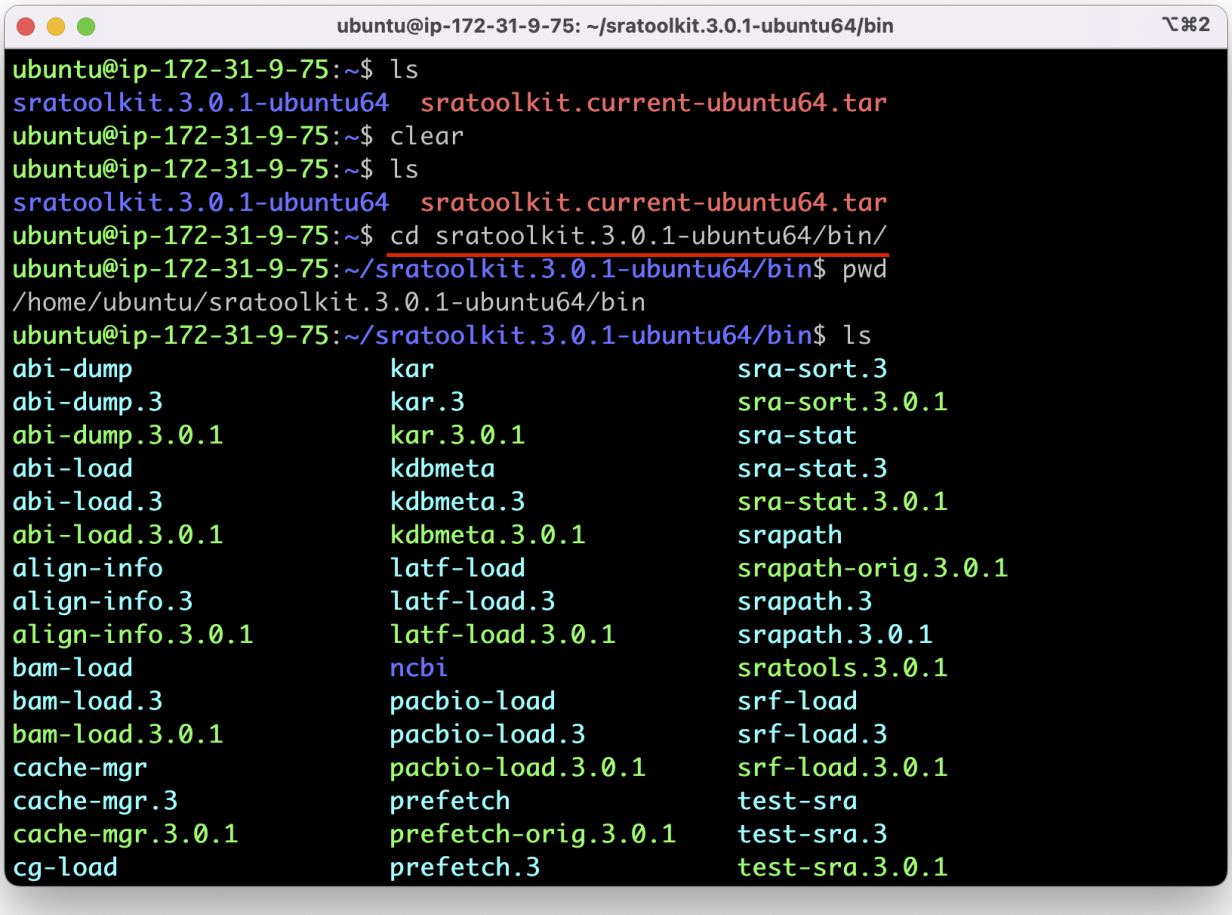
```
On your AWS instance
```

```
# Download
curl -O https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz

gzip -d sratoolkit.current-ubuntu64.tar
tar -xvf sratoolkit.current-ubuntu64.tar
```

> Note the name of the directory that tar created. The name of this directory changes with each release and varies by platform, i.e. it follows the pattern `sratoolkit.<release>-<platform>` e.g. at the time of writting a directory called `sratoolkit.3.0.1-ubuntu64` was produced.

There should be a new directory called `sratoolkit.3.0.1-ubuntu64` (or a similar name with a higher number as the version changes in the future). Check for this directory with your regular `ls` command. Then `cd` into the associated `bin/` directory, e.g.



On AWS

```
ls
cd sratoolkit.3.0.7-ubuntu64/bin
pwd
```

In the `bin/` directory there are lots of programs that we can use to work with SRA data. We will focus on two main programs - **prefetch** and **fastq-dump**.

The first of these, **prefetch**, retrieves compressed data from SRA. The second, **fastq-dump**, reconstructs the actual FASTQ file(s) from the output of prefetch that we want to analyze.

We can check if these programs are working by running them with the `--version` or `--help` options.

For example:

On AWS

```
$ ~/sratoolkit.3.0.1-ubuntu64/bin/prefetch --version

/home/ubuntu/sratoolkit.3.0.1-ubuntu64/bin/prefetch : 3.0.1
```

Note that even though I can use the TAB key to help it is still cumbersome to type the full PATH to the prefetch program every time we want to run it. To avoid needing to do this we can add it's location to a special `$PATH` environment variable:

So in the command below, it is saying add /home/ubuntu... to the main $PATH directory (which has lots of extra commands like ls that doesn't require you to be in a particular directory). Then call the whole thing path again (making a new vector)

On AWS In terminal Type the below

```
$ export PATH=$PATH:/home/ubuntu/sratoolkit.3.0.7-ubuntu64/bin
```

Now we will not need to type the absolute PATH every time we want to run one of the sra-tools programs from it's `bin/` directory, e.g. this will now just work

On AWS

```
$ prefetch --version

prefetch : 3.0.7
```

With this setup complete we are now ready to get some sequencing data to work with.

A mini example: from a given paper to raw reads

Let's say we wanted to obtain the ChIP-seq data from an ovarian cancer DNA sample, such as that cited in:

> N Chapman-Rothe et al, "Chromatin H3K27me3/H3K4me3 histone marks define gene sets in high-grade serous ovarian cancer that distinguish malignant, tumour-sustaining and chemo-resistant ovarian tumour cells" Oncogene 32(38):4586 (2013).

Visit the full text link for this manuscript from the link above. Note that the manuscript lists an **SRA accession** for the project of '*SRP016075*' under the **Accession codes** section.

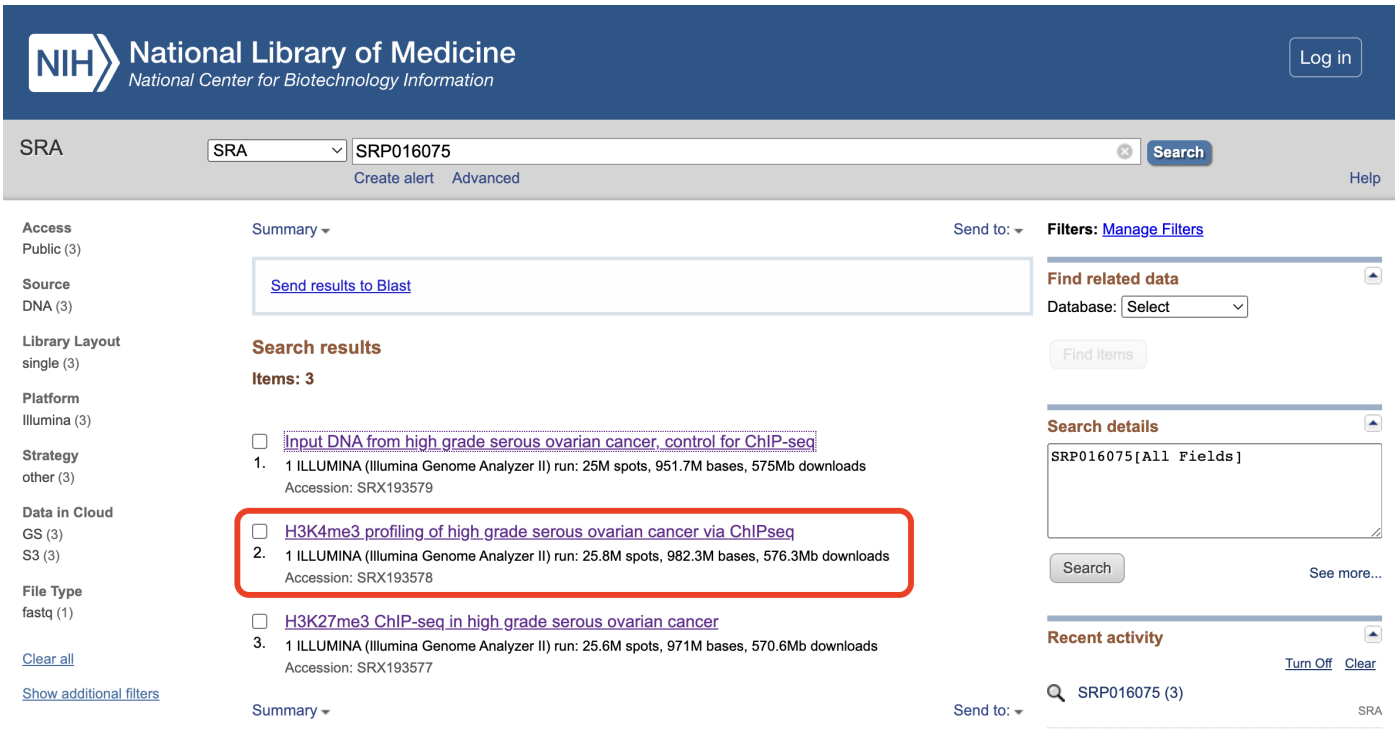
Searching for the corresponding record on the SRA database, we will find a list of the sequencing runs associated with this manuscript.

Some other commands:

$ ^Z # this command will put the currently running program to sleep

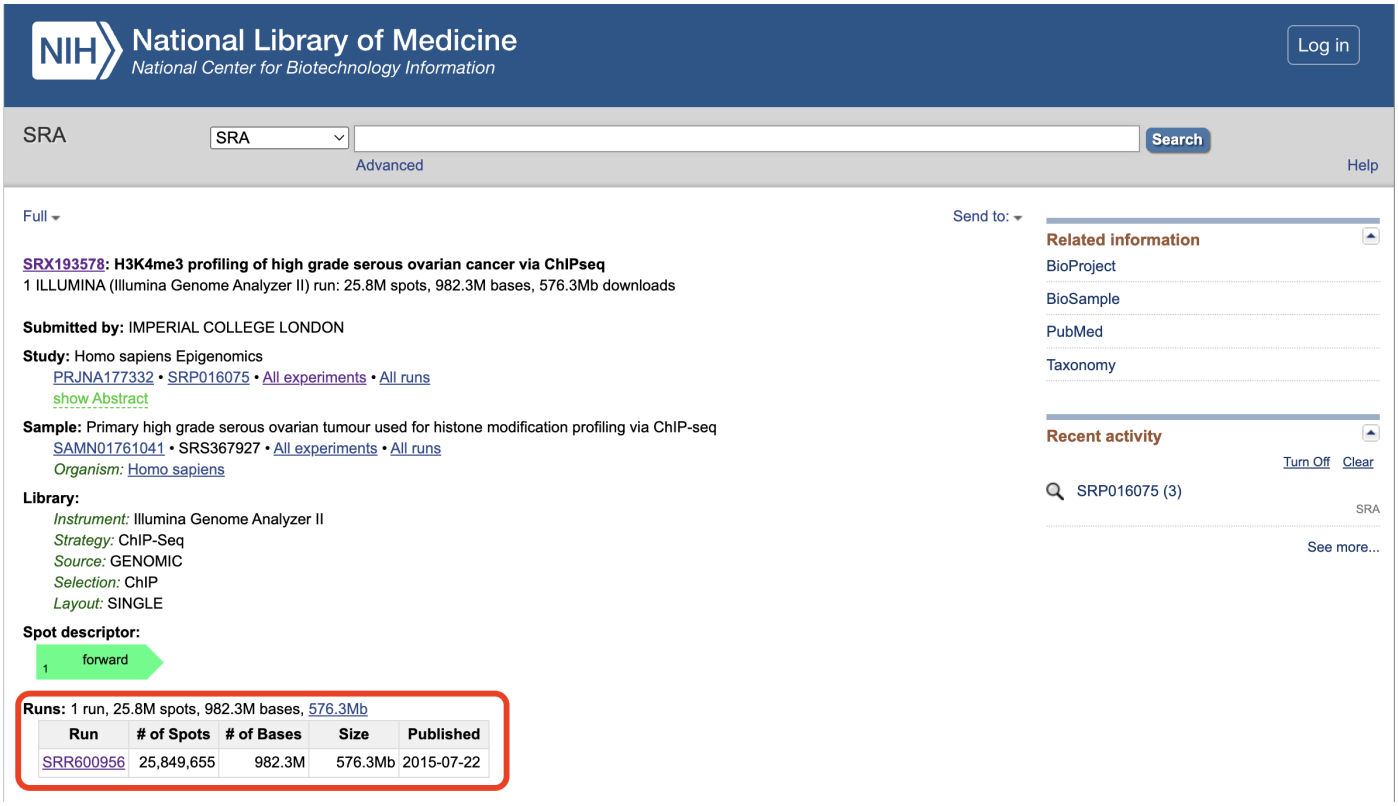
$ ps # this command will show current programs running

$ bg # this command will run things in the background.



< >

Specifically look at the ChIP-seq with the H3K4me3 antibody, which can be found at: https://www.ncbi.nlm.nih.gov/sra/SRX193578. See below image:



From the corresponding web page (image above), there is a run identifier listed: '**SRR600956**'. This is the SRA identifier we need to obtain the raw sequencing data with the `prefetch` program.

To download the corresponding data, call the `prefetch` program (using either it's absolute path `~/sratoolkit.3.0.1-ubuntu64/bin/` or if you have followed the instructions above simply it's name) with the run accession number as its only argument:

On AWS

```
$ cd
$ pwd
/home/ubuntu

$ prefetch SRR600956

2023-02-27T20:38:08 prefetch.3.0.1: Current preference is set to retrieve SRA Normalized Format f
2023-02-27T20:38:08 prefetch.3.0.1: 1) Downloading 'SRR600956'...
2023-02-27T20:38:08 prefetch.3.0.1: SRA Normalized Format file is being retrieved, if this is dif
2023-02-27T20:38:08 prefetch.3.0.1:  Downloading via HTTPS...
2023-02-27T20:38:27 prefetch.3.0.1:  HTTPS download succeed
2023-02-27T20:38:28 prefetch.3.0.1:  'SRR600956' is valid
2023-02-27T20:38:28 prefetch.3.0.1: 1) 'SRR600956' was downloaded successfully
```

Run an `ls` to see the resulting directory - note that the contents here is compressed and not yet usable for further analysis.

To btain usable raw data we first need to reconstruct the FASTQ file(s) with the **fastq-dump** program:

On AWS

```
$ fastq-dump SRR600956
```

This will take some time but eventually result in a FASTQ file, appearing in your current working directory, which constitutes the raw data from the corresponding sequencing run.

You can use the `head` command to see the start of the resulting FASTQ file.

On AWS

```
$ head SRR600956.fastq

@SRR600956.1 HWI-EAS486_0002:3:1:1382:1342 length=38
GTGTTCCAAATGCTGCAAATGGGTGTCAATGTATGTTA
+SRR600956.1 HWI-EAS486_0002:3:1:1382:1342 length=38
D?BCCA?BDBDBACD@=??BAAC>CBBBBBCBBBD?%%
@SRR600956.2 HWI-EAS486_0002:3:1:1382:5487 length=38
GATGATAGTTTCTTTTGCCGTTAGCACAATTTTTCCAA
+SRR600956.2 HWI-EAS486_0002:3:1:1382:5487 length=38
DCEECEAECEFFDFECEEEFFFFDB?BBADEEEEE???
@SRR600956.3 HWI-EAS486_0002:3:1:1382:4694 length=38
TGTAGGCTCCACCTCTGGGGGCAGGGCACAGACAAACA
```

SRR600956.1 means that it is the first gene.
SRR600956.2 means that it is the second gene.
ect.

Note that each new read starts with the pattern `@SRR600956`. Let's use `grep` with this pattern to determine how many total reads are in this file:

```
On AWS grep will find everything with a particular pattern. the -c tells it to count it
```

```
$ grep -c "@SRR600956" SRR600956.fastq

25849655
```

If you do tail -5 SRR600956.fastq it will show you the last 5 lines in the fastq file. the # after the SRR600956. will tell you the last number of genes which should match the number you see here (25849655).

Working with RNA-Seq data

In this section we will move on to an arguably more interesting RNA-Seq dataset from a human cancer cell line: two replicates of a cell line treated with CRISPR-Cas9 targeting a gene of interest, and two replicates of the cell line treated with a control construct as described in:

> Zhang et al. "Identification of focally amplified lineage-specific super-enhancers in human epithelial cancers". Nat Genet 2016 Feb;48(2):176-82. PMID: 26656844

We will start with their SRA accession numbers, download their raw data and follow up with a complete analysis from raw reads to gene specific counts, principal component analysis and differential expression analysis.

The GEO entry for the study in question can be found at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE72001

Let's use the SRA toolkit to download the study's data and extract it into FASTQ format:

> **Side-note**: With single-end reads we will have only one file for each sample, with paired-end reads (which are more common in RNA-seq) we will have two files for each sample as we have here.

```
On AWS
```

```
$ prefetch SRR2156848

2023-02-27T20:47:07 prefetch.3.0.1: Current preference is set to retrieve SRA Normalized Format f
2023-02-27T20:47:07 prefetch.3.0.1: 1) Downloading 'SRR2156848'...
2023-02-27T20:47:07 prefetch.3.0.1: SRA Normalized Format file is being retrieved, if this is dif
2023-02-27T20:47:07 prefetch.3.0.1:  Downloading via HTTPS...
2023-02-27T20:47:14 prefetch.3.0.1:  HTTPS download succeed
2023-02-27T20:47:15 prefetch.3.0.1:  'SRR2156848' is valid
2023-02-27T20:47:15 prefetch.3.0.1: 1) 'SRR2156848' was downloaded successfully
2023-02-27T20:47:15 prefetch.3.0.1: 'SRR2156848' has 0 unresolved dependencies
```

Recall that the `prefetch` program prepares temporary files for obtaining sequence data efficiently from the SRA. With this run, you can use `fastq-dump` to extract the data in FASTQ format to a file in the current

directory.

On AWS

```
$ fastq-dump --split-3 SRR2156848

Read 2959900 spots for SRR2156848
Written 2959900 spots for SRR2156848
```

Since we have paired end reads we need to split the file into two (one for each read direction). If you don't add --split-3 then it will merge it to a single file. As for why it is --split-3, that is because the people who made this program decided that name. There is another split file option if the -3 part bothers you.

Note that as this sequencing run is a paired-end library, this command uses the `--split-3` argument to indicate that we wish to extract separate files for the different mate-pairs. Now if we look at the contents of the current directory, we should see the following:

On AWS

```
$ ls

SRR2156848/
SRR2156848_1.fastq
SRR2156848_2.fastq
SRR2156848/
SRR2156848.fastq
```

> Q. How would you check that these files with extension '.fastq' actually look like what we expect for a FASTQ file? You could try printing the first few lines to the shell standard output:

On AWS

```
# Fill in the blank!
head SRR2156848_1.fastq
```

You can also use grep - c on both files to make sure that they are the same lenght

> Q. How could you check the number of sequences in each file?

On AWS

```
# Fill in the blanks
$ grep -c "@SRR" SRR2156848_1.fastq
2959900
```

> Q. Check your answer with the bottom of the file using the `tail` command and also check the matching mate pair FASTQ file. Do these numbers match? If so why or why not?

- Now let's download the other 3 datasets (`SRR2156849`, `SRR2156850` and `SRR2156851`) we need for our analysis, first with `prefetch` and then process with `fastq-dump`

On AWS

```
$ prefetch SRR2156849 SRR2156850 SRR2156851
$ fastq-dump --split-3 SRR2156849 SRR2156850 SRR2156851
```

- Check you have pairs of FASTQ files for all four datasets.

On AWS

```
# fill in the blank
ls *.fastq
```

Transcript quantification via pseudoalignment

A fast approach to analysis of RNA-seq data makes use of the fact that if you only wish to quantify the expression of a gene, you don't need to know where in the gene the read maps to. In fact, it is enough to know which genes a read might have come from. This is the principle behind **Kallisto**. In brief, Kallisto compares each RNA-seq read against an indexed table of transcript sequences to find the sets of transcripts that each read could have come from. These sets are then used to estimate the most likely set of transcript counts over the whole RNA-seq library.

- To begin with we need to install kallisto software on our AWS instance:

On AWS

```
wget https://github.com/pachterlab/kallisto/releases/download/v0.44.0/kallisto_linux-v0.44.0.tar.

# Unzip and Untar the resulting file
gzip -d kallisto_linux-v0.44.0.tar.gz        Then $ tar -xvf kallisto_linux-v0.44.0.tar
```

We can check the program works by calling the `kallisto` executable with it's full PATH:

On AWS

```
$ ~/kallisto_linux-v0.44.0/kallisto

kallisto 0.44.0

Usage: kallisto <CMD> [arguments] ..

Where <CMD> can be one of:

    index         Builds a kallisto index
    quant         Runs the quantification algorithm
    pseudo        Runs the pseudoalignment step
    h5dump        Converts HDF5-formatted results to plaintext
    inspect       Inspects and gives information about an index
    version       Prints version information
    cite          Prints citation information
```

```
Running kallisto <CMD> without arguments prints usage information for <CMD>
```

- For ease of use and to avoid having to type the full PATH each time we want to use the program we can add it's location to our $PATH environment variable as we did previously for the **sra-tools** package:

On AWS

```
# Fill in the blank
$ export PATH=$PATH:/home/ubuntu/kallisto_linux-v0.4
```

Now we should be able to run it by simply calling `kallisto`.

> Q. Can you run `kallisto` to print out it's citation information?

Building a transcript index

Let's pull the **hg19 Ensembl** reference transcriptome directly from the ENSEMBLE ftp site:

- We can use the `wget` command or `curl` command for this.

On AWS

```
$ wget ftp://ftp.ensembl.org/pub/release-67/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh37.67.cdna.a
```

- Now we need to unzip the downloaded file:

On AWS

```
$ gunzip Homo_sapiens.GRCh37.67.cdna.all.fa.gz
```

- Now we can use Kallisto's index program to build the transcriptome index (N.B. this can take some time):

On AWS

```
$ kallisto index -i hg19.ensembl Homo_sapiens.GRCh37.67.cdna.all.fa
```

This command specified that we give this transcriptome index the name `hg19.ensembl`. *This will take a little while*.

Once complete we will have an *indexed human genome* to match the reads against, we can then use Kallisto for fast transcript quantification on our FASTQ files.

Quantifying transcripts

To quantify transcripts with Kallisto we use the `quant` command option, specifying a transcript index (with the `-i` flag, an output directory (with the `-o` flag) in which to write results to file, and a list of our Fastq

input files:

- Run transcript quantification for the pair of SRR2156848 FASTQ files:

On AWS asking tho quantify, telling it to output it into -o SRR2156848_quant. Then the two files you want in it

```
$ kallisto quant -i hg19.ensembl -o SRR2156848_quant SRR2156848_1.fastq SRR2156848_2.fastq
```

This again will take some time (but not nearly as long as the alignment based mapping we did on our previous galaxy lab).

- Once run correctly, there should be three new files in the `SRR2156848_quant` directory:

On AWS

```
ls SRR2156848_quant
abundance.h5  abundance.tsv  run_info.json
```

We can now run the quantification for the remaining samples:

On AWS you can also make a nano text file using $ nano name. Then you can copy and paste these. If you add an & at the end of each line it will run them sequentially for you. Then can type $ sh text file name

```
# You complete these steps
$ kallisto quant -i ___ -o SRR2156849_quant SRR2156849_1.fastq SRR2156849_2.fastq

$ kallisto quant -i hg19.ensembl -o SRR2156850_quant ___.fastq ___.fastq

$ kallisto quant -i hg19.ensembl -o SRR2156851_quant SRR2156851_1.fastq SRR2156851_2.fastq
```

You should now have a `*_quant/` directory for each sample.

> Q. Have a look at the TSV format versions of these files to understand their structure. What do you notice about these files contents?

If you accidentally quit, just go back to your terminal and enter $ ssh -i ~/Downloads/"bioinf_krysten.pem" ubuntu@ec2-34-223-23-207.us-west-2.compute.amazonaws.com.

Transfer your results

With the transcript quantification completed, we are now finished our UNIX intensive work on AWS.

We can thus transfer our main results files to date (the various `abundance.tsv` files) back to our laptop for further analysis.

- On your computer (i.e. open a new Terminal) navigate to a directory where you want to store your results and use `scp` to transfer all your abundance results to this directory.

On YOUR computer

```
# Make sure you are accessing your keyfile and instance
```

Open new terminal by clicking on the word terminal and choosing "new" from the dropdown. Then make sure to move back to the class 18 directory (which we put on the desktop) using the $ cd command.

```
scp -r -i "~/Downloads/bimm143w23.pem" ubuntu@ec2-34-219-113-54.us-west-2.compute.amazonaws.com:~
```

For mine, its going to be scp -r -i ~/Downloads/bioinf_krysten.pem $aws:~/*quant .
always source to destination, so the downoads folder is our source, since we want to save to the same then you can just use "." instead of typing the same destination again. "-r" search for all the files in the directory to copy.

Downstream analysis

Back on our laptop we can now use R and Bioconductor tools to further explore this large scale dataset.

For example there is an R function called `tximport()` in the **tximport** package, which enables straightforward import of Kallisto results

With each sample having its own directory containing the Kallisto output, we can import the transcript count estimates into R using:

```
# BiocManager::install("tximport")
```

```
library(tximport)

# setup the folder and filenames to read
folders <- dir(pattern="SRR21568*")
samples <- sub("_quant", "", folders)
files <- file.path( folders, "abundance.h5" )
names(files) <- samples

txi.kallisto <- tximport(files, type = "kallisto", txOut = TRUE)
```

```
1 2 3 4
```

```
head(txi.kallisto$counts)
```

```
                SRR2156848 SRR2156849 SRR2156850 SRR2156851
ENST00000539570          0          0    0.00000          0
ENST00000576455          0          0    2.62037          0
ENST00000510508          0          0    0.00000          0
ENST00000474471          0          1    1.00000          0
ENST00000381700          0          0    0.00000          0
ENST00000445946          0          0    0.00000          0
```

We now have our estimated transcript counts for each sample in R. We can see how many transcripts we have for each sample:

```
colSums(txi.kallisto$counts)
```

```
SRR2156848 SRR2156849 SRR2156850 SRR2156851
   2563611    2600800    2372309    2111474
```

And how many transcripts are detected in at least one sample:

```
sum(rowSums(txi.kallisto$counts)>0)
```

```
[1] 94561
```

Before subsequent analysis, we might want to filter out those annotated transcripts with no reads:

```
to.keep <- rowSums(txi.kallisto$counts) > 0
kset.nonzero <- txi.kallisto$counts[to.keep,]
```

And those with no change over the samples:

```
keep2 <- apply(kset.nonzero,1,sd)>0
x <- kset.nonzero[keep2,]
```

Principal Component Analysis

We can now apply any exploratory analysis technique to this counts matrix. As an example, we will perform a PCA of the transcriptomic profiles of these samples.

Now we compute the principal components, centering and scaling each transcript's measured levels so that each feature contributes equally to the PCA:

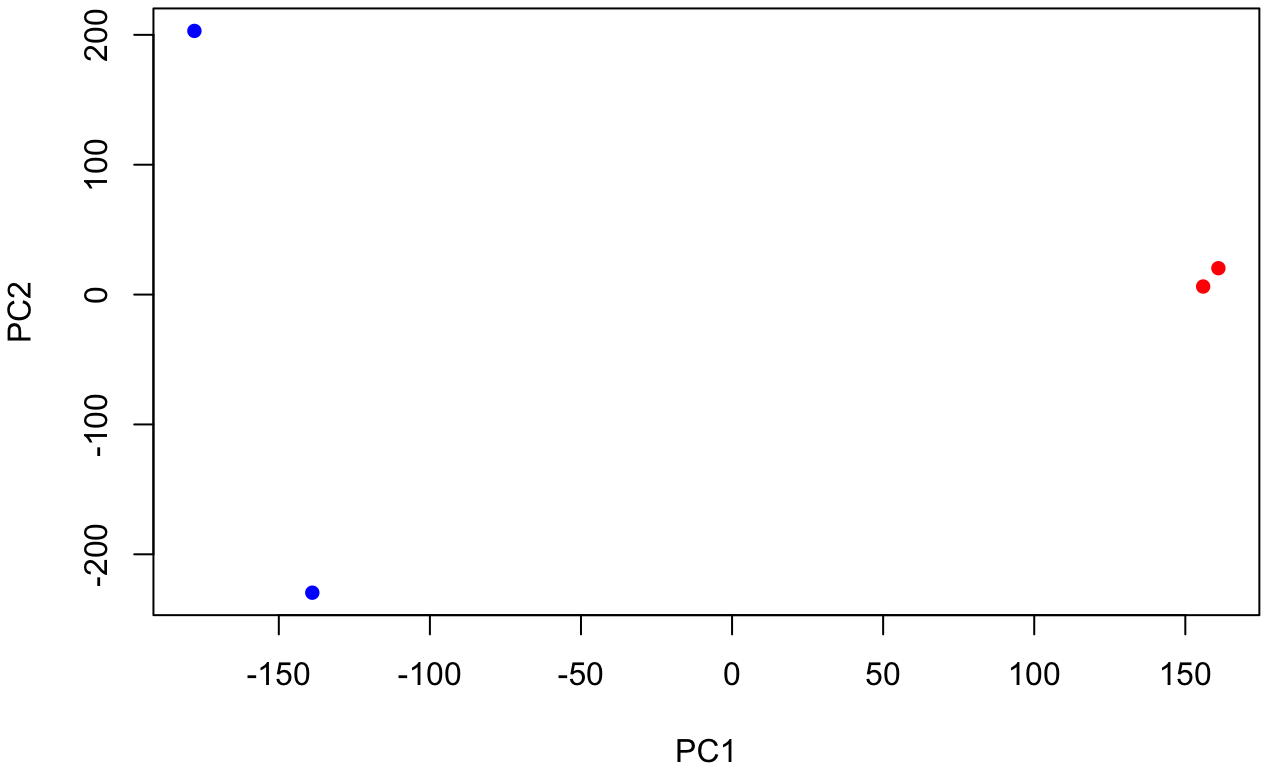
```
pca <- prcomp(t(x), scale=TRUE)
```

```
summary(pca)
```

```
Importance of components:
                          PC1      PC2      PC3   PC4
Standard deviation     183.6379 177.3605 171.3020 1e+00
Proportion of Variance   0.3568   0.3328   0.3104 1e-05
Cumulative Proportion    0.3568   0.6895   1.0000 1e+00
```

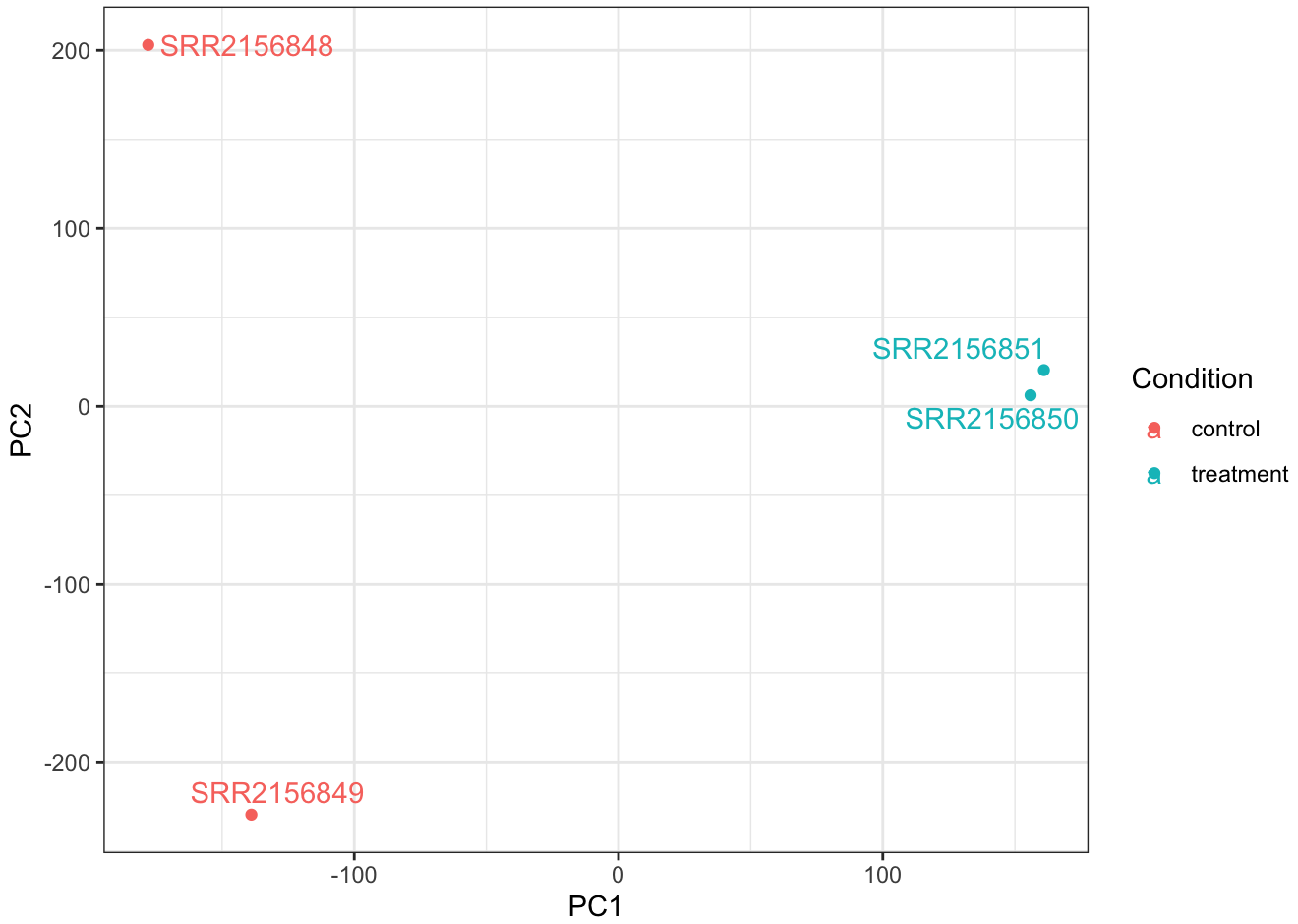
Now we can use the first two principal components as a co-ordinate system for visualizing the summarized transcriptomic profiles of each sample:

```
plot(pca$x[,1], pca$x[,2],
     col=c("blue","blue","red","red"),
     xlab="PC1", ylab="PC2", pch=16)
```



> Q. Use ggplot to make a similar figure of PC1 vs PC2 and a seperate figure PC1 vs PC3 and PC2 vs PC3.

▶ Code



The plot makes it clear that PC1 separates the two control samples (SRR2156848 and SRR2156849) from the two enhancer-targeting CRISPR-Cas9 samples (SRR2156850 and SRR2156851). PC2 separates the two control samples from each other, and PC3 separates the two enhancer-targeting CRISPR samples from each other. This could be investigated further to see which genes result in these separation patterns. It is also at least slightly reassuring, implying that there are considerable differences between the treated and control samples.

OPTIONAL: Differential-expression analysis

We can use **DESeq2** to complete the differential-expression analysis that we are already familiar with:

An example of creating a DESeqDataSet for use with DESeq2:

```
library(DESeq2)
```

```
sampleTable <- data.frame(condition = factor(rep(c("control", "treatment"), each = 2)))
rownames(sampleTable) <- colnames(txi.kallisto$counts)
```

```
dds <- DESeqDataSetFromTximport(txi.kallisto,
                                sampleTable,
                                ~condition)
```

```
using counts and average transcript lengths from tximport
```

```
# dds is now ready for DESeq() see our previous classes on this
```

```
dds <- DESeq(dds)
```

```
res <- results(dds)
head(res)
```

```
log2 fold change (MLE): condition treatment vs control
Wald test p-value: condition treatment vs control
DataFrame with 6 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENST00000539570  0.000000             NA        NA        NA        NA
ENST00000576455  0.761453       3.155061   4.86052 0.6491203  0.516261
ENST00000510508  0.000000             NA        NA        NA        NA
ENST00000474471  0.484938       0.181923   4.24871 0.0428185  0.965846
ENST00000381700  0.000000             NA        NA        NA        NA
ENST00000445946  0.000000             NA        NA        NA        NA
                     padj
                <numeric>
ENST00000539570        NA
ENST00000576455        NA
ENST00000510508        NA
ENST00000474471        NA
ENST00000381700        NA
ENST00000445946        NA
```

These results could go on to be visualized and subjected to pathway analysis etc. as we have done in previous classes.

IMPORTANT: Stop your instance!

Please **Stop** your AWS instances at this point as we want to conserve funds for future work. To stop your instance:

- Retuen to the AWS console by visiting https://awsed.ucsd.edu/,
- Select your running instance,
- Click “Instance State” > “Stop” to suspend and stop being charged for resources.